# Neuromorphic photonic systems for information processing

Nickson Mwamsojo

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAS002

Thèse de doctorat

TELECOM SudParis

# Neuromorphic Photonic Systems for Information Processing

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale
n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Réseaux, informations et communications

Thèse présentée et soutenue à Palaiseau, le 15 Fevrier 2023, par

**NICKSON MWAMSOJO**

Composition du Jury :

| | |
|---|---|
| Prénom Nom<br>Statut, Établissement (Unité de recherche) | Président |
| Laurent Larger<br>Professeur, Université de Franche-Comté (FEMTO-ST) | Rapporteur |
| Sylvain Barbay<br>Directeur de Recherche (C2N) | Rapporteur |
| Serge Massar<br>Professeur, Université Libre de Bruxelles | Examinateur |
| Frederic Lehmann<br>Professeur, Télécom SudParis (SAMOVAR) | Directeur de thèse |
| Kamel Merghem<br>Maître de Conférences, Télécom SudParis (SAMOVAR) | Co-directeur de thèse |
| Yann Frignac<br>Ingénieur de Recherche, Huawei Technologies France | Invité |

# Neuromorphic Photonic Systems for Information Processing

NICKSON MWAMSOJO

June 6, 2023

*In loving memory of my father, Robert Pomboma Mwamsojo*

*"Mwanyandeee, Pomboma, Mwakandulu!"*

# Acknowledgements

So many people have been part of my journey in different ways that my efforts to thank everyone may fall short. But I will try to be as inclusive as I can.

I acknowledge the guidance and support of my supervisor and director Prof. Frederic Lehmann. I have learnt so much from him as we faced challenges together and search routes to take to advance our works. From fostering and encouraging my ideas to the assistance with the mathematical tools and formulations, he made this journey enjoyable and fruitful.

I want to thank my co-supervisor Dr. Kamel Merghem for his valuable support in general and particularly in the experimental aspects of this thesis. He answered my questions and offered suggestions throughout my thesis. I also thank Prof. Yann Frignac and Prof. Badr-Eddine Benkelfat also for equally being a part of this journey by offering advice, recommendations and their interest in our undertakings.

I acknowldge the fruitful discussions and assistance from Prof. Mounim A. El Yacoubi and Dr. Christian Kahindo as we worked on the Alzheimer's disease detection.

Many thanks to Prof. Laurent Larger. He is part of the motivation for my embarking on this journey. His papers led to my discovery and admiration of neuromorphic computing. He even offered me an experimental setup to play with, and Oh I played!

I also thank Prof. Laurent Larger and Prof. Sylvain Barbay for reviewing my thesis; and Serge Massar for his acceptance of being part of my defense jury.

I want to thank my father, a man and a half, Mr. Robert a.k.a Mwananchi. He taught me the most valuable of lessons, from how to count and write to manhood, duty, and responsibility. I want to thank my mother, Tumpale, a tender-hearted warrioress. She made even the most difficult of times bearable. I thank my lovely sister Suma for always taking care of me, and my brother Oswald - one heck of a dude. I thank my brother Godfrey, he sat the bar high and we all followed. The support and prayer from brothers David, Elimu; the Upendos, and the rest of my family never went unnoticed.

I thank my colleagues friends for their company and listening ears during the complex and challenging times.

# Abstract

This thesis lies at the frontier between Machine Learning, particularly Artificial Neural Networks, and Photonics. It explores the implementation of neural networks on unconventional photonic hardware to propose and highlight low-complexity and efficient solutions to the existing challenges.

We are in the Information Age characterized by large and ever-growing amounts of data in various domains and forms. The vast amounts of data contain knowledge that, once extracted, could be useful in industrial, financial, and medical scenarios. Artificial Neural Networks gained traction as tools for extracting knowledge from data and their subsequent application. The general tendency has, so far, been to focus solely on improving the performance of the resultant models with little regard to the investment in energy costs and computation effort for the reported performances. We have, however, come to a realization that this is unsustainable, detrimental to the environment, and prone to cause severe bottlenecks in the near future.

We explore the applications of neural architectures implemented using photonic hardware as a direction toward breaking the aforementioned trend. We consider Reservoir Computing and Coherent Ising Machines. The two are frameworks for traditional Recurrent Neural Networks with unique characteristics. Reservoir Computing emerged about two decades ago as a robust framework simplifying traditional recurrent networks' rather complicated training. Ising machines are autonomous ground-state-seeking recurrent networks suitable for combinatorial optimization applications.

In this thesis, we focus on low complexity and efficiency in various applications. We begin by proposing a new low-complexity hyperparameter tuning technique for hardware Reservoir Computers from a system-level perspective. From an application perspective, we suggest new applications of neuromorphic photonic approaches. We complement the performance results of each investigation with the energy and computation cost analysis to accentuate the potential for less demanding and eco-friendly

implementations of neuromorphic methods. We elaborate on our exploration below.

We introduce a low complexity stochastic gradient-based method for tuning the hyperparameter of Reservoir Computers in Chapter 3. Our proposal approximates the gradients using the finite-difference method, alleviating the requirement for the functional relationship between the parameters and the system's performance. Removal of this constraint makes the automatic tuning of parameters in hardware setups not only feasible since the functional relationship is seldom known but also resistant to drifts in time for the physical settings. Our theoretical analysis, simulation, and experimental results substantiate the fact that our approach finds the optimal parameters quicker and more efficiently than the exhaustive search and Simulated Annealing, two well-known methods. We compare the computation effort and complexity of attaining the optimal performance for three tasks, the Spoken Digit, Control Charts, and Wafer classification tasks and obtain consistent results.

As an extension of our interest in low-complexity solutions, we also propose Reservoir Computing for Early Stage Alzheimer's disease detection in numerical and hardware implementations for the first time. To justify our proposal, we benchmark the performance of our Reservoir Computer with that of Bidirectional Long Short-Term Memory, Convolutional Neural Network, and the state-of-the-art method, the $k$-Medoids. Detection is carried on the full handwriting dynamics from the writing of four sets of cursive-$\ell$ on a digital tablet. Our implementation of Reservoir Computing yields $85\%$ in detection accuracy outperforming the Convolutional Neural Network and the state-of-the-art. While it under-performs by $3\%$ relative to Bidirectional Long Short-Term Memory, it does so with huge savings in terms of energy efficiency.

Furthermore, we quantify the energy costs ($kWh$), Carbon-dioxide emissions ($kg$), computation duration, and the number of floating point operations for each model. We assess these costs for the three stages i.e. optimization, training, and inference stages. We find that the Reservoir Computer requires the least amount of effort in each stage by a significant margin. For instance, the $3\%$ gain in accuracy by the Bidirectional Long Short-Term Memory comes with $8$ times the electric energy cost and the mass of $CO_2$ and equivalent Green House Gases emission according to the grid information at Palaiseau, France, where the experiments were run. This makes Reservoir Computing a more reasonable approach overall, but even more so, if the models are to be run on mobile devices, for instance, on the same tablets used for data acquisition, Reservoir Computing will give an extended battery discharge cycle. We also propose a hardware implementation that is more energy efficient as compared to the digital Reservoir computer, though with a slight penalty in terms of performance.

Withal, We investigate the Coherent Ising Machines' application to image process-

ing. In particular, we reformulate a statistical image denoising task as the minimization of an Ising Hamiltonian that can be solved using Hopfield Networks that can be implemented using either as a digital or a hardware Coherent Ising Machine. Compared to standard Simulated Annealing, we show that the proposed approach leads to a favorable tradeoff in terms of runtime/complexity vs. probability of successfully reaching the ground state.

In the final chapter, we conclude our findings and discuss the potential direction of explorations in the future.

# List of Publications

## Peer-reviewed journals

1. N. Mwamsojo, K. Merghem, M. A. El–Yacoubi, Y. Frignac, B. Benkelfat, A. Rigaud, and F. Lehmann, "Reservoir Computing for Early Stage Alzheimer's Disease Detection," in IEEE *Access*, vol. 10, pp. 59821-59831.

2. N. Mwamsojo, K. Merghem, Y. Frignac, B. Benkelfat, A. Rigaud, and F. Lehmann, "A stochastic optimization technique for hyperparameter tuning in reservoir computing," in (*Under review*)

3. N. Mwamsojo, K. Merghem, Y. Frignac, B. Benkelfat, A. Rigaud, and F. Lehmann, "Optoelectronic Coherent Ising Machine For Combinatorial Optimization Problems," in (*Under review*)

## Conferences

1. N. Mwamsojo, K. Merghem, M. A. El–Yacoubi, Y. Frignac, B. Benkelfat, A. Rigaud, and F. Lehmann, "Optoelectronic Reservoir Computer for Early Stage Alzheimer's Disease detection," in Conference on Lasers and Electro-Optics (Optica Publishing Group, 2022), paper ATh4I.5.

2. N. Mwamsojo, F. Lehmann, K. Merghem, Y. Frignac, B. Benkelfat, and, B. Benkelfat, "Image Restoration Using Coherent Ising Machine," in Conference on Lasers and Electro-Optics, Technical Digest Series, (*submitted*).

3. Q. Zou, K. Merghem, N. Mwamsojo. Optical-injection-induced period-one oscillation in semiconductor lasers with coherent optical feedback. Optique Nice 2022, Jul 2022, Nice, France.

4. Qin Zou, Kamel Merghem, Nickson Mwamsojo. Period-doubling bifurcation and intermittency route to chaos in semiconductor lasers with dual feedback. Optique Nice 2022, Jul 2022, Nice, France.

5. Qin Zou, Kamel Merghem, Nickson Mwamsojo. First Hopf bifurcation of semiconductor lasers with delayed optoelectronic feedback produced by weak external optical feedback. Optique Dijon 2021, La Société Française d'Optique, Jul 2021, Dijon, France.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Intelligence in inanimate objects has tantalized human minds for ages. There were Greek myths about an intelligent robot named *Talos* made from bronze possessing human qualities such as emotions and wisdom. Another Greek mythology speaks of Daedalus, a craftsman, who created bronze sculptures that could see, move and speak. In ancient Indian culture, there were stories of a 'mechanical doll' named *Yantraputraka* that was similar to a human, so much so that it fooled and seduced a painter. In Tanzania, there are stories about *Chautope*, a life-like girl made from mud and sticks. The ancient intelligent systems often include the ingenious design, say of a metallic body, and, as should be expected, supernatural abilities of the designers and creators. Also, these stories arguably stem from the human fear of the inevitable death and philosophical questions about what it means to be human beyond mundane existence. They, nonetheless, underline our continued interest in developing human-like forms of intelligence outside the human mind.

The idea of intelligent machines was entertained for millennia, and in the mid $20^{th}$ century, Isaac Asimov wrote the book *'I, Robot'* and brought forth the recommendations of acceptable human-robot interactions. His ideas, however, just as in the Greek and Indian myths, were purely fictional. At about the same epoch and independently, after inventing what is commonly known as the first computer for deciphering the German's Enigma code, Alan Turing started pondering about intelligent machines [1]. Unlike the Greeks, Indians, Tanzanians, and Asimov, the basis of his contemplation was rooted in scientific inquiry rather than fiction. He reformulated what it means to think and designed a test for intelligent systems. He alluded that: should a system be deemed intelligent, it must generate the test results such that a human observer fails to decipher whether the test results are from a human or machine subject. His works inspired the development of intelligent systems as we know them today.

## 1.1 Artificial Intelligence and Machine Learning

A decade ensued, and the ideas spawned by Turing and the developments in computing systems witnessed the outlook on intelligent systems gaining traction in the scientific community and entering the mainstream. Governments and organizations seduced by the idea poured funds and support into the research in the field. The general masses split between curious excitement and the fear of the potential tyrannical takeover of robots. In 1956, John McCarthy and Marvin Minsky organized a workshop named *Dartmouth Summer Research Project on Artificial Intelligence*, coining human-made thinking as Artificial Intelligence (AI) for the first time [2]. Years succeeding the workshop, the excitement of AI led to bold statements from prominent figures, to cite a few :

*"Within ten years, a digital computer will be the world's chess champion unless the rules bar it from competition."*
- H. A. Simon (Nobel Prize laureate) and Allen Newell (Turing Award) in 1958,

*"Within a generation, the problem of creating artificial intelligence will substantially be solved,"*
and
*"In from 3 to 8 years, we will have a machine with the general intelligence of an average human being."*
- Marvin Minsky in 1967 and 1970, respectively.

Unfortunately, these predictions grossly exaggerated the prospects of AI, and failure to deliver incited doubts in the field and encouraged critics. For instance, intense criticism came from influential figures such as James Lighthill, who wrote a very pessimistic assessment of the developments made in the field in a report to the UK Science Research Council. He accused the researchers of failing to deliver the promises made and painted the field as merely science fiction inept of real-world applications [3]. His report diminished the initial enthusiasm for research and the promise of AI, resulting in the withdrawal of support from the British government and other governments. This report resulted in a period termed the $1^{st}$ AI winter, where little progress was made in AI research.

The tumble of AI was, to a large extent, also due to the implementation of 'intelligence' based on the assumption that intelligence can be formulated in a top-down manner as an extensive collection of '*if-then*' commands [4]. These are essentially not intelligent systems but expert systems relying on expertise rather than knowledge. One such system was the MYCIN system comprising 600 rules for diagnosing blood infection [5]. These systems, however, excel at simple problems or problems that are easy

to formulate. However, some tasks, like speech recognition, that seem to be tackled easily by human brains, are difficult for expert systems because an exhaustive top-down formulation is not feasible.

Fortunately, a shift from the programming of expert systems to *learning* from experience was embraced in AI research. Instead of attempting to fully formalize a problem as a sequence of instructions in a Turing-like manner, the system deduces knowledge from experiences. This is, in essence, how our brains learn how to process seemingly complex tasks ranging from facial to speech recognition by interacting with the world. Similarly, for machines, learning requires a means to extract knowledge from observations - an algorithm to teach the system and a good set of experiences. Depending on the teaching mechanism, such algorithms can be supervised or unsupervised. Supervised learning translates to the system getting to experience the examples and their expected outcomes/classes and adjusting to map the two together. In an unsupervised scheme, the system learns not in the example-target manner but by extracting statistical relationships between the given unlabeled examples and grouping together those that show the most similarities.

Machine Learning (ML) as a process of extracting knowledge from data revived interest in AI research and the shift from under-achieving expert systems. Furthermore, essential breakthroughs such as honing backpropagation for training artificial neural networks reignited the hope for AI. However, most of these new advances were slow and quiet, avoiding re-inciting the previous negative publicity.

## 1.2 Neural Networks

Today, a considerable amount of effort is dedicated to research and developments in the field of Artificial Neural Networks (ANN). Of the different avenues of IA, ANNs represent the most sought solution in many domains of application and are our central focus. In this section, we will introduce essential notions of ANN that are of interest for subsequent discussions in this thesis.

### 1.2.1 The brain

We dare argue that the human brain is the most astounding wonder of evolution. It is a vast neural network of 100 Billion biological neurons and an equally surprisingly large number of glial cells, all carefully organized to form a massively complex structure. Even more astonishing is the fact that trillions upon trillions of connections are created linking the neurons, neither randomly nor arbitrarily, but systematically to form a structure capable of unimaginable feats. No wonder brain-inspired scientists like Warren

McCulloch and Walter Pitts came up with the first mathematical framework for artificial neurons, the McCulloch-Pitts model [6], to explain the brain's functioning. Later, Hebb proposed that neuronal pathways strengthen with learning [7], a hypothesis that sparked interest in neuronal computing leading to the earliest designs of artificial neural networks.

### 1.2.2 Artificial Neural Networks

The McCulloch-Pitts neuron model takes in multiple binary inputs, computes the sum, passes the sum to some activation function, and then compares the result to a threshold to decide whether or not to generate an output. Since the outputs (also referred to as activations) generated are either 0s or 1s, the neuron can thus be considered a Heaviside function of the sum of the inputs (see Figure 1.1). This model had several limitations addressed by Frank Rosenblatt and Marvin Minsky and Seymour Papert [8]. The resultant model was the Perceptron Model, which, unlike the McCulloch-Pitts model, introduced weighing of the inputs before summing. Further improvements resulted in the modern mathematical model of the artificial neuron. Suppose $x_1, x_2, ...x_n$ are inputs and $w_1, w_2, ...w_n$ are the weights, the output of a neuron will be a continuous variable $y$ defined as:

$$y = f_{NL}(w_1x_1 + w_2x_2 + ... + w_nx_n) \tag{1.1}$$

where $f_{NL}$ is a nonlinear activation function (typically a sigmoid, hyperbolic tangent, or ReLu, as shown in Figure 1.1). Notice, unlike the Perceptron, there is no thresholding, and the output is analog, taking a range of values and not just in the set $\{0, 1\}$.

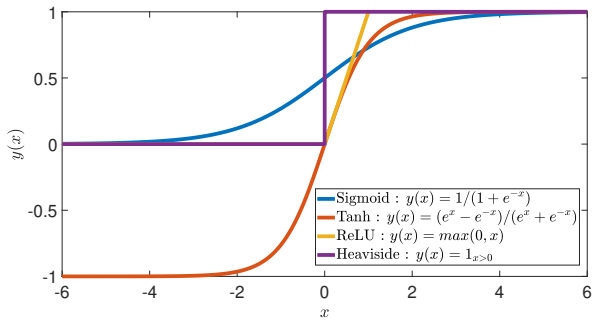



Figure 1.1: Common activation function for the artificial neuron mathematical model

As interesting as this model is, a neuron does not accomplish much beyond a weighted sum and a nonlinear output of the aggregate. However, like in the brain, a network of many neurons possesses more interesting computational properties. These networks are know as Artificial Neural Networks (ANNs) and have amassed remarkable success in signal processing, facial recognition, and pattern generation in commercial and

non-commercial settings. ANNs can approximate any function [9], which means that given enough sets of observations for any system, there is a neural network capable of mimicking the system's behavior with arbitrary precision.

The generic architectural organization of ANNs is that of multiple layers, the input layer for handling the input data, the hidden layer(s) for computation, and the output layer are tasked with generating the prediction or the system's approximation. If the ANN has only one (or very few) hidden layer (s), it's called a shallow network; otherwise, it is a deep neural network. Deep neural networks perform better than shallow ones as they introduce hierarchical learning [10]. Depending on how the layers and neurons are organized, the nature and function of the neurons, and whether or not cyclic connections exist, neural networks can be grouped into several categories, as we discuss below.

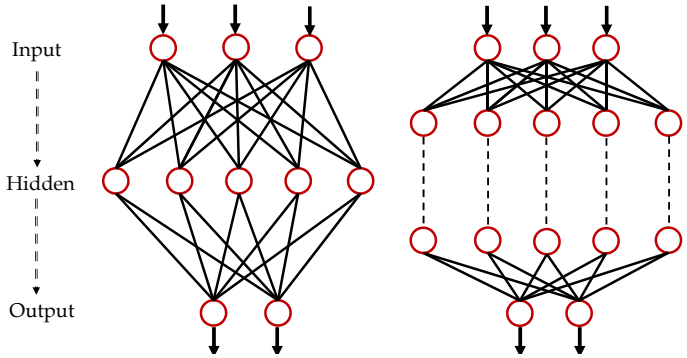



Figure 1.2: Shallow vs. Deep ANNs

### 1.2.3 Feedforward and recurrent processing

Artificial Neural networks are categorized into two major classes: Feedforward Neural Networks (FNN) and Recurrent Neural Networks (RNNs). The FNNs are essentially the class we have already generally evoked in the previous section with a similar architecture to those in Figure1.2. They are characterized by a unidirectional flow of information from the input to the output layer without any cyclic paths in the structure. These networks are universal function approximations provided certain conditions are met. ANNs falling in the FNN category are Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs). As their names suggest, a MLP (resp. CNN) is obtained when each layer consists of a weighted sum (resp. convolution) of the previous layer's output, followed by an activation function. MLPs are sometimes utilized to loosely imply FNNs in general, irrespective of the neuron activation function. In this thesis, a CNN will be proposed and studied in Chapter 4. On that account, we will briefly explain CNNs below.

### Convolutional Neural Networks

CNNs are a particular class of ANN notorious for image processing. They consist of stacked layers performing convolution operations with several filters automatically learned from the data during training. The layers extract various discriminatory features from images. These features are then sent to the dense layer(s) for neural processing and classification, as shown in Figure 1.3. CNNs have enjoyed the most success in the commercial realm due to their numerous applications. They can be used for automatic facial recognition, pattern recognition, or digit classification on bank checks and postal addresses. A few examples of standard CNNs models are:

- **LeNet-5** : For digit recognition from images like the MNIST dataset [11].
- **AlexNet**: This model is much deeper than LeNet-5 and won the 2012 ImageNet ILSVRC challenge by a significant margin. It has *60* million parameters and *650,000* neurons, employing the dropout technique to reduce overfitting [12].
- **VGG**$-16$: An even bigger model, taking inspiration from AlexNet, for image classification with *138 million* parameters designed by the Visual Geometry Group requires $500Mb$ of storage space. This model secured first and second places in the localization and classification tracks in ImageNet Challenge 2014 [13].
- **ResNet** : A very deep network with *152* layers. ResNet took 1st place on the ILSVRC 2015 classification task [14].

It is worth noting that, after AlexNet's victory, the subsequent CNNs kept getting more extensive and profound to beat the previous models' accuracy. Later in this thesis, by virtue of comparison, we implement a CNN architecture to accentuate the shortcoming of this way of thinking.

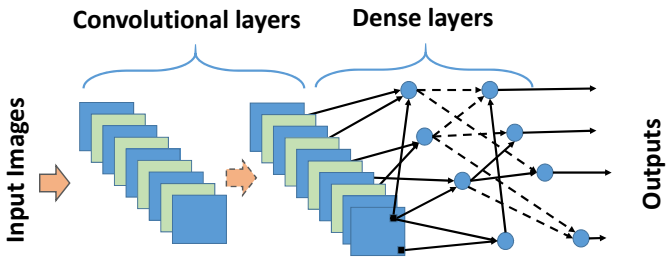



Figure 1.3: Block diagram illustrating the CNN working principle.

### Recurrent Neural Networks

As you read this manuscript, your brain processes individual words and extracts their meanings. Yet, this is insufficient to understand the rather long sentences and paragraphs. For this, you must recall previously read words and their order to capture context. Therefore, processing data sequences requires incorporating the history of previous entries in the processing of the recent inputs. And this is where RNNs, such as your brain, excel. FNNs may be universal approximators but are less adapted for modeling dynamical systems whose states depend on the current and historical inputs. Being dynamic, they capture the temporal/sequential dependence of the processed inputs to extract both meaning and context. As a result, RNNs stand-out in performance for tasks such as trajectory learning, speech recognition, time series classification, and forecasting. Under conditions, they are universal approximators for dynamical systems [15]. In this thesis, we deal with non-standard RNNs called Bidirectional LSTMs described below.

Long Short-Term Memory (LSTM) neural networks are a generalized class of RNNs with special gates that can store, read and reject information from recent history and further back in time (short and long memories) [16]. Bidirectional LSTM (BiLSTM) is a variant of LSTM that simultaneously exploits the past and future (hence bidirectional in time) for computation which makes it well-suited for specific tasks. Fig. 1.4 shows the principle of operation with the classifier having the serial information ($x_i$) processed in both past-future and future-past directions to give the output ($d_i$) for processing in the subsequent dense layers. BiLSTM is extensively studied in the literature (we direct the reader to [17–19]). In our works, we use Python's Tensorflow to implement the BiLSTM model in Chapter 4 for pathology detection.

However, unlike FNNs, RNNs come with a hefty price. Training FNNs is relatively simpler and can be done using linear approaches. The layered structure permits the backpropagation of gradients layer-wise for adjusting the weights and biases. The addition of recurrent connections significantly complicates the task of training RNNs via backpropagation through time, which notoriously suffers from the vanishing gradient problem. In Chapter 2 of this thesis, we discuss Reservoir Computing (RC), a discrete-time nonlinear dynamical framework to circumvent the RNNs training difficulties. We will detail the working principle, explain various implementations, and recount success stories and the promising future of Reservoir Computers.

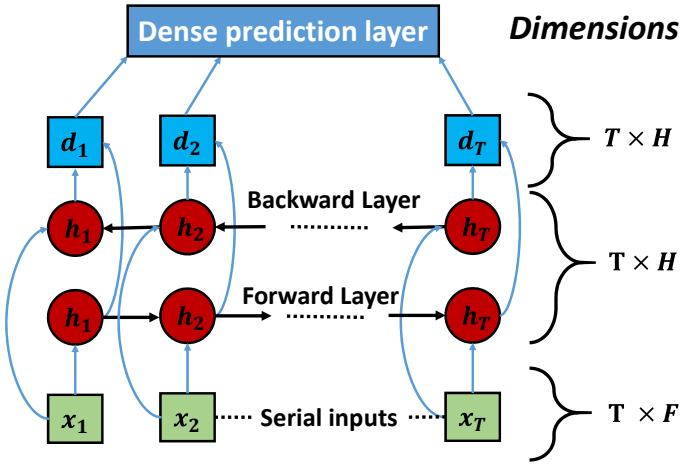



Figure 1.4: Block diagram illustrating the BiLSTM working principle. The serial inputs features are introduced to the hidden layers both in past-future (right arrows) and future-past (left arrows) directions.

## 1.3 Energy trends in training ANNs

Learning is a crucial step for Machine Learning (ML) approaches. Thanks to the dawn of powerful silicon processing chips, groundbreaking methods for training and the availability of ever-increasing large datasets, more complex models are trained for inference. The tendency is to design bigger and deeper networks able to capture potentially complicated structures inside vast amounts of data. We previously discussed this trend when discussing the winners of image classification challenges. In this section, we quantitatively elaborate on this trend by analyzing the costs associated with training several models in practice today. We will define Green AI as a sustainable alternative to Red AI and point out that the works presented in this thesis took this greener (hence more environment-friendly) direction of exploration.

### 1.3.1 Red and Green AI

Colossal computational resources are required to design and train modern-day deep models. To attain that top-notch accuracy, researchers create even larger networks, and a profusion of hyperparameters is fine-tuned for days, weeks, and sometimes even months. The authors of [20] made an interesting observation by examining the environmental impact of training on the environment. They calculated the mass in pounds (*lbs)* of $CO_2$ gas emitted to generate the energy consumed for training several well-known models. Among the models studied is the Transformer model for machine translation [21]. The *big* Transformer model has *213* Million parameters to be optimized through training. We plot in Figure 1.5 the data from [20] to visually illustrate the ludicrous

amount of $CO_2$ released into the atmosphere for training this model. On the plot, emission per passenger for a flight across the USA from New York (East) to San Francisco (West), average emission for a single human in 1 year, average emission for a single American in 1 year, emission from a car in a *lifetime* and the emission for training the Transformer ANN are illustrated. At 626,155 *lbs* (284019 *kg*), the model's impact on the environment is equivalent to that of five cars in their entire lifetime. This was an eye-opening discovery for us. A car burning fossil fuel seemed more reprehensible than a model stored as matrices of weights and biases in computer memory.

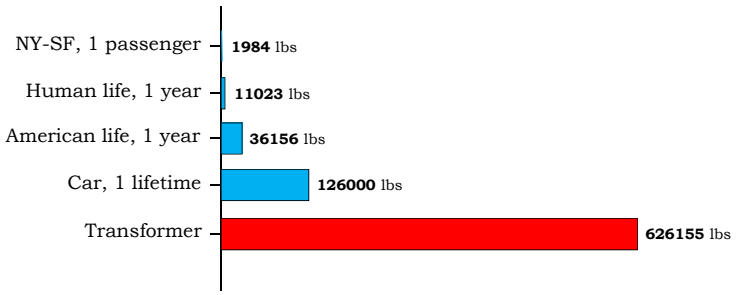



Figure 1.5: A bar plot showing environmental impact in pounds of $CO_2$ emitted in training an ANN model being five times that of a car in its lifetime.

We have pointed out that the success and progress of ANNs can be attributed mainly to the increase in their depth and complexity, with a significant increase in the cost of training them. Researchers tend to focus solely on reaching or surpassing state-of-the-art accuracies with little attention paid to the costs incurred. Often, an insignificant increase in accuracy comes at a substantial increase in training costs. This way of thinking is not environmentally friendly and has been coined the term Red-AI [22]. Red-AI not only increases the cost of running the models but also complicates the implementation of the ANN models on mobile devices such as telephones or tablets, which could be useful for certain applications [23], such as the pathology detection task in Chapter 4. The greener approaches, on the other hand, termed Green-AI, should incorporate and consider the energy efficiency analysis as an added metric for both model selection and optimization, as we do in this thesis.

Reducing the environmental impact of training ANNs is crucial, even more so now that effort is directed to this goal in many other fields. Moreover, focusing on boosting resources solely for performance is detrimental to scientific progress [24]. Some ideas win not because they are superior but because they are suited and adapted to the existing software and hardware. The term *Hardware Lottery* was proposed to describe the biased interest for ideas well adapted to already existing resources [24]. It is thus

very important to renew and question ourselves continuously to generate a necessary revitalization of ideas and thought patterns.

Furthermore, complex and expensive solutions limit this area of scientific potential to the industry labs with access to the necessary resources. For instance, the hardware budget for an ImageNet classifier proposed [25] is 1.2 million USD, and a similar implementation by Facebook costs 4.1 million USD [26]. Such vast sums of money leave creative staff in academic institutions behind, not for the lack of better and more creative innovative ideas, but for lack of proper funding. The consequence is the discouragement of creativity, skewing exploration and hindering progress for certain directions of interest with the potential to change the future radically.

### 1.3.2 A humbling Benchmark: The brain

We previously called the brain a miracle of evolution and maintain that stance. While the costs of training and maintaining ANNs are skyrocketing, the brain does an even more impressive job at meager demands. The brain constantly recognizes patterns, interprets multiple sensory signals, and predicts. It accomplishes all these at the cost of only 20 *Watts*. A typical computer, not nearly as complex nor powerful, runs at about 100 *Watts*. The *Sandberg and Bostrom* report estimated the computation effort necessary to emulate a human brain at 20 PFLOPS [27] (other papers gave higher estimates). Assuming the brain is emulated on an NVIDIA Tesla K20 GPU running at 2 GFLOPS per Watt [28], we would require 10 *MWatts* of electricity. You need a small nuclear power plant to power an artificial brain while we do it at 20 *Watts*. How, then does the brain accomplish so much so cheaply? The answer to this question is, in part, the motivation for our works in this thesis. In the next section, we will discuss efforts to answer this question.

## 1.4 Neuromorphic computation

There is increasing talk about neuromorphic computation, and this is no accident (See Figure 2 in [29]). During the AI inception days, computers were projected to become majestic thinking machines even surpassing the human minds. The general outlook was that the brain is slow and ineffective and that it will not take long for computers to take over. However, even with much more powerful computers today, we are yet to challenge the computational capabilities of the brain. Our attempts thus far, as we pointed out in the preceding section, have been towards increasing the size and number of parallel hardware to do better and better. The outlook, nevertheless, is shifting towards

looking closer into the brain for inspiration. Neuromorphic systems imitate the internal workings of the analog brain units and replicate by using unconventional hardware such as electronic, photonic, mechanical, biological, and, as we will show in Chapter 2 even liquid substrates. fIn this section, we will look at motivations for neuromorphic computing and introduce two classes of neuromorphic RNN systems of interest in this thesis: the non-autonomous (i.e. input driven) RNNs known as Reservoir Computers (Chapters 2, 3, and the 4 ) and the autonomous RNNs (i.e. RNNs evolving in time without the driving input) known as Coherent Ising Machines (Chapter 5).

### 1.4.1 Motivations

In his invited paper [30], Mead investigated the reasons for the disparity between silicon chips and the brain. He alluded that, for electronics, there is a factor of $100$ in thermal losses from heating in the wires and a factor of $10000$ from the use of multiple transistors to do one operation. All together, resulting in a factor of a *million* in losses for computation in digital systems. He also argued that the brain uses elementary physical phenomena as computation primitives using analog rather than digital signals that complicate efficient designs. Mashing up all computation tasks to a digital implementation consisting of 0s and 1s complicates the computation of most real-world analog phenomena in digital electronics. He coined the word *Neuromorphic* Electronic Systems, to describe specialized systems that consider the brain's analog principles when modeling the capabilities of neural networks found in biological systems. His views are valid today; with the surge of data to be processed, constricting analysis to 0s and 1s in digital systems is about to hit the wall. Dedicated ANNs hardware can lead to groundbreaking efficient computing machines, with gains in speed and considerable savings in terms of energy efficiency.

To justify the overexertion of silicon electronics, we will discuss two important laws, Moore's law and Koomey's Law. Moore observed that the number of transistors in integrated circuits (computation power) doubles every two years. However, the rise of AI computing has accelerated our computational needs beyond Moore's law. The amount of computing for AI now doubles every *3.4* months as shown by the fit of several popular models in Figure 1.6 [31]. While our computation needs to increase, some developments in the supporting hardware show asymptotic progress. Koomey had observed a trend of doubling in computations carried per joule of heat dissipated in about 1.57 years. However, a growing gap between the physical hardware and our computational needs will drive us to an efficiency wall, as shown by the plot in Figure 1.7 [32]. Hence the term 'hit the wall.'

Digital computing systems use the standard Von Neumann architecture, as shown in

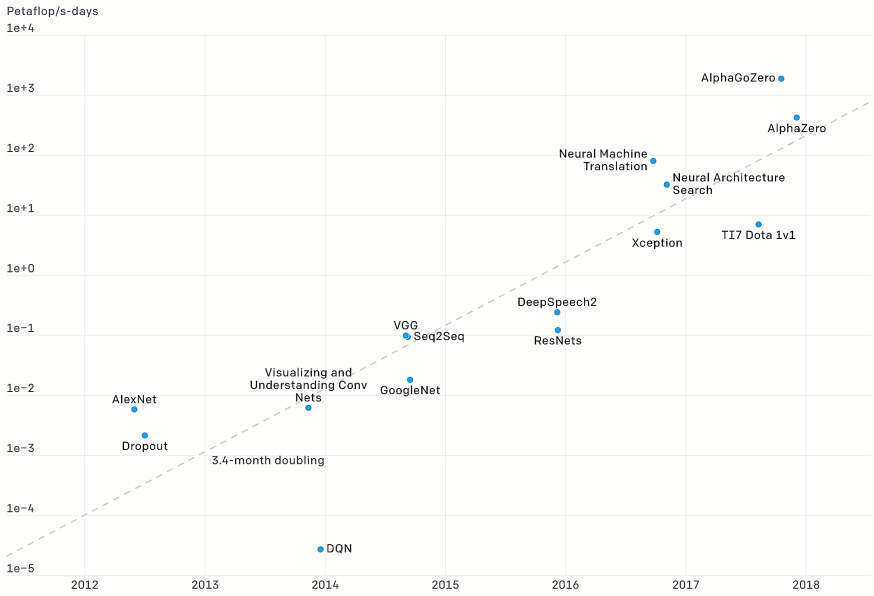



Figure 1.6: Evolution of computational power needs in PFLOPS-days from 2012 to 2018 for popular AI models. The trend shows a doubling of the requirements in 3.4 months.

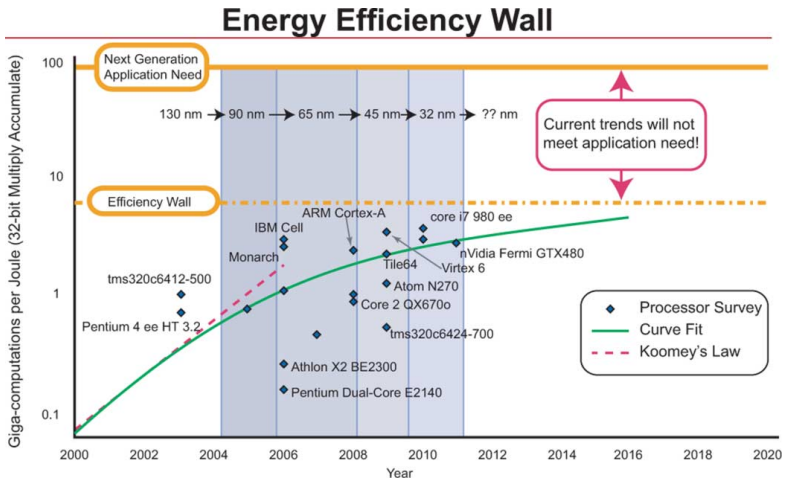



Figure 1.7: Evolution of the amount of heat dissipated in Joules per giga-computations from 2000 to 2020 showing an asymptotic trend and a lag gap for next generation needs.

Figure 1.8. Memory units and computation units are delocalized. As a result, information has to be dispatched from the memory units for computation and back for storage or output. Over the years since his proposal, processors have remarkably increased in computation speeds whereas the data transfer speeds lag. This results in a bottleneck where processors spend more extended idle periods resulting in losses. Attempts to remedy this by introducing faster memories, such as caches and registers have reduced the idle times to some degree. However, these memories are generally small and

only used to store small immediate data for computation. Most AI tasks require large amounts of memory so the problem of memory access overhead persists. The brain, on the contrary, uses associated memory stored concurrently with the computation units, thus eradicating this overhead problem. In short, implementing ANNs employ computational concepts that are fundamentally different from the traditional computational workhorse, and this is largely responsible for their inefficient power consumption.

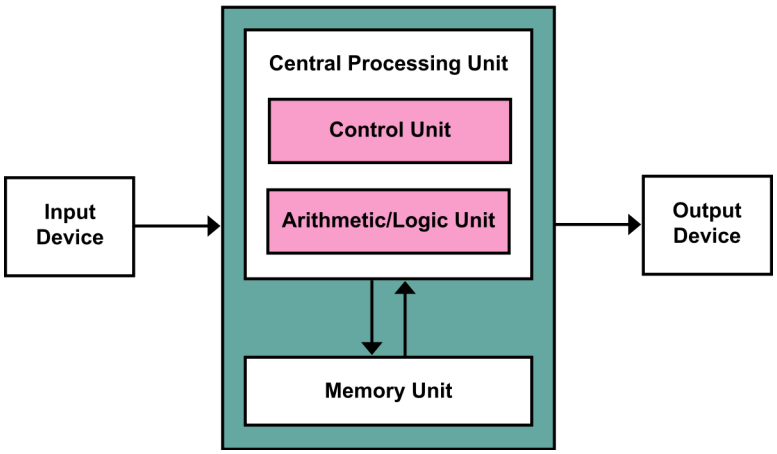



Figure 1.8: The John Von Neumann architecture. The memory and computation units are separate.

### 1.4.2 Reservoir Computing

In this thesis, we study a recurrent neuromorphic system with demonstrated potential for hardware feasibility called Reservoir computing (RC). RC is a powerful RNN framework for processing sequential data characterized by the simplification of the training procedure while maintaining high computational power. Motivation for the RC originates from the known difficulties in training the traditional ANNs (particularly RNNs). RC computing offers an alternative approach by limiting training to the outermost layer alone and randomly generating most neurons in the hidden reservoir layer. This approach drastically simplifies training and introduces little to no performance penalty. Moreover, the randomness of the reservoir reduces constraints in the hardware implementations of RCs resulting in a plethora of hardware implementations. In-depth discussion and examples of RCs are detailed in Chapter 2 where we define the important notions and discuss applications and physical implementations of RC. In Chapters 3 and 4 we will optimize RCs and run several applications.

### 1.4.3 Ising machines

Neuromorphic systems are interesting for finding solutions to Combinatorial optimization problems. These problems are notoriously hard to solve in the conventional algorithmic approaches and using the Von Neumann approaches. The difficulty stems from the NP-hard nature of these problems resulting in exponential explosions in the number of evaluations necessary in the search for extrema with a meager increase in the search space.

Imitating the computational properties of the connection of a large number of simple and coupled biological elements is an attractive avenue for these types of problems. A class of neuromorphic autonomous RNNs known as Hopfield Neural Networks has demonstrated great potential for these applications [33]. The network's energy function monotonically decreases with each neural state update. The natural ability of the RNN to arrive at an extremum can be leveraged for optimization problems by mapping the desired solution to the extrema of the system. Hardware solvers for the Ising model offer an interesting alternative for optimization problems, especially as we become more mindful of energy efficiency as we search for their solutions. In Chapter. 5 we discuss Ising machines in more detail and present our findings on the subject.

### 1.4.4 Neuromorphic photonics

A few decades ago, electronics drove every aspect of technology while photonics was still in its infancy in many domains of application. Progressively advances in optics and photonics penetrated different fields ranging from research, medicine, security, and communications. Today we have witnessed a massive shift into optics. For instance, most of our communication passes as photons at some point. The takeover can be attributed to many attractive properties of light: low energy cost, parallelism, multiple degrees of freedom, speed, large bandwidth, and low interference. The progress in photonic computing, however, has been rather slow. The reason is in part due to the difficulty in harnessing the optical nonlinearities for computation in the digital computing fashion. The advent of neuromorphic analog computing has, nonetheless, sanctioned the use of optics for computation. The works presented in this thesis are photonic in nature making use of off-the-shelf components intended for optical telecommunications.

## 1.5 Goals of this thesis and chapter overview

In this introductory chapter, we have given a brief history of computing with emphasis on the non-Turing, brain-like style of computation. The works presented in the rest of

this manuscript follow a common philosophy of low complexity and environmentally friendly solutions. As we propose novel solutions and applications, we consider the efficiency of our proposals on top of their brute performances. We provide a general overview of the chapters below to give a glimpse of our train of thought.

### 1.5.1 Chapter 2 : Reservoir Computing

We introduce the Reservoir Computing (RC) concept in this chapter. Due to the well-known difficulties of training traditional RNNs, a special framework for processing sequential data was proposed that drastically simplifies training. The simplification alleviates most of the challenges hindering convergence and efficient training of RNNs. Two variants of RC: Echo State Networks (ESNs) and Liquid State Machines (LSMs) are introduced. The ESNs are the variant of interest in this thesis along with their implementation in simulations and physical hardware. We describe several properties of RCs, such as the Echo State Property, Memory Capacity, and Computational Capacity. Various parameters control these properties, and tuning them prescribes different modes of operation. Moreover, since RCs are versatile, many neuromorphic implementations using unconventional hardware and substrates for neuron reservoirs have been proposed in the literature. Consequently, we elaborate on their theoretical footing and various design simplifications, such as the delay-based reservoirs. We describe the various implementations from the Water Bucket reservoir, and mechanical mass oscillators, to electronic and photonic implementations. The utility of RC is brought to light by the rich variety of applications and their respective domains. By reading this chapter, the theory, advantages, versatility, and potential avenues for applications are made apparent.

### 1.5.2 Chapter 3 : Parameter optimization Reservoir Computing

We put forward a low-complexity method for hyperparameter optimization for RCs in this chapter. Several parameters prescribe the stability, memory, and computational behavior of RC as a recurrent system. Various methods are employed for finding the best combination of hyperparameters that correspond to the RC's optimal performance for the task at hand. In the literature, some proposed Exhaustive Search and evolutionary approaches, and others proposed Stochastic Gradient Descent (SGD) based methods.

Our proposal is based on SGD as well, however, we relax the constraint on the knowledge of the functional relationship between the cost function and the parameters to be tuned. This is particularly interesting for the physical neuromorphic systems where the exact characteristics of the constituent devices are unknown and can also

drift in time. This renders gradient computations impossible for the SGD methods, a challenge we are going to address. Suppose $\boldsymbol{\theta}_i$ is a vector of the $p$ parameters at the $i^{th}$ iteration and that $L(.)$ is the cost function, then the gradient can be approximated via finite difference as follows:

$$\Delta_i = \frac{L(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i, \mathbf{W}_{i-1}^{out}) - L(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i, \mathbf{W}_{i-1}^{out})}{2h}, \tag{1.2}$$

where $h$ constant step, $\mathbf{W}^{out}$ is the readout matrix obtained using the training data and $\Delta_i$ is the noisy approximation of the gradient for the parameter combination. To minimize the loss function, we employ an iterative procedure to update each component of $\boldsymbol{\theta}$, selected randomly one at a time, according to

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu\mathbf{d}_i\Delta_i, \tag{1.3}$$

where $\mu$ is a constant adaptation step-size. The proposed iterative parameter optimization scheme is summarized in Algorithm 1. Rather than updating all components of $\theta$ in each iteration, which requires $2p$ cost function evaluations, the update direction is randomly and independently chosen such that only one hyperparameter is updated per iteration. This reduces the complexity of our method and improved convergence.

Our results indicate, that our method consistently converged after relatively fewer iterations compared to the benchmark alternatives, i.e., Exhaustive search and Simulated Annealing (SAN). We show that the complexity of each algorithm is proportional to the number of iterations hinting at reduced complexity with our approach. This observation is consistent when we employ our algorithm to search for optimal parameters for the spoken digit recognition benchmark and two other tasks not found in RC literature: the Control Chart classification task and the Wafer classification task. These two tasks have important industrial applications. We also study the transient state and the steady state behaviors, both theoretically and numerically, as the systems converge to the optimum. The steady-state behavior showed agreement between the theoretical and numerical evaluations thus validating our theoretical analysis of the method.

### 1.5.3 Chapter 4 : Reservoir Computing for Early-Stage Alzheimer's disease Detection

In this chapter, we propose Reservoir Computing (RC) for the Early Stage Alzheimer's disease detection task in both numerical and hardware implementations for the first time. Alzheimer's disease is a neurodegenerative disorder caused by the progressive destruction of nerves in the brain. Its slow and progressive nature is responsible for

its insidious onset which makes early diagnosis difficult. As a result methods for diagnosing the disease are either biased or intrusive and expensive, whereas early diagnosis is crucial for intervention measures. The disease impairs patients' fine motor control, making handwriting (HW) a potential biomarker for pathology. The handwritten data in this work was collected at Broca Hospital in Paris. Diagnosed patients and healthy controls were asked to write a sequence of cursive-$\ell$ letters on a digital tablet. With this data and using RC, we obtained a classification accuracy of $85\%$, an improvement of $11\%$ over the state-of-the-art.

We benchmark the performance of our RC with that of Bidirectional Long Short-Term Memory (BiLSTM), *k*-Medoids, and the Convolution Neural Network (CNN) on the full HW dynamics taken as a time-series for the classification. By considering accuracy as the sole metric, BiLSTM beats RC by $3\%$ in accuracy. However, extending the comparison by gauging the computation efforts and environmental costs of obtaining the reported performances we get a more complete picture. Our assessment indicates that the $3\%$ gain in accuracy for the BLSTM comes at the cost of $8$ times the electric energy cost and the mass of $CO_2$ and equivalent Green House Gases released according to the grid information at Palaiseau, France, where our experiments were run. We interpret this as an advantage of RC over all other approaches for repeated use. More so, if the models are to be run on a mobile device, for instance, the same battery-powered devices used for data acquisition, RC will give the most extended battery discharge cycle. Moreover, our hardware implementation implied roughly more energy efficiency with a slight penalty in performance with respect to the digital RC.

### 1.5.4 Chapter 5 : Coherent Ising Machines for Combinatorial Optimization

So far in our discussions, we have considered RC, which is non-autonomous (input driven) and trained in a supervised manner. In this chapter, we consider an autonomous class of neuromorphic systems that will evolve down to the lowest possible energy state without being driven by an external input signal. We interest ourselves, particularly in Coherent Ising Machines (CIMs). They are extremum-seeking neural networks suitable for combinatorial optimization problems. CIMs imitate the ferromagnetic behavior, i.e. the temperature-dependent alignment of polarization of the magnetic moments (spins) of the constituent atoms in the same direction at the microscopic level. Above a critical temperature, $T_{cr}$ the spins become randomly oriented resulting in zero net magnetic fields and vice versa for the temperature below. The total energy of the system at any time depends on the alignment of the spins. The goal is to formulate the given

problem such that the solution to the problem coincides with the ground state of the Ising Hamiltonian of the system, defined as:

$$E(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{(s,t)\in\mathcal{N}} J_{s,t}\sigma_s\sigma_t - \sum_{s\in\Omega} b_s\sigma_s, \tag{1.4}$$

where the problem is defined over an $n \times n$ square lattice. Any $s = (l, c) \in \Omega$ can be assimilated to a position in the lattice with line (resp. column) coordinate $l$ (resp. $c$), where $1 \leq l, c \leq n$. $\forall s \in \Omega$, we let the spin $\sigma_s$ be a random value in $\{-1, +1\}$, $J$ is the matrix dictating the interaction between the spins and $\mathcal{N}$ denotes all couples of neighboring nodes with end-around boundary conditions.

Following [34] in order to minimize Equation 1.4, we numerically implement a generalized Hopfield network, whose discrete-time difference equation at instant $k$ has the form

$$\begin{aligned} x_s(k) &= f\left(\alpha x_s(k-1) + \beta\left(\sum_{t:(s,t)\in\mathcal{N}} J_{s,t}x_t(k-1) + b_s\right)\right) \\ \hat{\sigma}_s(k) &= \text{sign}(x_s(k)), \quad \forall s \in \Omega \end{aligned} \tag{1.5}$$

where $f(.)$ is a nonlinear activation function, $\alpha$ and $\beta$ are scaling coefficients controlling the self-coupling and feedback strength affecting the neuron output $x_s(k)$, while $\hat{\sigma}_s(k)$ is the spin estimate of pixel $s$. We study the ability of the CIM to find solutions to several problems such as the Antiferromagnetic Ising Model and Maximum A-Posteriori (MAP) image denoising and Traveling Salesman. We gauge the system's performance by the success rate/probability of the system evolving sufficiently close to the optimum, i.e., to energy below a certain threshold.

Our contributions in this chapter are three-fold; first, from an application perspective, we demonstrate the potential for the CIM for statistical image denoising. For this, we map the systems such that the ground state of the CIM corresponds to the cleaned image. Secondly, we compare the proposed mixed digital/hardware system with online processing to the standard digital implementation of Hopfield Networks and Simulated Annealing (SAN) in terms of their ground state attaining probabilities, computational complexity, and energy consumption.

The obtained results indicate that SAN is more powerful with the success probability of attaining the ground state of $98.9\%$ and $100\%$ for the Antiferromagnetic Ising Model and the MAP denoising tasks, respectively. Our experiments for the TSP are ongoing; therefore, some results will not be included in this manuscript. Our digital CIM (resp. hardware CIM) attained the ground state at $91.4\%$ (resp. $90\%$) and $89\%$

(resp. $86.75\%$) for the Antiferromagnetic Ising Model and the MAP image denoising tasks, respectively. Again, considering the probability of reaching the ground state as the sole metric, CIMs are outperformed by SAN. However, considering also runtime, FPOs count, and energy consumption, we will see that CIMs offer an interesting performance vs. energy efficiency tradeoff.

# 2 Reservoir Computing

## 2.1 Introduction to Reservoir Computing

Training ANN models requires data and/or a teacher that adjusts the weights and biases to appropriately map the input signals to expected outputs. A teacher in this case is an algorithm used to train the model by systematically adjusting the weights and biases of neurons in the model. The hebbian, perceptron and delta learning rules [35] are considered one of the earliest methods to train a model. A newer method, called Back-Propagation (BP) revolutionized the development of ANN and re-ignited the interest in AI after the Second "AI Winter" [36]. BP enjoys remarkable success in the training of feedforward neural networks such as Multi-Layer Perceptrons and Convolutional Neural Networks. However, it struggles in the training of RNNs due to the added complexity of feedback loops that may sometimes prevent convergence unless properly accounted for [37, 38]. And even when RNNs converge with iterative gradient-based methods, the convergence is slow, prone to local minima, and computationally demanding.

Several methods have been proposed to circumvent the convergence issues of training RNNs using gradient-based methods [39–41]. For instance, Backpropagation Through Time (BPTT) unfolds RNNs in time resulting in multiple FNNs with common weights that can be trained more easily using BP. However, the emergence of bifurcations renders gradient calculations intractable and unreliable [42]. The increasing computational load of the updates with feedback loops and the difficulty of iteratively tracking dependence when long-range memory is in play further limits the success of these methods. Also, some of these methods are computationally demanding and may require adaptive tuning of learning rates and other hyperparameters which requires skills and expertise. An interesting observation in RNN training is that the backpropagated errors impact the outermost layer the most and the impact fades or degrades inwards. That is the outermost layers adapt quickly while the internal layers evolve relatively slowly during

training. This fact is exploited in techniques such as the Atiya Parlos Recurrent Learning (APRL) [43] and Back-Propagation DeCorrelation (BPDC) [44], in an attempt to mitigate the aforementioned challenges to a certain extent, outperformed BPTT significantly.

About two decades ago, a framework of RNNs called Reservoir Computing(RC), which drastically simplifies RNN training was introduced in two flavors: Liquid State Machines (LSMs) [45] and Echo State Networks (ESNs) [46]. LSMs originated from a computational neurosciences background hence the proposed neurons closely mimic the spiking integrate-and-fire neurons in the brain. The design of LSM includes inhibitory and excitatory neurons i.e. brain-like microcircuits and employs more biologically plausible spiking neurons. The term liquid in LSM refers to the analogy in operation between this model and a liquid surface generating ripples in response to external perturbations.

The other variant, the ESN, stems from an engineering background, employing the classical non-spiking Sigmoid-based neurons (the mean firing rate approximation). The design of ESN constraints does not incorporate the sophistication of the brain-like microcircuits. The design employs the activation functions introduced in Section 1.2.2 and shown in Figure 1.1. The hardware feasibility is not as complex and the availability of many off-the-shelf photonic devices has allowed many scientists to propose various creative contributions. In this thesis, we interest ourselves in ESNs and dedicate this chapter to the expounding of various notions associated with them.

For Reservoir Computing, unlike traditional RNNs, training is implemented at the readout layer alone. The readout layer is the outermost layer that computes a linear combination of the states of randomly connected recurrent neurons in the middle layer. Figure 2.1 shows a visual contrast between training in RNNs and RC. The simplification of training comes with a significant reduction in the number of parameters to be tuned for optimal performance. For classical ANNs, for instance, the training computation effort increases in a polynomial manner with the increase in the number of neurons using gradient-based methods. Training RNNs using the Back-Propagation Through Time (BPTT), for example, necessitates $\mathcal{O}(N^2)$ whereas RC usually takes $\mathcal{O}(N)$ where $N$ is the number of neurons.

Lower complexity for RCs translates to cheaper computations and the ability to increase $N$ for richer dynamics with a lesser increase in computation times. Richer dynamics imply the projection of the input information to a much higher dimensional space. This space is large enough to allow for easier separation of different inputs by the hyperplanes that separate different classes. The benefit here is that the hyperplanes can be computed using computationally cheap linear methods. What's more, the connec-

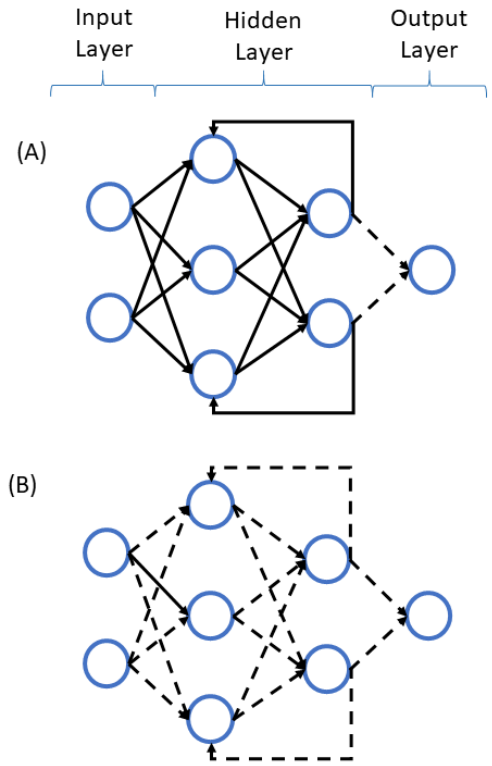



Figure 2.1: Differences in training between RC (A) and classical RNN (B) architectures: synaptic link updated during training (dashed) - synaptic links fixed to their initial pseudo-random value (solid). Note that for the case of RC in (A) the hidden layer with fixed weights is called the reservoir layer.

tions in the reservoir layer of RC are usually sparse, further reducing the computation costs of projections in the reservoir, especially when sparse matrix computation methods are applied in practical implementations.

## 2.2 Reservoir Computing: The Echo State Network model

The basic model of an ESN consists of three layers: The input layer for formatting and injecting the signals to be processed to the second layer, called the reservoir layer. The reservoir layer is the reservoir of randomly connected processing units responsible for the nonlinear expansion of the input signals to a higher dimensional space. The third layer is the output layer that takes the reservoir node states (*node* in this manuscript is used synonymously with *neuron*) and predicts the target output. Figure 2.2 shows the three distinct layers in an ESN.

The basic model of the ESN exploits the fact that, if $\mathbf{x}(n)$ is the vector of node states of neurons in an RNN at a discrete time $n$ then the $\mathbf{x}(n)$ can be expressed as a function of the input history (metaphorically 'echo') $\mathbf{u}(n)$, $\mathbf{u}(n-1)$,... [46] The node states in

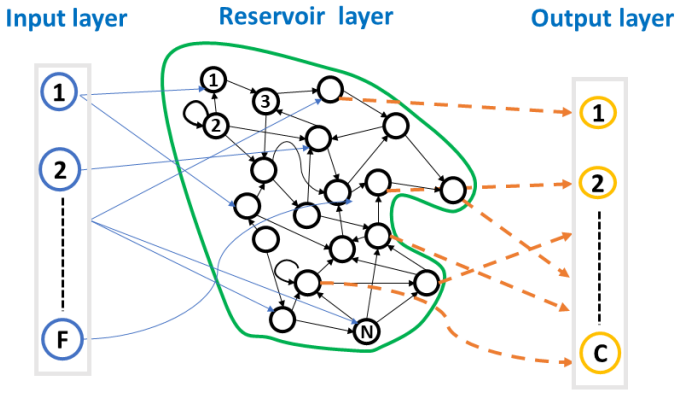



Figure 2.2: Basic Spatiotemporal architecture of a reservoir computer showing the three layers

the reservoir are a function of previous states and the current input as described by the discrete-time nonlinear dynamical equation below [46]

$$\mathbf{x}(n) = f_{NL}(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}^{res}\mathbf{x}(n-1)), \tag{2.1}$$

where $n$ is the discrete-time variable, $\mathbf{x}$ is the vector of node states, $\mathbf{u}$ is the vector of the input activation signal, $\mathbf{W}^{in}$ is the matrix of random input weights also called the input mask, $\mathbf{W}^{res}$ is the random connectivity matrix for the reservoir, and $f_{NL}$ is the nonlinear activation function. Also for the rest of this thesis, $F$ denotes the size of the input vector $\mathbf{u}$, $N$ denotes the size of the reservoir layer (i.e. length of vector $\mathbf{x}$) and $C$ denotes the size of the output.

Equation 2.1 assumes the neurons are memoryless. It is also possible to introduce memory leaks in the neurons with the leaking rate $\xi$ such that

$$\mathbf{x}(n) = (1-\xi)\mathbf{x}(n-1) + \xi(f_{NL}(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}^{res}\mathbf{x}(n-1))) \tag{2.2}$$

The output is generated at the output layer as a linear weighted combination of the node states $\mathbf{x}$ to give a prediction vector $\hat{\mathbf{y}}$ as follows:

$$\hat{\mathbf{y}}(n) = f^{out}(\mathbf{W}^{out}[1; u(n); \mathbf{x}(n)]). \tag{2.3}$$

Here, $\mathbf{W}^{out}$ is the read-out matrix, and $\hat{y}(n)$ is the estimated output vector at time step $n$ to be compared to the expected output vector $\mathbf{y}(n)$ for the supervised learning case. Training consists of computing $\mathbf{W}^{out}$ in a recurrence-free fashion hence avoiding the complexities of loops previously discussed. Unless otherwise stated, the output function $f^{out}$ (resp. $\xi$) will be set to an identity function (resp. 1) in the sequel.

## 2.3 Designing a Reservoir Computer

From the discussion in Section 2.2, it can be deduced that generating a reservoir computer generally boils down to the choice of the tuple of matrices ($\mathbf{W}^{in}$, $\mathbf{W}^{res}$, $\mathbf{W}^{out}$). These matrices represent the input, the reservoir, and the output layer respectively. The general guidelines for generating a good RC are [42]:

- **Large enough reservoir layer** (i.e. the larger the size $N$) imply richer dynamics and this could lead to better performance of the model due to easier signal separation in the bigger phase space.
- **Loose coupling between the neurons** (sparsity in $\mathbf{W}^{res}$) to limit the mixing of the processed signal to a reasonable extent and also moderate the dynamics.
- **Random Connections** (random elements in $\mathbf{W}^{res}$) to ensure varied, hence richer, interactions in the reservoir yield richer reservoir outputs.

These recommendations are indeed general, depending on the task at hand, more fine-tuning may be necessary for optimal performance. The design needs for each layer are different as described in the Sections below.

### 2.3.1 The input layer

The input layer is responsible for interfacing between the brute real-world input signal and the reservoir layer by a process sometimes referred to as *masking*. It is generally characterized by its matrix $\mathbf{W}^{in}$. The matrix is called the input mask or just the input matrix. It is a randomly generated matrix from a given zero mean distribution and is typically dense. Some implementation makes $\mathbf{W}^{in}$ sparse to optimize computation. Masking allows maximizing the dimensionality of the input signals, which may as well be uni-dimensional when the sample and hold are applied. Masking, therefore, generally enriches the system's response to the inputs.

The choice of the elements of $\mathbf{W}^{in}$ has an impact on the dynamics of the reservoir layers as the values can drive the dynamics to one side of the activation function. For sigmoid functions, this could be around the center, where they are mostly linear, or to the ends where saturation makes the neurons act more nonlinear. Moreover, the distribution from which the elements of $\mathbf{W}^{in}$ are sampled impacts the system's performance especially when realistic noise-stricken RC implementations are concerned. This is investigated in [47] where performance of elements sampled from a binary distribution $\{-1, 1\}$ and six-valued distribution $\{1.5, 0.9, 0.3, -0.3, -0.9, -1.5\}$ are compared. They concluded that the binary distribution performed poorly in face of

quantization noise. Setting up to six values, however, increased RC's noise robustness, and beyond the 6 values, no significant improvement should be expected. In another work, general recommendations for setting up the optimal input pattern derived from the concepts of maximum length sequences were proposed by maximizing the variability of the node states while minimizing the number of elements in the mask [48]. The authors of [49] took a more traditional approach and instead of randomly setting up the input mask $\mathbf{W}^{in}$, they proposed input mask optimization by a gradient-based approach similar to other ML techniques. In our work in this thesis, we generate this mask randomly by selecting multiple values from specified distributions.

### 2.3.2 The reservoir layer

This layer computes the nonlinear dynamics that are responsible for the expansion of the input. The neurons are randomly connected to form a random graph, whose adjacency matrix corresponds to the non-zero elements in $\mathbf{W}^{res}$. The adjacency matrix is also called the reservoir connectivity matrix. Like $\mathbf{W}^{in}$ the chosen $\mathbf{W}^{res}$ impacts the extent of the nonlinearity depending on how it drives the activations. It also controls the memory capacity of the reservoir. Depending on the spectral radius of $\mathbf{W}^{res}$, the reservoir can recall more or less the history of the input. Too much memory can drive the system to instabilities making the projections intractable and chaotic. The matrix should be tuned depending on the needs of the task at hand. Tuning $\mathbf{W}^{res}$ is explained in detail in the discussion on the Echo State Property in Section 2.4.1.

### 2.3.3 The output layer

This is the only layer optimized through training. Generally, training consists of computing the optimal $\mathbf{W}^{out}$ that minimizes the error between the expected $\mathbf{y}(n)$ and predicted $\hat{\mathbf{y}}(n)$ outputs. Predictions are carried as shown in Equation 2.3. We use the state vector $\mathbf{x}(n)$ instead of the compound vector $[1; \mathbf{u}(n); \mathbf{x}(n)]$ which can be considered as a special case that does not alter our general discussion. Taking a column-wise concatenation of $\mathbf{x}(n)$ and $\mathbf{y}(n)$ for all $n = 1...T$ as $\mathbf{X} \in \mathbb{R}^{N \times T}$ and $\mathbf{Y} \in \mathbb{R}^{C \times T}$ respectively. From Equation 2.3, we compute $\mathbf{W}^{out}$ by minimizing the squared Frobenius norm between the reservoir outputs and their expected values:

$$||\mathbf{Y} - \mathbf{W}^{out}\mathbf{X}||_F^2 \tag{2.4}$$

The matrices $\mathbf{X}$ and $\mathbf{Y}$ are can be very large depending on the time steps $T$. Inverting large matrices is taxing in terms of memory and computation. A workaround is to

multiply the transpose of $\mathbf{X}^T$ on both sides to obtain :

$$\mathbf{W}^{out}\mathbf{X}\mathbf{X}^T = \mathbf{Y}\mathbf{X}^T \tag{2.5}$$

The solution of which is :

$$\mathbf{W}^{out} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \tag{2.6}$$

In the formulation of Equation 2.4, $\mathbf{Y}\mathbf{X}^T$ and $\mathbf{X}\mathbf{X}^T$ are independent of the input length $T$ making the complexity of the computation of the inverse independent of $T$. Optimizing $\mathbf{W}^{out}$ from the high dimensional $\mathbf{X}$ and $\mathbf{Y}$ is highly prone to overfitting due to the curse of dimensionality and/or the small size of some training datasets. Overfitting can be reduced by penalizing large coefficients in $\mathbf{W}^{out}$. This can be done by introducing a regularization parameter, say $\lambda$, in the Equation 2.5 to obtain :

$$\mathbf{W}^{out} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1} \tag{2.7}$$

where $\mathbf{I}$ is the identity matrix of dimension $N$.

## 2.4 Properties of Echo State Networks

ESNs have certain properties responsible for their usefulness. These properties can be controlled by varying certain parameters of the reservoir as we will describe in Chapter 3. In this section, we explain the properties associated with RC and their contribution to RC computational abilities.

### 2.4.1 Echo State Property (ESP)

ESNs rely on their fading memory to process serial input correctly and capture context. Jaeger [46] coined the term Echo State Property, the ability of reservoirs to retain a fading recollection of input history while at the same time asymptotically eliminating distant information in time. The property ensures that the current state of the reservoir is a result of the recent input information and is insensitive to the initial conditions of the reservoir and the distant history of the input. Without the ESP, the system's state would be dependent on the initial and every transient state visited, which would in turn make computation unreliable. With ESP, the dynamical RC systems yield a uni-modal solution i.e. only one stable solution exists. At a given discrete time $n$, the state $\mathbf{x}(n)$ must be uniquely defined for the given left-infinite inputs ..., $\mathbf{u}(n-2)$, $\mathbf{u}(n-1)$ and $\mathbf{u}(n)$ [50].

ESP is strongly related to the connectivity matrix of the reservoir $\mathbf{W}^{res}$. It has been demonstrated that large values of the spectral radius $\rho(\mathbf{w}^{res})$ can cause the existence of multiple fixed points or even chaotic behavior. A widespread misconception is that, to guarantee ESP, the spectral radius of $\mathbf{W}^{res}$ must be less than a unit and that this is a sufficient and necessary condition [42, 51]. The authors of [50] point out the flaw of this assumption and demonstrate the new sufficient conditions for the echo state property.

The values of the activations $\mathbf{u}(n)$ and the input mask $\mathbf{W}^{in}$ can also impact the ESP of a reservoir. Depending on how far the non-zero input drives the system from the 0 where the sigmoidal function has a more linear slope to regions that squash the activations. This removes the necessity of $\rho(\mathbf{W}^{res}) < 1$ for ESP in some applications. Also, in the case of the generalized RC introduced in Eq. (3) of [52], the ESP is related to the notion of effective spectral radius taking into account the presence of leaky neurons. The effective spectral radius is computed taking into account the nature of nonlinearities, input data, and the input mask; all of which could impact the ESP.

### 2.4.2 Memory Capacities (MC)

Temporal signal analysis requires the ability of the computing system to more or less recall previous information. In engineering, this can be accomplished by employing a delay line. RNNs, however, have an inherent ability to store this history in the transient states. The measure of this amount of recalled history is called Memory Capacity (MC). For ESNs, one way to estimate MC is to evaluate how well the input $u(n-k)$ can be reconstructed at discrete time $n$. The inputs $u(n)$ are independently and identically distributed random variables. The higher the memory the higher the number of times steps $k$ from which we can reconstruct $u(n-k)$. The upper bound of the MC was found to not exceed the size of the reservoir. Other measures of MC were proposed in [53–55] for continuous and discrete time RCs.

The authors of [56] observed that MC depends on the spectral radius of the connectivity matrix $\rho(\mathbf{W}^{res})$. The smaller values of $\rho(\mathbf{W}^{res})$ mean the lower the number of steps $k$ for which we can reconstruct $u(n-k)$ whereas larger values of $\rho(\mathbf{W}^{res})$ reverses the effect. Therefore, tuning $\rho(\mathbf{W}^{res})$ is crucial to adapt RC to the memory needs of the task at hand.

The memory capacity introduced in [51] is called linear memory capacity quantifying the ability to reconstruct $u(n-k)$ assuming a linear relationship. If we let the reconstructed variable from $k$ steps be $v_k(n)$ then this assumption means $v_k(n) = u(n-k)$. In [57] the notion of memory capacity is extended by introducing non-linear memory and cross-memory capacities.

The quality of the reconstruction of $u(n-k)$ is measured as a function of the Nor-

malized Mean squared error (NMSE) by the capacity $C$ calculated as follows :

$$C(v_k) = 1 - NMSE(v_k) \tag{2.8}$$

such that a perfect reconstruction with $NMSE(y) = 0$ will yield a capacity of 1. To compute the total capacity $C$, we sum the capacities overall for all delays $k$.

$$C = \sum_k C(v_k) \tag{2.9}$$

Generalizing the results of [51], the sum of the linear, quadratic, cross, and higher order memory capacities were found to be upper bound by the size $N$ of the reservoir.

### 2.4.3 Operation at the edge of chaos

Dynamical systems such as the ESNs are prone to exhibit chaotic behavior under certain conditions. The setting of different parameters plays an important role in determining the regime of operation as either chaotic or ordered. At certain regions of parameters, the systems are at the frontier between the two operating regimes. This region is called the edge of chaos (or edge of stability for some). There are claims that the computation power of dynamical systems such as RC on time series operating at the edge of chaos is superior [58–60]. This is attributed to the very high memory capacity of the system at the edge of chaos, hence the input history does not die out quickly which is favorable for tasks that require long history and may be detrimental to those that do not. However, although the term edge of chaos is generally accepted, it can be misleading as the recurrent systems tend to pass through a number of stable bifurcation before the onset of chaos.

## 2.5 Physical RC implementations

The randomness of the reservoir connections offers the potential for the hardware feasibility of the RC. The randomness relieves the strict constraint of having to precisely design and tune the physical weights and biases of the neurons. As a result, many dynamical systems capable of yielding nonlinear responses $\mathbf{x}(n)$ to the inputs $\mathbf{u}(n)$ can be used as reservoirs [61, 62]. RCs' versatility due to loose requirements for producing a physical reservoir has attracted attention from various fields such as electronics, mechanics, biology, and photonics to cite a few. The computation units in RC can be emulated by various substrates which have led to numerous *exotic* reservoir designs as we will discuss in this chapter. To design a good hardware reservoir, the proposed

system has to fulfill the following conditions [61]:

- Fading memory (previously described as Echo State Property): This allows for stability and ensures that only the recent input (i.e. limited history) is a determinant of the reservoir output. The hardware can therefore capture context while avoiding instabilities.
- High dimensionality - The hardware reservoir must be able to provide a large number of independent representations of the input information. This way the output benefits from the richer representation of the input signal.
- Non-linearity - The physical systems needs to have some source of nonlinearity. This allows the output system to linearly separate the inputs that would otherwise be linearly inseparable. Non-linearity is essential for the projections to higher dimensions to represent computation,
- Separation property - This property guarantees that two different inputs generate sufficiently different outputs in the much higher dimensional space regardless of the noise and small perturbations that may affect the RC system,
- Approximation property - This property guarantees that sufficiently similar inputs generate sufficiently similar outputs in the much higher dimensional space and remain robust in the face of noise and small perturbations that may be present in the system.

The demands for high dimensions translate to a large number of interacting physical nonlinear components. Typically, simulated RCs have hundreds to thousands of neurons. From the hardware point of view, this translates to the requirements of a large number of nonlinear elements, with all the cabling/linking to one another to form a network; and if powered, cabling to electrical sources for each element. Albeit with difficulty, this can be feasible for smaller reservoirs with few nodes. For instance, in their pioneering work, the authors of [63] implemented a spatiotemporal network of 25 Semiconductor optical amplifiers as a reservoir. Some applications, however, require a large number of nodes whose hardware feasibility is unrealistic. Fortunately, a simpler workaround was introduced by the authors of [64] to alleviate this design impediment.

The proposals in [64, 65] exploit the already well-known fact that, delayed nonlinear systems, that are ubiquitous in nature, have interesting properties linked to the duration and the strength of the delay. These systems were initially considered a nuisance and could now be used for interesting computation applications. They proposed the simplest delay dynamical system consisting of a single nonlinear node coupled to a single

delay line, of say, duration $\tau$. This system is known to be infinite-dimensional since the state of the system at time $t$ depends on the states at previous times taking values from infinite possibilities in the interval $]t-\tau, t]$. This property coupled with the attenuation (fading memory) makes such a simple system adaptable for reservoir computing. Their results encouraged a large number of researchers from various domains to exploit different physical systems for reservoir computation. Below we discuss a few examples of domain-specific RC implementations with varying choices of substrates, nonlinearity, and architectures.

### 2.5.1 Mechanical Reservoirs

#### Water bucket RC

Taking the metaphor quite literally, the authors of [66] took inspiration from Liquid State Machines and Reservoir Computing and set up an experiment using a *reservoir* (a bucket) filled with *liquid* (water). This setup is rather interesting and the exact architecture is shown in Figure 2.3. The serial inputs are fed to the system via four mechanical arms that vibrate in response to a motor. The motor movements are modulated by a current proportional to the input signals generated by a computer. The arms, therefore, represent a low-dimensional mechanical representation of the signal to be processed. The movements of the arm generate ripples on the surface of the water whose interaction generates high-dimensional patterns. The patterns are projected on an anti-reflective surface below after illumination by a projector placed directly above the reservoir. A webcam records the patterns and sends them to the computer for post-processing. The interaction between neighboring neurons is emulated by local nonlinear interaction between close water molecules as they respond to external disturbances. The nonlinear interaction and response to perturbations are the major source of nonlinearity in the system. This setup achieved low classification error on the XOR task and Spoken Digit Recognition, two important benchmarks for machine learning methods.

#### Coupled oscillators RC

In [67] a network of coupled mechanical springs is used as a reservoir of nonlinear nodes. The scheme consists of 400 inertial masses ($N = 400$) coupled with nonlinear springs as shown in Figure 2.4. The masses are under a constant high-frequency periodic disturbance from a mechanical arm on which the low-frequency input envelope modulates the forcing disturbance proportional to the inputs. The position of the masses represents the output states to be recorded for subsequent post-processing. Equation 2.10 governs the evolution of the states of the masses (node states $x_i(t)$) in

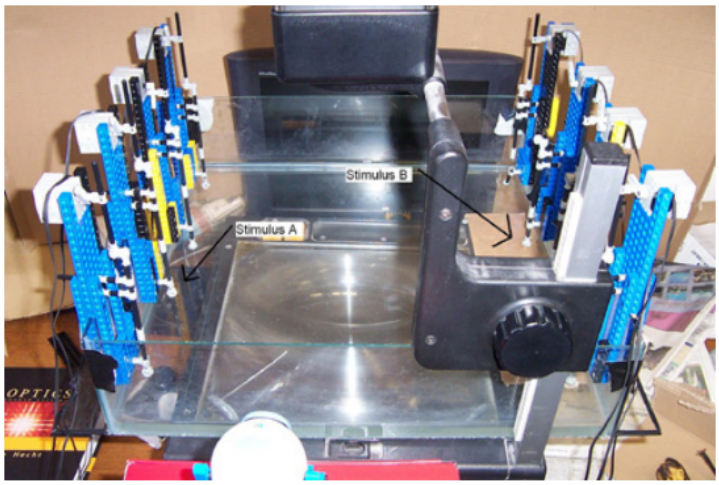



Figure 2.3: Reservoir computing using a reservoir of water

time.

$$\frac{d^2x_i(t)}{dt^2} = \frac{\omega_0}{Q}\frac{dx_i}{dt} - \omega_0^2 x(t) - \beta_i x(t)^3 + A[1+\Delta_i u(t)]\cos(\Omega t) + \omega_1^2[x_{i-1}(t) - 2x_i(t) + x_{i+1}(t)] \tag{2.10}$$

where $i$ is the mass index, $u(t)$ is the time-variant low-frequency input perturbation, $\omega_0$ is the fundamental frequency, and $\Omega$ is the constant high-frequency perturbation. Other parameters are system constants.

The setup, though mechanical can be fabricated in a compact manner using micro-electromechanical technologies resulting in a very high energy-efficient operation.

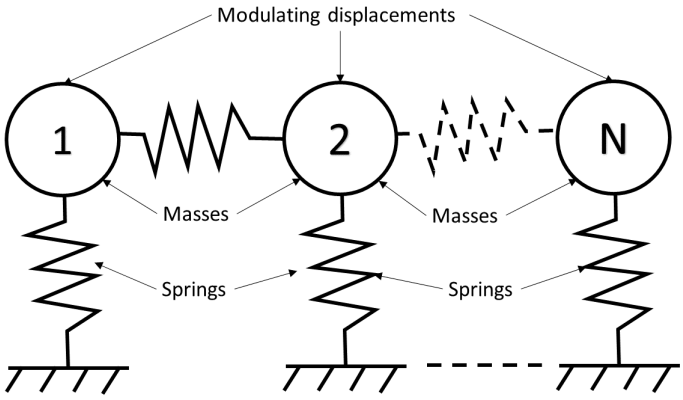



Figure 2.4: Reservoir computing coupled oscillating springs

### 2.5.2 Electronic Reservoirs

#### Analog RC

The simplest RC in this class consists of a single nonlinear electric node coupled to a delay line as a reservoir [64, 68]. The first single-node reservoir was proposed by the authors of [64]. The node states were virtualized as the smaller time intervals along the delay line. The input is adapted from the digital form to a staircase signal from the sample-and-hold operation. This approach simplifies the spatiotemporal architecture making physical implementation more feasible. The input mask is emulated by a periodic signal whose weights are assigned at each symbol duration. The equation describing the temporal evolution of nodes states is the input-driven Mackey-Glass nonlinear equation

$$\frac{dx(t)}{dt} = -x(t) + \beta \frac{x(t-\tau) + \alpha u(t)}{1 + (x(t-\tau) + \alpha u(t))^p} \tag{2.11}$$

where $x(t)$ and $u(t)$ are the node state and input information at time $t$, $\alpha$ is the input scaling and $\beta$ is the feedback scaling (strength) and $p$ controls the nonlinearity in play.

The system is driven by the input signal $u(t)$ yielding states $x(t)$ that are linearly combined to produce the output. The output weights are obtained by the postprocessing step done digitally in an offline manner. Data converters are used to transform to and from digital and analog signals. Other studies proposed different types of single-node electronic reservoirs for analog processing such as the use of multiple reservoirs [69] and emulation of spiking non-sigmoidal neurons [70].

#### FPGA RCs

Field Programmable Gate Arrays (FPGAs) are electronic devices that are essentially a matrix of configurable logic blocks (CLBs) connected via programmable intra- and interconnects. The FPGAs are reprogrammable circuits that can be configured to complete certain digital processing tasks at high speeds. Their reconfigurability and ability to run concurrently make them suitable for RC. They can be used to implement the reservoirs and/or be employed for the online processing of readouts [71]. Although sometimes used for single node RCs [72], the Concurrence allows for FPGAs to generate network-based RC (non-delay based).

The authors of [73, 74] proposed a stochastic bitstream neuronal network that uses an FPGA. The traditional neurons use a weighted sum of the input excitations to generate an output. This translates to multiple Multiply-And-Add operations that are not hardware friendly. To counteract this shortcoming they used stochastic arithmetics, that is every value of the network is converted to its statistical representation. For example,

if the output is uni-polar, that is $x \in [0, 1]$ then it can be represented as a probability of having a 1 such that $P(x' = 1) = x$. The simplified design was used for benchmark tasks such as the reproduction of a sine wave and generating the phase-shifted version of a sinusoidal signal with promising results.

Some others implemented the more sophisticated spiking neural networks (SNNs) on FPGAs. The SNNs are essentially LSMs processing spikes instead of the sigmoidal outputs. They were explored for various tasks such as isolated spoken digit recognition [75] and image processing [76]

### Memristive RCs

A memristor, the name etymologically derived from *Memory Resistor* is a two-terminal component whose resistance depends on the current that has flowed through it [77, 78]. This interesting characteristic has been exploited for neuromorphic computation systems and more specifically for reservoir generation. The memristors can be used as physical synapses with their conductance tuned to emulate synaptic weights of artificial neuronal connections [79–82]. In these works, the neurons are either spiking or sigmoidal and the memristors serve as random synaptic weights for the reservoir networks, as input mask weights, or as the output weights for linking the reservoir layer to the output layer. Other studies considered memristors in a more central role and deployed them as the nonlinear nodes responsible for computation. They exploit the fact that memristors exhibit nonlinear dynamics that depend on the history of the traversing currents and can, therefore, be used for reservoir design [83–85]. The use of memristors for neuromorphic computing is attractive because they require very low power to function even for network-like reservoirs offering efficient alternative components for hardware neuromorphic computation [86].

### Very Large Scale Integration (VLSI) RCs

VLSI refers to an integrated circuit technology with numerous devices incorporated on a single chip. Neuromorphic computing usually consists of a large number of interconnected elements that can be fabricated in a VLSI fashion. For instance, some explored an Application specific architecture, a VLSI they called a *hard liquid*, with analog and digital processing parts [87]. Others employed a platform with Leaky Integrate and Fire (LIF) neurons with random connectivity on a re-configurable platform [88] and used it for the biomedical detection of epileptic seizures. Integration is a promising avenue for electronic RC since it allows the fabrication of compact and energy-efficient reservoirs.

### 2.5.3 Photonic Reservoirs

#### Optical node array RCs

This category consists of reservoir designs with multiple interconnected optical nodes resulting in a spatiotemporal architecture. The first proposal for this type of reservoir was proposed in a simulation experiment [63] followed by experimental realization in [89] consisting of an array of $4$ Semiconductor Optical Amplifiers (SOAs) as a reservoir. Others proposed reservoirs based on photonic crystals [90] and microring resonators [91]. Some studies explored free space such as the diffractive network of semiconductor lasers [92]. They used an $8 \times 8$ square lattice of single-mode vertical-cavity surface-emitting lasers (VCSELs). Light originating from the lattice is passed through the Diffractive Optical Element (DOE) before being subsequently imaged on a Spatial Light Modulator (SLM). The loop is completed with feedback from the SLM whose strength is controlled by the adjustable attenuation controlled by the SLM's grayscale values.

Light propagating through any medium gets scattered in random locations and directions. This can be quite an inconvenience in some domains of applications such as tissue imaging for example. However, the rich randomness of the scattering forms an image called a speckle figure that can be exploited as the random reservoir connectivity matrix. The authors of [93] exploited this by expanding a coherent light to cover the surface of a Digital Micromirror Device (DMD). The DMD micromirrors determine the reservoir state by the on and off state of individual mirrors. The on-mirrors reflect light to the SLM and a camera records the image and uses it as feedback to the DMD completing the loop. The scattering mediums provide the random coupling and the nonlinearity of the system stems from the square modulus detection of the camera. The computational power of the device is verified on the regression task of the Mackey-Glass time series.

#### Delay-based optical RCs

Of the proposed photonic reservoir, the delay-based implementations represent a vast majority of the studied architectures. This is the consequence of the availability of off-the-shelf constituent components in the laboratories and the versatility of the resultant implementations. The spatiotemporal architectures previously discussed are limited in number as they suffer routing congestion when physical implementations are envisioned. The delay-based architecture is a ring-type topology with signals evolving as they circulate in a loop. This loop has openings, one for the input injection and the other for the readout as shown in Figure 2.5. The neurons in this case correspond to smaller

time intervals of an equal duration along the delay line. For a delay line of duration $\tau_D$, $N$ virtual nodes correspond to the $\tau_D/N$ smaller intervals distributed along the line. In the literature, we find two major classes of optical delay RCs, again depending on the choice of the constituent devices and the nonlinearity. First, the fully optical reservoir implementations whose nonlinearity and reservoir states are generated and mixed with the input fully in the optical domain [94, 95]. The second class includes reservoirs necessitating the Electrical-Optical conversions for the nonlinearity, information injection, and/or mixing of the feedback [96].

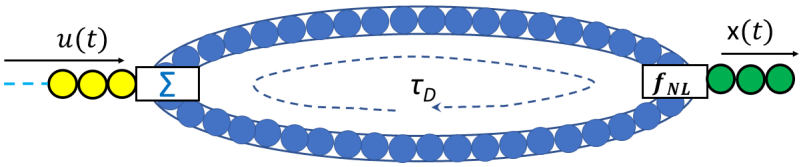



Figure 2.5: Schematic representation of delay reservoir architecture. The delay line is of duration $\tau_D$: discrete-time inputs (yellow circles), discrete-time nodes of duration $\tau_D/N$ (blue circles), discrete-time read-outs (green circles).

The authors of [95] demonstrated a fully optical reservoir using a semiconductor laser as both the light source and the source of nonlinearity. The delay line is a standard single-mode fiber of length equating to a light propagation duration of $77.6ns$ along which 388 neurons of $200ps$ each are obtained. The input signal is introduced into the reservoir via two proposed openings: one is through the electrical current modulating the laser and another is through optical injection into the laser via the feedback loop. A part of the light in the loop is extracted by an optical coupler and photo detected to provide the readout signal for postprocessing. This setup is tested on spoken digit classification task and the chaotic time-series prediction yielding good results.

Another example of a fully optical implementation of RC is the study that employed a Semiconductor Optical Amplifier (SOA) as a source of nonlinearity [94]. For a nonlinear operation, the SOA is driven to saturation. With a delay of $7.9437\mu s$ they obtained $50$ neurons of $155.76ns$ each. For local coupling (mixing) of neurons, the desynchronization between the input injection and round-trip time was used since the nonlinearity, in this case, is instantaneous. This setup yielded good performance for several tasks: the radar prediction task, the isolated spoken digit prediction task, and the nonlinear channel equalization task.

These implementations of the all-optical reservoir are groundbreaking as they gave a glimpse into the long-standing question of all-optical computing. However, fully optical implementations are fewer compared to optoelectronic ones since optical nonlinearity

is hard to harness for computation. The use of SOA in [94] necessitated driving the SOA to saturation which brought along its high levels of noise impacting the performance of the system.

### Delay-based optoelectronic RCs

The first delay-based photonic system was an optoelectronic reservoir as the delay loop with optical-electrical-optical conversions. Larger et al. [96] designed an optoelectronic setup that explores the Ikeda-type nonlinearity described by the equation:

$$\frac{dx(t)}{dt} = -x(t) + \pi\mu[1 + 2B\cos(x(t-\tau) - x_0)] \tag{2.12}$$

where $\mu$ is a system constant, $\pi$ is the ratio of the circumference of a circle to its diameter, $\tau$ is the duration of the delay line, $x(t)$ is the state of a node, and $x_0$ is the initial state. This equation introduces the cosine function that can be emulated in the hardware implementation by a Mach-Zehnder Modulator (MZM).

The experimental setup consists of a DFB laser in CW operation. The emitted light goes through an MZM whose RF arm is modulated by an electronic circuit. The resultant modulated signal is delayed in a fiber spool of duration $20.86\mu s$ before being photo detected and sent to an electronic circuit. This circuit acts as a low pass filter allowing some local coupling between the neurons that are essentially time intervals. It also amplifies the signal and sends a part to the processing units and combines a part with the incoming information to complete the loop. The preprocessing and postprocessing were carried out offline on a computer. The computation power of the setup tested on isolated spoken digit benchmark yields error rates as low as $0.5\%$. They also studied the impact of nonlinearity on computation by varying the bias voltage of the MZM. This setup is also the setup of interest for our experiments in this thesis. We provide further details of its components and mode of operation in Section 2.6.

A similar optoelectronic RC implementation was studied by the authors of [97]. They incorporated an MZM for nonlinearity and modulation hence the setup experiences the Ikeda-type nonlinearity as well. However, unlike [96] where the electronic circuit had some intrinsic time scale responsible for local coupling of the nodes, this setup had instantaneous non-linearity. This means there is no mixing and that the reservoirs are one-dimensional. To introduce coupling they employed a sample and hold procedure with the hold duration $\tau$ slightly less than the delay line duration $\tau$. This allows each part of the signal to be used for the generation of states in the near future. Also, the readout was done by tapping the delay line using a coupler and photodetecting the signal for subsequent processing. They tested their setup on several known

benchmarks for time series computation. For the Nonlinear Auto Regressive Moving Average (NARMA) equation of order 10 driven by white noise, they obtained a Normalized Mean Square Error $NMSE = 0.168 \pm 0.015$ which was similar to the digital implementation of the system. The setup also was tested for the nonlinear channel equalization of a wireless communication channel. The signal distortion is due to multiple propagation paths of the transmitted signal before reaching the nonlinear channel. The experimental setup yielded a very low error rate of $1.3 \times 10^{-4}$ falling within the ranges reported by more complex methods. They also tested the setup on the spoken digit recognition benchmark and obtained a Word Error Rate (WER) of $0.2\%$.

In [98] the versatility of RC is further explored. The authors pushed the limits by processing three different tasks on a single circuit. They interleaved three *virtual* reservoirs in a single delay line and employed each to process a certain task. By doing so, they were able to increase the spectral efficiency from $12\%$ to $36\%$ of the total $125Mhz$ bandwidth they had. The delay line was $\tau = 69.96\mu s$ meaning $N = 424$ nodes are possible. However, for the proposed tasks, reservoirs of 50 nodes sufficed. As a result, only $150$ of $424$ nodes were used. Their idea consisted of interleaving three internal variables of duration $\theta = 165ns$ each. Each variable represents the node state for 3 reservoirs. They processed three benchmark tasks, the Nonlinear Channel Equalization task, the RADAR signal forecasting task, and the NARMA10 task described in Section 2.7. Their findings showcase the possibility of running several reservoirs concurrently without hurting the performances of each. The difficulty is that the hyperparameters required for each task, say the input and feedback scalings, are different. The problem can be solved by introducing modulators in the delay loop and the input allowing the tuning of these parameters separately to their optimal values. These solutions were implemented and studied in their second work detailed below.

The power of analog computing in photonic hardware implementation suffers a serious bottleneck. The need for input signal-mask multiplication (Equation. 2.1) at the input layer and the internal states-readout matrix multiplication (Equation. 2.3) at the output layer imposed the use of digital electronic devices that are slower. This motivated the authors of [99] to propose and study the setup with an analog readout. They tested their experimental setup on a nonlinear wireless channel equalization and obtained results less accurate than the digital counterpart. The reason is that analog processing is more prone to noise than digital domains. Extending on [98, 99] a proof of concept fully analog RC was implemented with the input masking operation and linear combination to generate output accomplished *online* in an analog manner [100]. The reservoir design consists of an SMF fiber of $1.7km$ coupled to an MZM modulator responsible for the system's sinusoidal nonlinearity. The spool was divided to have 47

internal variables. The dynamic of the system can be described by :

$$x(t) = \sin(\alpha x(t-\tau) + \beta m(t)u(t)) \tag{2.13}$$

where $x(t)$ are the node states along the delay, $u(t)$ is the input information, $m(t)$ is the periodic mask function which is implemented in an analog manner and $\alpha$ and $\beta$ are the scaling parameters. For the input layer, the goal is to generate an optical signal $I(t)$ by sample and hold procedure using an Arbitrary Waveform Generator (AWG) whose output, $u(t)$ modulates the RF arm of MZM as follows :

$$I(t) = I_0 u(t) \tag{2.14}$$

Where $I(t)$ is the time-dependent intensity and $I_0$ is the maximum amplitude. A second AWG, with the same baud rate, generates the input mask $m(t)$ that modulates the second MZM whose input light is the output of the first MZM ($I(t)$). The resultant optical signal to be injected into the reservoir layer $I_{in}$ is, therefore :

$$I_{in}(t) = I(t) \times m(t) = I_0 u(t) m(t) \tag{2.15}$$

For the output layer, $30\%$ of the light in the loop goes through an MZM modulated by a third AWG that sends an analog readout matrix ($w(t)$) through a balanced photodiode setup for detection. The use of a balanced photodiode setup instead of a single photodiode allows negative components to be computed. The fully analog RC performed slightly worse compared to the one where inputs and output layers are implemented digitally for the nonlinear equalization task and even worse for the NARMA10 and RADAR tasks. This can be attributed to the noise tolerance of classification tasks as opposed to regression tasks. The other two are regression tasks requiring more precise computations hence the noise has a more profound impact on performance. However, the system yielded promising results in the direction of fully analog computing and paves way for fully analog implementation of RC has the potential for high-speed computing.

A very fast RC developed by authors of [101] can process one million words (spoken digits) per second. The speed stems from the use of high bandwidth of the electro-optic phase delay setup. It consists of two Electro-Optic Phase modulators (EO PM), the first one modulated the light's phase proportional to the input information whereas the second modulates the light proportional to the feedback. The resultant accumulated phase change is therefore

$$\phi(t) = x_i(t) + \rho u_i(t) \tag{2.16}$$

where $u(t)$ and $x(t)$ are input and feedback respectively with the input scaled by $\rho$. Notice the similarity with the general state evolution Equation 2.1. It had a delay fiber spool of total duration $\tau = 63.33ns$. It also has a passive fiber-based Mach Zehnder Interferometer with a time imbalance of $\delta T = 402.68ps$ that nonlinearly converts the signal phase variation to intensity variations according to the standard two-wave intensity interference function as follows

$$f_{NL}(\phi) = \kappa\{\cos(\phi(t) - \phi(t - \delta T) + \Phi_0) - \cos^2(\Phi_0)\} \tag{2.17}$$

where $\kappa$ is the gain and $\delta T$ is much greater than the smallest time scale of the system. Two amplified photodiodes are employed to convert the optical fluctuations to electrical ones for the feedback and output post-processing respectively. The nodes experience local coupling thanks to different timescales with filtering but also distant mixing according to the $\delta T$ of the interferometer. A filter of the characteristic response time of $\tau_R = 284ps$ slows down the $10Ghz$ bandwidth for node coupling and also for the electronic AWG slower sampling rate. The resultant equations governing the dynamics of the system are :

$$\begin{aligned} \tau_R \frac{dx(t)}{dt} &= -x(t) + \frac{y(t)}{\theta} + f_{NL}\big[\psi(t - \tau_D)\big] \\ \frac{dy}{dt} &= x(t) \end{aligned} \tag{2.18}$$

The setup was optimized numerically and experimentally for two Isolated Spoken digit datasets, the cleaner TI46, [102] and the more noisy and distorted Aurora-2. The test performances yielded perfect detection ($0\%$ error) for the former dataset. As for the Aurora-2, the performances were $4.5\%$ and $8.9\%$ respectively for numerical and experimental implementations. The reported experimental performance was obtained at an impressive processing speed of $1$ million words per second highlighting the power and speed of RCs.

## 2.6 The optoelectronic setup by Larger et al.

In this Section, we detail a setup of particular interest in some of our subsequent presented works. We work with the optoelectronic compact setup provided to us by Laurent Larger one of the authors of [96] with FEMTO-ST in Besançon, France. It is an Eletcro-Optic (EO) intensity chaos setup shown in Figure 2.6 consisting of off-the-shelf components that we mounted in our laboratory (laser, photodiode, and the modulator).

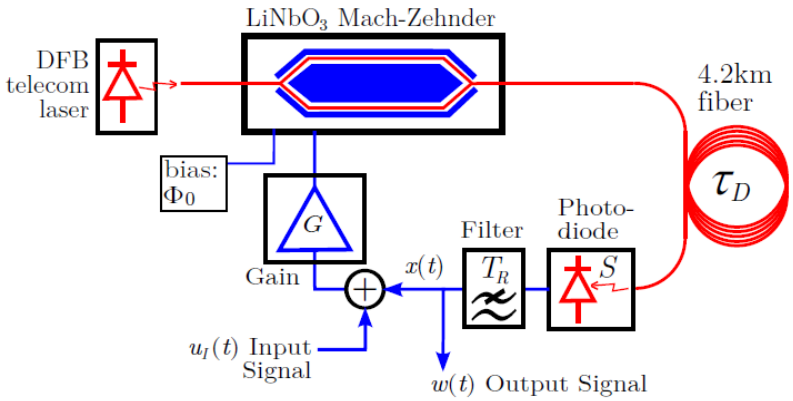



Figure 2.6: Setup 1 from FEMTO-ST (Reprinted with permission from © The Optical Society).

We describe the constituent components as follows :

- A distributed feedback (DFB) laser diode (with output power up to $20mW$) emits light at wavelength $1550nm$. The injection current pumping the laser is an important parameter to tune the overall gain of the optoelectronic system [103].
- A *LiNbO*$_3$ Mach-Zehnder Modulator (MZM) receives the emitted light. The transfer function of the MZM is a sinusoidal function. By appropriately setting the bias voltage ($V_{bias}$ or $\phi_0$) we fix the point around which the nonlinear dynamics fluctuate. The light from the laser is modulated by a signal proportional to the continuous-time input signal $u_I(t)$ to be processed[104].
- A $4.2$ km fiber spool holds the light for an equivalent duration of $\tau_D \approx 20.87\mu s$ constituting the delay of the system.
- A photodiode receives and converts the optical intensity variations to electrical variations generating an electric signal proportional to the feedback signal or the state of the system $x(t)$ [105].
- An electronic feedback circuit completes the optoelectronic loop and has the following functionalities :
  - Amplifies the electrical signal from the photodiode this is important to obtain an appropriate voltage swing necessary for the MZM.
  - Acts as a low-pass filter with characteristic response time $T_R$ slowing down the dynamics of the system and inducing neuronal local coupling
  - Splits the electric signal in a part for the readout recording for the postprocessing stage.

- Enables the summation of the feedback photodetected signal $x(t-\tau_D)$ with the input $u_I(t)$ to give a single electric signal.
- Sends the signal sum to the RF arm of the MZM to complete the EO loop.

In this way, $N = 400$ virtual hidden nodes are created via time-division-multiplexing, where the delay-spacing between consecutive nodes is $T_s \approx 52.18ns$ [96]. The filter response time $T_R = 4.6T_s$ allows for local coupling between nodes emulating the sparse connections in $\mathbf{W}^{res}$. The DDE equation governing this nonlinear delay system is given by

$$T_R \frac{dx(t)}{dt} + x(t) = \kappa \sin^2(\alpha u_I(t-\tau_D) + \beta x(t-\tau_D) + \phi_0), \tag{2.19}$$

where $\phi_0$ is the MZM bias and $\kappa$ is the nonlinearity gain.

In addition, we use an AWG for the generation of the continuous time input $u_I(t)$ from the discrete-time samples $\mathbf{W}^{in}\mathbf{u}(n)$ via a sample-and-hold procedure. Finally sampling the electronic feedback circuit output with a DAC $N$ times at the sampling period $T_s$ reconstitutes the hidden state vector $\mathbf{x}(n)$. The linear output layer can then be implemented via digital postprocessing on a computer.

## 2.7 Applications

The importance of RC is highlighted by the number and significance of the various tasks it processes which has drastically increased recently. RCs have been studied and employed for applications whose data is obtained from various sources, be it simulated data or real noisy and distorted signals from sensors. Often a fixed random reservoir can be applied to multiple applications with minimal to no adjustments by only training the output layer. Moreover, the low complexity and simplicity of the concept have attracted ML experts and non-experts from various domains. In this Section, we present and discuss the nature, importance, and performances in various domains of applications. In the literature, the common applications employing reservoir computing can be categorized into the major classes described below.

### 2.7.1 Pattern generation

Pattern generation is the process of creating a signal with certain properties that can occur repeatedly given the same settings. Patterns can represent physical systems, living organisms, machines, or even designs. In some applications it can be useful to generate signals of certain properties and RC can be useful in such cases. A simple task of generating a sine wave time series of a certain frequency (or period) is studied in [106].

The goal is to generate an output of the form:

$$y(t) = \sin(\nu t) \tag{2.20}$$

where the pulsation $\nu$ has its value limited by the memory of the reservoir and the limits of their experimental setup. Due to the limited memory, the authors observed the success of the generation fell to zero when very long periods were considered. Note that output (or readout) feedback is needed in such a case.

In many scenarios, random number generation finds interesting applications. The authors of [106, 107] study this relatively more complex task compared to the previous one, which consisted of repeated generation of a sequence of random numbers. The results show that this is heavily dependent on the memory of the system as it aims to recall the sequence in essence. Apart from the memory, noise has more impact on the performance of the proposed setup. They obtained similar observations for the other pattern generation task: the Mackey-Glass series generation (see Equation. 2.11) and the well-known Lorenz Chaotic Series Generation show dynamics are defined by :

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= -xy + rx - y \\ \frac{dy}{dt} &= xy - bz \end{aligned} \tag{2.21}$$

where $\sigma$, $r$ and $b$ are real and positive.

### 2.7.2 Time series forecasting

Time series forecasting is the process of making a scientific prediction based on previous historical sequences. The goal is to train a model via supervised learning and then use it to predict future trends. The necessity of predicting future trends is ubiquitous and has been applied in various domains such as weather, finance, health, and communications. We discuss a few examples of time series forecasting tasks below.

Jaeger and Haas studied the use of RC for chaotic series prediction and nonlinear channel equalization [108]. They studied the Mackey-Glass System governed by Equation. 2.11 which is a seemingly irregular complex series. They trained the system using a teacher sequence to obtain a readout $\mathbf{W}^{out}$ which was then used to forecast the system's outputs several time steps ahead. Moreover, the authors of [96] evaluated RC's ability in the prediction of chaotic behavior of a far-infrared laser operating in a chaotic state. The ability to predict chaotic behavior is important since complex systems are

prevalent such as the weather and systems such as lasers subjected to feedback.

Reservoir computers have also been employed in modeling and predicting complex financial stock markets and other financial market trends [109, 110]. Using the daily closing stock prices of the S&P500 Index, New York Stock Exchange Composite, Dow Jones Industrial Average, Nasdaq Composite Index, Financial Times Stock Exchange 100 Index, Nikkei 225 Index, and Shanghai Stock Exchange Index the authors of [109] successfully trained a reservoir to predict the future trends in the stock market indices. Their results were competitive when compared to larger and deeper neural networks. In [110] an RC-based financial model is studied and used to forecast the long-term behavior of the financial system with high accuracy. The RC model also proved to be robust to rapidly changing trends and other fluctuating dependencies. These are interesting applications since the ability to predict behavior in the stock or other financial markets is crucial. Predictions drive the decision-making and policy formulations in favor of the practitioners. It is worth noting that RC won the financial time series challenge [111] highlighting the potential of RC in financial analysis, modeling, and prediction.

Other works considered the RADAR signal forecasting application. Traditionally RADAR was mostly considered in defense systems but today it has applications in navigation, weather forecasting, space exploration, and pollution control. In [98, 100, 112] RC is employed in the prediction of the RADAR signal from 1 to 10-time steps in the future from the radar signal back-scattered from the ocean's surface. There are many other instances of RC use in the forecasting of time series with for diverse applications.

### 2.7.3 Time Series Classification

Time series classification is a process of attributing serial data to a group with similar properties. This is done by supervised learning using labeled sequential data before predicting on the new unseen data. In many domains, it is important to group observations into smaller groups. The classification aids in taking the right measures in the functioning of machines in industrial settings or decision-making for financial or social situations. In RC literature, we find many researchers that have designed and implemented RCs for various classification applications. Classification tasks are attractive to RC because, unlike the regression counterparts, performances suffer less degradation by noise-infested analogy computing. We provide and discuss a few examples of classification tasks below.

In communication systems, the transmission of a signal from the transmitter to the receivers faces many distortive channel imperfections that can potentially cause the misrepresentation of the intended message. For wireless channels, the degradation originates from the involved components' thermal noise, the interference with parasite sig-

nals, multi-path propagation which leads to a superposition of adjacent symbols, and nonlinear distortion induced by the sender's and/or receiver's amplifiers. For optical channels also a plethora of degrading influences and forces exist such as the optical fiber nonlinearity at high powers, the dispersion by the spreading out of a light pulse in time as it propagates down the fiber, and noise from the components. RC has been successful in restoring the intended message from the distorted signals for wireless [98, 100, 112, 113] and optical systems [114–118].

Some studies considered the visual data processing both in the form of still images [119] or sequential images in form of video [119–121]. The authors of [119] employed RC an unconventional task of still images processing, which is not an essential time series task. They used the well-known handwritten dataset, the MNIST dataset for visual digit classification. To convert the time-independent images to sequences adapted for RCs, they scanned the images pixel-wise from left to right forming time-series-like sequences. For this task, RC yielded a competitively low error rate of only $0.92\%$ for the MNIST dataset. They extended their experiments to include actual video data, where self-made videos of a door being closed or opened are used to train the RC. For this, the RC yielded $1.9\%$ prediction accuracy.

Furthermore, RC has been considered in a stand-alone [120] and composite fashion with a CNN as a feature extractor [121] for visual spatiotemporal applications. In [120] the CNNs employed were the pre-trained VGG-16 and ResNet-50, which are already trained to extract relevant features from images. The extracted features are sent to the RC after passing through the dense layer frame by frame. The input video sequences considered are: DogCentric activity dataset i.e. a video stream from a camera mounted on a dog's back as it does different movements and the second is the UECPark dataset with recordings of a human either jogging, twisting, or resting with a camera mounted on his forehead. The combination resulted in $77.2\%$ and $78.7\%$ in accuracy for the Dogcentric and UECPark tests respectively. The standalone setup was tasked to classify human action from sequences of videos of people either walking, jogging, running, boxing, hand waving, or hand clapping. The photonic RC performed with an accuracy of $91.3\%$ and with a promise for higher speeds compared to the state-of-the-art. Video processing tasks are crucial for different applications, be it: for security through surveillance video processing, computer vision, or robotics. The challenge is the requirement of speed and computation power since videos contain large amounts of data to be processed. As a consequence, low complexity, low consumption, and hardware RC are very attractive research directions in this domain.

Automation of spoken digit recognition is an attractive solution as it simplifies communication between people and machines in the form of sound. Digits represent a sig-

nificant part of our exchanges. As a result, we find interesting applications in settings such as the exchange of bank information, bills, taxes, social security numbers, and addresses. It is for this reason that Spoken Digit Recognition (or Speech Recognition in general) was established as an important benchmark for machine learning methods. It is also one of the most utilized benchmarks in the RC literature for testing and reporting the reservoir's predictive performance [96, 97, 122], versatility [66, 67] and speed [95, 101].

Another family of applications belongs to the biomedical field where various time-varying signals, from sensors or electrodes, are used to either provide a diagnosis or information about the underlying medical conditions. The electrical signals from the heart known as the ElectroCardioGram (ECG) have been processed by RC to identify various heart conditions [69]. Other studies involved the electroencephalogram (EEG) from the intra-cranial of rats [88, 123]. Of these studies, one employs RC for the task of classifying two types of signals: those signals corresponding to the absence of seizures from the Genetic Absence Epilepsy Rats from Strasbourg (GAERS) and others corresponding to tonic-clonic seizures from the kainate-induced temporal-lobe epilepsy rats. We also studied the early-stage detection of Alzheimer's disease from handwriting by focusing on accuracy [124] and efficiency [125]

| ***Domain*** | ***Studies*** |
|---|---|
| *Robotics* | Event detection [126] |
| *Security* | Crack chaos-based cryptographic protocols [127] |
| *Financial* | Stock price prediction [109] and Financial Modeling [110] |
| *Visual* | [119–121] |
| *Communications* | Wireless [98, 100, 112, 113] and Optics [114–118, 128, 129] |
| *Audio* | Spoken Digit Classification [66, 67, 95–97, 101, 122] |
| *Biomedical* | Heart Conditions [69], Epileptic seizures, [88, 123] and Alzheimer's disease [124, 125] |

Table 2.1: Various application domains for RCs in the literature

We have described several applications of RC analog computing in this section. Table. 2.1 summarizes different classes of applications we found in the literature.

# 3 Parameter optimization Reservoir Computing

*Part of the results and discussions in this chapter are presented in our journal article titled "A stochastic optimization technique for hyperparameter tuning in reservoir computing" that, by the time of this writing, has been submitted and is under review.*

## 3.1 Introduction

The power and versatility of Reservoir Computing stems from the fact that most reservoir parameters, such as elements of the input mask and connectivity matrices, can be chosen randomly, as discussed in Chapter 2. This relieves the strict requirement to adjust a large number of weights and biases of the recurrent system, unlike the case of traditional RNNs. However, fully random-generated reservoirs seldom yield optimal performances [42]. Some parameters, although much fewer than conventional RNNs, require tuning to obtain the best performance for the task at hand.

In addition, we pointed out that with the randomness in the reservoir layer comes the possibility for hardware implementation with unconventional devices. However, the real-time processing capability of such devices brings a potpourri of challenges nonexistent in digital implementations. The characteristics of constituent physical devices are not always perfectly known and can even drift with time, unlike the case of digital deterministic variants. The consequence is the impossibility of having neither the full knowledge of the reservoir's feedback weights nor the exact nonlinearity of the system hinders the direct application of fast deterministic gradient-based methods such as [52, 130, 131]. Brute force parameter selection uses grid search [132] but requires fine-grained resolution, which, as we will show, can be expensive. More efficient alternatives are based on stochastic optimization, such as genetic algorithms (GA) [133], particle swarm optimization (PSO) [134], or simulated annealing (SAN) [135]. These methods, nevertheless, are usually very time-consuming because they need to repeat

similar steps a large number of times to find quasi-optimal parameters.

To address the aforementioned issues, we propose an optimization technique for tuning a $p$-dimensional vector of RC parameters based exclusively on noisy loss function evaluations. Our algorithm strikes a balance between the fast convergence of deterministic gradient-based methods and the efficiency of stochastic approximations. This is useful in contexts where the exact functional relationship between the parameters and loss function values is unavailable, as is the case for hardware implementations of RC.

Similar methods building on the Robbins-Monro algorithm for root finding [136], procedures for finding the optimum of a loss function based on stochastic finite-difference gradient approximations have been proposed. For instance, the original Kiefer-Wolfowitz algorithm [137] needs $2p$ noisy function evaluations per iteration, which is costly. It was later simplified to employ the random direction [138, p. 58-59], resulting in only a pair of function evaluations per iteration. At each iteration, the latter method creates a two-sided gradient approximation by applying a random perturbation vector sampled uniformly and independently from a $p-$hypersphere in the parameter space. To guarantee convergence, such procedures usually require strong assumptions on the perturbations, the decreasing learning step-sizes, and the decreasing finite-difference steps [139].

Taking into account the shortcomings mentioned above, we simplify the random direction method in several ways to extend the applicability by :

- generating perturbations as random standard unit direction vectors in the hyper-parameter space,
- keeping the finite-difference step constant, and
- keeping the learning step-size constant.

These modifications are essential for two reasons: first, they are responsible for the low complexity of our approach, and secondly, they increase the numerical robustness in the case of hardware RC, where the noise-ridden loss function computation and slow device time-variations require an ability to track parameter drift. We derive a local convergence analysis based on quadratic approximations of the loss function. Although these convergence results are weaker than the original method, our results show that they hold in practice. In particular, successful hyperparameter optimization will be demonstrated both for the digitally simulated and experimental hardware RC.

**Notations** used throughout-out this chapter is as follows; bold letters indicate vectors and matrices while $\mathbf{I}_m$ is the $m \times m$ identity matrix. For any vector $\mathbf{a}$, $\mathrm{diag}\,(\mathbf{a})$ is the diagonal matrix whose principal diagonal is $\mathbf{a}$ while off-diagonal entries are zero. For

any matrix, $\mathbf{B}$, $\mathrm{diag}(\mathbf{B})$ is the diagonal matrix, whose diagonal entries are the elements on the principal diagonal of $\mathbf{B}$ while off-diagonal entries are zero. Also $\Lambda(\mathbf{B})$ denotes the set of eigenvalues of $\mathbf{B}$. A sequence of vectors $\{\mathbf{v}(t)\}_{t=m}^{n}$, i.e. vectors from discrete time $m$ to $n$, stacked columnwise in a matrix is denoted by $\mathbf{v}(m:n)$. The Frobenius norm of matrix $\mathbf{M}$ is denoted by $||\mathbf{M}||_F$. The matrix operators $\otimes$, $\circ$, $\mathrm{vec}(.)$ and $\rho(.)$ denote the Kronecker product, the Hadamard (elementwise) product, vectorization, and the spectral radius, respectively.

**Chapter outline and summary**

- In Section 3.3, we introduce our stochastic gradient optimization method.
- Then, in Section 3.4 , we tackle the performance analysis issue by providing a convergence criterion and the expression of the residual error covariance.
- In Section 3.5, the effectiveness of the proposed hyperparameter optimization method is investigated in three realistic applications i.e. speech recognition application, Control Charts classification, and Wafer Classification both for a simulated RC and in an experimental optoelectronic hardware implementation.
- Finally, in Section 3.6 , we conclude our study of the proposed stochastic gradient descent approach. We highlight the advantages of our system over similar optimization algorithms and the perspectives for further research.

## 3.2 System model

Most of the RC system notions have already been addressed in Chapter 2. Here, we adapt the previous to set the stage for the proposed optimization scheme. For instance, in this chapter, we introduce two parameters ($\alpha$ and $\beta$) and rewrite the discrete state Equation 2.1 as follows

$$\mathbf{x}(n) = f_{NL}(\alpha \mathbf{W}^{in}\mathbf{u}(n) + \beta \mathbf{W}\mathbf{x}(n-1)), \tag{3.1}$$

where $f_{NL}$ denotes, the nonlinearity applied componentwise. $\mathbf{W}^{in} \in \mathbb{R}^{N\times F}$ and $\mathbf{W} \in \mathbb{R}^{N\times N}$ are the sparse random matrices. The hyperparameters $\alpha$ and $\beta$ stand for the input and feedback scaling factors. They significantly impact the system's dynamics because they scale the activations and the spectral radius (thus memory), respectively, and must therefore be fine-tuned [46]. Let us collect all hyperparameters into the vector

$\boldsymbol{\theta}$. A natural choice for $\boldsymbol{\theta}$ is $[\alpha, \beta]^T$, but more hyperparameters could also be included if needed. The linear output layer subsequently generates the $n$-th readout, $\hat{\mathbf{y}}(n)$, as prescribed in Equation 2.3 When RC performs a prediction (resp. a classification) task, $\hat{\mathbf{y}}(n)$ is typically in $\mathbb{R}$ (resp. in $\mathbb{R}^C$, where $C$ is the number of classes).

Learning a RC model based on a training dataset $\{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^T$ consists in minimizing the loss function [46]

$$L(\boldsymbol{\theta}, \mathbf{W}^{out}) = ||\mathbf{y}_{train}(1:T) - \mathbf{W}^{out}\mathbf{x}(1:T)||_F^2 + \lambda||\mathbf{W}^{out}||_F^2 \tag{3.2}$$

where Tikhonov regularization with parameter $\lambda$ is used to limit the effect of noise and overfitting [42]. The computational complexity of a single loss function evaluation is $\mathcal{O}(CT(N+2)+CN)$. This can be obtained from the asymptotic complexity of matrix subtraction and multiplication in [140]. This learning problem consists in solving

$$\frac{\partial L(\boldsymbol{\theta}, \mathbf{W}^{out})}{\partial \mathbf{W}^{out}} = \mathbf{0} \tag{3.3}$$

and admits a unique hyperparameter-dependent closed-form solution expressed as

$$\hat{\mathbf{W}}^{out}\left(\boldsymbol{\theta}, \{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^T\right) = \mathbf{y}_{train}(1:T)\mathbf{x}(1:T)^T\left[\mathbf{x}(1:T)\mathbf{x}(1:T)^T + \lambda\mathbf{I}_N\right]^{-1} \tag{3.4}$$

whose computational complexity is $\mathcal{O}(CN(N+T)+N^3)$ per hyperparameter vector and training set.

## 3.3 The proposed stochastic gradient method

In this Section, we introduce the proposed stochastic approximation method to estimate a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$, to minimize the loss function. Each time we evaluate an RC hidden layer on a limited training dataset under new hyperparameters, we obtain a noisy value of the loss function. Since stochastic approximations can iteratively solve optimization problems using only noisy function evaluations, minimizing the loss function using such methods makes sense. While the random direction method [138, p. 58-59] uses only a pair of loss function evaluations per iteration for the sake of finite-difference gradient approximation, almost sure convergence results need specific conditions on the perturbations, the decreasing learning step-size, and the finite-difference step [139]. These conditions cannot always be met in practice. In particular, letting the finite-difference step converge to zero with an increasing iteration index is not advisable for the sake of numerical stability. Also, decreasing the learning step-size is unsuitable for applications where adaptive tracking of a drifting solution is needed. Therefore, in

the proposed method, we let both steps remain constant while also updating the coordinates of $\boldsymbol{\theta}$ one at a time to reduce the complexity by a factor of $p$ during each update.

Let $\mathbf{e}_c$ be the standard unit vector in the direction of the $c$-th coordinate (i.e. $\mathbf{e}_c$ is the length-$p$ vector containing 1 in its $c$-th coordinate and 0 elsewhere), for $c = 1, \ldots, p$. Starting from an initial guess $\boldsymbol{\theta}_0$, direction vectors $\{\mathbf{d}_i\}$ are uniformly, independently and identically distributed (u.i.i.d.) over $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$, where $i$ is the iteration index. Gradient descent, commonly used for optimization in machine learning [141], is not feasible in our setting since gradient calculation needs complete knowledge of the functional relationship between $\boldsymbol{\theta}$ and $L(.,.)$ that was assumed to be unavailable. Instead, a stochastic approximation of the gradient of the loss function in the random direction $\mathbf{d}_i$, is obtained as

$$\Delta_i = \frac{L(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i, \mathbf{W}_{i-1}^{out}) - L(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i, \mathbf{W}_{i-1}^{out})}{2h}, \tag{3.5}$$

where a central finite-difference approach with constant step $h$ has been used. Note that on top of the error inherent to the finite-difference method, the numerator in Equation 3.5 involves two evaluations of an RC loss function using only a limited number of training data, thus giving rise to noisy evaluations. For the sake of simplicity, the iterative procedure to minimize the loss function updates each component of $\boldsymbol{\theta}$ one at a time according to

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \Delta_i, \tag{3.6}$$

where $\mu$ is a constant learning step size. The proposed parameter optimization procedure is summarized in Algorithm 1.

**Algorithm 1** Parameter optimization procedure

**Require:** $\mu, h, \boldsymbol{\theta}_0$
  Initialize $\mathbf{W}_0^{out}$ by minimizing $L(\boldsymbol{\theta}_0, \mathbf{W}^{out})$
  **for** $i = 1, 2, \ldots,$ **do**
    Pick uniformly at random $c \in \{1, 2, \ldots, p\}$

    Set the direction vector $\mathbf{d}_i = \mathbf{e}_c$
    Compute $\Delta_i$ according to Equation 3.5
    Parameter update: $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \Delta_i$
    Update $\mathbf{W}_i^{out}$ by minimizing $L(\boldsymbol{\theta}_i, \mathbf{W}^{out})$
  **end for**

## 3.4 Performance analysis

In this Section, we give a detailed analysis of the behavior of our proposed optimization algorithm around the optimal value, $\boldsymbol{\theta}_{opt}$ under standard assumptions. We begin by approximating the recursion for the proposed method around the optimal value of $\boldsymbol{\theta}$ in Section 3.4.1, based on a quadratic approximation of the loss function. In Section 3.4.2 , we study the convergence in probability of the transient, provided that the learning step-size is smaller than some upper bound. We do so by taking into account the stochastic nature of the proposed method. Finally, in Section 3.4.3, we derive the steady-state covariance of the parameter vector due to noise terms affecting the stochastic approximation of the gradient.

### 3.4.1 Linearized behavior around the optimum

Let $\boldsymbol{\theta}_{opt}$ be the optimal parameter vector and let $\hat{\mathbf{W}}^{out}(\boldsymbol{\theta}_{opt})$ be the corresponding output layer weights. We only consider the variations of the loss function w.r.t $\boldsymbol{\theta}$ arount $\boldsymbol{\theta}_{opt}$, which leads to studying the recursion

$$\boldsymbol{\theta}_i \approx \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \frac{C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h}, \tag{3.7}$$

where $C(\boldsymbol{\theta}) = L(\boldsymbol{\theta}, \hat{\mathbf{W}}^{out}(\boldsymbol{\theta}_{opt}))$. Assume the Hessian matrix of $C(\boldsymbol{\theta})$ (i.e. the matrix of continuous second-order partial derivatives), $\mathbf{HC}(\boldsymbol{\theta})$ exists. Linearizing Equation 3.7 around $\boldsymbol{\theta}_{opt}$ leads to the recursion

$$\boldsymbol{\theta}_i - \boldsymbol{\theta}_{opt} = \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt}) - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{n}_i, \tag{3.8}$$

where $\mathbf{n}_i$ is a noise vector accounting for potential noisy loss function evaluations in our method. The derivation is postponed to Appendix A.1. Since (3.8) has the structure of a linear time-variant system, it can be seen as the superposition of the stochastic transient response.

$$\begin{cases} \boldsymbol{\zeta}_0 = \boldsymbol{\theta}_0 \\ \boldsymbol{\zeta}_i = \boldsymbol{\theta}_{opt} + \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt}), \end{cases} \tag{3.9}$$

and the noise contribution

$$\begin{cases} \boldsymbol{P}_0 = \mathbf{0}_{p\times 1} \\ \boldsymbol{P}_i = \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) \boldsymbol{P}_{i-1} - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{n}_i. \end{cases} \tag{3.10}$$

### 3.4.2 Transient response behavior

**Lemma 3.4.1** $\lim_{i\to\infty} E[\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}] = \mathbf{0}$ *if and only if* $0 < \mu < \frac{2p}{\rho(\mathbf{HC}(\boldsymbol{\theta}_{opt}))}$.

**Proof 3.4.2** *Regarding the mean of* $\{\boldsymbol{\zeta}_i\}$ *conditional on the previous direction vectors, according to Equation 3.9, we have*

$$\begin{aligned} &E[\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt} | \ldots, \mathbf{d}_{i-2}, \mathbf{d}_{i-1}] = \\ &\left(\mathbf{I}_p - \mu E[\mathbf{d}_i \mathbf{d}_i^T] \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt}) | \ldots, \mathbf{d}_{i-2}, \mathbf{d}_{i-1}], \end{aligned} \tag{3.11}$$

*using the fact that the* $\{\mathbf{d}_i\}$ *are u.i.i.d. Applying the law of total expectation, for all* $i \geq 0$*, we have*

$$E[\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}] = \left(\mathbf{I}_p - \mu E[\mathbf{d}_i \mathbf{d}_i^T] \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E[\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt}]. \tag{3.12}$$

*By induction and using* $E[\mathbf{d}_i \mathbf{d}_i^T] = (1/p)\mathbf{I}_p$*, we have*

$$E[\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}] = \left(\mathbf{I}_p - \frac{\mu}{p} \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)^i (\boldsymbol{\theta}_0 - \boldsymbol{\theta}_{opt}). \tag{3.13}$$

*From [142, p. 119],* $\lim_{i \to \infty} E[\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}] = \mathbf{0}$ *if and only if* $\rho\left((\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) < 1$*. Using the fact that* $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ *is symmetric positive-definite completes the proof.*

**Lemma 3.4.3** *There exists* $\mu^* > 0$*, such that for all* $0 < \mu < \mu^*$*,*
$\lim_{i \to \infty} \operatorname{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T)]\right) = \mathbf{0}$*.*

**Proof 3.4.4** *We first establish the matrix difference equation (see Appendix A.2)*

$$\begin{aligned} &\operatorname{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T)]\right) \\ &= \mathbf{D}_C(\mu) \operatorname{vec}\left(E[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T)]\right). \end{aligned} \tag{3.14}$$

*whose dynamics are governed by the matrix*

$$\begin{aligned} \mathbf{D}_C(\mu) =& \mathbf{I}_{p^2} - \frac{\mu}{p} \left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) \\ &+ \frac{\mu^2}{p} \operatorname{diag}\left(\operatorname{vec}(\mathbf{I}_p)\right) \left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right). \end{aligned} \tag{3.15}$$

*Using [143, Thm. 4.4.5],*

$$\begin{aligned} &\Lambda\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) \\ &= \{2\lambda, \forall \lambda > 0 \in \Lambda(\mathbf{HC}(\boldsymbol{\theta}_{opt}))\}, \end{aligned} \tag{3.16}$$

*consequently,* $\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})$ *is also symmetric positive-definite so that in turn*

$$\begin{aligned} &\Lambda\left(\mathbf{I}_{p^2} - \frac{\mu}{p}\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)\right) \\ &\quad = \left\{1 - \frac{2\mu}{p}\lambda, \forall \lambda > 0 \in \Lambda(\mathbf{HC}(\boldsymbol{\theta}_{opt}))\right\}. \end{aligned} \tag{3.17}$$

*Applying the Bauer-Fike theorem [140, p. 321] to Equation 3.15, there exists* $\mu^* > 0$*, such that for all* $0 < \mu < \mu^*$ $\rho(\mathbf{D}_C(\mu)) < 1$.

*Finally,* $\lim_{i\to\infty} \text{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T)]\right) = \mathbf{0}$ *if and only if* $\rho(\mathbf{D}_C(\mu)) < 1$ *[142, p. 119]. Noting that* $0 < \mu < \mu^*$ *is a sufficient condition for* $\rho(\mathbf{D}_C(\mu)) < 1$ *completes the proof.*

We are now ready to state our main convergence result regarding the transient behavior.

**Theorem 3.4.5** *When* $0 < \mu < \mu^*$*, the transient response random vectors* $\boldsymbol{\zeta}_i \xrightarrow[i\to\infty]{} \boldsymbol{\theta}_{opt}$ *in probability.*

**Proof 3.4.6** *Applying the definition of convergence in probability in the multivariate case [144, p. 530], we must show that*

$$\lim_{i\to\infty} P\left(\sqrt{(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})} \geq \epsilon\right) = 0, \quad \forall \epsilon > 0. \tag{3.18}$$

*Applying Markov's inequality, for any* $\epsilon > 0$*, we have*

$$\begin{aligned} &P\left(\sqrt{(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})} \geq \epsilon\right) \\ &\quad = P\left((\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}) \geq \epsilon^2\right) \\ &\quad \leq \frac{E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})\right]}{\epsilon^2} \\ &\quad \leq \frac{\text{trace}\, E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T\right]}{\epsilon^2}. \end{aligned} \tag{3.19}$$

*Letting* $i \to \infty$ *when* $0 < \mu < \mu^*$*, lemma 3.4.3 leads to the desired result.*

### 3.4.3 Steady-state covariance

We assume the $\{\mathbf{n}_i\}$ to be white zero-mean with covariance matrix $\boldsymbol{\Sigma}$ and independent of the direction vectors $\{\mathbf{d}_i\}$.

Regarding the mean of $\{\boldsymbol{P}_i\}$ conditional on the previous direction vectors, according to Equation 3.10, we have

$$\begin{aligned} E[\boldsymbol{P}_i|\ldots,\mathbf{d}_{i-2},\mathbf{d}_{i-1}] = \\ \left(\mathbf{I}_p - \mu E[\mathbf{d}_i\mathbf{d}_i^T]\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E[\boldsymbol{P}_{i-1}|\ldots,\mathbf{d}_{i-2},\mathbf{d}_{i-1}] \\ - \mu E[\mathbf{d}_i\mathbf{d}_i^T]E[\mathbf{n}_i], \end{aligned} \tag{3.20}$$

using the fact that the $\{\mathbf{d}_i\}$ are u.i.i.d. Applying the law of total expectation and using the fact that $E[\mathbf{n}_i] = \mathbf{0}$ and $E[\mathbf{d}_i\mathbf{d}_i^T] = (1/p)\mathbf{I}_p$, for all $i \geq 0$, we have

$$E[\boldsymbol{P}_i] = \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E[\boldsymbol{P}_{i-1}]. \tag{3.21}$$

Thus $E[\boldsymbol{P}_i] = \mathbf{0}$, for all $i \geq 0$, since $E[\boldsymbol{P}_0] = \mathbf{0}$.

It can also be shown that the covariance of $\{\boldsymbol{P}_i\}$ is obtained via the recursion (see Appendix A.3)

$$\begin{aligned} \operatorname{vec}\left(E[\boldsymbol{P}_i\boldsymbol{P}_i^T]\right) =& \mathbf{D}_C(\mu)\operatorname{vec}\left(E[\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T]\right) \\ &+ \frac{\mu^2}{p}\operatorname{vec}\left(\operatorname{diag}\left(\boldsymbol{\Sigma}\right)\right). \end{aligned} \tag{3.22}$$

When $0 < \mu < \mu^*$, then $\rho(\mathbf{D}_C(\mu)) < 1$ and the steady-state solution is given by [145, p. 27].

$$\begin{aligned} &\lim_{i\to\infty} \operatorname{vec}\left(E[\boldsymbol{P}_i\boldsymbol{P}_i^T]\right) \\ &= \frac{\mu^2}{p}(\mathbf{I}_{p^2} - \mathbf{D}_C(\mu))^{-1}\operatorname{vec}\left(\operatorname{diag}\left(\boldsymbol{\Sigma}\right)\right). \end{aligned} \tag{3.23}$$

## 3.5 Numerical and experimental results

We now assess the performances of the proposed hyperparameter optimization algorithm using both a simulated (see Section 3.5.3) and a hardware implementation (see Section 3.5.4) of RC. As practical classification tasks, we consider the spoken digit recognition application described below a well-established reference to evaluate RC performances. We also add two new tasks in ML literature: the Control Chart and Wafer classification tasks. Section 3.5.2 describes the considered generic setup, which is valid for several optoelectronic and optical hardware implementations of RC.

### 3.5.1 Time Series Classification tasks

#### Spoken digits recognition

As we pointed out in Section 2.7.3, the spoken digit classification is the most commonly used benchmark in the RC literature. And among the spoken digit datasets, the TI46 dataset[102] is the most common benchmark and contains clean and well-formatted utterances of digits $0-9$. However, in our work, we applied the TIDIGITS LDC93S10 [146] dataset that, unlike the TI46, contains utterances in more realistic settings infested with environmental noise, recorder distortions, and considerable speaker variation. The dataset consists of 326 speakers of different ages and gender, uttering the digits from 0 to 9 so that $C = 10$. The individual files are of variable length and are recorded at a sampling frequency of 20kHz. The speakers are English speakers, and the utterances are in English. From this dataset, we choose 1500 sound files with equal distribution between gender and age.

The sound files have an average length of $10k$ samples after trimming the silent ends for the longer files and padding the slightly shorter files with zeros. Although this was done to homogenize the file length, we note that varied input lengths are also feasible in RC. For more robustness in a speech recognition task, it is customary to incorporate preprocessing for feature extraction before the process of word classification or speaker identification. The extracted features incur reduced distortions and format the data in a way that maximizes the contributions of the actual speech data. In our experiments, the raw sound files undergo preprocessing to extract the Mel-frequency Cepstral Coefficients (MFCCs) [147] as input features for the RC processing. The process of extracting the coefficient can be summarized as follows :

- Pre-emphasis by applying a high-pass filter to the waveform to compensate for the attenuation of the high frequencies in human speech production.

- Framing by dividing the entire sound wave into smaller overlapping fixed length segments of duration $20-30ms$ with an overlap of up to $15ms$. In our case, we generated $69$ frames of $25.6ms$ every $7.2ms$ (with an overlap of $18.4ms$).

- Windowing by multiplying each segment to a window function in the spectral domain. The window function tappers each end of a frame producing a smoother spectrum with fewer artifacts. We use Hanning windowing in our experiments.

- Fast Fourier Transform is applied to each window to generate spectral coefficients; in our case, we used a 512-point FFT.

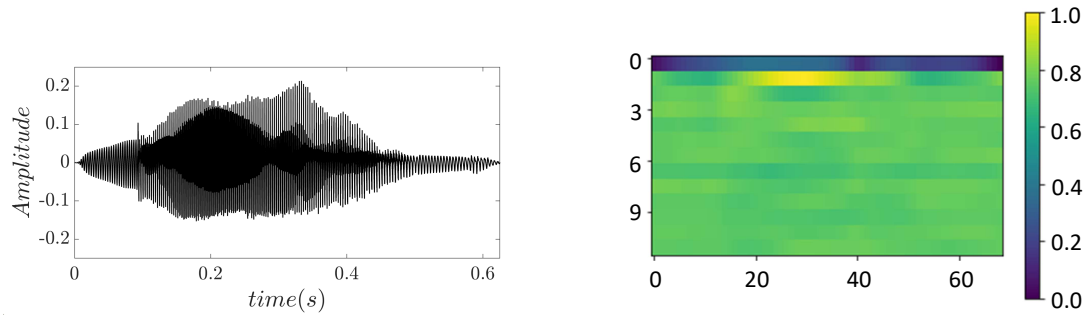
- Filter bank values at Mel Scale permit the reduction of a large number of coefficients to fewer meaningful ones. In our case, we generated 13 coefficients per frame (the typical number is 12 coefficients) by multiplying cross-wise the FFT coefficients with 13 triangular weighing functions centered at $F_{MEL}$ computed as follows :

$$f_{MEL} = 2595 \log_{10} \left(1 + \frac{f_{LIN}}{700}\right) \tag{3.24}$$

  where $f_{LIN}$ corresponds to the cepstral frequencies in the linear domain.

- Environmental compensation is an optional step included in our preprocessing to compensate for the multiplicative distortion of the environment. Assuming invariance, these can be significantly reduced by subtracting the average value of the MFCCs. This removes any time-invariant multiplicative effects that have now become additive due to the logarithmic operation.

With the above preprocessing steps, the input sound whose temporal trace is shown in Figure 3.1a is converted to a $13 \times 69$ matrix of cepstral coefficients displayed in the form of an image in Figure 3.1b. The $1500$ sound files are decomposed into two sets,




(a) (b)

Figure 3.1: (a) A sound profile corresponding to the pronunciation of the digit $5$ with a visual representation of the MFCCs in (b).

the train-validation set, and the test set, composed of $90\%$ (1350 files) and $10\%$ (150 files) of the total number of files, respectively. The train-validation sets are iteratively split into two, the validation set and the training set, at $11.11\%$ (150 files) and $88.89\%$ (1200 files), respectively. Consequently, $T = 1500 \times 69 \times 90\% = 93150$ discrete time instants.

### Control chart pattern recognition

In industrial, financial, and medical settings, the need to classify observed time series patterns produced by equipment or sensors often arises to detect anomalies and guar-

antee continuous operation. The goal is to establish similarities between the system-generated temporal signals and known normal or abnormal signals to narrow down the necessary mitigative measures. However, determining the similarity between time series is not a trivial task. The typical approach is the computation of some distance measure, such as the euclidean distance as a dissimilarity measure. It attributes the patterns characterized by shorter distances between themselves to the same subgroup. For instance, a self-organizing network was proposed [148] and trained to cluster networks in different clusters yielding a classification accuracy of $96.67\%$ employing a distance metric. However, as pointed out by the authors of [149], distance metrics are ineffective as they tend to miss-classify certain series that are visually similar. Their study explored the extraction of features from time series data and carry computation of distances in a different reduced space. The state-of-the-art performance is by a technique named Collective of Transformation-Based Ensembles (COTE) [150] yielded $99.92\%$ of classification accuracy by transforming the representation into a different space before distance computation. Despite the high accuracy, we observed in our study [125] that processing TSC tasks by dissimilarity measures are usually computationally intensive, even for small datasets.

RC is an attractive avenue for the classification of Control Charts, as it avoids distance computations; hence we will explore the application in the context of RC optimization. We work with the dataset of 600 charts representing operation trends of a system whose tendencies belong to one of six classes: Normal, Cyclic, Increasing, Decreasing, Upward shift, and Downward shift [151]. Figure 3.2 shows the more visually apparent examples of observed trends for each class. The $600$ sound files come in two

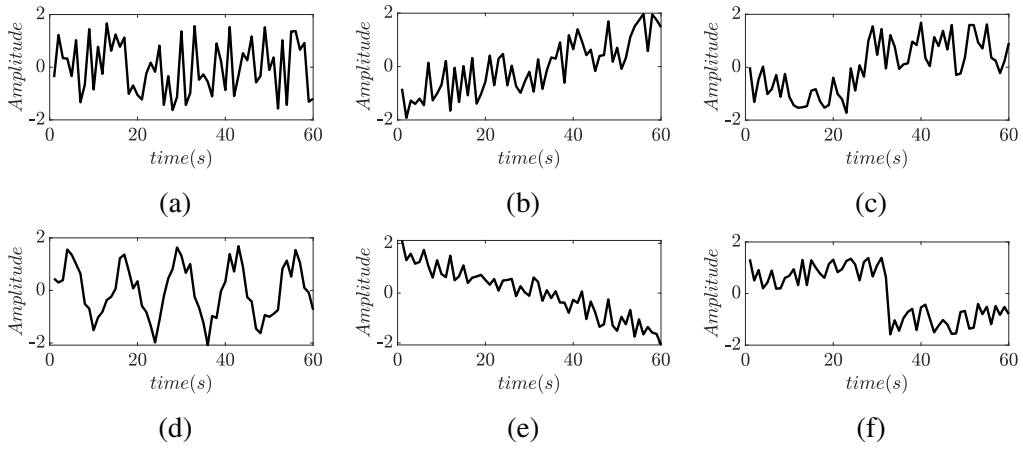



Figure 3.2: Control chart trends : (a) Normal (b) Increasing (c) Upshift (d) Cyclic (e) Decreasing (f) Downshift

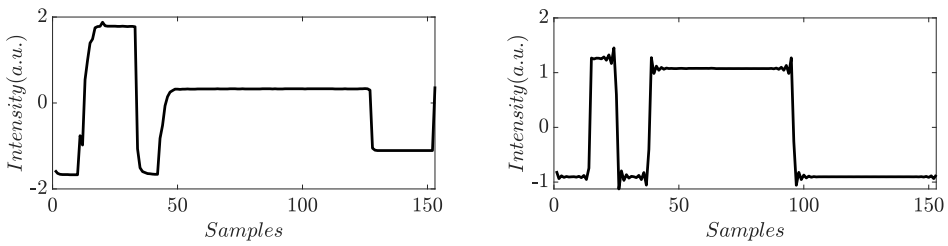
sets, the train and test set composed of $300$ examples each. The training data is iteratively split into two, the train subset and the validation subset, the validation set, and

the training set at 60 ( $20\%$ ) and 240 ($80\%$) files, respectively.

**Semiconductor microelectronics fabrication: Wafer classification task**

The process of manufacturing semiconductor microelectronics is very complex, involving hundreds of steps. Various layers or materials are added to a silicon wafer and carefully removed through etching to form circuit patterns. For instance, during the process, a layer of a very thin photosensitive material called a photoresist is deposited on the surface of the wafer. This layer is selectively exposed to UV light in accordance with the targeted circuitry; being photosensitive, these layers undergo chemical changes. A chemical is introduced on the surface to remove areas unaffected by the UV light leaving the rest intact; the process is called photolithography. Then a reactive plasma is also applied to remove the remaining unprotected regions leaving out the intended circuit pattern [152].

This process is prone to many imperfections that may cause malfunctions, unreliability, and performance issues during post-production. To control fabrication quality, several sensors are included in the whole process. In this work, we use the data from a sensor for the plasma emission at $405$ *nm*, deemed more efficient for faulty detection [152]. With readings from the sensor, the task is to discriminate between the normal and abnormal wafers referring to non-faulty and faulty wafers, respectively. Unfortunately, the dataset has a significant class imbalance, with the abnormal class accounting for only $10.7\%$ of the train data and $12.1\%$ of the test data, respectively. We employed the SMOTE technique to restrain the overfitting of the majority class, avoid underfitting the minority class, and improve overall generalization [153]. With this tool, we undersample the majority class slightly, and synthetic examples are generated to add more samples for the minority class. Figure 3.3a and 3.3b show the readings for normal and abnormal wafers, respectively. The train set contains 1000 files that we split into 900 ($90\%$) for training and 100 ($10\%$) for validation. The performances are reported on the test set, which contains 6164 files.




(a) Normal wafer (b) abnormal Wafer

Figure 3.3: Plots for the plasma emission sensor readings at 405nm.

### Classification and costs

Let us consider the spoken digit recognition task as our running example. Classification of the output signal is implemented at the output layer. The ouput layer training in the Equation 2.7 relies on the fact that if $c \in \{0, 1, \ldots, C-1\}$ is the class corresponding to the $n$-th training feature vector $\mathbf{u}_{train}(n)$, the corresponding target RC readout $\mathbf{y}_{train}(n)$ will be $+1$ in its $c$-th coordinate and $0$ elsewhere. In our experiments, to classify a data entry, the node states vectors $\mathbf{x}(t)$ are column-wise concatenated to form an $N \times 69$ matrix $\mathbf{X}$ shown in Figure 3.4a ($69$ is the number of time steps in a single digit for speech, changes for other tasks). The obtained matrix is multiplied as shown in Figure 3.4 to get the predicted output matrix (See Figure 3.4b). The predicted output is expected to be close to the expected output matrix (See Figure 3.4c) To decide the

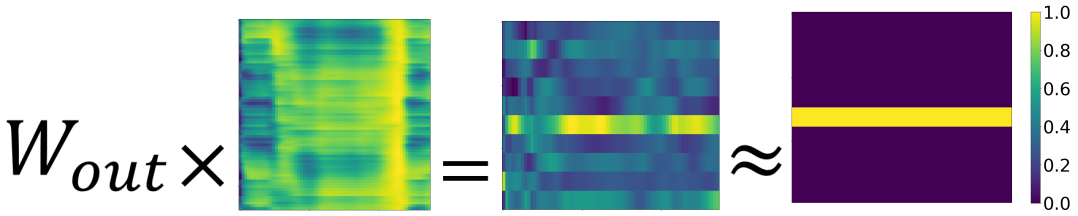



Figure 3.4: Graphical representation of the read-out operations where the readout matrix (size $10 \times 400$) is multiplied by the $400 \times 69$ node states matrix (a) to obtain the $10 \times 69$ predicted matrix (b), that is, after successful training, is an approximation of the expected $10 \times 69$ target matrix (c)

prediction class, we sum the rows of the output matrix and identify the row $c$ that has the maximum value. This procedure is illustrated in Figure 3.5 with the predicted matrix placed alongside a bar plot of the row-wise sum.

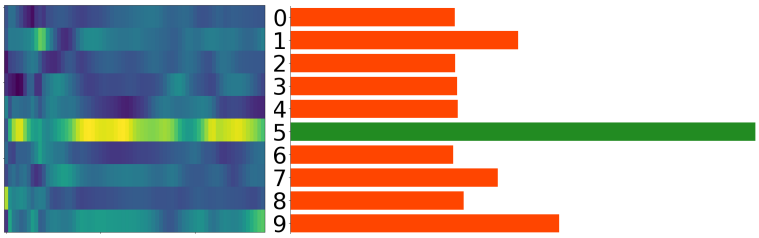



Figure 3.5: Graphical representation of the class decision process for digit 5.

In addition to the loss function in Equation 3.2, we use the Classification Error Rate (CER) to gauge the model performances. CER measures the extent of misclassification in percentage $(\%)$ and computed as:

$$CER\,(\%) = \frac{\text{Number of wrongly classified entries}}{\text{Cardinality of the test set}} \times 100 \tag{3.25}$$

Note that for the particular case of the spoken digit task, the CER is also commonly referred to as the Word Error Rate (WER), since the inputs are classified as words.

### 3.5.2 Proposed Setup

We consider the delayed feedback loop optoelectronic and optical RC implementations from [96, 98, 100, 101], which can be modeled as the discrete-time RC model (Equation 3.1) after digital-to-analog conversion (DAC) and analog-to-digital conversion (ADC) at the input and output of the circuit, respectively. Since the operating principle of such physical implementations relies on serializing the internal (hidden) nodes using time-multiplexing, the connectivity matrix $\mathbf{W}$ has non-zero coefficients only on the first lower diagonal. In our numerical simulations, these coefficient are sampled from discrete coefficients $h_i$ of a first order low-pass filter $h(t) = e^{-t/T_R}$ of a response time $T_R \approx 240ns$ where $h_i = h(i\theta)$, $\theta = 52.18ns$ and $i$ discrete time instants. The connectivity matrix $W$, therefore, can be written using the $h_i$ coefficients as follows :

$$\mathbf{W} = \begin{bmatrix} h_0 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ h_1 & h_0 & 0 & 0 & 0 & . & . & . & 0 \\ h_2 & h_1 & h_0 & 0 & 0 & . & . & . & 0 \\ h_3 & h_2 & h_1 & h_0 & 0 & . & . & . & 0 \\ h_4 & h_3 & h_2 & h_1 & h_0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ & . & . & . & h_4 & h_3 & h_2 & h_1 & h_0 \end{bmatrix} \tag{3.26}$$

In hardware implementations, the exact values of these coefficients depend on the characteristic time scale of the system, which is not precisely known. Although characterization of the filter's coefficients could be attempted, this would not only need a complex experimental setup on its own but also account for drift in time due to thermal instability which seems very difficult in practice.

The elements of the input matrix $\mathbf{W}^{in}$ at positions $(l, c)$ may be generated from any distribution. In this work, they are sampled independently from the probability distribution defined as:

$$P\left(\mathbf{W}^{in}(l,c) = w\right) = \begin{cases} \frac{1}{5}, & \text{if } w = -1, 1. \\ \frac{3}{5}, & \text{if } w = 0. \end{cases} \tag{3.27}$$

Moreover, $f_{NL}$ in Equation 3.1 stands for the squared transfer function of a Mach-Zehnder Modulator (MZM) (where squaring is due to the action of a photodiode), i.e

$$f_{NL}(.) = \sin^2(. + \phi). \tag{3.28}$$

The value of the parameter $\phi$ controls the nonlinearity in play by varying from highly linear operation regimes (e.g., around point C in Figure 3.6) to highly nonlinear regimes (e.g., point A or point E in Figure 3.6). Hence, by varying the value of $\phi$, the nonlinearity can be optimized to meet the needs of the task at hand. We select $\phi = 3$ rad in our simulations, which is located near an extremum of $f_{NL}$ but slightly leaning towards the negative slope, which gave the best results for the task of spoken digit recognition in [96]. Finally, hyperparameter vector $\boldsymbol{\theta} = [\alpha, \beta]^T$, where $\alpha$ and $\beta$ are as defined in

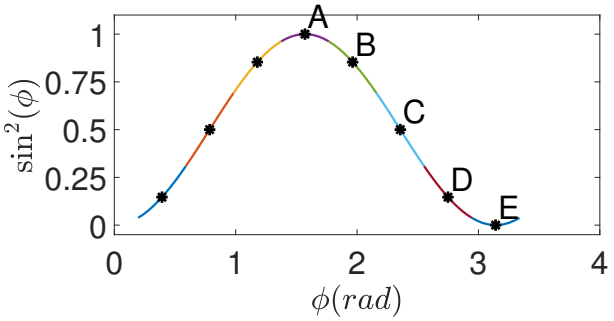



Figure 3.6: Squared Mach-Zehnder transfer function (Operating point A, E: highly nonlinear regimes. Operating point C: essentially linear regime).

Equation 3.1, needs to be fine-tuned since it plays a vital role in controlling the dynamical regime in the reservoir by ensuring the so-called echo state property [46].

### 3.5.3 Numerical Simulations

We aim to optimize the vector $\theta = [\alpha, \beta]^T$, but we have two global parameters to set, namely the size of the reservoir, $N$, and the regularization parameter $\lambda$. Regarding the choice of $N$, a rule of thumb in [52] states that $N$ should be approximately one-tenth of the training sequence's length, and other RC properties can be optimized for smaller reservoirs and transferred to larger ones. Therefore, in this work, we use $N = 400$, the size also used in [96]. Regarding $\lambda$, we fix it at $10^{-8}$ after running trials manually.

We consider the optimization of the computer simulations of the RC system described in Section 3.2. Grid search and simulated annealing (SAN) are the benchmark methods against which we assess the performances of our optimization method. Grid-search consists in retaining the hyperparameter value corresponding to the lowest loss function value (Equation 3.2) after an exhaustive search over the discretized variable

$\boldsymbol{\theta}$. Table 3.1 lists the parameters for the grid-search algorithm showing the resolution necessary to find a near-optimal operating point. On the other hand, SAN is a stochastic optimization algorithm that emulates materials' slow and controlled cooling until the lowest energy state is reached. Our implementation of SAN follows [144, 154–156], with parameters listed in Table 3.2 for the Spoken digit recognition task. Note that to counteract the effect of noisy (instead of deterministic) evaluations of the loss function, we use the acceptance threshold $\tau$ introduced in [144, p. 115], in the sense that we only update when the new loss function improves the previous loss function by not less than $\tau$. For the other two tasks, only the acceptance threshold needed to be adjusted, and the rest of the parameters remained the same.

The loss function is evaluated at each iteration using a random data split into a train-validation couple to avoid being stuck in local minima for the proposed method. We list the parameters of the method in Table 3.3. We set the finite difference step $h = 5 \times 10^{-3}$ manually, whereas the choice of learning step-size $\mu$ is through the theoretical analysis described later in this section. We follow a similar procedure for the other two tasks and list their parameters in Table 3.3 as well. The upper bound $\mu^*$, corresponds to the maximum value guaranteeing $\rho(\mathbf{D}_C(\mu)) < 1$ and convergence when $0 < \mu < \mu^*$, for the Hessian $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ computed near the optimum value of $\boldsymbol{\theta}$. The details of the estimation of $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ near optimal are also detailed in this section. The plot of $\rho(\mathbf{D}_C(\mu))$ on Figure 3.11 (3.13 for the other two tasks) reveals the upper bound $\mu^*$. Therefore, in our experiments, we can safely set $\mu$ to conservative values equal to $5 \times 10^{-6}$, $1 \times 10^{-6}$, and $1.5 \times 10^{-5}$ for the Spoken Digit, Control chart, and the Wafer classification tasks respectively.

The complexity of the three aforementioned methods is determined by the number of repeated steps $S$ (see Table 3.4). Note that for the grid-search approach (resp. the proposed and SAN approaches), $S$ represents the number of grid points (resp. number of iterations and the number of temperature changes times the number of epochs until convergence).

| **Parameter** | **Range** | **Discretization step** |
|---|---|---|
| Input scaling $\alpha$ | $(10^{-2}, 1)$ | $6.6 \times 10^{-3}$ |
| Feedback scaling $\beta$ | $(0, 1)$ | $10^{-2}$ |

Table 3.1: Exhaustive search parameters.

We started with the grid-search algorithm that evaluates the cost function for every possible combination of $\alpha$ and $\beta$ in the ranges and resolutions provided in Table 3.1. We maintained the same discretization steps for the three tasks in our study. At the end of

| **Parameter** | **Value** |
|---|---|
| Initial temperature ($T_0$) | *1* |
| Cooling scheme | *Geometric* |
| Cooling rate | *0.9* |
| Epoch | *100* |
| Threshold ($\tau$) | $1.8e-2$ |
| Random perturbation of the solution | *Lorentzian distribution* |

Table 3.2: Parameters of SAN.

the experiments, we selected the optimal combinations. We reported their performance on test sets in the second column of Tables 3.5, 3.6 and 3.7 for the Spoken Digit, Control chart, and the Wafer classification tasks, respectively.

For the sake of fair comparison, the proposed method and SAN were both initialized at ($\alpha = 0.25$, $\beta = 0.25$) for all the tasks of interest. Sample trajectories of optimization vs. the number of steps for SAN and the proposed method are shown in Figures 3.7, 3.8 and 3.9 for the Spoken Digit, Control chart, and the Wafer classification tasks, respectively. We place the convergence plots for SAN and our SGD method side by side for an easier visual comparison of the evolution for $\alpha$, $\beta$, and the cost function (Equation 3.2). Note that the seemingly erratic behavior of SAN before convergence comes from the non-zero probability of acceptance of hyperparameter values that increase the loss function, especially at high temperatures. In practice, a stopping criterion (for instance, detecting marginal loss function changes) would be implemented to detect the convergence of SAN and the proposed method. Notice, SAN stops exploring at low temperatures maintaining seemingly constant values whereas our method seems erratic since it continues updating with noisy gradients around optimum.

The performance measures of the three methods after RC optimization for the spoken digit recognition task are shown in Table 3.5. Note that the proposed method and SAN have similar WERs after convergence for this application. We could also improve the grid-search WER after optimization, but at the expense of higher complexity, by using a finer grid of values. Moreover, we observe that the computational complexity, being essentially proportional to the number of steps $S$ defined previously, according to Table 3.5 the proposed algorithm outperforms the two competing methods at least by a factor of $13.6$ in this respect. We also compare the variance of the values taken by $\alpha$ and $\beta$ at convergence, as shown in Table 3.9. We observe that the parameters'

**Spoken Digit Recognition**

| Parameter | Value |
|---|---|
| Learning step-size ($\mu$) | $5 \times 10^{-6}$ |
| Finite-difference step ($h$) | $5 \times 10^{-3}$ |

**Control Chart Classification**

| Parameter | Value |
|---|---|
| Learning step-size ($\mu$) | $1 \times 10^{-6}$ |
| Finite-difference step ($h$) | $5 \times 10^{-2}$ |

**Wafer Classification**

| Parameter | Value |
|---|---|
| Learning step-size ($\mu$) | $1 \times 10^{-5}$ |
| Finite-difference step ($h$) | $5 \times 10^{-2}$ |

Table 3.3: Parameters of the proposed hyperparameter optimization method of digital RC simulation. Top, Middle, and Bottom for the Spoken digit, Control Chart, and Wafer classification tasks, respectively.

empirical and theoretical variances (derived from Equation 3.23) have the same order of magnitude at convergence. In order to do so, $\boldsymbol{\Sigma}$, the covariance of the noise affecting the gradient approximation in all directions around $\boldsymbol{\theta}_{opt}$, is reliably estimated by computing the sample covariance of

$$\begin{bmatrix} \frac{C(\boldsymbol{\theta}_{opt}+h\mathbf{e}_1)-C(\boldsymbol{\theta}_{opt}-h\mathbf{e}_1)}{2h} \\ \vdots \\ \frac{C(\boldsymbol{\theta}_{opt}+h\mathbf{e}_p)-C(\boldsymbol{\theta}_{opt}-h\mathbf{e}_p)}{2h} \end{bmatrix}. \tag{3.29}$$

over $1000$ Monte Carlo trials.

| | ***Complexity*** |
|---|---|
| **Grid Search** | $\mathcal{O}(S(2CNT + N^3))$ |
| **SAN** | $\mathcal{O}(S(2CNT + N^3))$ |
| ***Proposed*** | $\mathcal{O}(S(3CNT + N^3))$ |

Table 3.4: Complexity order. Note that the definition of the number of steps $S$ depends on the optimization algorithm.

| | ***Grid Search*** | ***SAN*** | ***SGD*** |
|---|---|---|---|
| $\alpha$ | 0.06 | 0.05 | 0.05 |
| $\beta$ | 0.13 | 0.13 | 0.13 |
| **WER** (%) | 2.10 | 1.88 | 1.80 |
| **Number of steps** ($S$) | 15000 | 3400 | 250 |

Table 3.5: Optimization results show the hyperparameters, WER, and number of steps $S$ for the spoken digits task.

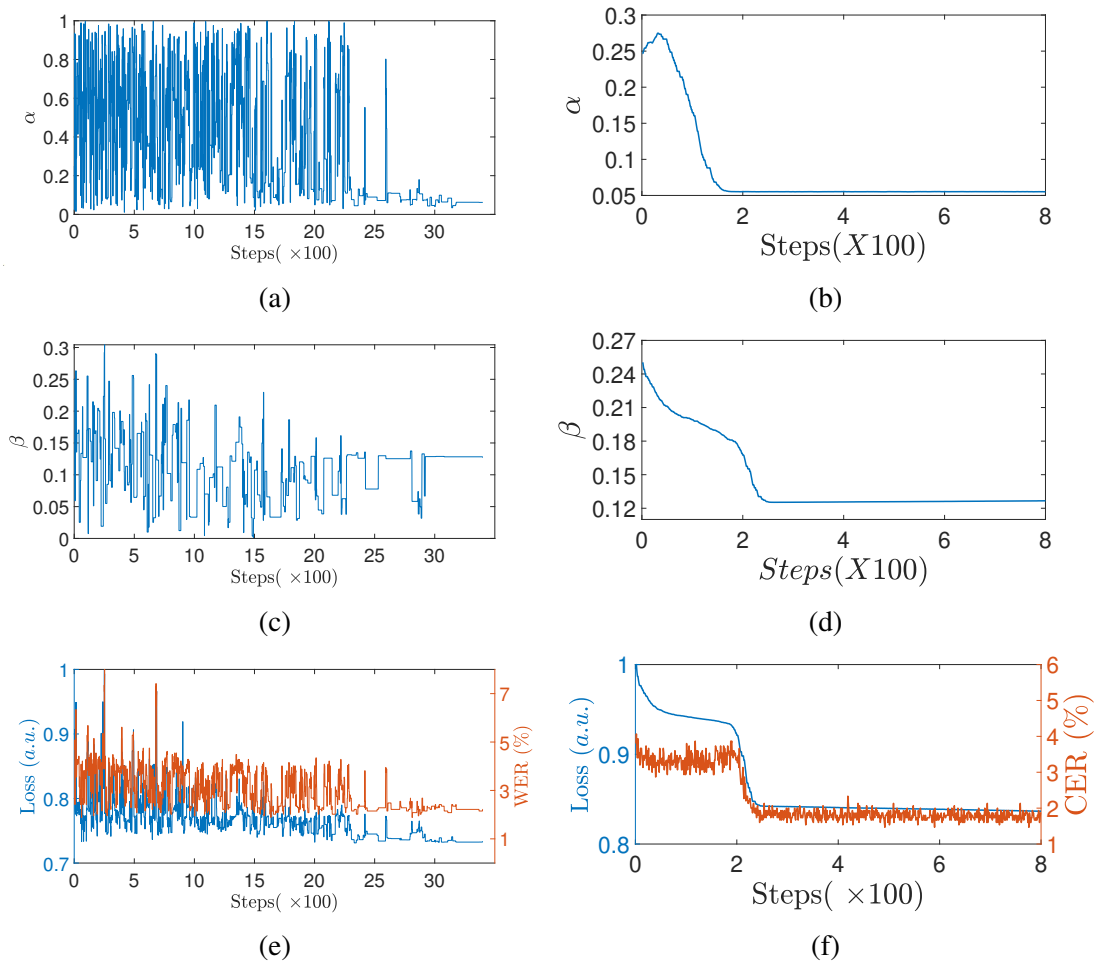



Figure 3.7: SAN (left column) and SGD (right column) plots for simulated RC. (a) and (b) show the Input scaling ($\alpha$) evolution, (c) and (d) show the Feedback scaling ($\beta$) evolution, and (e) and (f) show the Loss and WER evolution, respectively for the spoken digit recognition task.

| | ***Grid Search*** | ***SAN*** | ***SGD*** |
|---|---|---|---|
| $\alpha$ | 0.19 | 0.18 | 0.18 |
| $\beta$ | 0.14 | 0.14 | 0.14 |
| **CER** (%) | 3.33 | 1.97 | 2.00 |
| **Number of steps** ($S$) | 15000 | 3550 | 1300 |

Table 3.6: Optimization results show the hyperparameters, WER, and number of steps $S$ for the Control Charts task.

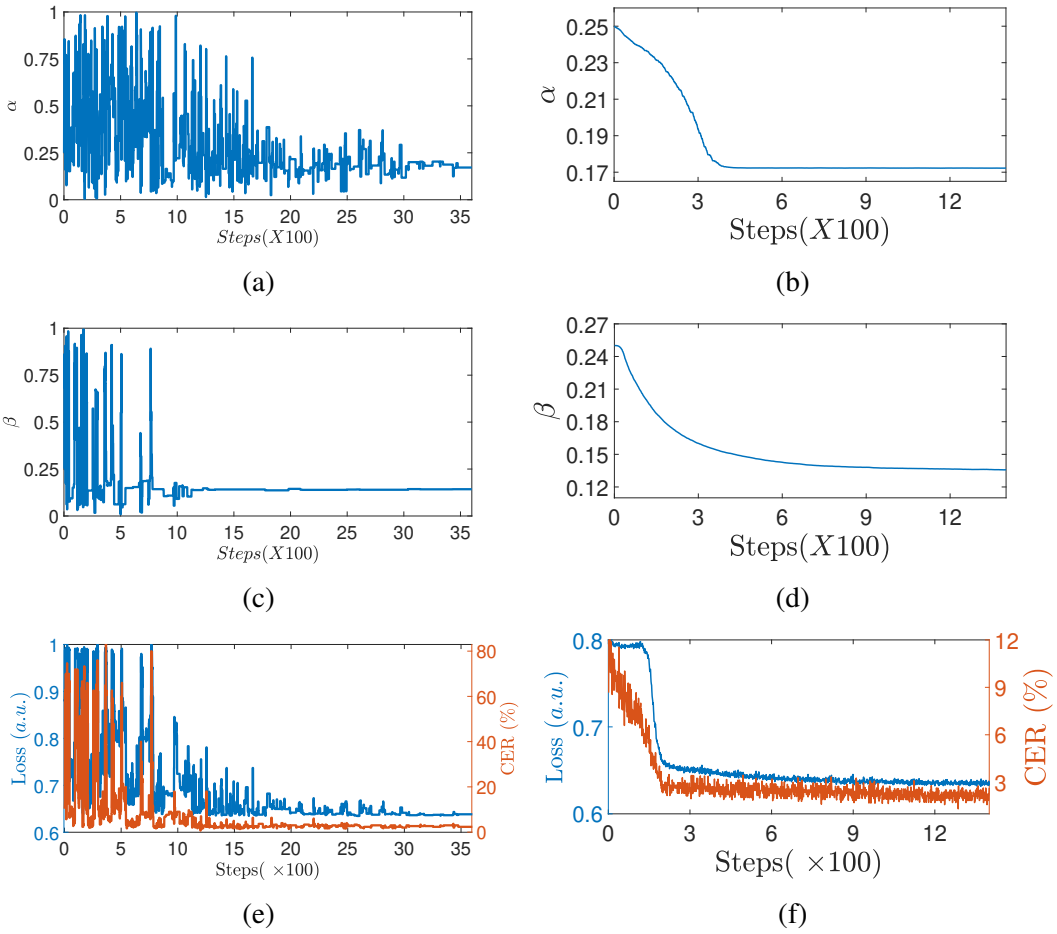



Figure 3.8: SAN (left column) and SGD (right column) plots for simulated RC. (a) and (b) show the Input scaling ($\alpha$) evolution, (c) and (d) show the Feedback scaling ($\beta$) evolution, and (e) and (f) show the Loss and CER evolution, respectively for the Control Chart task.

| | ***Grid Search*** | ***SAN*** | ***SGD*** |
|---|---|---|---|
| $\alpha$ | 0.840 | 0.838 | 0.836 |
| $\beta$ | 0.383 | 0.388 | 0.392 |
| **CER** (%) | 8.73 | 8.71 | 8.67 |
| **Number of steps** ($S$) | 15000 | 4200 | 1200 |

Table 3.7: Optimization results show the hyperparameters, WER, and the number of steps $S$ for the Wafer classification task.

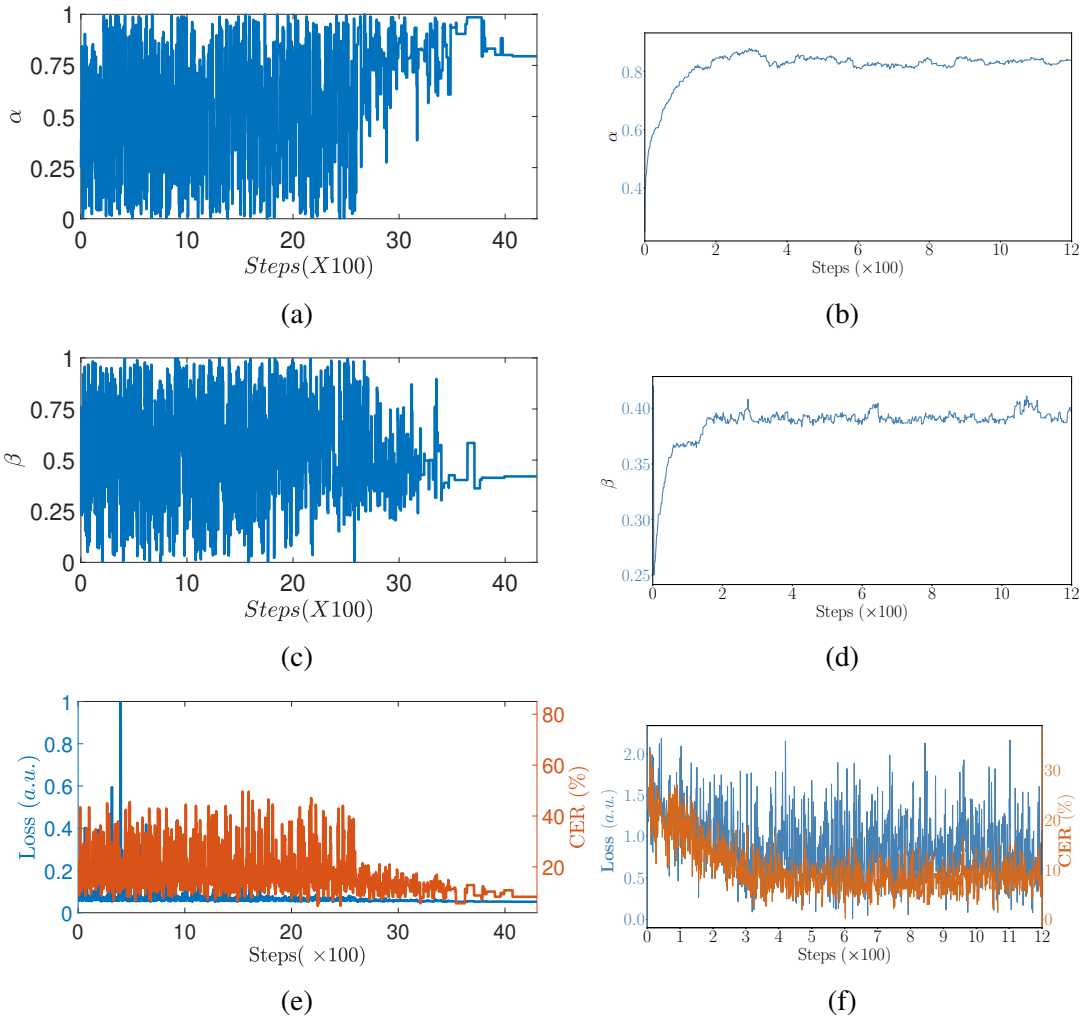



Figure 3.9: SAN (left column) and SGD (right column) plots for simulated RC. (a) and (b) show the Input scaling ($\alpha$) evolution, (c) and (d) show the Feedback scaling ($\beta$) evolution, and (e) and (d) show the Loss and CER evolution, respectively for the Wafer classification task.

A similar analysis is applied to the other tasks under consideration: the Control Chart and Wafer Classification tasks. The parameters $\alpha$ and $\beta$ are initialized at the same values as for the speech task. We examine the performance measures of the three optimization methods for the Control Chart and Wafer Classification tasks which are given in Tables 3.6 and 3.7 respectively. The results are:

- For the Control Chart task, the reported error rates belong to the grid search, proposed method, and SAN. Notice a similar observation to that of the Spoken Digit task. Finer resolution may improve the results of Grid-search at the price of increasing the computational complexity. To attain convergence of up to $CER \approx 2\%$, SAN requires $3550$ steps, whereas our method requires $1300$. This corresponds to a factor of $2.73$ in complexity reduction by our method. The variance of $\alpha$ and $\beta$ at convergence are shown in Table 3.9. We observe an agreement between the empirical and the theoretical variances to the same order of magnitude at convergence.

- For the Wafer Classification, the reported error rates belong to the grid search, proposed method, and SAN. To attain convergence of up to $CER \approx 2\%$, SAN required $4200$ steps, whereas our method requires $1200$. This corresponds to a factor of $3.5$ in complexity reduction by our method. The variance of $\alpha$ and $\beta$ at convergence are also shown in Table 3.9 for this task. Notice, again, the agreement between the empirical and the theoretical variances to the same order of magnitude at convergence.

**Empirical transient response behavior for each task**

Following the theoretical analysis detailed in Section 3.4, we study the behavior around the optimum found by our method for each task. The optima for each task are:

- Spoken Digit Recognition : $\boldsymbol{\theta}_{opt} = [\alpha, \beta] = [0.0521, 0.131]$,
- Control Chart Classification : $\boldsymbol{\theta}_{opt} = [\alpha, \beta] = [0.18, 0.139]$,
- Wafer Classification : $\boldsymbol{\theta}_{opt} = [\alpha, \beta] = [0.836, 0.392]$.

We compute costs around these optimal points ($\boldsymbol{\theta}_{opt}$) and estimate the quadratic approximation of the loss function. We also study the convergence in probability of the transient and derive the steady-state covariance of the parameter vector due to noise affecting the computation of costs. With the computed costs around $\boldsymbol{\theta}_{opt}$, we can deduce the elements of $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ of the Equation 3.8. The cost function around the optimum

has a bowl shape and can be approximated as a quadratic function of $\alpha$ and $\beta$ of the form of the Equation 3.30.

$$C(\alpha, \beta) = a_0 + a_1\alpha + a_2\beta + a_3\alpha\beta + a_4\beta^2 + a_5\alpha^2 \tag{3.30}$$

We approximate the coefficients $a_0, ..., a_5$ by regression for the speech recognition task. Then by computing the second-order partial derivatives of $C$ we obtain elements of $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ as follows :

$$HC(\boldsymbol{\theta}_{opt}) = \begin{bmatrix} \frac{\partial C}{\partial \alpha^2} & \frac{\partial C}{\partial \alpha \partial \beta} \\ \frac{\partial C}{\partial \beta \partial \alpha} & \frac{\partial C}{\partial \beta^2} \end{bmatrix} = \begin{bmatrix} 4.97 \times 10^6 & 3.08 \times 10^6 \\ 3.08 \times 10^6 & 5.71 \times 10^7 \end{bmatrix} \tag{3.31}$$

Utilizing the estimated $\mathbf{HC}(\boldsymbol{\theta}_{opt})$, We plot the actual and estimated surface plots in Figure 3.10. The *actual* surface here is the average of 20 noisy surface plots averaged around the optimum. The surface plots show that the estimated $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ allows us to model the behavior of the cost function near convergence closely. Satisfied with the fit, we estimate the upper bound $\mu^*$ that guarantees convergence of our approach to the optimum. The results of this study are plotted in Figure 3.11 showing the eligible values of the learning step-size $\mu$ and the upper bound $\mu^*$ that guarantees convergence.

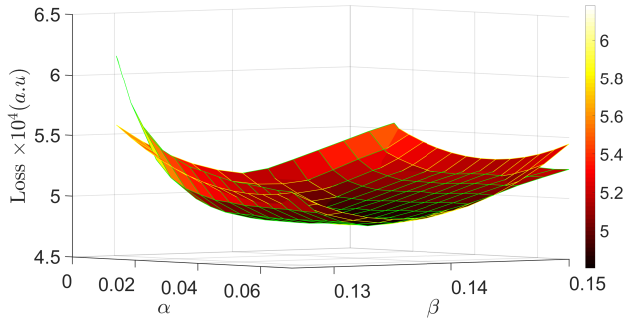



Figure 3.10: Actual and fitted bowl-shaped surface plot near the optimal $\boldsymbol{\theta}_{opt}$ for the spoken digit task

The procedure elaborated above is repeated for the other two tasks: The Control Chart and Wafer classification generating, and the surface plots are shown in Figure 3.12a and Figure 3.12b respectively. The obtained HC matrices are shown in Table 3.8 and the estimation for the maximum learning step-size $\mu^*$ is deduced from Figure 3.13a and Figure 3.13b. In our experiments, We set the $\mu$s to their conservative values of $1 \times 10^{-6}$ and $1.5 \times 10^{-5}$ respectively.

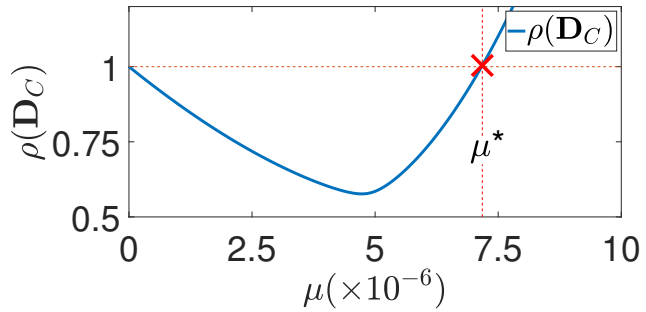
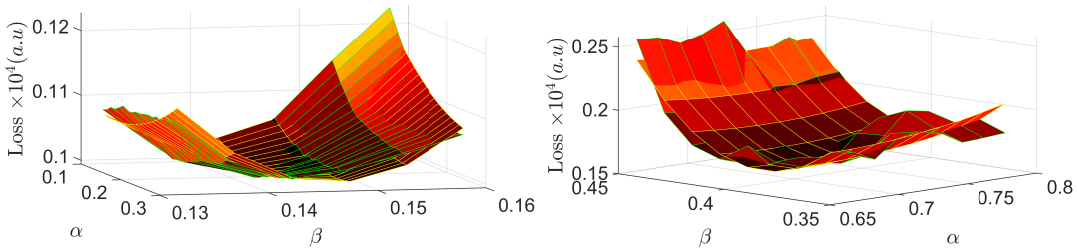



Figure 3.11: Spectral radius of $\mathbf{D}_C(\mu)$ plotted versus $\mu$ for the numerically simulated RC applied to the spoken digit recognition task.




(a) (b)

Figure 3.12: Actual and estimated surface plots for (a) Control charts and (b) Wafer classification tasks around their respective optimal parameters ($\boldsymbol{\theta}_{opt}$).

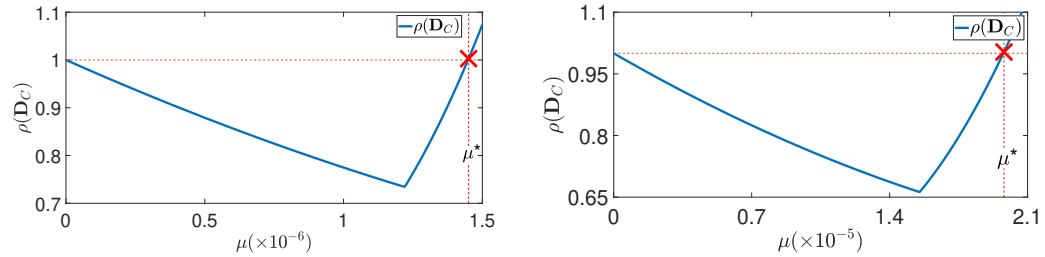
| | **Control charts** | **Wafers** |
|---|---|---|
| $\mathbf{HC}(\theta_{\mathbf{opt}})$ | $\begin{bmatrix} 2.61 \times 10^3 & -5.2 \times 10^4 \\ -5.2 \times 10^4 & 1.38 \times 10^6 \end{bmatrix}$ | $\begin{bmatrix} 2.79 \times 10^4 & 2.81 \times 10^4 \\ 2.81 \times 10^4 & 1.01 \times 10^5 \end{bmatrix}$ |

Table 3.8: The Hessian matrices for the control chart and the Wafer applications describe behavior around the optimal value.

### Steady-state mean and covariance

In this section, we study the steady-state behavior at convergence. First, we simulate the recursion formula developed for the expectation of $\boldsymbol{E}[\boldsymbol{\theta} - \boldsymbol{\theta}_{opt}]$ (Equation 3.21) for 1000 trajectories. Figure 3.14a and 3.14b show the plots of $\alpha - \alpha_{opt}$ and $\beta - \beta_{opt}$ respectively, showing the expected empirical values converge towards 0, save the noise deduced empirically from the covariance at $\boldsymbol{\theta}$, as predicted in the theoretical analysis for the Spoken Digit task. Similar plots are simulated for the Control Chart (Figure 3.14c and 3.14d) and Wafer classification (Figure 3.14e and 3.14f) tasks.

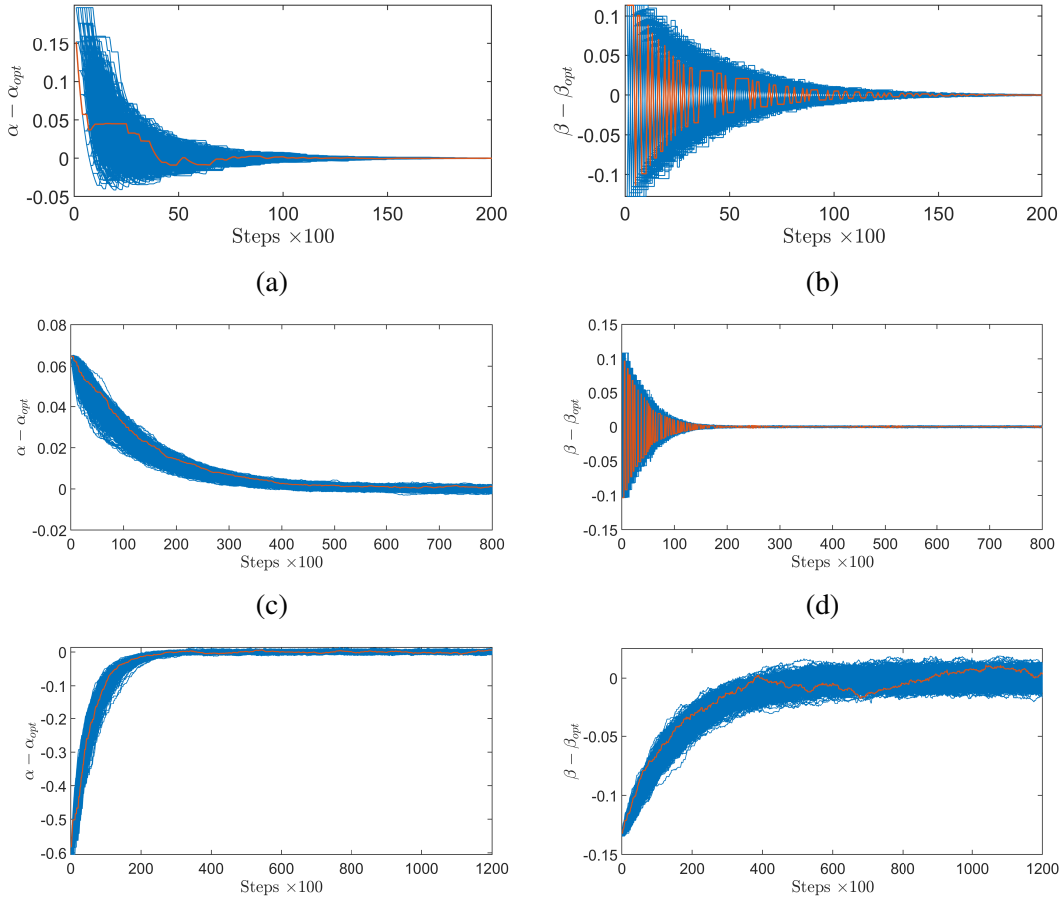
We also compute the average covariance of the perturbation on $\alpha$ and $\beta$ for the 1000 trajectories as the algorithm approaches convergence following the theoretical analysis




(a) (b)

The s

Figure 3.13: Spectral radius of $\mathbf{D}_C(\mu)$ plotted versus $\mu$ for the numerically simulated RC for the Control Chart task (a) and the Wafer task (b).




(a) (b)

(c) (d)

(e) (f)

Figure 3.14: The evolution of $\boldsymbol{\theta} - \boldsymbol{\theta}_{opt}$ for 1000 trajectories is shown in blue for the proposed hyperparameter optimization. For clarity, one trajectory is plotted in orange to contrast it with the 999 others. The first, second, and third-row show $\alpha - \alpha_{opt}$ (left) and $\beta - \beta_{opt}$ (right) for the Spoken Digit, Control charts, and Wafer Classification tasks respectively. As the theoretical and empirical evolution of $\boldsymbol{\theta}$ approaches $\boldsymbol{\theta}_{opt}$ these plots approach $0$.

**Spoken Digit Classification**

| | var($\alpha$) | var($\beta$) |
|---|---|---|
| Theoretical | $7.94 \times 10^{-9}$ | $14.9 \times 10^{-9}$ |
| Empirical | $8.471 \times 10^{-9}$ | $16.4 \times 10^{-9}$ |

**Control Chart Classification**

| | var($\alpha$) | var($\beta$) |
|---|---|---|
| Theoretical | $5.8 \times 10^{-8}$ | $2.1 \times 10^{-8}$ |
| Empirical | $6.1 \times 10^{-8}$ | $1.8 \times 10^{-8}$ |

**Wafer Classification**

| | var($\alpha$) | var($\beta$) |
|---|---|---|
| Theoretical | $8 \times 10^{-6}$ | $25.7 \times 10^{-6}$ |
| Empirical | $8 \times 10^{-6}$ | $28.6 \times 10^{-6}$ |

Table 3.9: Theoretical and empirical variances of $\alpha$ and $\beta$ at convergence for the Spoken Digit (Top), Control Charts (Middle), and Wafer classification (Bottom) tasks.

that led to Equation 3.23. Figure 3.15 tracks the evolution as the perturbation attains a steady state whose purely theoretical (Equation 3.23) and empirical covariance values are given in Table 3.9. We plot the Frobenius norm of the perturbation covariance matrix at each iteration. The empirical and theoretical analyses agree closely and show a small perturbation of the parameters at convergence, hinting at stability. The more rugged surface of the Wafer task in Figure 3.12b manifests itself in the noisy nature of the $\beta - \beta_{opt}$ in Figure 3.14f.

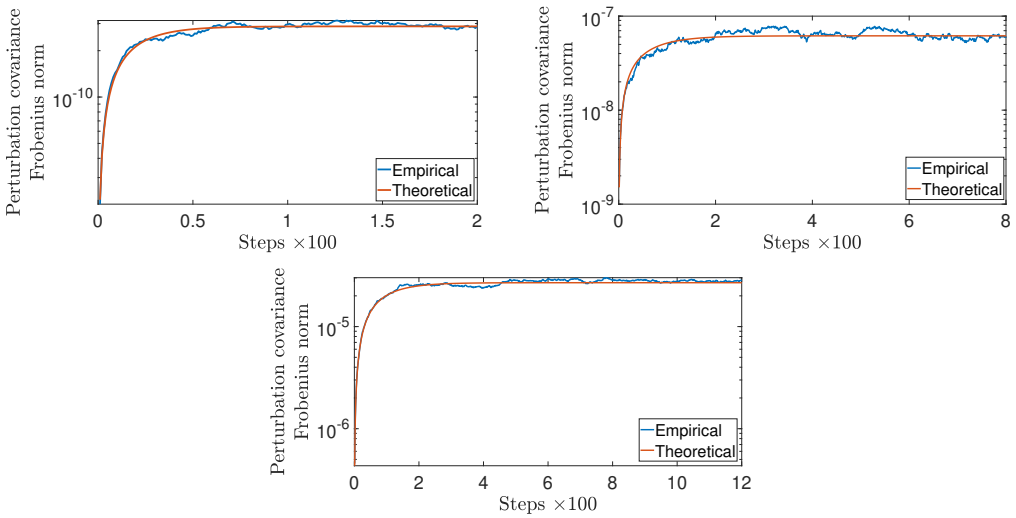



Figure 3.15: The evolution of the perturbation covariance of the proposed hyperparameter optimization method for the Spoken Digit (Top Left), Control charts (Top Right), and Wafer Classification (Bottom) tasks, respectively.

### 3.5.4 Hardware RC optimization

The RC system described in Section 3.2 now uses a hardware implementation that also needs hyperparameter optimization. Like most physical RCs, many of the parameters of the setup are unknown, for example, the filter coefficients and many other details of the electronic circuit components. The optimizable parameters crucial to RC performance are $\phi$ (set as in our previous simulations) and the nonlinearity gain $\kappa$ (set by adjusting the input power of the laser). Referring to Equation 2.17 and Figure 2.6, we manually set $\phi$ and $\kappa$ at the optimal values obtained in [96]. We use the proposed method to optimize the amplitude of the signal $u_I(t)$ generated with an Arbitrary Waveform Generator (AWG) [157] by varying $\alpha$, which is the gain of the AWG. An external computer running our algorithm updates $\alpha_i$ at iteration $i$ and communicates the corresponding value to the AWG through an interface to serve as the new amplitude range of the MZM input voltage. This way, the proposed algorithm updates $\alpha$ without human intervention.

The parameters of the proposed algorithm for the hardware implementation are given in Table 3.10. We set the learning step-size in a manner similar to the digital RC case described in Section 3.5.3. The upper bound for learning step-size is deduced from the plots in Figure 3.16 for each task. The finite difference step size is manually set to $h = 2 \times 10^{-2}$ for the Spoken Digit and Control Charts tasks and $h = 5 \times 10^{-2}$ for the Wafer Task. for comparison, the empirical and theoretical variances for $\alpha$ at convergence for each task are given in Table 3.12 similar to the simulated RC. These curves are different from those of simulated RC because the numerical simulation isn't an exact map of the experimental setup hence the system's function is different. Also, $\alpha$ and $\beta$ are optimized in simulations whereas we only optimize $\alpha$ for hardware. Both of these reasons would change the shape of the spectral radius curves.

The results of this experiment are plotted in Figure 3.17 showing the trajectories of $\alpha$ and that of the loss function (and WER or CER) as they converge to the optimal values for all the studied task, respectively. The value of alpha scales the amplitude of the output signal from the AWG. We initialized the amplitude at $1.5$ Volts (the maximum output for the AWG). We allowed the proposed optimization algorithm to find the optimal maximum amplitude based solely on the noisy loss function measurements in the digital domain. For each iteration, the AWG is given the value of $\alpha$, which it uses to scale its output signal, which is the signal modulating the MZM. After a run, the new value of $\alpha$ is computed on a computer and passed to the AWG for the next run. The result of these experiments after the convergence of each task is shown in Table 3.11 below. There is a minor degradation in classification accuracy with respect to the digital RC. The cause may be the noisy nature of the experimental setup and the limited fine-tuning of other parameters. Also, interestingly, we observe in Fig. 3.17(f) that a

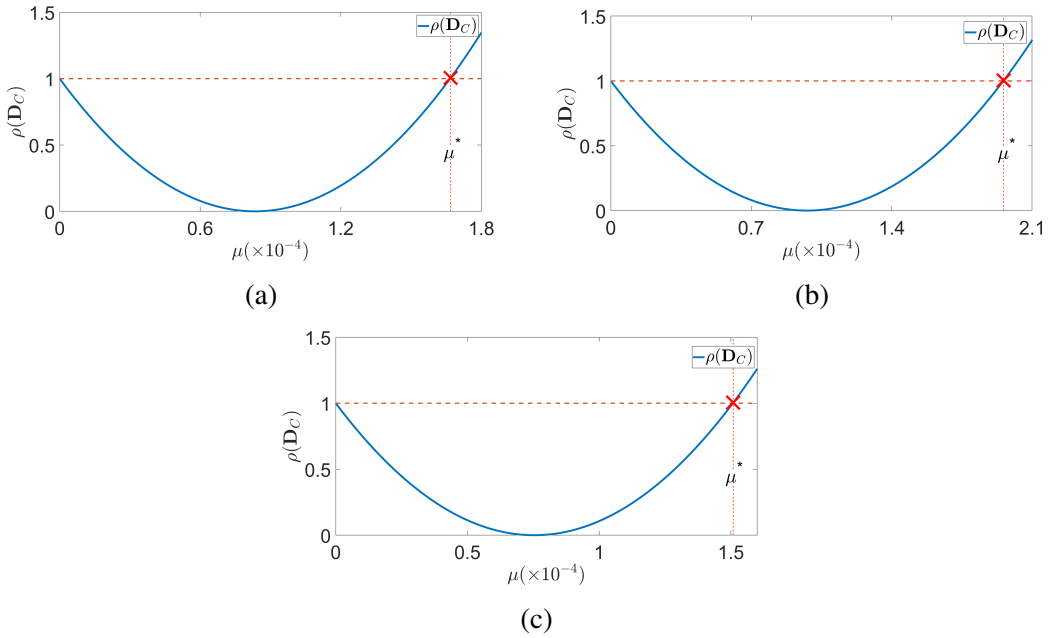



Figure 3.16: The spectral radius of $\mathbf{D}_C(\mu)$ plotted versus $\mu$ for the (a) Spoken Digit, (b) Control Chart, and (c) Wafer classification tasks with the hardware RC.

**Spoken Digit Recognition**

| | |
|---|---|
| Learning step-size ($\mu$) | $1.5 \times 10^{-4}$ |
| Finite-difference step ($h$) | $2 \times 10^{-2}$ |

**Control Chart Classification**

| **Parameter** | **Value** |
|---|---|
| Learning step-size ($\mu$) | $1.8 \times 10^{-4}$ |
| Finite-difference step ($h$) | $2 \times 10^{-2}$ |

**Wafer Classification**

| **Parameter** | **Value** |
|---|---|
| Learning step-size ($\mu$) | $1 \times 10^{-4}$ |
| Finite-difference step ($h$) | $5 \times 10^{-2}$ |

Table 3.10: Parameters of the proposed hyperparameter optimization for hardware RC. Top, Middle, and Bottom for the Spoken digit, Control Chart, and Wafer classification tasks, respectively.

| **Task** | $\alpha$ | **CER or WER** (%) | **steps** |
|---|---|---|---|
| Spoken Digit Recognition | 0.32 | 2.6 | 30 |
| Control Chart Classification | 0.18 | 4 | 52 |
| Wafer Classification | $5 \times 10^{-2}$ | 12 | 90 |

Table 3.11: Result for the hardware experiments showing the optimal parameters, classification error, and the number of steps required to attain convergence.

| | **var($\alpha$)** | | |
|---|---|---|---|
| **Type** | **Spoke Digit** | **Control Charts** | **Wafers** |
| Theoretical | $5.87 \times 10^{-6}$ | $3.77 \times 10^{-7}$ | $3.34 \times 10^{-4}$ |
| Empirical | $5.22 \times 10^{-6}$ | $3.32 \times 10^{-7}$ | $3.15 \times 10^{-4}$ |

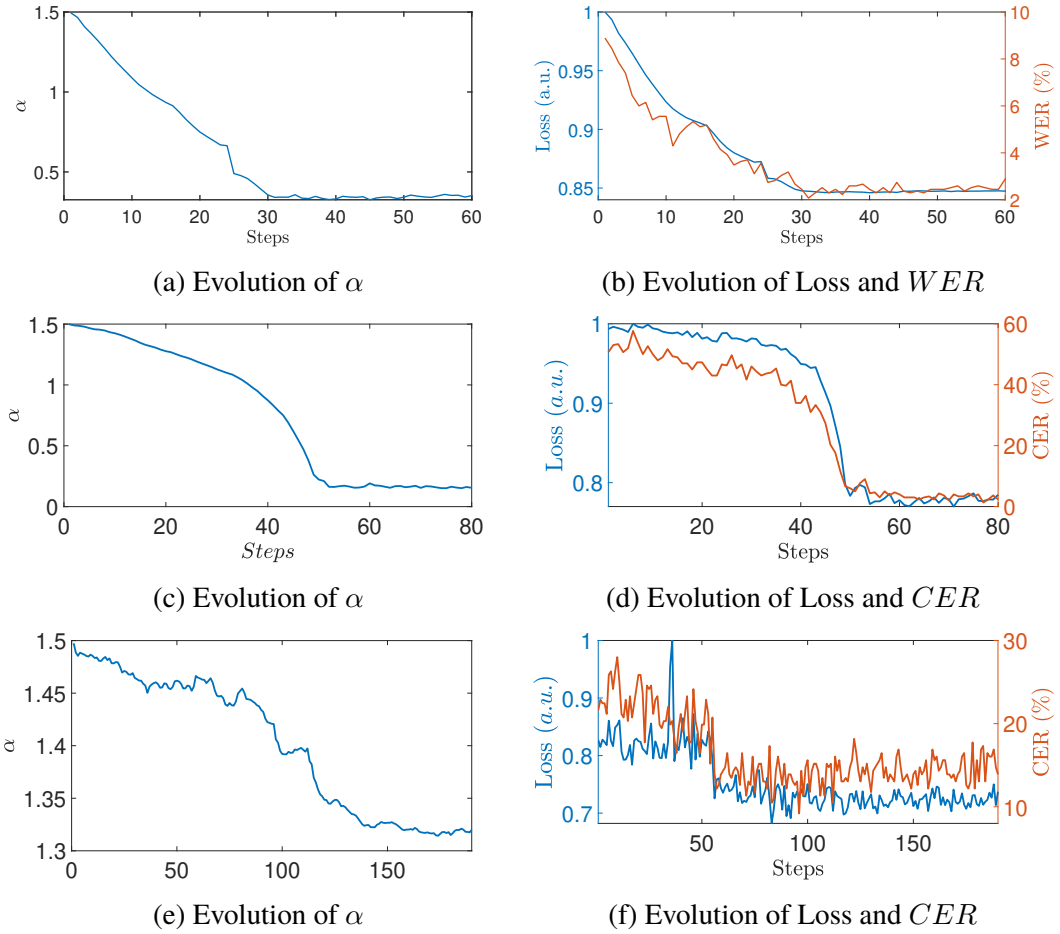
Table 3.12: Theoretical and empirical values of variances of $\alpha$ for the proposed hyperparameter optimization in the optoelectronic hardware experiment.

lower loss function does not always convert to a lower CER. Our interpretation is that the optimum obtained by minimizing the chosen loss function in Equation. 3.2 need not be the same as the one obtained by minimizing some other cost function such as the CER. Nevertheless, it can be checked on all used datasets that a low value of the loss function is consistent with a low CER.

## 3.6 Conclusion

In this chapter, an algorithm for automatic tuning of the hyperparameters of RC is introduced. The proposed method is a stochastic optimization version of gradient-descent on a loss function, where gradients are approximated with finite differences so that the functional relationship between the desired hyperparameters and the value of the loss function need not be known. Unlike similar methods, the procedure can track the time fluctuations of the hyperparameters and is numerically robust since it is governed by a constant learning step-size and a constant finite-difference step. Under suitable hypotheses, the convergence analysis shows that the transient behavior converges to the optimum value. The effect of noise affecting the loss function can also be predicted.

All aforementioned features constitute a valuable advantage in practical RC hard-




(a) Evolution of $\alpha$

(b) Evolution of Loss and $WER$

(c) Evolution of $\alpha$

(d) Evolution of Loss and $CER$

(e) Evolution of $\alpha$

(f) Evolution of Loss and $CER$

Figure 3.17: The proposed hyperparameter optimization method for hardware RC device: (a), (c), and (e) show input scaling ($\alpha$) evolution and (b), (d), and (f) Loss and CER (WER) evolution for the Spoken Digit, Control Chart Chart, and Wafer Classification tasks on the left and right column respectively.

ware implementations. Indeed, the successful application of an optoelectronic implementation demonstrated the validity of the proposed approach to the speech recognition, Control Chart, and Wafer Classification tasks.

# 4 Reservoir Computing for Early-Stage Alzheimer's disease Detection

*The results presented in this chapter are also, in part, presented in our papers titled "Optoelectronic Reservoir Computer for Early Stage Alzheimer's Disease detection" and "Reservoir Computing for Early Stage Alzheimer's Disease Detection" accepted and published in Conference on Lasers and Electro-Optics, Technical Digest Series (Optica Publishing Group, 2022) and the IEEE Access respectively.*

## 4.1 Introduction

Chapter 1 elaborated on the necessity for energy efficiency considerations while proposing new solutions. In conjunction with proposing an energy evaluation workflow, we dedicate this chapter to studying Alzheimer's disease detection. Alzheimer's is a brain disease that causes slow but gradual destruction of memory and a decline in behavioral and social skills. This results from the progressive destruction of nerve cells in various parts of the brain. The disease falls under dementia, an umbrella term describing the symptoms associated with the decline of an individual's ability to think, learn, and memorize information. Alzheimer's disease interferes with patients' work and social lives making them incapable of handling everyday tasks. Of the various causes of dementia, Alzheimer's is the most common type and accounts for between $60 - 80\%$ of all neurodegenerative diseases [158]. There is no known cure and the disease starts with mild memory loss and then worsens over time. In severe forms, the disease can cause death especially when the patients become incapable of responding appropriately to their environment. In fact, on average the patients live for about 4-8 years post-diagnosis of the severe symptoms.

Like all other causes of dementia, the majority of Alzheimer's cases are found in

people aged 65 and older. The strong correlation with age makes the disease more common among the senior members of the population. The advances in medicine have brought about an increase in life expectancy, especially in developed countries, and as a result; Alzheimer's disease is becoming an alarming cause of death and dependency in the developed world. For instance, the prevalence of the disease is as high as $13.8\%$ in the USA (the year 2021) and $17.8\%$ (the year 2015) in France for people aged above 75 years of age and older, respectively [158, 159]. However, there are numerous cases of an early-onset of the disease for people much younger than 65 years of age and certain rare genes are thought to be the cause. The pathology has no cure but some medicines, such as *Aducanumab* are effective when applied in the early stages, in slowing down the disease by targeting and removing the toxins accumulating in the brain.

**Chapter outline and summary**

- In Section 4.2 we explain briefly the known causes and symptoms associated with Alzheimer's disease. We describe its nature and state-of-the-art diagnosis techniques. We further elucidate the strengths and weaknesses of each method.

- In Sections 4.3 and 4.4 we describe the impact of the pathology on the fine motor control of planned movements. We describe Handwriting as an important biomarker for the detection of not only Alzheimer's but also other brain pathologies such as Parkinson's and Huntington's. We explain the two types of handwriting data acquisition: the dynamic and offline/paper-based types. We also give examples of writing and/or drawing tasks used for detection.

- In Section 4.4 we present the methods already studied in the literature for neurodegenerative disease detection from handwritten features. We highlight their findings and point out potential areas of improvement.

- In Section 4.5 we constrict our discussion to focus on ANNs-based approaches for diagnosis of the neuropathologies. We discuss the utilization of BiLSTM for Parkinson's disease detection from voice recordings and also, in combination with CNN, from handwriting. We point out the elevated complexity of these ANNs and the limitations of small datasets. We propose RC as a low-complexity alternative and the study to validate our proposal.

- In Section 4.6 we introduce the dataset of interest in this thesis, obtained from Broca Hospital in Paris. We elucidate the different features extracted from the cursive-$\ell$ writing task and the preprocessing steps employed to format the data. We also discuss the methods used to measure the performance of the models

under study, i.e., BiLSTM, CNN, k-Medoids, and RNN. We also detail the estimation of computation effort as a number of Floating point Operations (FPOs), the energy consumed in $kWh$, and the environmental impact in terms of the mass in $kg$ of $CO_2$ and equivalent Green House Gasses (GHG) emitted by the models.

- In Section 4.7 we explain our experiments and present their results. We begin by comparing prediction accuracy as is customary in the community then extend the comparison by examining the costs incurred in model optimization and selection, training, and inference to accentuate the efficiency and suggest reasonable compromises.

- Finally, in Section 4.8 we give our conclusive remarks based on our observations. We point out that energy efficiency makes RC attractive for implementation on mobile devices and gives potential perspectives for the application of RC for handwriting processing in pathology detection applications.

## 4.2 Symptoms and diagnosis

The toxic processes in the brain that lead to Alzheimer's disease are the formation of amyloid plaques and tau tangles. The latter interferes with the function of the neurons, blocks their interconnections, and causes the shrinking of the brain tissue. Unfortunately, these processes are slow and the disease may take more than a decade for any alarming symptoms to show. In the early stages, it is almost asymptomatic, then it passes through a spectrum of mildly symptomatic pre-dementia phase known as Mild Cognitive Impairment (MCI) before evolving into more severe forms. This slow progression complicates the early detection of the disease [160] while late diagnosis, in turn, reduces the effectiveness of the treatments for slowing the development of severe symptoms. The need for Early Stage Alzheimer's Detection (ES-AD) is, therefore, crucial to warrant early intervention.

Several methods have been proposed for the detection of the disease [161] such as :

- Clinical examinations: that consists of the analysis of the memory capabilities of the patient, family history, deterioration of language and motor skills, altered behavioral patterns, insomnia, depression, and hallucinations [162]. They may also include Neurological and Cognitive status checks. These methods are prone to bias and offer poor repeatability of the test.

- Paraclinical examinations: these consist of more accurate methods such as :

- Neuroimaging with Magnetic Resonance Imaging (MRI) by targeting the hippocampus, a complex part of the brain structure responsible for learning and memory. The images from MRI help physicians rule out many other diseases that may cause symptoms similar to Alzheimer's. The images can reveal abnormal liquid build-ups, tumors, and trauma in the head. By ruling out these other potential causes, Alzheimer's as the actual culprit may be made more apparent.
- Positron Emission Tomography (PET) of brain amyloid, the protein build-up is known as a cause of Alzheimer's disease. The method consists of injecting a radiotracer into the part to be examined then a detailed 3D scan of the tissue is obtained. From the images, abnormalities such as high levels of beta-amyloid can suggest a patient has Alzheimer's disease. This approach is more accurate compared to clinical tests but is expensive and not readily available.
- Lumbar puncture or spinal tap is another approach for detection by extracting the cerebrospinal fluid and examining it. Several biomarkers found in the fluid correlate strongly with diseases offering a much higher sensitivity to the pathology. However, necessitating the insertion of a needle in between two lumbar bones to extract the fluid, this approach is intrusive and also expensive [163].

The methods above for the diagnosis of the pathology have their strengths and weaknesses. In practice, a combination of more than one method is needed to give an accurate diagnosis. However, due to the high costs and expertise, these methods are available in the developed world and inaccessible to poorer nations where they are also needed.

## 4.3 Handwriting as a biomarker

As Alzheimer's progresses it affects the frontal lobe, the brain region responsible for executive functions such as controlled behavior and voluntary motion planning [164]. As a result, neurodegenerative diseases like Alzheimer's can be characterized by observing their impact on movement. The study [165] shows that the pathology can be characterized by analyzing the fine motor control and coordination of patients. Writing is a task that requires planning, coordination, and fine motor control hence the impact of the disease manifests itself in the handwriting (HW) of patients. The handwriting kinematic patterns are, therefore, important bio-markers that are discriminatory for certain brain pathologies. The literature contains investigations on using handwritten features

for characterizing Alzheimer's [165–172], Parkinson's [173–175], and Huntington's [176, 177] diseases, among many others.

These investigations of handwriting for pathology detection fall into two major categories based on how the features were recorded. Firstly, the offline handwriting data processing, [178] and secondly the online variant [179]. The offline methods consist of paper-based acquisition where semantical [180] or lexical [181] deterioration in handwriting is investigated. The offline approaches record only the positional information of the trajectory generating still image-like features. This requires a subsequent manual and visual extraction of various features from the handwriting on paper. For a summary of the methods and the corresponding offline methods, we encourage the reader to read [161].

The dynamic HW acquisition means that the features are recorded in parallel with the temporal information of their evolution at every point. The addition of temporal information gives rise to certain writing kinematics not available in the paper-based counterparts such as the speed and the jerk of the pen during the writing task. The online approaches have the benefit of richer data due to the addition of subtle temporal patterns to the positional HW features. Richer information results in interesting HW profiles that may be discriminatory for the pathology.

In this thesis, tasks involving both writing and drawing are indiscriminately referred to as handwriting tasks. Researchers use various acquisition techniques and patterns such as :

- Drawing shapes such as circles or squares [182, 183]
- Drawing straight horizontal lines, straight vertical lines, and/or spirals [165, 175]
- Copying of numbers and texts [169, 184]
- Drawing a cursive-$\ell$ pattern on a tablet of numbers and texts [171, 172]

## 4.4 Handwriting classification techniques in the state-of-the-art

In the literature, most quantitative studies on AD are based on statistical tests of the HW. The authors of [165] processed a data set of concentric circles drawn on a digital tablet by entering the data into the Statistical Package for the Social Sciences (SPSS) and carrying out the subsequent statistical analysis. Their methodology includes the analysis of variance (ANOVA), analysis of covariance (ANCOVA), and the minimum

mean square error (MMSE) estimator on the extracted $x$ and $y$ coordinates of the pen trajectory. The authors of [167] use a similar approach, they analyze the movement time (MT) and Movement Jerk which is the third derivative of positional features with respect to time. They computed the mean values of the two features before using them to classify patients. The jerk parameter proved to be more discriminatory for the three profiles of patients considered MCI, Alzheimer's Patients (APs), and healthy controls (HCs).

Other studies considered Machine Learning-based classification methods. The authors of [169] combined techniques such as ANOVA and linear discrimination analysis on dynamic text copying tasks. In [184] Linear Regression is used to process handwritten letters using a digital pen. They process both the dynamic and the static features of the handwriting and concluded that kinematic features are indeed more discriminatory. In [168], the Linear Discriminant Analysis method is employed to classify patients using copied, dictated, and own written sentences. They used the kinematic features extracted from the dynamic dataset.

The aforementioned works, however, impose a heavy assumption on the HW profiles for AD, MCI, and HC. They assume that there is a unique HW style associated with Alzheimer's disease. With this idea, the authors approached the Alzheimer's Disease detection task by extracting global kinematic parameters (e.g. average velocity, average jerk, average time etc.). This assumption is challenged by the authors of [171] who proved that it denies classifiers the subtle temporal tendencies that could improve pathology detection. Their results indicate the advantage of exploring the full dynamics of the raw data and highlight the limits of working on the global kinematic parameters. They allowed the emergence of multimodal patterns in their work by exploring the full HW dynamics using unsupervised learning techniques and studying the distribution of different profiles in the clusters obtained.

Building on [171], authors of [172] process the full kinematics of HW time series. Their approach consists of two layers. The first layer generates different clusters from the HW time series using Dynamic Time Warping (DTW) as a dissimilarity measure and k-Medoids as a clustering technique. The second layer, employs the probabilistic classifier, the Bayesian Classifier, to aggregate the contribution of clusters uncovered in the previous layer in different possible outcomes of the detection. The conditional probabilities were used to predict whether a participant has the pathology or not. They obtained a high accuracy of $74\%$ which is state-of-the-art performance.

## 4.5 Artificial Neural Networks for Alzheimer's detection

We have witnessed the advent of Artificial Neural Networks (ANNs) and their applications in information processing in various domains. However, in general, the pathology detection domain has seen relatively slow progress when it comes to the application of ANNs. The reasons for the slow progress are the smaller and imbalanced datasets and the general high complexity of ANNs training leading to poor generalizations. Nevertheless, some studies have recently applied ANNs for processing small datasets by employing data augmentation techniques and/or models with lower complexities. In the literature, we find fewer studies harnessing the power of ANNs for brain pathology diagnosis.

For instance, a standalone implementation of Bidirectional Long Short-Term Memory (BiLSTM) is proposed in [185] for the detection of Parkinson's disease from the voice recording time series of the participants. Parkinson's disease is known to cause a decline in cognitive functions, motor skills, and more importantly here, speech. The speech time series considered in this study consisted of features such as frequency variations (jitter), amplitude variations (shimmer), harmonicity etc. Their results indicated superior detection performances compared to the traditional machine learning approaches such as Logistic regression, Support Vector Machine (SVM), Decision Tree, k-NN and Ensemble bagged tree.

BiLSTM has also been considered in combination with Convolutional Neural Networks (CNNs), for Parkinson's disease detection [186] using handwritten features. They used dynamic HW features from online recordings of tasks such as cursive-$\ell$, triangular wave, rectangular wave, and repetitive writing of certain words. This study focused on comparing various visual representations of the original HW time series data. The time series to image conversion consisted of computing the Gramian angular field and spectrogram images of the time series to obtain a series of images. The series of images obtained is processed by the CNN, where several features are learned and extracted, before being injected, as serial images in the BiLSTM for further processing. They also compared performance obtained from different combinations of recorded HW features such as Pressure, $x-$coordinate, $y-$coordinate, Altitude, and Azimuth.

In line with our discussions in Section 1.3.1, we point out that BiLSTM and CNN are both high complexity solutions requiring rather sophisticated algorithms and large quantities of data to train them and attain acceptable convergence. The high training complexity and a large number of trainable parameters make them prone to over-fitting and poor generalization especially when the number of training samples is small, as is

usually the case for biomedical applications. Moreover, as pointed out in Section 1.3.1, high complexity models require more computation effort on the processors running them which, in turn, translates into more energy required for training and inference. There is an obvious need for methods that can reduce the penalties caused by the small datasets while maintaining relatively acceptable performance. Not only that but also finding alternative approaches to maintaining low model complexity in terms of trainable parameters and the complexity of the training algorithms.

The low complexity and linear regression training of Reservoir Computer make them candidates for the aforementioned end-goals. In this chapter, we will quantitatively measure and analyze the advantages of using RC on time series classification tasks over more complex BiLSTM and CNNs using the ES-AD task. We gauge the performances of all the models under study using prediction accuracy and extend the comparison to include energy consumption estimates. The goal is to compare and propose a solution that is not only accurate but also the one that has the best energy efficiency.

## 4.6 Dataset and methodology

### 4.6.1 Dataset

The HW data in this study were acquired at Broca Hospital in Paris by the authors of [161] and were processed in numerous other studies [171, 172, 187]. It consists of three groups of participants, all aged above 60 years of age, such as:

- Healthy Controls (HC): This group consists of members from two senior homes: the OLD-UP and Génération 13 in France. They were carefully selected following the neuropsychological examinations. Their diagnosis showed normal cognitive profiles for their respective ages with no signs of Alzheimer's in the early or late stages.

- Mild Cognitive Impairment (MCI): This is the group that was selected based on the diagnosis following the recommendation of [188] by carrying the general cognitive tests and non-memory tests to detect MCI.

- Early-Stage Alzheimer's (ES-AD): This group consists of patients who scored a Minimum Mean Squared Error (MMSE) of over 20 after filling a questionnaire of 30 points that evaluates attention, orientation abilities, computation abilities, language, and other cognitive tests.

Participants from each group were seated in front of a Wacom Intuos Pro Large digital tablet with a Wacom Inking pen. The tablet records different parameters relative

to the pen movement both on the surface of the tablet and its on-air trajectory at 125 Hertz. Participants were asked to perform various tasks using the pen and the tablet and recorded the time series of the features described in Table 4.1. The interface was designed using C++ programming language and the tablet had a resolution of 5080 pixels/inch (dpi).

| **Feature** | **Symbol** | **Description** |
|---|---|---|
| $x-$coordinates | $X$ | The position of the pen tip along the x-axis of the tablet screen. |
| $y-$coordinates | $Y$ | The position of the pen tip along the y-axis of the tablet screen. |
| Pressure | $P$ | Pressure of the pen tip on the tablet screen |
| Altitude | $Al$ | The angle between the pen and the tablet surface (See Figure 4.1) |
| Azimuth | $Az$ | The angle between the pen and the tablet's x-axis (See Figure 4.1 |

Table 4.1: Description of recorded HW features on the tablet.

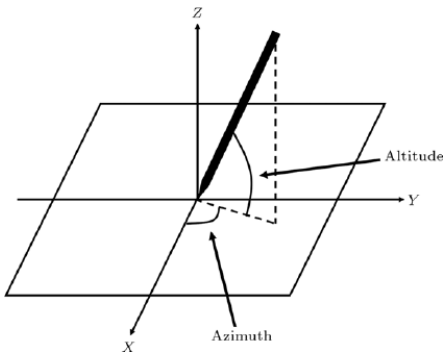



Figure 4.1: Visual representation of the difference between the Azimuth and Altitude of the pen.

Each participant was required to complete several tasks such as :

- Maintain the pen tip on a point for 15 seconds
- Drawing a spiral
- Drawing a circle
- Writing a text, fill a form or a check (imposed text and by self-chosen text)
- Writing four sets of four cursive letters as shown in Figure 4.2

The writing task of interest in this thesis is the cursive $\ell$ task where participants were asked to write four sets of cursive $\ell\ell\ell\ell$ to form a pattern shown in Figure 4.2. In this thesis, only two groups among the aforementioned three will be considered for classification, the Early-Stage Alzheimer's Disease group, and the Healthy Controls group.

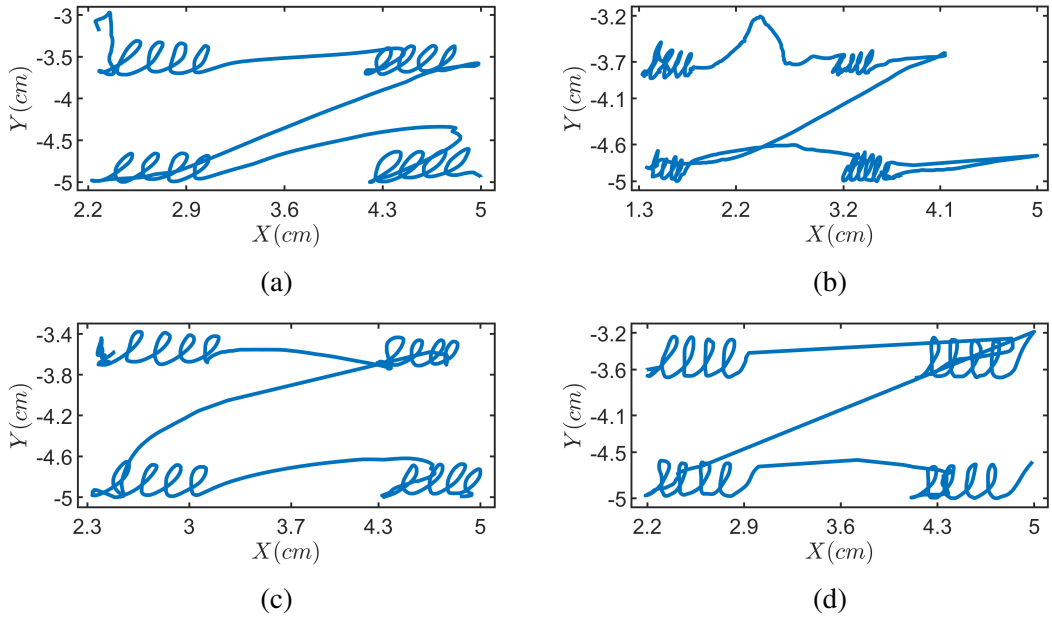



Figure 4.2: (a) and (c), the first column, are sets of $\ell$ drawn by participants from the group confirmed to be in the early stages of Alzheimer's disease while (b) and (d), the second column, are drawn by participants in the Healthy Control group.

### 4.6.2 Data preprocessing

The two groups of interest consist of only 54 participants. To properly train an ANN model one needs to have a sufficient amount of examples to train the model so as to attain acceptable performances. Fortunately, each of the 54 participants drew 16 cursive-$\ell$s that can each be considered a separate data point [161]. A loop here is defined as the combination of the upward stroke and the downward stroke. Strokes correspond to segments of the drawing between two points where the velocity in the $y-$axis direction, $V_y = 0$. The segmentation of the loop is accomplished as follows :

1. The velocity of the pen is not recorded by the tablet, it can however be deduced from the positions $X$ and $Y$ and the time (t). Figure 4.3a shows a zoom on the first four loops written by a patient with the velocity profile in the $y$-axis ($V_y$) plotted in Figure 4.3b. In the figure, one can notice the tremors in the HW.

2. To determine the point where $V_y = 0$ the velocity was low-pass filtered with the cut-off frequency set to its fundamental frequency. This removed the high-frequency tremors and the smoother profile obtained is shown in Figure 4.3d.

3. The different strokes constituting individual loops are obtained by selecting sets of two consecutive points where $V_y = 0$ on the smooth version. The points are marked by the red crosses in Figure 4.3d.

4. Two consecutive strokes, one corresponding to $V_y > 0$ and the other corresponding to $V_y < 0$ are joined together to form a single loop and Figure 4.3c shows four such loops after segmentation of those in Figure 4.3a

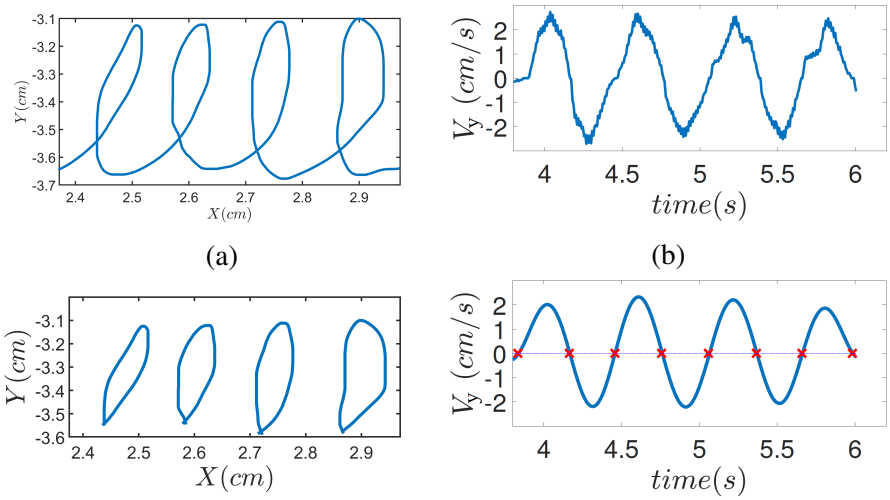
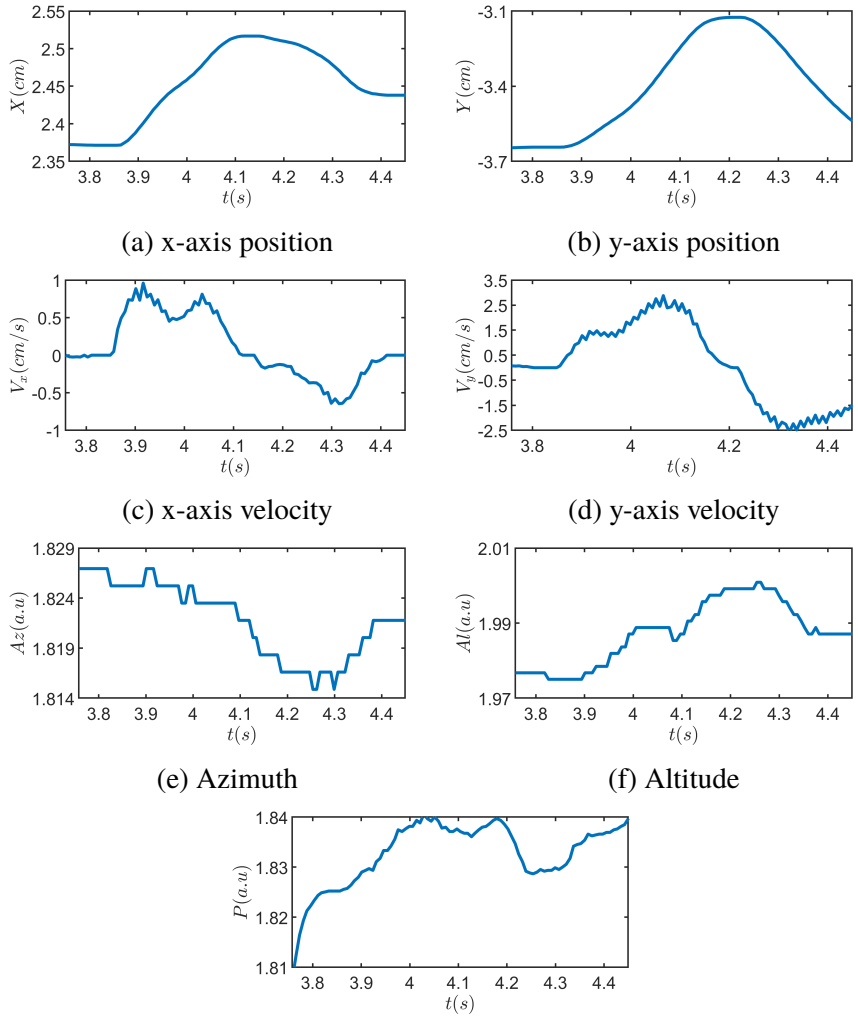



Figure 4.3: Preprocessing of HW data. Showing $X$ and $Y$ coordinates for four loops in (a) and the Y-velocity in (b). By extracting points with $V_Y = 0$ after filtering in (d) we can extract individual loops shown in (c).

After the four steps, a total of 866 loops were obtained. These correspond to 16 loops produced by each of the 54 participants ($54 \times 16 = 864$) and two extra loops produced by two participants who made 17 loops each. The velocity features along each axis were added to the set of features mentioned in Table 4.1. Each loop, therefore, has a new set of time-series features associated to it : x-position $(X)$, y-position $(Y)$, x-velocity $(V_x)$, y-velocity $(V_y)$, pen-pressure $(P)$, pen-azimuth $(Az)$, pen-altitude $(Al)$ as shown in Figure 4.4 and a corresponding label (AD or HC).

### 4.6.3 Time-series data as images for CNN

The Convolutional Neural Network is adapted for processing images hence additional processing is necessary, to convert the time series data to a form compatible with the ap-




(a) x-axis position

(b) y-axis position

(c) x-axis velocity

(d) y-axis velocity

(e) Azimuth

(f) Altitude

(g) Pressure

Figure 4.4: Time plots of different HW feature from a single loop after segmentation at $V_y = 0$.

proach. To this end, we employ the technique proposed by the authors of [189] that generates images called Gramian Angular Summation/Difference Fields (GASF/GADF) from univariate time series, while maintaining the maximum temporal relationship between the successive points. The idea is to rescale the times series samples into $[-1, 1]$ or $[0, 1]$ and to convert the rescaled cartesian-coordinate version of the input signal, say $\mathbf{U} = \{u_1, u_2 ... u_n\}$, in the form of polar coordinates such that:

$$\theta_i = \arccos(u_i), -1 \leq u_i \leq 1, u_i \in \mathbf{U} \tag{4.1}$$

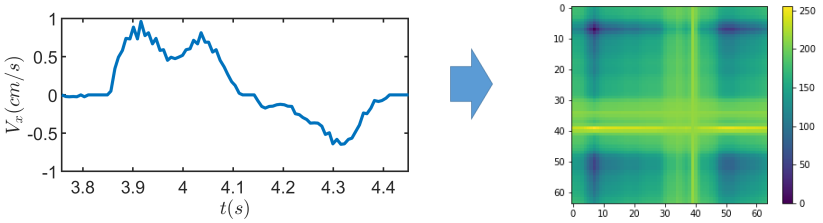
$$r_i = t_i/T, t_i \leq T \tag{4.2}$$

where $t$ is the discrete-time of $i_{th}$ discrete sample $u_i$ and $T$ the total duration. From the coordinates above, the images are obtained as follows :

$$GASF = \cos(\theta_i + \theta_j) \text{ for } (i,j) \in \{1...T\} \times \{1...T\} \tag{4.3}$$

$$GADF = \sin(\theta_i - \theta_j) \text{ for } (i,j) \in \{1...T\} \times \{1...T\} \tag{4.4}$$

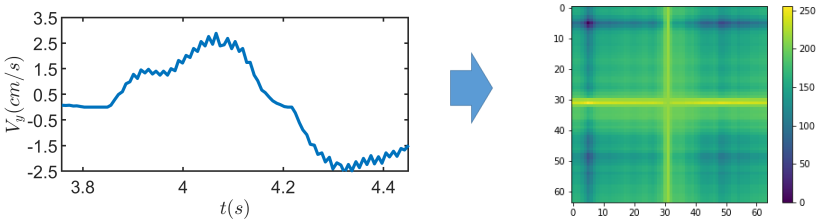
Examples of the obtained images under this processing are shown in Figure 4.5 below.




(a) x-axis velocity time-series



(b) GASF image for $V_x$




(c) y-axis velocity time series



(d) GASF image for $V_y$

Figure 4.5: Conversion of time series to static image representation for the HW velocities.

### 4.6.4 Model energy efficiency

In line with our discussion on the environmental impact of AI in the Introduction, we refrain from making the simplistic conclusion that a model is the best model based solely on prediction accuracy. We extend our studies to include the complexity and computation effort required to obtain each of the reported performances. To this end, we examine and quantify the computational cost incurred during the optimization, training, and inference phases to attain the reported performances.

The complexity of an algorithm is intuitively proportional to the time, the number of operations, and the energy consumed in the completion of computation. However, gauging the energy consumption of an algorithm running on a processor is no trivial task. Fortunately, with the advent of new processors, there are several approaches for estimating the cost incurred by running algorithms on electronic processors such as :

- The energy consumed by the devices running the computation [190, 191]
- The time it takes to run the algorithm on a CPU or GPU [190–192],
- The number of Floating Point Operations (FPOs) carried on the processors [191, 193].

### Energy, Carbon Dioxide, and Green House Gases Emission

To incentivize responsible research in Machine Learning, the authors of [194] developed an efficient framework for estimating the cost of running machine learning algorithms. They named the framework: the *experiment-impact-tracker*, as it tracks the operations of the target algorithm and estimates :

- The energy in $kWh$ is required to power the system during the runtime. This framework only estimates the power consumed by DRAM, CPUs, and GPUs ignoring all other components. To account for the other components that contribute to the total consumed power, the power consumed by CPU, GPU, and DRAM is rescaled using the power usage effectiveness (PUE) factor. This factor allows the estimation of the overhead due to cooling, network hardware, power conversions, etc.
- The Carbon Dioxide gas ($CO_2$) and the equivalent Green House Gasses (GHG) using the unit: ($CO_{2eq}$) emissions. The amount of these gases released into the atmosphere depends on the nature of the electricity sources supplied for the grid. It can vary from greener sources to the most pollutant sources. Therefore, to get a good estimation of the emission, the frameworks utilize the models from [195] that take into account the nature of the power lines deployed in the selected geographical location.

### Floating Point Operations and Runtime

The energy and $CO_2$ emission estimation previously discussed, gives more accurate results for an algorithm running for a relatively extended duration (many minutes or many hours). The tracker is therefore useful for estimating the time-consuming tasks such as parameter optimization of the models we considered in our case. To estimate the relative cost of training and inference, tasks that take a much shorter duration, we employ the Floating Point Operations (FPOs) count, length of runtime, and the number of trainable parameters of the models. To count the FPOs running on the processors throughout the task, we use a tool called PAPI [196].

PAPI (Performance Application Programming Interface) is a platform and operating system independent interface for various counters in the systems such as CPUs, GPUs, File Systems, Memory, and I/O systems. It does so by tracking the software and hardware events almost in real-time. PAPI allows access to sets of registers that count events in the hardware giving detailed information necessary for deducing the performance and reliability metrics. Among the counters, PAPI gives the FPOs count of the running computation.

To obtain a rough estimate of the incurred energy costs for the RC hardware implementation, we compute the sum of the powers consumed by various components of the setup to estimate the energy consumption when running the experiment on hardware. The given estimate is a rough average estimation of the consumed power on repeated runs but sheds light on the orders of magnitude of the cost.

## 4.7 Classification experiments

The objective is to diagnose the state of the patient as either sick (AD) or not (HC). We define this as a classification problem and propose models that will predict the patient's state given their set or subset of HW features exemplified in Figure 4.4. We propose three models for this task the BiLSTM, CNN, and RC models described in Section 1.2.3 and 2.2 respectively. The RC is implemented both as a computer simulation and in the hardware optoelectronic implementation described in Section 2.6. We also re-implement the state-of-the-art approach by the authors [172] of based on clustering by $k$-Medoids using Dynamic Time Warping as a dissimilarity measure followed by the Bayesian classifier for aggregating the contribution of clusters and carrying predictions. To gauge classification performance in our experiments we use the following metrics :

- Accuracy (**Acc**): The overall percentage of correctly classified individuals
- Sensitivity (**Sens**): The percentage of correctly classified Alzheimer's patients
- Specificity (**Spec**): The percentage of correctly classified healthy control (HC) individuals.

To minimize over-fitting and incorrect reporting of performances in our experiments, we employed techniques known in the machine learning community. We adopted the Nested-Cross Validation (NCV) technique [197, 198]. This consists of splitting the dataset in a way to avoid bias in the model selection, i.e., to avoid reporting the accuracy of parameters selected and tested on the same data. NCV consists of two loops, the inner and outer loops. In the inner loop, the normal Cross Validation is carried and the best

parameters are selected. The outer loop supplies the inner loop with data after holding out the test set. The test set is held to be used only once in the testing phase. The informal visual representation of NCV is shown in Figure 4.6.

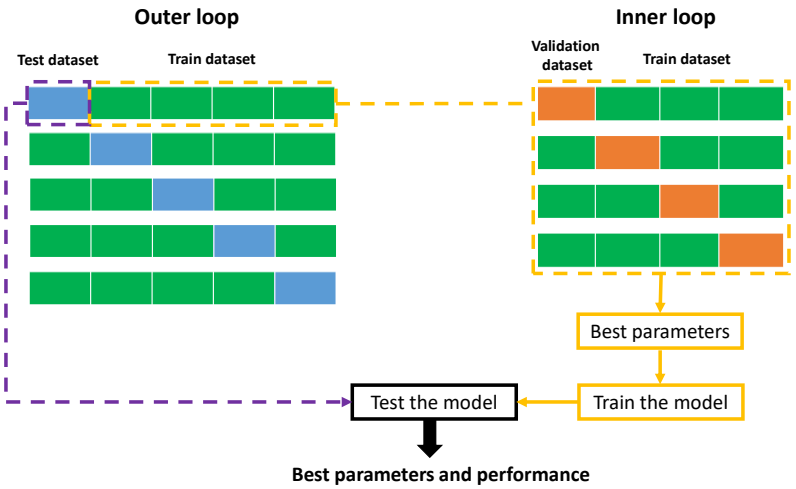



Figure 4.6: Nested cross-validation scheme.

In the experiments, the data was split into $\frac{7}{9}$-$\frac{1}{9}$-$\frac{1}{9}$ subsets. The $\frac{7}{9}$ (42 subjects) and $\frac{1}{9}$ (6 subjects) are for the inner cross-validatory loop to be used for training and validation respectively. The remaining $\frac{1}{9}$ (6 subjects) of the data is reserved for testing and final model selection by the outer loop. This guarantees that the $\frac{1}{9}$ (6 subjects) held out for the test are never used for the optimization of the hyperparameters in the inner loop offering more unbiased performances.

For the splits, we used another technique known as stratified sampling to ensure class homogeneity. To this end, we maintained the same number of examples for each class in the subsets used for testing in the inner and outer loops. Of the 6 randomly chosen test individuals reserved for testing, 3 are AD and 3 are HC. Since the model predictions are carried from features of a single loop and not the individual, the prediction of an individual class is obtained through the maximum voting technique. With this technique, prediction is done on all of the participant's loops (16 loops) and the class representing the majority of the predictions is considered the true diagnosis of the person.

### 4.7.1 Classification results in prediction accuracy

For each of the models under study, we optimize and select the final model using the NCV described above in an exhaustive search fashion. We note that, in addition to the model's hyper-parameters, we also search for the best combination among the seven

features to be fed into the proposed predictive models. To this end, we let $k$ denote the index of a feature combination among the ones listed in the **Parameters** column of Table 4.5.

For the RC model, we optimize the size ($N$), the input scaling ($\alpha$), the feedback scaling ($\beta$), and the parameter combination $k$. Table 4.2 shows the characteristics of the resultant RC model. The reservoir size of $N = 300$ yields the best accuracy with the optimal input features combination being the two ($F = 2$) velocity features alone ($k = 4$ in Table 4.2). The predefined number of discrete time-steps obtained after the preprocessing of the dataset is $T = 100$, therefore the input to RC is a $F \times T$ matrix. The optimal scaling factors, $\alpha$, and $\beta$ are found to be $0.6$ and $0.1$ respectively whereas $\phi$ is manually set to $0$ *rads* since it only defines the operating point of the RC system - and when combined with the amplitude it defines the extent of non-linearity experienced by the system as explained in Section 3.5.2. Finally, the total number of parameters to be optimized through training is $600$, that is, the elements of the read-out matrix $W^{out}$.

| **Layer type** | **Output shape** | **Trainable Parameter** |
|---|---|---|
| Input | (N = 300 , T = 100) | 0 |
| Reservoir | (N = 300, T = 100) | 0 |
| Output | (C = 2, N = 300) | 600 |

Table 4.2: Resultant digital RC model architecture

The BiLSTM model's characteristics obtained after optimization are summarized in Table 4.3. The optimal learning rate is $10^{-3}$ using a Sigmoid function. The size of the recurrent BiLSTM layer to be $H = 32$ and the dense layer responsible for final prediction has the size of $D = 128$. The predicting layer utilizes a *binary-cross entropy function*. The total number of parameters to be optimized through training is $409728$ accounting for all the weights and biases in the network.

The CNN architecture, on the other hand, is not deduced through the model selection procedure similar to RC and BiLSTM described above. It is instead adapted from the model that already proved successful for Parkinson's disease detection in [175] consisting of three Convolutional layers, two Max pooling layers, and a Dense layer. The number of trainable parameters through training sums up to $1168417$ as shown in Table 4.4

The performances in accuracy are reported in Table 4.5 are the best accuracies for

| Layer type | Output shape | Trainable parameters |
|---|---|---|
| Bidirectional | (T = 100, H = 32) | 2432 |
| Flatten (Flatten) | (3200) | 0 |
| Dense (Dense) | (D = 128) | 409728 |
| Dense | (1) | 129 |

Table 4.3: Resultant BiLSTM model architecture

a given $k$ (input feature combination), that is, we retain the highest accuracy and the corresponding hyper-parameters combination for the model. From these results three important facts are revealed:

- Confirming the important observation of [172], the $V_X$ and $V_Y$ velocity features yield the best accuracy. The use of velocity features alone (i.e. $k = 4$) outperformed all other feature combinations across all the models, i.e., the RC, BiLSTM, and CNN (Table 4.5).
- The $Az$, $Al$, and $P$ features do not add enough discrimination regarding the classification of Alzheimer's vs. Control across all three models. Their addition to $X(Y)$ or $V_X(V_Y)$ contributes little accuracy improvement for the models and sometimes hurts the performance.
- The use of all features for prediction tends to yield the least performances across all the models. The increased features enlarge the space representing the data which in turn causes an exponential increase in the amount of data needed for the model's proper generalization. In general, the increase in the number of irrelevant features negatively affects the training on a limited amount of data similar to our case, this is known as the curse of dimensionality.

We summarize the prediction performance results in Table 4.6 and complete the table with other metrics, Specificity, and Sensitivity. We also report the accuracy of $74\%$ obtained by the authors of [172] on a similar task using clustering with $k$-Medoids algorithm followed by a Bayesian Classifier. The recurrent models processing the input as a time series, i.e. BiLSTM and RC outperformed the static methods (CNN and k-Medoids). The BiLSTM attained the highest accuracy leading to the conclusion that

| Layer type | Output shape | Trainable parameters |
|---|---|---|
| conv2d (Conv2D) | (62, 62, 32) | 608 |
| MaxPooling2D | (31, 31, 32) | 0 |
| conv2d (Conv2D) | (27, 27, 128) | 102528 |
| MaxPooling2D | (13, 13, 128) | 0 |
| conv2d (Conv2D) | (11, 11, 64) | 73792 |
| Flatten (Flatten) | (7744) | 0 |
| Dense (Dense) | (128) | 991360 |
| Dropout | (128) | 0 |
| Dense | (1) | 129 |

Table 4.4: The CNN model architecture.

it is the most sensitive and most accurate model among the studied models. At $88\%$ accuracy it surpasses RC and CNN by $3\%$ and $5\%$ respectively. CNN is the least accurate of the three studied models yielding an accuracy of $83\%$ and also provides the least sensitivity to the pathology.

### 4.7.2 Classification results in model efficiency

In the previous discussions, we concluded that the BiLSTM is indeed the best model among those studies in this thesis by looking at the classification accuracy. Indeed BiLSTM surpasses RC by $3\%$, CNN by $5\%$, and the state-of-the-art $k$-Medoids by $14\%$ in classification accuracy. In this Section, we complement the previous results by quantifying the costs incurred to obtain the aforementioned accuracies. Then we analyze the efficiency to determine if the reported gains in accuracy between the models come at a reasonable price.

| $k$ | **Parameters** | | | **Accuracy**(%) | | |
|---|---|---|---|---|---|---|
| | | | | **RC** | **BiLSTM** | **CNN** |
| 0 | $X$ | $Y$ | - | 83 | 83 | 81 |
| 1 | $X$ | $Y$ | $P$ | 83 | 81 | 75 |
| 2 | $X$ | $Y$ | $Az$ | 81 | 77 | 79 |
| 3 | $X$ | $Y$ | $Al$ | 81 | 70 | 77 |
| 4 | $V_x$ | $V_y$ | - | **85** | **88** | **83** |
| 5 | $V_x$ | $V_y$ | $P$ | 81 | 78 | 81 |
| 6 | $V_x$ | $V_y$ | $Az$ | 83 | 71 | 79 |
| 7 | $V_x$ | $V_y$ | $Al$ | 79 | 85 | 77 |
| 8 | **All parameters** | | | **77** | **72** | **77** |

Table 4.5: Feature combination results for the digital RC, BiLSTM and CNN.

### Cost of hyperparameter tuning and model selection

Hyperparameter tuning and final model selection are important steps in finding models suitable for a given task. For the ES-AD task, the exhaustive search is used to scan the discretized parameters space to select the tuple yielding the best results. As mentioned in the Section 4.7.1 we used the Nested cross-validation technique with $\frac{7}{9}$-$\frac{1}{9}$-$\frac{1}{9}$ splits. This translated to the outer NCV loop running $8$ times and the NCV inner loop running $7$ times. Inside the inner loop, there is a loop scanning the $8$ features combinations followed by the loops for each model's hyperparameters. This gives the total train-test cycles for a model around $8 \times 7 \times \times 8 \times Z$ where $Z$ is the number of runs necessary to exhaust the specific model's parameters.

To track the costs incurred, the *experiment-impact-tracker* framework introduced previously is used. We estimate the costs for optimization and selection of the final model for the digital RC and BiLSTM as they are the two models we designed from scratch and they are the two best methods in terms of accuracy in Table 4.6. We will also elaborate on the complexity of the CNN and $k$-Medoids in the sequel since they are adapted from already established works hence only a few of the model's hyperpa-

| **Method** | **Acc** (%) | **Sens** (%) | **Spec** (%) |
|---|---|---|---|
| BiLSTM | 88 | 96 | 79 |
| Digital RC | 85 | 88 | 83 |
| CNN | 83 | 75 | 91 |
| k-Medoids [172] | 74 | 75.6 | 72.2 |

Table 4.6: Comparison of accuracies for all models under consideration for the ES-AD task.

rameters are to be optimized. The results for the consumption and environmental impact are summarized in Table 4.7. The consumed electric energy is expressed in $kWh$ consumed and the corresponding mass of carbon dioxide released by our optimization experiments is expressed in kilograms of the $CO_2$ or equivalent GHG. To scan and select the optimal parameters for the BiLSTM approach $9.312$ *kWh* of electric energy was required. The RC approach required only $1.156$ *kWh*. Therefore, the BiLSTM model costs more than $8$ times in electricity compared to the digital RC approach and takes 44 *hours* (compared to 21 *hours*) to complete the optimization on our Intel Xeon $E5-1603$ processor.

These results are based on the electric grid information for *Palaiseau* City in France where the experiment was carried out. Therefore a total of $0.586$ *kg* of $CO_2$ and equivalent GHG is released into the atmosphere by our experiments. Of which, $0.521$ *kg* comes from optimizing the BiLSTM whereas the RC contributed $0.065$ *kg* into the atmosphere. Consequently, the optimizing RC has a relatively lower environmental impact compared to BiLSTM due to lower emissions are also shown in Table 4.7.

| **Method** | **$CO_{2eq}$** (kg) | **Energy** ($kWh$) | **Duration** (hours) |
|---|---|---|---|
| BiLSTM | 0.521 | 9.312 | 44 |
| Digital RC | 0.065 | 1.156 | 21 |

Table 4.7: Energy consumption for hyperparameter tuning and model selection for the RC and BiLSTM.

### Costs of training the model

The energy consumption and emission results presented previously include a large number of train-test cycles. But how much does training alone cost relatively between the models running on the same processor? Unfortunately, the *experiment-impact tracker* could not be used to estimate the energy and emissions directly during training since the training times were short for the models under consideration. However, we compute the number of parameters and the Floating Point Operations using *PAPI* to have a glance at the relative costs of training each model. As previously stated, the number of FPOs provides an idea of how much computation effort an algorithm exerts on the processor. The results of this experiment are presented in Table 4.8.

| **Method** | **Trainable Parameters** | **Billion Floating Point Operations** |
|---|---|---|
| CNN | 1168417 | 1313.09 |
| k-Medoids | 750016 | 83.98 |
| BiLSTM | 412289 | 18.93 |
| Digital RC | 600 | 11.92 |

Table 4.8: Estimates of FPO counts required for training.

In this experiment, the CNN had the largest number of parameters and required the highest computation effort. Relative to the RC, CNN has $1947$ times more parameters requiring optimization through training. The optimization necessitated more than $110$ times the FPOs required by RC to attain optimality. The $k$Medoids is the second most expensive approach despite having the least accuracy. The training required about $7$ times more computation effort compared to RC having about $1250$ times more parameters requiring optimization. It is worth noting that, the most expensive part of the approach was the clustering part, more specifically, the computation of DTW distances between all the 866-time series features to obtain an $866 \times 866$ matrix of distances.

The two least expensive methods are also the two most accurate for the task. BiLSTM, the most performant model surpassed RC by $3\%$ in prediction accuracy. However, this gain in performance was attained by optimizing more than $687$ times the parameters for RC. The computational effort in FPOs for optimizing RC parameters is only $63\%$ of the effort required by BiLSTM implying $37\%$ savings.

**Cost of using the trained model (inference)**

The final trained model is now ready to be deployed to serve as a predictive model. This means the model will be repeatedly supplied with inputs from numerous new patients to output a diagnosis. A good model for this case should be the one that will have lower costs on repeated use while maintaining reasonable accuracy. To measure this, we analyze the number of FPOs required by the models for inference. We count the FPOs covering the loading of the saved model, loading the data, and the whole process resulting in a single prediction. The summary of the results for this experiment is in Table 4.9. Notice that, even in inference, the digital RC and BiLSTM require the least computation effort. For instance, RC requires $15.7\%$ the number of FPOs necessary for BiLSTM to give a prediction. This implies that RC provides a prediction by carrying significantly fewer computations on the processor hinting at lower energy requirements. However, one can notice a significant difference in effort between training and inference for RC (i.e. at $15.7\%$ w.r.t BiLSTM in inference and $63\%$ for training). The reason for the difference is due to two facts:

- Before the actual training by regression for the RC, a large amount of computation is required for the multiple projections in the reservoir (Equation 2.1) to obtain high dimensional states for all time steps $n$. The projection accounts for more than $49\%$ of the effort required for training.

- Training for RC depends on the inversion and multiplication of large matrices such as $\mathbf{X}$ and $\mathbf{X}$ after columnwise concatenation for all $n$ in Equation 3.4). These account for more than $50\%$ of the effort required for training.

  Because these calculations account for most of the energy cost in training, their absence in inference reduces the number of FPOs drastically. These results sell the RC as an even better candidate when implementing the models on mobile devices running on batteries (i.e the very tablets on which the patterns are drawn). Running repeatedly the model for predictions leads to longer-lasting charge-discharge cycles on mobile devices but will also incur lower costs of running tests on a distant server in the cloud or edge computing [199].

Once again for inference, the CNN model required the most computation effort followed by $k$-Medoids. However, the gains for RC relative to other models are much higher here since the prediction is now simplified to a single simple matrix multiplication operation.

| Method | Million Floating Point Operations |
|---|---|
| CNN | 79.8 |
| k-Medoids | 54 |
| BiLSTM | 8.64 |
| Digital RC | 1.36 |

Table 4.9: Estimates of FPO counts are required for inference.

### 4.7.3 Hardware performance

We analyzed the larger gain in inference compared to training in Section 4.7.2 and discovered that $49\%$ of the computation effort goes into the projection of the input, that is, in computing the reservoir states as described in Equation 2.1. The hardware feasibility of RC offers the possibility to *save* these FPOs by outsourcing the projection operations to the dedicated optoelectronic reservoir setup described in Section 2.6. We called this the hardware RC and the said projection will be carried in a faster analog manner. By doing so, only the regression operation to compute the $\boldsymbol{W}^{out}$ is implemented on a digital processor for the hardware RC similar to the digital variant. For a fair comparison and since the hardware RC is a close imitation of the digital RC, we use the best features combination already found for the digital RC (i.e. the one indexed by $k = 4$ in Table 4.5. We estimate the average energy consumed by the processor and the optoelectronic setup for the projection and compare them in Table 4.10. For the hardware setup, the power consumed is roughly estimated by summing the energies consumed by the setup's constituent active components. For the digital RC, the energy consumed for projection is obtained by the repeated running of the projections and computing the average energy consumed for a single run. Although the estimated values are rough estimates, they hint at lower energy costs for the setup. However, there is a marginal reduction in accuracy by $2\%$ for the hardware RC compared to the digital one. This can be caused by either the noisier nature of the hardware experiments and/or the limited and difficult fine-tuning of the physical components. A more compact setup, say integrated hardware RC would not only consume less than our setup but may even improve the prediction accuracy.

| - | **Energy** ($kWh$) | **Acc** ($\%$) | **Sens** ($\%$) | **Spec** ($\%$) |
|---|---|---|---|---|
| Digital RC | $1.147 \times 10^{-4}$ | 85 | 88 | 83 |
| Hardware RC | $8.3 \times 10^{-5}$ | 83 | 92 | 74 |

Table 4.10: Estimated end-to-end energy consumption in the reservoir layer.

## 4.8 Conclusion

In this chapter, we have proposed for the first time the use of Artificial Neural Networks (ANNs) for Early-Stage Alzheimer's Detection from handwritten (HW) temporal data. We have approached the problem by studying the trade-off between accuracy and efficiency (number of parameters, number of FPOs, and energy consumed) for four models RC, BiLSTM, CNN, and $k$-Medoids. We found that BiLSTM and Reservoir Computing are the best approaches for the task, compared to alternative methods using k-Medoids or CNNs. Both methods provided an improvement in accuracy compared to state-of-the-art with the digital RC yielding a classification accuracy of $85\%$ whilst that of BiLSTM is $88\%$, that is, an increase in $3\%$ in accuracy for the BiLSTM. Further analysis unveils that, with the digital RC, we incur significantly lower costs in optimization (8 times less energy), training (only $63\%$ of FPOs), and inference (only $15.7\%$ of FPOs) when compared to BiLSTM. The lower energy requirements for optimization and training make RC the more efficient and the more environmentally friendly approach, especially when the small performance penalty is tolerable. Moreover, the digital RC's lower inference energy cost makes it ideal to run on the same mobile devices used to record the HW pattern running on battery power for a longer period in between recharges compared to running the more costly BiLSTM method. If even slightly lower performances can be tolerated in favor of lower energy costs, the hardware RC implementations can provide a good route to even greener solutions by reducing the computation load on power-hungry electronic processors.

# 5

# Coherent Ising Machines for Combinatorial Optimization

*The results presented in this chapter are also, in part, presented in our papers titled "Image Restoration Using Coherent Ising Machine" and "Optoelectronic Coherent Ising Machine For Combinatorial Optimization Problems" submitted at the Conference on Lasers and Electro-Optics, Technical Digest Series (Optica Publishing Group, 2022), and Optics Letters respectively.*

## 5.1 Introduction

In the previous chapters, we have presented bio-inspired physical systems made from a large number of interconnected simple elements that possess interesting computational properties. We have argued that such systems represent an interesting avenue of exploration in light of increasingly overexerted Turing approaches functioning in a traditional Von Neumann fashion. They can therefore be employed for unconventional computation schemes yielding interesting results at unprecedented speeds and efficiency. Our discussions in the previous chapters were only concerned with non-autonomous implementations of neuromorphic systems that are driven by an input signal and require a technique for training (supervision) for their computation to be deemed useful. However, neuromorphic systems have other properties that make them attractive even in the autonomous mode of operation i.e. without the driving input signal. In this chapter, we explore the unsupervised variant of recurrent neuromorphic systems both in digital and hardware platforms. This area lacks comparisons to other near-global optimization techniques as well as evaluations of the energy efficiency of the proposed systems and we will contribute to that.

Following the philosophy of this thesis, we study a hardware architecture of a Coherent Ising Machine (CIM) based on off-the-shelf telecommunication components, i.e. a semiconductor laser coupled to a Mach-Zehnder Modulator (MZM), a photodiode, an Arbitrary Waveform Generator (AWG), an Analog-to-Digital Converter (ADC) and an FPGA board. Our study considers the implementation of such a system in a digital simulation followed by a physical setup. In addition to the hardware architecture, the main novelties of our exploration are as follows: from an application perspective, we demonstrate the potential of the CIM for statistical image denoising - from a system-level perspective, we compare the proposed mixed hardware/digital system to a standard digital implementation of Hopfield networks and simulated annealing (SAN) [144] in terms of the probability of reaching the average ground state, computational complexity, and energy consumption.

## 5.2 Hopfield neural networks and Ising machines

In his seminal work, Hopfield introduced networks of neuron-like elements and studied their usefulness in combinatorial optimization problems expressed as the minimization of a quadratic energy function, both in continuous [200] and discrete time [201]. While theoretical guidelines to achieve feasible solutions were investigated in [202], several practical improvements have also been published to escape local minima including tailoring the energy function and its hyperparameters [203], incorporating simulated annealing heuristics [204], random noise [205] or transient chaotic behavior [206]. A generalization to highly nonlinear energy functions has also appeared in [207].

The introduction of efficient hardware architectures of Hopfield networks and their generalizations during the last decade has sparked renewed interest in this field. These implementations, known as Ising machines, are based on various physical principles such as quantum [208]-[209], nanomagnetic [210], memresistive [211] or laser [212] and photonic [34] technologies. Before defining Ising machines, we will briefly introduce the Ising model which is at the heart of the working principle.

At sufficiently low temperatures, i.e. temperatures below a critical value $T_{cr}$, some metals become magnetized at a macroscopic level. This is known as a ferromagnetic behavior and is the result of the polarization alignment of the magnetic moments (spins) of the constituent atoms in the same direction at the microscopic level. Above $T_{cr}$ the moments or spins become randomly oriented resulting in zero net magnetic fields. This is the basis of the Ising model named after *Ernst Ising*. The model assumes a fixed lattice structure and that atoms are fixed and the only degree of freedom they possess is the orientation of the spins.

We describe the model formally by letting $\Omega$ be a square $n \times n$ lattice. Any pixel $s = (l, c) \in \Omega$ can be assimilated to a position in the lattice with line (resp. column) coordinate $l$ (resp. $c$), where $1 \leq l, c \leq n$. $\forall s \in \Omega$, we let the spin $\sigma_s$ be a random variable in $\{-1, +1\}$. The random vector $\boldsymbol{\sigma}$ is obtained by raster scanning the spins columnwise. A particular Markov random field (MRF) defines the probability mass function (pmf) of $\boldsymbol{\sigma}$ as

$$P(\boldsymbol{\sigma} = (\sigma_s)_{s\in\Omega}) \propto e^{-E(\boldsymbol{\sigma})}, \tag{5.1}$$

where the system's energy function has the form of an Ising Hamiltonian

$$E(\boldsymbol{\sigma}) = -\frac{1}{2} \sum_{(s,t)\in\mathcal{N}} J_{s,t}\sigma_s\sigma_t - \sum_{s\in\Omega} b_s\sigma_s, \tag{5.2}$$

where $J_{s,t}$ is the coupling parameter between the spins indexed by $s$ and $t$ while $b_s$ is the bias associated with the spin indexed by $s$, and $\mathcal{N}$ denotes all couples of neighboring pixels with end-around boundary conditions. Note that finding the most probable spin configuration (the so-called ground state), which is equivalent to the combinatorial optimization problem consisting in minimizing Equation 5.2, has found many practical applications [213].

## 5.3 Combinatorial Optimization problems

Systems dependent on multiple parameters to characterize their operation are abundant in many real-world scenarios. As a result, the determination of the parameter tuples corresponding to a certain desired mode of operation requires means enabling practitioners to search for the right set of parameter combinations. Combinatorial Optimization consists of searching for a solution in a discrete set (possibly multidimensional) such that an objective function is optimized (maximized or minimized).

The search space for the combinatorial optimization problems explodes rapidly with the increase in the number of parameters prolonging the search duration. In computation complexity terms, they are NP-hard problems. A straightforward method to address combinatorial optimization problems is the Brute Force method, capable of attaining exact solutions but ill-famed for coping with the exponential increase in search space causing complexity outbursts. The elevated computational demand attracted the application of computationally cheaper heuristics to find near-optimal solutions in a reasonable amount of time.

However, some problems remain complex even for the less explosive approaches, therefore, compromises on the exactitude of the solution are made by settling for good-

enough approximations. Luckily, for most practical applications, good-enough solutions obtained in a reasonable time are more desirable than the exact solutions requiring much more resources. Here, we will discuss two scenarios requiring combinatorial optimization solutions to divulge the variety and importance of applications that can benefit from such models.

### 5.3.1 MAXCUT optimization

This problem consists of finding the split of a graph into two sub-graphs such that the number (or sum of their weights) of edges cut is maximized. Suppose $\mathbf{G} = [\mathbf{V}, \mathbf{E}]$ is a graph with vertex set $\mathbf{V}$ and edge set $\mathbf{E}$. A cut is a partition of $\mathbf{G}$ into $\mathbf{G_1} = [\mathbf{V_1}, \mathbf{E_1}]$ and $\mathbf{G_2} = [\mathbf{V_2}, \mathbf{E_2}]$, such that $\mathbf{V_1} \cap \mathbf{V_2} = \emptyset$. Finding the cuts corresponding to the maximum possible number of cut edges, known as a MAXCUT problem, is one of the first challenges identified as NP-complete problems. The number of possible cuts increases abruptly with meager increases in the graph size. Fortunately, the problem can be reformulated such that searching for a solution becomes equivalent to the minimization of the Ising Hamiltonian [214].

### 5.3.2 Image Denoising

The advent of cameras, powerful telescopes, and microscopes has brought about mushrooming of images taken to capture memories, study galaxies, and observe microorganisms. Unfortunately, such devices suffer degradation from the noise or imperfections of the photo sensors. Further noise can be introduced in transmission channels or during compression of obtained images. Noise is a high-frequency addition to the image, suggesting a low-pass filtering to restore the image. However, edges and textures are high-frequency components as well therefore imprudent filtering will result in blurring these important details. The noisy image problem can be formulated by supposing the clean image $y$ is subjected to additive noise $\eta$ such that the noisy image $\hat{y}$ is defined as :

$$\hat{y} = y + \eta \tag{5.3}$$

Image denoising, defined as the restoration of $y$ from $\hat{y}$ has remained a challenging problem and multiple solutions have been proposed such as filtering [215], Tikhonov Regularization [216], and CNNs [217]. We consider black and white images corrupted with Salt and Pepper noise and formulate them such that each bright (dark) pixel is an upward (downward) spin. In Section 5.5.2 we detail the image model and demonstrate the denoising capability of our Ising machine.

## 5.4 Experiments

We implement the Coherent Ising Model in a digital simulation running on a CPU and the equivalent optoelectronic hardware. The details of these two implementations are provided below.

### 5.4.1 Digital Ising Machine

Following [34], we numerically implement a generalized Hopfield network to minimize Equation 5.2 at discrete time instant $k$ has the form

$$\begin{aligned} x_s(k) &= f\left(\alpha x_s(k-1) + \beta\left(\sum_{t:(s,t)\in\mathcal{N}} J_{s,t}x_t(k-1) + b_s\right)\right) \\ \hat{\sigma}_s(k) &= \text{sign}(x_s(k)), \quad \forall s \in \Omega \end{aligned} \tag{5.4}$$

where $f(.)$ is a nonlinear activation function (for instance a sigmoid, periodic or clipped nonlinearity as suggested in [218]), $\alpha$ and $\beta$ are scaling coefficients controlling the self-coupling and feedback strength affecting the neuron output $x_s(k)$, while $\hat{\sigma}_s(k)$ is the spin estimate of pixel $s$. The digital CIM executes a Python implementation of the algorithm described by Equation 5.4 on an Intel Xeon E5-1603 processor. The nonlinear function is the $\sin^2(.)$ imitating the transfer function of the MZM followed by a photodiode used in the hardware implementation. The nonlinear function is shown in Figure 5.3 together with the span of observed neuron states.

### 5.4.2 Proposed optoelectronic architecture

Compared to a conventional Von-Neuman architecture for the digital implementation in Section 5.4.1, mixed analog/digital processing in the form of a CIM can lead to savings in terms of achievable processing speeds and energy consumption. High-speed solutions to combinatorial optimization problems are very attractive for applications requiring real-time processing or adaptation to dynamically changing environments.

We solve the algorithm-architecture adequacy problem by selecting only commercially available telecommunication components in the optoelectronic oscillator setup shown in Figure 5.1. The details of the components of our setup are as follows:

- A Distributed Feedback (DFB) laser diode emitting light at $1.55\mu m$.
- A Mach-Zehnder Modulator (MZM) modulates the optical phase of the light emitted by the laser. The phase is modulated by the feedback signal allowing

for the *EO* conversion of the system's state. The MZM also provides the nonlinear activation function in the optical domain thanks to its $\sin^2(.)$ transfer function. The bias voltage of the MZM, $\mathtt{V_{bias}}$, is set such that the activation function after proper calibration becomes $f(x) = \sin(x + \pi/4 + \xi(k))^2 - 1/2$, where $\xi(k)$ accounts for photoreceiver and quantization noise.

- A 20 GHz photodiode for *OE* conversion necessary for the subsequent digital processing of the spins.

- The Zmod Scope $1410 - 125$ Analog-to-Digital Converter with $125MSa/s$ and 14-bit resolution [219]. This supplies the FPGA with the node states vector for the spin interactions computation according to Equation 5.4.

- The Eclypse-Z7 features a Field Programmable Gate Array (FPGA) board and two ZMod connectors allowing high-speed processing [220]. We implement via VHSIC Hardware Description Language (VHDL) the logic for high-speed data acquisition, matrix multiplication, and transfer of signals. We note that implementing the entire Ising machine in the electrical domain, that is implementing the activation function itself as a lookup table (LUT) inside the FPGA [221], would also be feasible.

- The Zmod AWG 1411: 2-channel 14-bit Arbitrary Waveform Generator (AWG) with a sample rate of $100MSa/s$ [219]. This allows for the conversion of the FPGA digital data to a continuous signal that modulates the phase of the MZM. It has two channels shown in Figures 5.1 and 5.2: channel CH1 sends data to the oscilloscope for visualization and CH2 completes the optoelectronic CIM loop.

- 10 GHz Analog RF Amplifier Driver with an output voltage of $9V_{pp}$ allowing proper signal scaling before modulating the MZM [103]. The node states in the digital CIM span the zone shown in Figure. 5.3a. In order to imitate a similar spanning of the node states (red dots on Figure. 5.3a) on the experimental activation function in the experimental setting, we tuned the bias ($V_{bias}$) of the MZM, the gain of the MZM driver and the scaling of the feedback signal ($x_s(k-1)$) to obtain experimentally a similar mapping illustrated on Figure 5.3b.

- Digital Serial Analyser (DSA) is the high-speed sampling oscilloscope connected to the CH1 of our DAC as shown in Figure. 5.2. We incorporated the DSA to visualize in real-time the evolution of the signals and spin formation in the loop.

The presented setup can implement Equation 5.4 only for a single spin, the feedback delay is decomposed into $n^2$ intervals along which the spins are multiplexed using Time

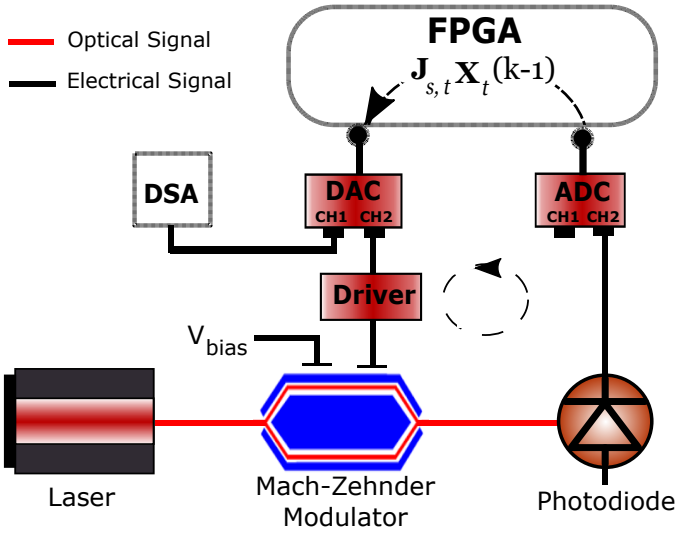



Figure 5.1: Experimental setup of the proposed opto-electronic CIM. $\mathtt{V_{bias}}$ is the bias voltage control of the MZM.

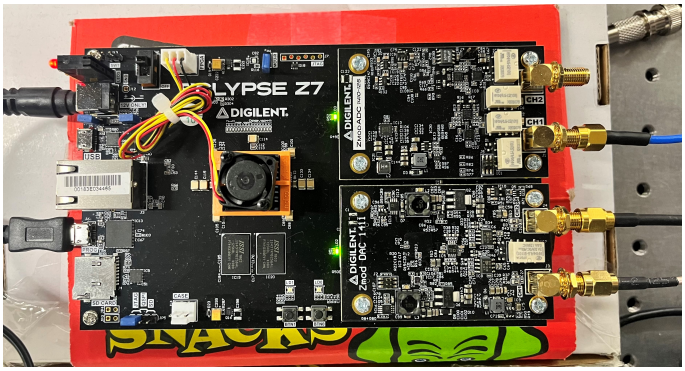
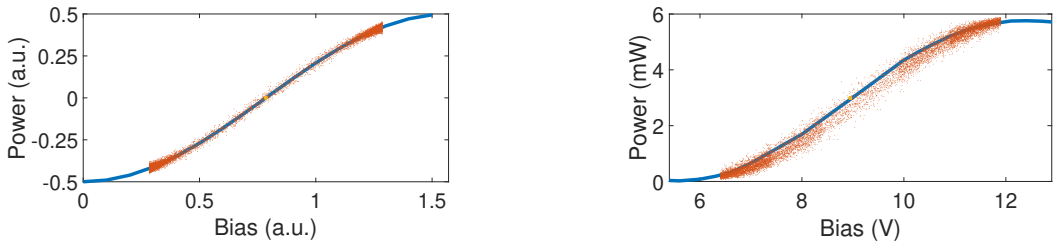



Figure 5.2: The image of our Eclypse Z7 FPGA board mounted with the dual-channel data converters (ADC and DAC).

Division Multiplexing (TDM). For each iteration, the FPGA waits for $n^2$ readings from the ADC before computing the resultant spins after spin interaction.

### 5.4.3 Simulated Annealing

As a benchmark, we implemented SAN similar to [222], arguably the most used heuristic method for gradual *cooling* of a 'high-temperature' problem to attain a frozen state that is, ideally, arbitrarily close to the solution of the problem. SAN is one of the most popular algorithms for this feat and has been implemented both in computer simulations as well as in dedicated hardware which provides parallelization of digital hardware accelerators and analog computing [223].




(a) (b)

Figure 5.3: Span of the node states on the $\sin^2(.)$ for the digital CIM (a) and on the MZM characteristic function for the optoelectronic CIM (b).

### 5.4.4 Performance metrics

To compare the optimization implementations under study we propose several metrics. Firstly, we evaluate the success probability of reaching (or approximating) the ground state for each method and compare them. We extend the analysis to incorporate estimation of computational complexity, that in the digital domain, is measured via the number of Floating Point Operations (FPOs), while an *experiment-impact-tracker* [194] running in parallel is used to report the consumed energy and the corresponding $CO_{2eq}$ emissions. For the hardware CIM, we estimate the average power consumption of constituent elements to obtain a rough estimate of the consumption. These metrics enable further analysis of the studied approaches by shedding light on the complexity vs. energy vs. performance trade-off, a subject that is mostly overlooked in the literature despite the recommendations for sustainable practices [22].

## 5.5 Results and discussion

In this section, we assess the proposed optoelectronic CIM architecture of Section 5.4.2 for the performance metrics introduced in Section 5.4.4 and we compare it to the digital CIM of Section 5.4.1, while near-global optimization using SAN based on the Gibbs sampler given in [224] is used as the benchmark method. Unless otherwise stated, $\mathcal{N}$ is chosen as the couple of $4$-point nearest neighbors in the lattice under end-around boundary conditions. For the digital and optoelectronic CIM, the initial spin configuration is chosen uniformly and independently at random, the activation function $f(.)$ is the one defined Section 5.4.2, the hyperparameters are set to $(\alpha = 0.25, \beta = 0.29)$, while the number of iterations is set to $N_{it} = 100$. Also for the sake of comparison with the benchmark method, we use SAN with an initial temperature of $2$, a geometric annealing parameter equal to $0.99$, and $200$ iterations [224]. Also, the adopted legend for all images is as follows: bright yellow for spin-up $(+1)$ and dark purple for spin-down $(-1)$.

We study two problems that can be formulated as the minimization of a Hamiltonian having the form given by Equation 5.2 such as the Antiferromagnetic Ising model and MAP image denoising using the optoelectronic Coherent Ising Machine (CIM) both in the simulations and hardware implementations.

### 5.5.1 Antiferromagnetic Ising model

We begin with a low-dimensional example where $n = 10$ with antiferromagnetic interactions, that is $J_{s,t} = -1\ \forall (s,t) \in \mathcal{N}$ and $b_s = 0\ \forall s \in \Omega$. It is well-known that the ground state corresponds to a checkerboard pattern (alternating spin-up and spin-down configuration with minimum energy equal to $-n^2$). The exact solution with this pattern is shown in Figure 5.4 for a $10 \times 10$ spin-lattice. In our experiments, Figure 5.5

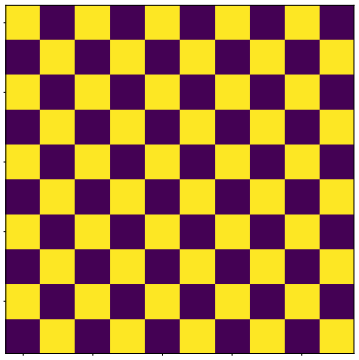
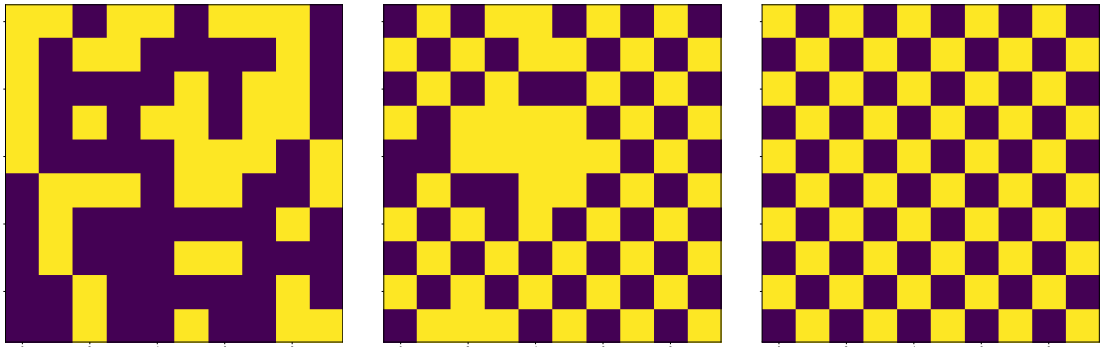


Figure 5.4: The exact solution for the $10$ by $10 \times 10$ Antiferromagnetic Ising model.

depicts the initial and final estimated spin configuration for a single run of the proposed optoelectronic CIM, that successfully converges to the ground state. In this figure, the $x$ and $y$-axis indices are pixel positions on the lattice. Over a single successful run, Figure 5.7 shows that the proposed optoelectronic CIM and the digital CIM have similar dynamics, with convergence reached typically after a few tens of iterations. Table 5.1 summarizes our performance metrics by repeating all the aforementioned experiments independently $1000$ times. Part of the TDM sequence of spins is copied on channel CH1 of DAC and sent to a Digital Serial Analyzer Sampling Oscilloscope (DSA). A screenshot in Figure 5.7 shows the resultant alternating up and down spins as a time series.

We observe that SAN converges later than both CIMs but reaches the ground state with approximately $98.9\%$ success probability compared to the approximately $91.4\%$ digital CIM. For this task, success means attaining exactly the theoretical ground energy state of $E(\boldsymbol{\sigma}) = -100$. From this standpoint, SAN is more performant. With further analysis, we noticed, however, that this win comes at a cost of $7.47$ times the



(a) Iteration = 0 (b) Iteration = 25 (c) Iteration = 50 to 200

Figure 5.5: The initial spins (Iteration = 0) are randomly and independently chosen for the digital CIM whereas for the optoelectronic CIM the system's noise initializes the spins. A checkerboard pattern appears (Iteration = 25) and stabilizes as the system converges (Iteration = 50 to 200).

| **Metric** | **Digital CIM** | **Optoeletronic CIM** | **SAN** |
|---|---|---|---|
| Success probability (%) | 91.4 | 90 | 98.9 |
| Time ($ms$) | 10 | 2.1 | 79 |
| FPOs ($\times 10^6$) | 0.76 | 0.32 | 5.68 |
| Energy ($\times 10^{-6}\ kWh$) | 0.6 | $3 \times 10^{-4}$ | 14 |
| $CO_{eq}$ ($\times 10^{-6} kg$) | 0.044 | $2.2 \times 10^{-5}$ | 1 |

Table 5.1: Average performance metrics for the antiferromagnetic model over 1000 runs.

number of FPOs and runtime required by the digital CIM. It, therefore, takes more computational effort to attain a solution with SAN than it does with digital CIM. Moreover, energy estimates with the *experiment-impact-tracker* show that SAN requires 23 times the energy of the digital CIM and the same factor for an increase in $CO_{eq}$ emissions. The optoelectronic CIM has a success probability of 90% - close to that of the digital CIM. Since the hardware CIM benefits from the speed of optics and the FPGA's programmable logic the energy analysis for the hardware CIM becomes interesting. We observe that the checkerboard solution shown in Figure 5.6 obtained in digital CIM costs 2017 times more than that from the optoelectronic CIM counterpart. What's more, the hardware CIM consumes $1/46667^{th}$ of the energy required by SAN. A significant gain in efficiency altogether.

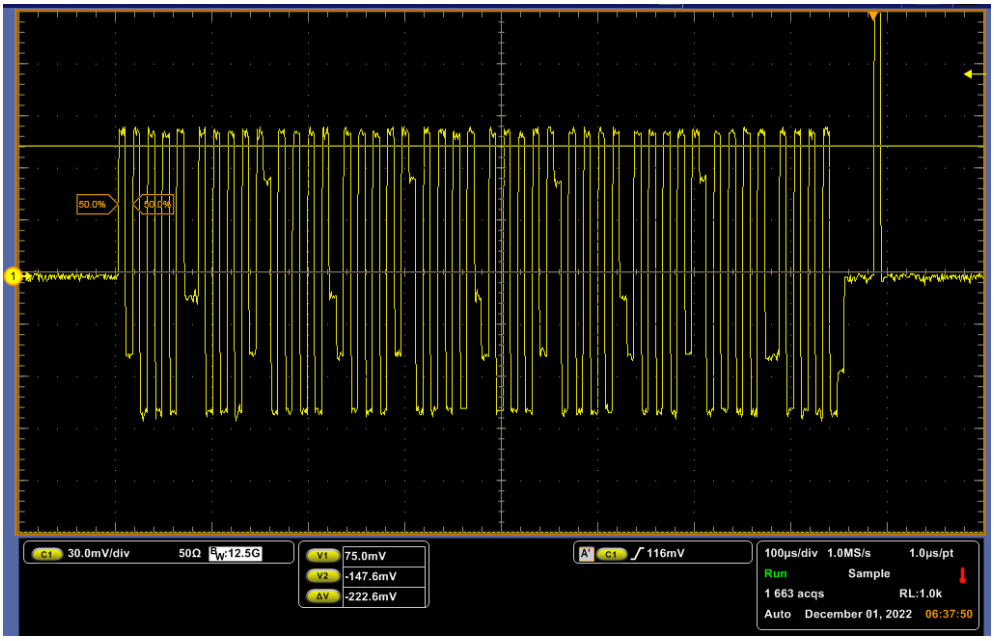



Figure 5.6: Digital Serial Analyzer screen capture showing the spins alternating in time after 100 iterations.

### 5.5.2 Maximum a posteriori image denoising

In this Section, we consider a hidden black and white image $(\sigma_s)_{s\in\Omega}$ to be restored from an observed image $(y_s)_{s\in\Omega}$ according to a salt and pepper noise model, i.e. $y_s = -\sigma_s$ with probability $p$ and $y_s = \sigma_s$ with probability $1-p$, independently for each pixel $s \in \Omega$. In our setting, $n = 64$ and the prior spin pmf is chosen as the MRF in Equation 5.1 with the coupling parameter between neighboring pixels set to 1 (ferromagnetic interactions). It is easily shown that maximum a posteriori (MAP) image denoising corresponds to selecting $J_{s,t} = 1\ \forall (s,t) \in \mathcal{N}$ and $b_s = -\frac{1}{2}\log(p/(1-p))y_s\ \forall s \in \Omega$ (as

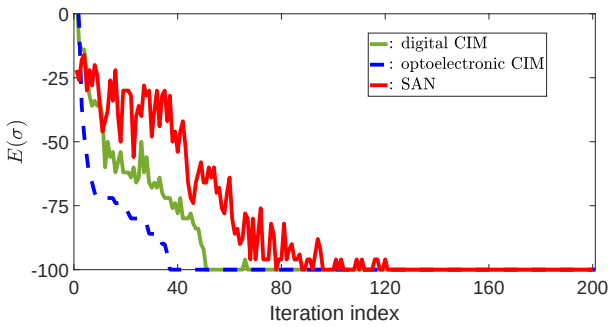
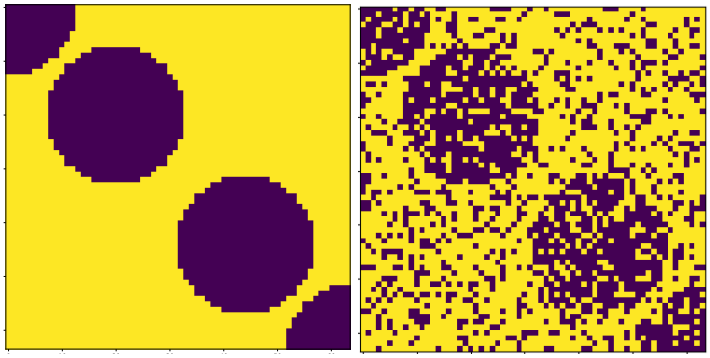



Figure 5.7: Energy evolution for a single run for the square-lattice of spins with anti-ferromagnetic interactions.

derived in Appendix B). In our experiments, we use $p = 0.2$ and we show the clean image and the resultant noisy image after being impacted by the noise in Figure 5.8.

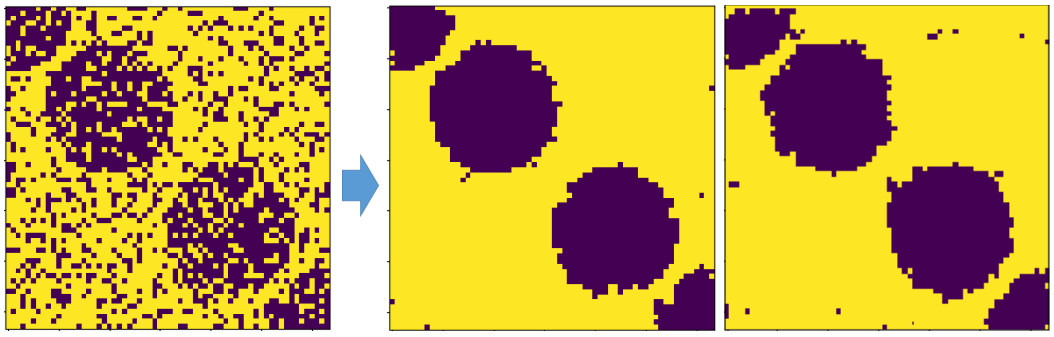


(a) (b)

Figure 5.8: The clean image (a) and the resultant noisy image (b) after the salt and pepper noise addition.

Figure 5.9 depicts the initial dirty image and the final image after running SAN, digital and optoelectronic CIM. In this figure, the $x$ and $y$-axis indices are pixel positions on the lattice as well. Over a single successful run, Figure 5.10 shows that the proposed optoelectronic CIM and the digital CIM have similar dynamics, with convergence reached typically after $15$ iterations. Table 5.2 summarizes our performance metrics - adding the pixel-wise classification error rate (PCER %) - by repeating all aforementioned experiments independently $1000$ times.

For this application, we observe that SAN attained the ground state for all the runs whereas digital and hardware CIM attained the ground state in $89\%$ and $86.75\%$ of the runs respectively. The success means considered as the system's falling within three standard deviations of the average ground state energy which is approximately $E(\boldsymbol{\sigma}) =$



(a) (b) (c)

Figure 5.9: A sample initial dirty image (iteration 1) shown in (a). After convergence (iteration 100), we obtain (b) and (c) for the digital and optoelectronic CIMs respectively.

$-5550$, thus corresponding to $98\%$ confidence interval. The clean images generated by SAN contain a PCER of $1.7\%$ whereas the digital and hardware CIMs converged to a PCER of $2.3\%$ and $3\%$ respectively. For this task as well, SAN excels over the digital and hardware CIMs in these convergence metrics. Nevertheless, further analysis reveals that SAN's performance comes at approximately $9.5$ times the execution time, 10 times the number of FPOs, and 12.22 times the energy (same factor for the $CO_{eq}$ emissions) consumed by the digital CIM. The $CO_{eq}$ emissions are reported taking into account the nature of electric grids in the *Palaiseau* city in *France*. The $11\%$ gain in convergence success probability of SAN costs us at least 10 times the computation cost of digital CIM by all measures. Following the observation with the Antiferromagnetic model we analyse the energy costs for the hardware CIM on this task as well. The energy consumption results for hardware CIM are reported in Table 5.2 showing a factor of $6216$ and $75970$ gain in energy and environmental impact.

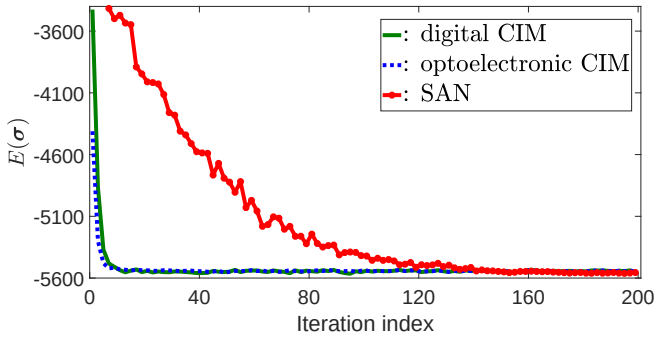



Figure 5.10: MAP image denoising energy evolution for a single run.

| **Metric** | **Digital CIM** | **Optoeletronic CIM** | **SAN** |
|---|---|---|---|
| Success probability $(\%)$ | 89 | 86.75 | 100 |
| PCER $(\%)$ | 2.3 | 3 | 1.7 |
| Time $(ms)$ | 4110 | 86 | 39130 |
| FPOs $(\times 10^9)$ | 0.7 | 0.0131 | 7.1 |
| Energy $(\times 10^{-6}\ kWh)$ | 90 | $1.45 \times 10^{-2}$ | 1100 |
| $CO_{eq}$ $(\times 10^{-6} kg)$ | 5 | $8 \times 10^{-4}$ | 62 |

Table 5.2: Average performance metrics for MAP image denoising over $1000$ runs.

## 5.6 Conclusion

Our experiments reveal the potential for CIMs in image denoising. SAN has consistently outperformed the CIMs in all of the studied tasks. It converged with the probability of $98.9\%$ and $100\%$ for the antiferromagnetic spin lattice model and image denoising respectively; compared to $91.4\%$ and $89\%$ of the digital CIM approach (slightly less for the hardware counterpart). The naive conclusion here is that SAN is indeed the better approach. However, we have presented the energy costs and computation effort to unveil the fact that SAN requires significantly more resources compared to the digital CIM implementation. The extra $11\%$ gain is convergence success probability by SAN requires 7.47 and 23 times the number of FPOs and energy required by digital CIM for the Antiferromagnetic Model respectively. For the MAP image denoising task, SAN required 9.5, 10, and 12.22 the runtime, FPOs count, and energy required by the digital CIM respectively.

With the severalfold increases in energy costs and computation efforts, a compromise on accuracy becomes reasonable. In most practical applications, a choice is often made to settle for less demanding solutions that yield acceptable performances. In this light, CIMs appear as the more reasonable and informed choice.

In the future, we aim to explore the more complex and challenging problem of a Traveling Salesman. This problem finds many practical and interesting applications in various domains of logistics, scheduling, and security. CIMs carry a promise for faster and more efficient solutions, this direction of exploration has the potential for interesting avenues.

# 6 Conclusions and perspectives

In this thesis, we focused on low-complexity approaches for various applications of neuromorphic methods. In this chapter we review the works presented in this thesis, accentuate the obtained results and give perspectives for future exploration building on our presented works.

The progress of bio-inspired approaches is threatened by the unmatched progress in speed and efficiency of the hardware they run on. In our presented works, we centered on neuromorphic approaches that possess the potential to deter the current trajectory predicted to hit the efficiency wall as we argued in Chapter 1. We studied Reservoir Computing and Coherent Ising Machines in parallel with proposing new applications. In Chapter 3 and 4 we focused on Reservoir Computing. First, we proposed and studied an efficient method based on stochastic gradient descent for hyperparameter tuning. Our theoretical and numerical analyses demonstrate the advantages of our method over SAN and the exhaustive search based on the quality and speed of convergence. Second, we assessed the energy saving and the performance penalties of using Reservoir Computing for a new task of early-stage Alzheimer's disease detection. Our results substantiate the fact that, at a reasonable compromise, significant energy savings can be made by being mindful of the costs associated with each proposed method.

Extending on our results with RC, we considered another class of hardware feasible bio-inspired approach: the Coherent Ising Machines. We proposed an image-denoising application for the first time and obtained good performances in terms of success and error rates. Furthermore, following the similar workflow we proposed for Reservoir computing, we demonstrated the potential for speed and efficiency by employing analog computing with an optoelectronic hardware setup.

Our results for Reservoir Computing and Coherent Ising Machines and their interpretations reveal that our proposals are both environment-friendly and theoretically sound.

As we carried on our work, several potential directions and perspectives became apparent and we will share them below.

In the optic of energy efficiency, parallel information processing has the potential to reduce consumption by a significant amount. In our presented works, artificial neurons were emulated using TDM inefficiently consuming the available optical bandwidth, and energy. However light has the potential for wavelength division multiplexing and multiple polarization which can allow the emulation of a large number of parallel artificial neurons. We aim to design and propose new architectures exploiting multiple degrees of freedom of optics while building on our work on efficiency. With these new architectures, we could increase the speed and efficiency of neuromorphic systems averting further the efficiency wall.

We also observed an increase in stability for lasers subject to delayed optoelectronic feedback produced by weak external optical feedback and published our results [225–227]. In this configuration, the laser is pumped by two sources: a constant current source and a current proportional to the intensity of the delayed electrical field from a distant mirror after an optical-to-electrical conversion. The latter makes the first feedback loop whereas the second loop consists of the part of the light from the same mirror fed back optically into the laser. As Reservoir Computers rely on their memory and computation capacities one could investigate how these properties will be impacted by such a dual setup. Also, the presence of dual feedback can allow for local neuron connectivity by one delay and virtual node generation by the other, therefore, studying such as system in an input and readout synchronous mode of operation could be of interest.

Our experimentation on image denoising with Coherent Ising Machines gave us the know-how necessary to confront even more complex problems such as the Traveling Salesman Problem. The practical importance of the solutions to this problem makes it an interesting avenue for future explorations on our efficient and fast optoelectronic implementations.

# A

# Supplementary material for Chapter 2

## A.1 Linearization of the proposed method in (3.7)

Assuming $C(\boldsymbol{\theta})$ is locally strictly convex close to its optimum, a second order Taylor expansion around $\boldsymbol{\theta}_{opt}$ is obtained as

$$C(\boldsymbol{\theta}) \approx C(\boldsymbol{\theta}_{opt}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta} - \boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}), \tag{A.1}$$

where the Hessian matrix $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ is symmetric positive definite [228], since all second-order partial derivatives were assumed to be continuous. The term $\nu(\boldsymbol{\theta})$ is a noise term accounting for potential noisy loss function evaluations in our method. Replacing $\boldsymbol{\theta}$ alternatively by $\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i$ and $\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i$, we get

$$\begin{aligned}
&C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) \approx C(\boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) + \\
&\frac{1}{2}(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} + h\mathbf{d}_i)^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} + h\mathbf{d}_i), \\
&C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i) \approx C(\boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i) + \\
&\frac{1}{2}(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} - h\mathbf{d}_i)^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} - h\mathbf{d}_i).
\end{aligned} \tag{A.2}$$

It follows that

$$\begin{aligned}
&\frac{C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h} \\
&\quad \approx \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt}) + \frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h}.
\end{aligned} \tag{A.3}$$

Let us define the random noise vector that would be obtained if a gradient approximation were sought along each direction $\mathbf{e}_c$, $c \in \{1, \ldots, p\}$ (instead of a single randomly

chosen $\mathbf{d}_i$ as in the proposed algorithm)

$$\mathbf{n}_i = \begin{bmatrix} \frac{\nu(\boldsymbol{\theta}_{i-1}+h\mathbf{e}_1)-\nu(\boldsymbol{\theta}_{i-1}-h\mathbf{e}_1)}{2h} \\ \vdots \\ \frac{\nu(\boldsymbol{\theta}_{i-1}+h\mathbf{e}_p)-\nu(\boldsymbol{\theta}_{i-1}-h\mathbf{e}_p)}{2h} \end{bmatrix}. \tag{A.4}$$

It follows that the noise term affecting (A.3) can be rewritten as

$$\frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h} = \mathbf{d}_i^T \mathbf{n}_i. \tag{A.5}$$

Now, injecting (A.3) into (3.7) results in (3.8).

## A.2 Proof of the matrix difference equation (3.14)

Starting from (3.9) and recalling that the direction vectors $\{\mathbf{d}_i\}$ are u.i.i.d.

$$\begin{aligned}
&E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T | \boldsymbol{\zeta}_{i-1}\right] \\
&= \frac{1}{p}\sum_{c=1}^{p} \left(\mathbf{I}_p - \mu \mathbf{e}_c\mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \left(\mathbf{I}_p - \mu \mathbf{e}_c\mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)^T \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \\
&\quad - \frac{\mu}{p}(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p} \mathbf{e}_c\mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \mathbf{e}_c\mathbf{e}_c^T,
\end{aligned} \tag{A.6}$$

where we have used the fact that $\sum_{c=1}^{p} \mathbf{e}_c\mathbf{e}_c^T = \mathbf{I}_p$ in the last equality. Applying the law of total expectation,

$$\begin{aligned}
&E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T\right] \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \\
&\quad - \frac{\mu}{p} E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\mathbf{I}_p \circ \left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T\right).
\end{aligned} \tag{A.7}$$

A straightforward application of the rules of vectorization in [142, p. 97-98] leads to the desired result in (3.14).

## A.3 Proof of recursion (3.22)

Starting from (3.10) and recalling that the direction vectors $\{\mathbf{d}_i\}$ are u.i.i.d. and independent from the zero-mean white noise process $\{\mathbf{n}_i\}$

$$
\begin{aligned}
&E\left[\boldsymbol{P}_i\boldsymbol{P}_i^T|\boldsymbol{P}_{i-1}\right] \\
&= \frac{1}{p}\sum_{c=1}^{p}\left(\mathbf{I}_p - \mu\mathbf{e}_c\mathbf{e}_c^T\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\left(\mathbf{I}_p - \mu\mathbf{e}_c\mathbf{e}_c^T\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)^T \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p}\mathbf{e}_c\mathbf{e}_c^T E[\mathbf{n}_i\mathbf{n}_i^T]\mathbf{e}_c\mathbf{e}_c^T \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T \\
&\quad - \frac{\mu}{p}\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p}\mathbf{e}_c\mathbf{e}_c^T\mathbf{HC}(\boldsymbol{\theta}_{opt})\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\mathbf{HC}(\boldsymbol{\theta}_{opt})^T\mathbf{e}_c\mathbf{e}_c^T, \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p}\mathbf{e}_c\mathbf{e}_c^T E[\mathbf{n}_i\mathbf{n}_i^T]\mathbf{e}_c\mathbf{e}_c^T,
\end{aligned}
\tag{A.8}
$$

where we have used the fact that $\sum_{c=1}^{p}\mathbf{e}_c\mathbf{e}_c^T = \mathbf{I}_p$, in the last equality. Applying the law of total expectation,

$$
\begin{aligned}
&E\left[\boldsymbol{P}_i\boldsymbol{P}_i^T\right] \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)E\left[\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\right] \\
&\quad - \frac{\mu}{p}E\left[\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\right]\mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\mathbf{I}_p \circ \left(\mathbf{HC}(\boldsymbol{\theta}_{opt})E\left[\boldsymbol{P}_{i-1}\boldsymbol{P}_{i-1}^T\right]\mathbf{HC}(\boldsymbol{\theta}_{opt})^T\right) \\
&\quad + \frac{\mu^2}{p}\,\mathrm{diag}(\boldsymbol{\Sigma}).
\end{aligned}
\tag{A.9}
$$

A straightforward application of the rules of vectorization in [142, p. 97-98] leads to the desired result in (3.22).

# Supplementary material for Chapter 5

## B.1 Black and white image denoising application

Consider a set of labels $\{x_i\}_{i\in\Omega}$, $x_i \in \{-1, 1\}$ where $\Omega$ is a $2D$ lattice of $L \times C$ pixels. Let $\mathbf{X}$ be the vector of labels obtained from $\{x_i\}_{i\in\Omega}$ after column-wise raster scanning.

We assume a Markov Random Field (MRF) model for $\mathbf{X}$ [224]:

$$\begin{aligned} p(\mathbf{X}) &= \frac{1}{Z(\beta)} \exp\Big(\beta \sum_{\substack{(n,m)\\ neighbors}} \delta(x_m - x_n)\Big) \\ &= \frac{1}{Z(\beta)} \exp\Big(\beta \sum_{\substack{(n,m)\\ neighbors}} \frac{1 + x_m x_n}{2}\Big) \\ &= \frac{1}{Z'(\beta)} \exp\Big(\frac{\beta}{2} \sum_{\substack{(n,m)\\ neighbors}} x_m x_n\Big) \\ &= \frac{1}{Z'(\beta)} \exp\Big(\frac{1}{2} \sum_{i<j} J_{i,j} x_i x_j\Big) \end{aligned} \tag{B.1}$$

where the matrix $\mathbf{J} = [J_{i,j}]_{\substack{1\le i\le L\\ 1\le j\le C}}$ is defined by:

$$J_{i,j} = \begin{cases} \beta & \text{if pixels indexed by i and j are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

Let $\{y_i\}_{i\in\Omega}$ be the observed image corrupted by salt and pepper noise with probability $p$ independently at random so that:

$$y_i = \begin{cases} x_i & \text{with probability } 1-p \\ -x_i & \text{with probability } p \end{cases} \tag{B.3}$$

That is:

$\Rightarrow$ the number of flipped pixels is $\sum_{i\in\Omega}\frac{1-x_iy_i}{2}$
$\Rightarrow$ the number of preserved pixels is $L\times C-\sum_{i\in\Omega}\frac{1-x_iy_i}{2}$
By the noise independence assumption:

$$\begin{aligned}
p(\mathbf{Y}/\mathbf{X}) &= \prod_{i\in\Omega} p(y_i/x_i) \\
&= (1-p)^{L\times C-\sum_{i\in\Omega}\frac{1-x_iy_i}{2}} \times p^{\sum_{i\in\Omega}\frac{1-x_iy_i}{2}} \\
\ln(p(\mathbf{Y}/\mathbf{X})) &= \left(L\times C-\sum_{i\in\Omega}\frac{1-x_iy_i}{2}\right)\ \ln(1-p) \\
&+ \left(\sum_{i\in\Omega}\frac{1-x_iy_i}{2}\right)\ \ln(p) \\
&= C^{+}-\frac{1}{2}\ln\left(\frac{p}{1-p}\right)\sum_{i\in\Omega}x_iy_i \\
&= \sum_{i\in\Omega} b_iy_i
\end{aligned} \tag{B.4}$$

where $C^{+}$ is a constant and:

$$b_i=-\frac{1}{2}\ln\left(\frac{p}{1-p}\right)y_i \tag{B.5}$$

By Bayes' theorem it follows that the posterior distribution of $\mathbf{X}$ can be written as:

$$\begin{aligned}
p(\mathbf{X}/\mathbf{Y}) &\propto p(\mathbf{Y}/\mathbf{X})p(\mathbf{X}) \\
&\propto \exp\left(\frac{1}{2}\sum_{i<j}J_{i,j}x_ix_j+\sum_{i\in\Omega}b_ix_i\right)\ \propto \exp(-E(\mathbf{X}))
\end{aligned} \tag{B.6}$$

where:

$$E(\mathbf{X})=-\frac{1}{2}\sum_{i<j}J_{i,j}x_ix_j-\sum_{i\in\Omega}b_ix_i \tag{B.7}$$

has the form of a classical Ising Hamiltonian. Thus finding the restored image $\mathbf{X}$ from the noisy image $\mathbf{Y}$ in the maximum a posteriori sense is equivalent to minimizing the energy function $E(\mathbf{X})$.

# C

# Introduction (Version française)

## C.1 Introduction

Aujourd'hui, une quantité considérable d'efforts est consacrée à la recherche et au développement dans le domaine des Réseaux de Neurones Artificiels (RNA). Parmi les différentes voies de l'IA, les RNA représentent la solution neuromorphique la plus recherchée dans de nombreux domaines d'application et sont au centre de notre attention. Motivé par la nécessité d'une intelligence artificielle respectueuse de l'environnement, dans cette thèse, nous étudions deux systèmes neuromorphiques récurrents appelés Reservoir Computing (RC) et machine de Ising cohérente (Coherent Ising Machine, CIM) qui présentent un potentiel de faisabilité matérielle.

### C.1.1 Le modèle Echo State Network de RC

Le Reservoir Computing (RC) est un cadre puissant de réseau de neurones récurrents (RNN) pour le traitement de données séquentielles. Il simplifie le processus d'entraînement tout en maintenant une grande puissance de calcul. La motivation derrière le RC vient des difficultés rencontrées dans l'entraînement des réseaux de neurones traditionnels, notamment les réseaux de neurones récurrents. Le RC offre une approche alternative en limitant l'entraînement à la couche externe et en générant aléatoirement la plupart des neurones dans la couche de réservoir. Cela simplifie considérablement l'entraînement sans compromettre les performances. De plus, le caractère aléatoire du réservoir ouvre la voie à de nombreuses implémentations matérielles.

L'architecture fondamentale d'un RC est constituée de trois couches distinctes, à savoir la couche d'entrée, la couche de réservoir et la couche de sortie. La couche d'entrée formate et injecte les signaux qui seront traités dans la couche de réservoir, qui est essentiellement une collection d'unités de traitement connectées aléatoirement qui étendent de manière non linéaire les signaux d'entrée vers un espace de dimension

supérieure. La couche de sortie prend les états des nœuds de la couche de réservoir, qui sont synonymes de neurones dans ce contexte, et les utilise pour générer des prédictions pour la sortie cible. L'architecture et la fonctionnalité de ces trois couches peuvent être observées dans la Figure C.1. Le modèle de base d'un RC exploite la propriété selon laquelle le vecteur d'états de nœuds dans un RNN, noté $\boldsymbol{x}(n)$, à un temps spécifique $n$ peut être exprimé comme une fonction de l'historique d'entrée (ou "écho"), $\boldsymbol{u}(n)$, $\boldsymbol{u}(n-1)$, etc. Les états de nœuds dans la couche de réservoir, en revanche, dépendent à la fois des états précédents et de l'entrée actuelle, comme l'indique l'équation dynamique non linéaire en temps discret:

$$\mathbf{x}(n) = f_{NL}(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}^{res}\mathbf{x}(n-1)), \tag{2.1}$$

où $n$ est une variable de temps discret, $\boldsymbol{x}$ est un vecteur représentant les états des nœuds, $\boldsymbol{u}$ est un vecteur représentant le signal d'activation d'entrée, $\boldsymbol{W}^{in}$ est une matrice de poids d'entrée aléatoires également appelée masque d'entrée, $\boldsymbol{W}^{res}$ est une matrice de connectivité aléatoire pour le réservoir, et $f_{NL}$ est la fonction d'activation non linéaire. $F$ représente la taille du vecteur d'entrée $\boldsymbol{u}$, $N$ représente la taille de la couche de réservoir (c'est-à-dire la longueur du vecteur $\boldsymbol{x}$), et $C$ représente la taille de la sortie. Les états des nœuds peuvent être utilisés pour générer la sortie $\boldsymbol{y}$ en utilisant l'équation:

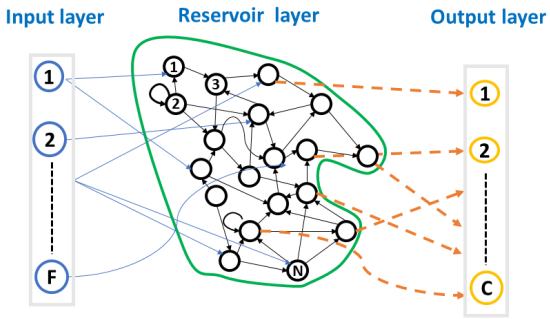



Figure C.1: Architecture spatio-temporelle de base d'un RC montrant les trois couches.

$$\hat{\mathbf{y}}(n) = f^{out}(\mathbf{W}^{out}[1; u(n); \mathbf{x}(n)]). \tag{2.3}$$

Ici, $\boldsymbol{W}^{out}$ est une matrice de lecture et $\hat{\boldsymbol{y}}(n)$ représente le vecteur de sortie estimé à l'étape de temps $n$, qui est ensuite comparé au vecteur de sortie attendu $\boldsymbol{y}(n)$ dans le cas de l'apprentissage supervisé. Le processus d'apprentissage consiste à calculer $\boldsymbol{W}^{out}$ de manière à éliminer la récurrence. Sauf indication contraire, la fonction de sortie ($f^{out}$) sera supposée être une fonction identité dans les discussions ultérieures.

### C.1.2 Machine de Ising cohérente

Nous considérons également une classe de systèmes neuromorphiques qui évolueront d'une façon autonome vers l'état d'énergie le plus bas possible sans être pilotés par un signal d'entrée externe. Nous nous intéressons particulièrement aux implementation optoéléctroniques des machines de Ising cohérentes (CIMs). Ce sont des réseaux neuronaux de recherche d'extrema adaptés aux problèmes d'optimisation combinatoire. Les CIM imitent le comportement ferromagnétique, c'est-à-dire l'alignement dépendant de la température de la polarisation des moments magnétiques (spins) des atomes constitutifs dans la même direction au niveau microscopique. L'énergie totale du système à tout moment dépend de l'alignement des spins.

L'objectif est de formuler le problème donné de manière à ce que la solution du problème coïncide avec l'état fondamental de l'hamiltonien de Ising du système, défini comme suit :

$$E(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{(s,t)\in\mathcal{N}} J_{s,t}\sigma_s\sigma_t - \sum_{s\in\Omega} b_s\sigma_s, \tag{1.4}$$

où le problème est défini sur une grille carrée de taille $n \times n$. Tout $s = (l, c) \in \Omega$ peut être assimilé à une position dans la grille avec une coordonnée de ligne (resp. de colonne) $l$ (resp. $c$), où $1 \leq l, c \leq n$. Pour tout $s \in \Omega$, nous posons le spin $\sigma_s$ comme une valeur aléatoire dans $\{-1, +1\}$, $\boldsymbol{J}$ est la matrice qui régit l'interaction entre les spins et $\mathcal{N}$ désigne tous les couples de nœuds voisins avec des conditions de bord périodiques.

Suivant [34] afin de minimiser l'équation 1.4, nous implémentons numériquement un réseau de Hopfield généralisé, dont l'équation aux différences en temps discret à l'instant $k$ a la forme :

$$\begin{aligned} x_s(k) &= f\left(\alpha x_s(k-1) + \beta\left(\sum_{t:(s,t)\in\mathcal{N}} J_{s,t}x_t(k-1) + b_s\right)\right) \\ \hat{\sigma}_s(k) &= \text{sign}(x_s(k)), \quad \forall s \in \Omega \end{aligned} \tag{1.5}$$

où $f(.)$ est une fonction d'activation non linéaire, $\alpha$ et $\beta$ sont des coefficients d'échelle qui contrôlent le couplage autonome et la force de rétroaction affectant la sortie du neurone $x_s(k)$, tandis que $\hat{\sigma}_s(k)$ est l'estimation du spin du pixel $s$.

Nous étudions la capacité du CIM à trouver des solutions à plusieurs problèmes tels que le modèle de Ising antiferromagnétique et la restoration d'images à au sens de MAP. Nous évaluons les performances du système en mesurant le taux/probabilité de réussite du système à évoluer suffisamment près de l'optimum, c'est-à-dire à une énergie inférieure à un certain seuil.

## C.2 Optimisation des paramètres pour RC

La puissance et la polyvalence du Reservoir Computing proviennent de la possibilité de choisir aléatoirement la plupart des paramètres du réservoir, tels que les éléments du masque d'entrée et les matrices de connectivité. Cela évite la contrainte stricte d'ajuster un grand nombre de paramètres du système récurrent, contrairement aux RNN traditionnels. Toutefois, les réservoirs générés entièrement de manière aléatoire donnent rarement des performances optimales [42]. Certains paramètres, bien moins nombreux que dans les RNN classiques, nécessitent un réglage pour obtenir les meilleures performances pour une tâche définie.

Nous proposons une technique d'optimisation pour régler un vecteur $p$-dimensionnel de paramètres du RC basée exclusivement sur des évaluations bruitées de la fonction de perte. Notre algorithme trouve un compromis entre la convergence rapide des méthodes déterministes basées sur les gradients et l'efficacité des approximations stochastiques. Cela est utile dans les contextes où la relation fonctionnelle exacte entre les paramètres et les valeurs de la fonction de perte n'est pas disponible, comme c'est le cas pour les implémentations matérielles du RC. Des méthodes similaires, s'appuyant sur l'algorithme de Robbins-Monro pour la recherche de racines [136] et des procédures visant à trouver l'optimum d'une fonction de perte basée sur des approximations de gradient stochastiques par différences finies, ont été proposées. Par exemple, l'algorithme original de Kiefer-Wolfowitz [137] nécessite $2p$ évaluations bruitées de la fonction du perte.

Nous simplifions les méthodes mentionnées précédemment de manière à étendre leur applicabilité de plusieurs manières:

- Générer des perturbations sous forme de vecteurs de direction aléatoires standard dans l'espace des hyperparamètres.
- Garder la taille du pas de différences finies constante.
- Garder la taille du pas d'apprentissage constante.

Ces modifications sont essentielles pour deux raisons : premièrement, elles sont responsables de la faible complexité de notre approche, et deuxièmement, elles augmentent la robustesse numérique dans le cas d'un RC matériel. En effet, le calcul de la fonction de perte, perturbé par le bruit, ainsi que les variations lentes du dispositif, nécessitent la capacité de suivre les dérives des paramètres. Nous effectuons une analyse de convergence locale basée sur des approximations quadratiques de la fonction de perte. De plus, nous démontrons une optimisation emperique des hyperparamètres à la fois dans le cadre de simulations numériques et d'expérimentations sur le RC matériel.

### C.2.1 La méthode proposée (SGD)

Nous introduisons deux paramètres ($\alpha$ et $\beta$) et réécrivons l'Equation d'état discrète 2.1 comme suit :

$$\mathbf{x}(n) = f_{NL}(\alpha \mathbf{W}^{in}\mathbf{u}(n) + \beta \mathbf{W}\mathbf{x}(n-1)), \tag{3.1}$$

ou $f_{NL}$ représente la non-linéarité par composante. $\mathbf{W}^{in} \in \mathbb{R}^{N\times F}$ et $\mathbf{W} \in \mathbb{R}^{N\times N}$ sont des matrices aléatoires creuses. $\alpha$ and $\beta$ sont des facteurs d'échelle d'entrée et de rétroaction qui influencent la dynamique du système. $\boldsymbol{\theta}$ est le vecteur regroupant les hyperparamètres. La couche de sortie linéaire génère $\hat{\mathbf{y}}(n)$ dans $\mathbb{R}^C$ où C est le nombre de classes.

Basé sur un ensemble de données d'entraînement $\{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^{T}$, apprendre un modèle RC consiste à minimiser la fonction de perte [46]:

$$L(\boldsymbol{\theta}, \mathbf{W}^{out}) = ||\mathbf{y}_{train}(1:T) - \mathbf{W}^{out}\mathbf{x}(1:T)||_F^2 + \lambda||\mathbf{W}^{out}||_F^2 \tag{3.2}$$

où la régularisation de Tikhonov avec le paramètre $\lambda$ est utilisée pour limiter le bruit et le surajustement [42]. La complexité computationnelle d'une évaluation de la fonction de perte est $\mathcal{O}(CT(N+2)+CN)$. Ce problème d'apprentissage consiste à résoudre l'équation suivante:

$$\frac{\partial L(\boldsymbol{\theta}, \mathbf{W}^{out})}{\partial \mathbf{W}^{out}} = \mathbf{0} \tag{3.3}$$

et a une solution unique sous forme analytique dépendant des hyperparamètres:

$$\hat{\mathbf{W}}^{out}\left(\boldsymbol{\theta}, \{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^{T}\right) = \mathbf{y}_{train}(1:T)\mathbf{x}(1:T)^T\left[\mathbf{x}(1:T)\mathbf{x}(1:T)^T + \lambda \mathbf{I}_N\right]^{-1} \tag{3.4}$$

dont la complexité computationnelle est de $\mathcal{O}(CN(N+T)+N^3)$ par vecteur d'hyperparamètres et ensemble d'entraînement.

Nous introduisons la méthode d'approximation stochastique proposée pour estimer un vecteur de paramètres $\boldsymbol{\theta} \in \mathbb{R}^p$ afin de minimiser la fonction de perte. Chaque fois que nous évaluons le RC sur un ensemble de données d'entraînement limité avec de nouveaux hyperparamètres, nous obtenons une valeur bruitée de la fonction de perte. Étant donné que les approximations stochastiques peuvent résoudre de manière itérative des problèmes d'optimisation en n'utilisant que des évaluations de fonction bruitées, il est logique de minimiser la fonction de perte en utilisant de telles méthodes.

Soit $\mathbf{e}_c$ le vecteur unitaire standard dans la direction de la $c$-ième coordonnée (c'est-à-dire que $\mathbf{e}_c$ est un vecteur de longueur $p$ contenant 1 dans sa $c$-ième coordonnée et 0 ailleurs), pour $c = 1, \ldots, p$. À partir d'une estimation initiale $\boldsymbol{\theta}_0$, les vecteurs de direction $\{\mathbf{d}_i\}$ sont distribués uniformément, indépendamment et de manière identique

(u.i.i.d.) parmi $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}$, où $i$ est l'indice d'itération. La descente de gradient, couramment utilisée pour l'optimisation en apprentissage automatique [141], n'est pas réalisable dans notre cas car le calcul du gradient nécessite une connaissance complète de la relation fonctionnelle entre $\boldsymbol{\theta}$ et $L(.,.)$ qui est supposée indisponible. Au lieu de cela, une approximation stochastique du gradient de la fonction de perte dans la direction aléatoire $\mathbf{d}_i$ est obtenue comme suit :

$$\Delta_i = \frac{L(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i, \mathbf{W}_{i-1}^{out}) - L(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i, \mathbf{W}_{i-1}^{out})}{2h}, \tag{3.5}$$

où une approche de différences finies avec un pas constant $h$ est utilisée. Notez qu'en plus de l'erreur inhérente à la méthode des différences finies, le numérateur dans l'Equation 3.5 implique deux évaluations d'une fonction de perte du RC en utilisant un nombre limité de données d'entraînement, ce qui donne lieu à des évaluations bruitées. Pour simplifier la procédure itérative de minimisation de la fonction de perte, chaque composante de $\boldsymbol{\theta}$ est mise à jour individuellement selon l'équation suivante :

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu\mathbf{d}_i\Delta_i, \tag{3.6}$$

où $\mu$ est une taille de pas d'apprentissage constante. La procédure d'optimisation des paramètres proposée est résumée dans l'algorithme 1.

**Algorithm 2** Procédure d'optimisation des paramètres

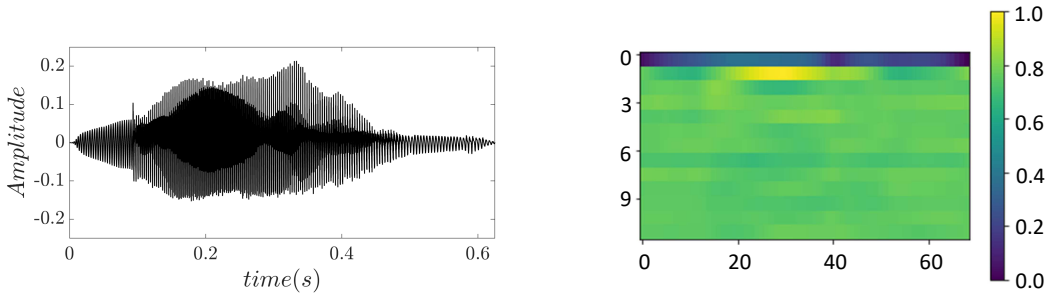
**Require:** $\mu, h, \boldsymbol{\theta}_0$
  Initialiser $\mathbf{W}_0^{out}$ en minimisant $L(\boldsymbol{\theta}_0, \mathbf{W}^{out})$
  **for** $i = 1, 2, \ldots,$ **do**
    Sélectionner aléatoirement une direction $c \in \{1, 2, \ldots, p\}$
    Définir le vecteur de direction $\mathbf{d}_i = \mathbf{e}_c$
    Calculer $\Delta_i$ selon l'Equation 3.5
    Mettre a jour les parametres: $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu\mathbf{d}_i\Delta_i$
    Mettre à jour $\mathbf{W}_i^{out}$ en minimisant $L(\boldsymbol{\theta}_i, \mathbf{W}^{out})$
  **end for**

### C.2.2 Les données et leur prétraitement

Nous évaluons les performances de l'algorithme d'optimisation d'hyperparamètres proposé à l'aide d'implémentations en simulation (Section C.2.1) et en matériel. En plus des tâches de classification pratiques, telles que la reconnaissance de chiffres parlés, nous introduisons deux nouvelles tâches dans la littérature sur l'apprentissage automa-

tique : la classification des diagrammes de contrôle et des anomalies dans la fabrication des wafers. Nos résultats et conclusions pour la tâche de reconnaissance des chiffres parlés sont cohérents avec les deux tâches. Par conséquent, dans ce résumé, dans un souci de concision, nous ne présenterons que les resultats la tâche de reconnaissance des chiffres parlés.

La classification des chiffres prononcés est couramment utilisée comme référence en recherche sur les RCs. Nous avons utilisé le jeu de données TIDIGITS LDC93S10 [146], qui comprend des enregistrements réalistes avec du bruit, des distorsions et des variations de locuteurs. Il compte 326 locuteurs enregistrant les chiffres de 0 à 9 en anglais. Nous avons sélectionné 1500 fichiers audio répartis également selon le sexe et l'âge. Les fichiers ont été tronqués à une longueur de 10k échantillons (voir Figure C.2a), permettant des longueurs d'entrée fixe. Un prétraitement a été appliqué pour extraire les coefficients MFCC [147] (voir Figure ).




(a) (b)

Figure C.2: (a) Un profil sonore correspondant à la prononciation du chiffre 5 avec une représentation visuelle des coefficients MFCC dans (b).

### C.2.3 Résultats expérimentaux

En plus de la fonction de perte dans l'Equation 3.2, nous utilisons le Taux d'Erreur de Classification (TEC) pour évaluer les performances du modèle. Le TEC mesure l'étendue des mauvaises classifications en pourcentage $(\%)$ et est calculé comme suit :

$$\text{TEC}\ (\%) = \frac{\text{Nombre d'entrées mal classées}}{\text{Cardinalité de l'ensemble de test}} \times 100. \tag{3.25}$$

Dans le cas particulier de la tâche des chiffres parlés, le TEC est également couramment appelé Taux d'Erreur de Mots (TEM), car les entrées sont classées comme des mots.

La Table C.1 présente les mesures de performance des trois méthodes après optimisation RC pour la tâche de reconnaissance de chiffres parlés. Il est important de mentionner que la méthode proposée et SAN obtiennent des TEMs similaires après conver-

| | ***Recherche en grille*** | ***SAN*** | ***SGD*** |
|---|---|---|---|
| $\alpha$ | 0.06 | 0.05 | 0.05 |
| $\beta$ | 0.13 | 0.13 | 0.13 |
| **TEM** (%) | 2.10 | 1.88 | 1.80 |
| **Nombre d'étapes** ($S$) | 15000 | 3400 | 250 |

Table C.1: Les résultats de l'optimisation montrant les hyperparamètres, le TEM et le nombre d'étapes $S$ pour la tâche des chiffres parlés.

gence dans cette application. Nous pourrions améliorer le TEM obtenu par recherche en grille après optimisation, mais cela se ferait au prix d'une complexité accrue en utilisant une grille de valeurs plus fine. De plus, nous observons que la complexité algorithmique, qui est essentiellement proportionnelle au nombre d'étapes $S$ défini précédemment, montre que l'algorithme proposé surpasse les deux méthodes concurrentes d'au moins un facteur de $13,6$ en termes d'efficacité algorithmique.

Pour la tâche des diagrammes de contrôle nous faisons une observation similaire à celle de la tâche des chiffres parlés. Pour atteindre une convergence d'environ $2\%$ de taux d'erreur, SAN nécessite $3550$ étapes, tandis que notre méthode nécessite $1300$ étapes. Cela correspond à une réduction de la complexité de notre méthode par un facteur de $2,73$. Egalement, pour la tâche de classification des wafers, un taux d'erreur d'environ $2\%$ est obtenu avec SAN après $4200$ étapes, tandis que notre méthode nécessite $1200$ étapes. Cela correspond à une réduction de complexité de notre méthode par un facteur de $3,5$.

## C.3 RC pour la détection de la maladie d'Alzheimer

Nous proposons une méthodologie d'évaluation du coût énergétique sur l'exemple de la détection de la maladie d'Alzheimer à une stage précoce. La maladie d'Alzheimer est une maladie du cerveau qui entraîne une destruction lente mais progressive de la mémoire et une détérioration des compétences comportementales et sociales. Cela résulte de la destruction progressive des cellules nerveuses dans différentes parties du cerveau. La maladie fait partie de la démence, un terme général décrivant les symptômes associés au déclin des capacités de réflexion, d'apprentissage et de mémorisation d'un individu. La maladie d'Alzheimer perturbe le travail et la vie sociale des patients, les rendant incapables d'accomplir des tâches quotidiennes. Elle affecte le lobe frontal, la région du cerveau responsable des fonctions exécutives telles que le comportement contrôlé et la planification volontaire des mouvements [164]. Par conséquent, les mal-

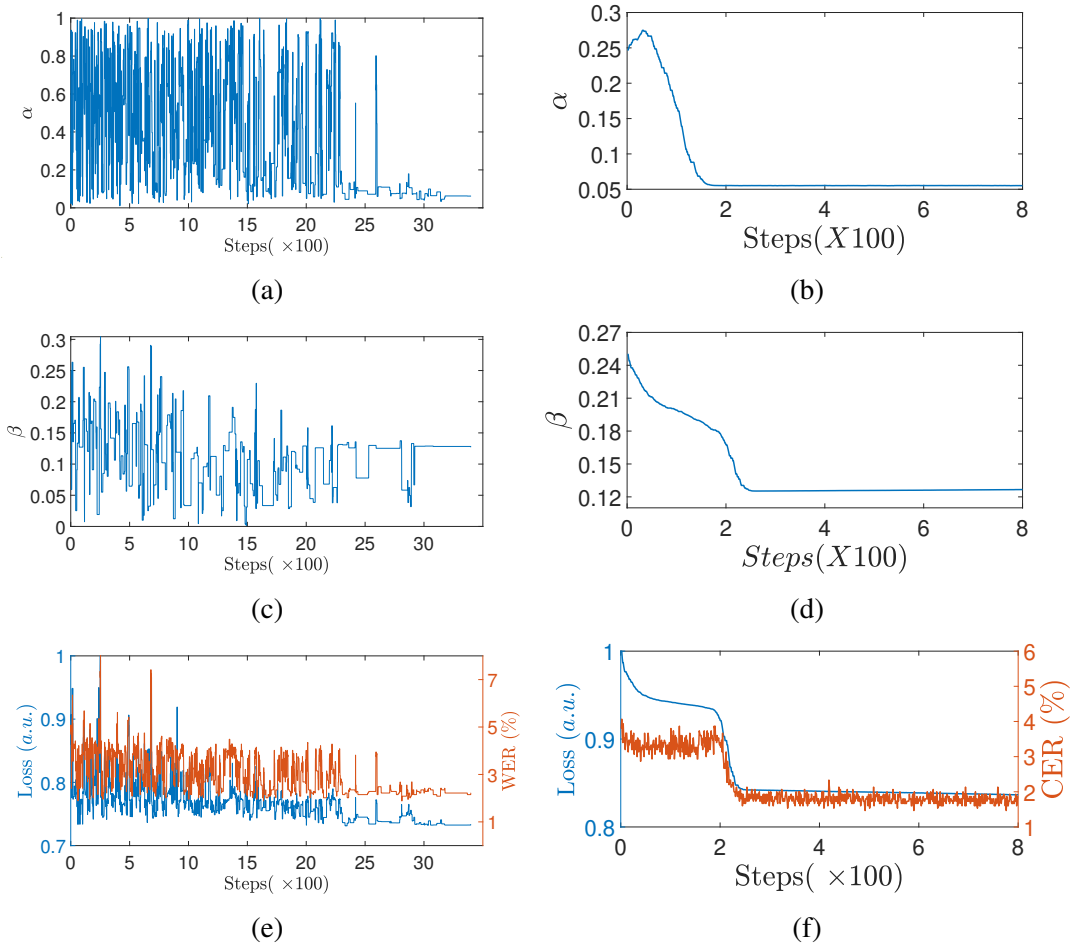



Figure C.3: Les graphiques pour SAN (gauche) et SGD (droite) pour le RC simulé sont présentés ci-dessous. (a) et (b) pour l'évolution de la mise à l'échelle de l'entrée ($\alpha$), (c) et (d) pour l'évolution de la mise à l'échelle de la rétroaction ($\beta$), et (e) et (f) montrent l'évolution de la fonction du coût et du TEM, respectivement, pour la tâche de reconnaissance de chiffres parlés.

adies neurodégénératives comme la maladie d'Alzheimer peuvent être caractérisées en observant leur impact sur les mouvements.

L'étude [165] montre que la pathologie peut être caractérisée en analysant le contrôle moteur fin et la coordination des patients. L'écriture est une tâche qui nécessite de la coordination et un contrôle moteur fin, alors l'impact de la maladie se manifeste dans l'écriture manuscrite des patients. L'écriture manuscrite est donc un biomarqueur important permettant de distinguer certaines pathologies cérébrales [165–177].

Nous analyserons quantitativement les avantages de l'utilisation de RC sur des tâches de classification de l'écriture manuscrite par rapport aux modèles plus complexes, BiLSTM et CNN, en utilisant les données de l'écriture manuscrite pour la detection de la maladie d'Alzheimer (Early-stage Alzheimer's disease detection, ES-AD). Nous évaluerons leurs performances en utilisant l'exactitude des prédictions et étendrons la com-

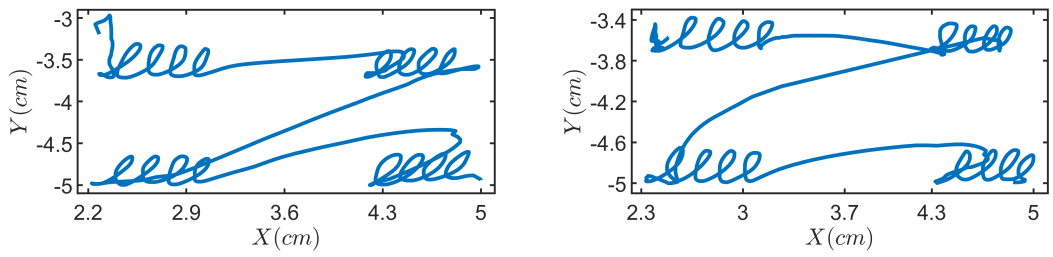
paraison pour inclure des estimations de la consommation d'énergie. L'objectif est de proposer une solution non seulement précise, mais aussi qui présente la meilleure efficacité énergétique.

### C.3.1 Les données

Les données de l'écriture manuscrite (Handwritting, HW) de cette étude ont été acquises à l'Hôpital Broca à Paris par les auteurs de l'article [161] et ont été traitées dans de nombreuses autres études [171, 172, 187]. Elles se composent de trois groupes de participants, tous âgés de plus de 60 ans, à savoir :

- Les témoins en bonne santé (Healthy controls, HC) : Ce groupe est composé de membres soigneusement sélectionnés après des examens neuropsychologiques. Leur diagnostic a montré des profils cognitifs normaux sans signes de la maladie d'Alzheimer aux stades précoce ou avancé.
- Les troubles cognitifs légers (Mild cognitive Impairement, MCI) : Il s'agit du groupe sélectionné sur la base du diagnostic selon les recommandations de l'article [188], en effectuant des tests cognitifs généraux et des tests non-mnémoniques.
- Les patients atteints d'Alzheimer à un stade précoce (AD) : Ce groupe est composé de patients ayant les signes précoces de la maladies.

Les participants ont été invités à écrire quatre ensembles de $\ell\ell\ell\ell$ en cursif pour former un motif présenté dans la Figure C.4. Dans cette thèse, seuls deux groupes parmi les trois mentionnés précédemment seront pris en compte pour la classification, le groupe des personnes atteintes de la maladie d'Alzheimer à un stade précoce et le groupe des témoins en bonne santé.




(a) (b)

Figure C.4: (a) et (b) sont des ensembles de $\ell$ tracés par un participant confirmé être respectivement à un stade précoce de la maladie d'Alzheimer et du groupe de contrôle.

Les deux groupes d'intérêt comprennent uniquement 54 participants. Pour entraîner correctement un modèle de réseau de neurones artificiels, il est nécessaire d'avoir un

nombre suffisant d'exemples pour entraîner le modèle afin d'obtenir des performances acceptables. Heureusement, chacun des 54 participants a dessiné 16 lettres $\ell$-cursives distinctes, qui peuvent être considérées comme des données séparées [161].

### C.3.2 L'efficacité computationnelle et énergétique

Conformément à notre discussion sur l'impact environnemental de l'IA dans l'introduction, nous évitons de tirer la conclusion simpliste qu'un modèle est le meilleur modèle uniquement en se basant sur la précision des prédictions. Nous étendons nos études pour inclure la complexité et l'effort de calcul nécessaires pour obtenir les performances rapportées. À cette fin, il existe plusieurs approches pour estimer le coût engendré par l'exécution d'algorithmes sur des processeurs électroniques:

- **L'énergie en kWh** nécessaire pour alimenter le système pendant l'exécution. On estime uniquement la consommation d'énergie par les DRAM, les CPUs et les GPUs en ignorant tous les autres composants.
- **Le dioxyde de carbone ($CO_2$) et les équivalents en gaz à effet de serre (GES) en utilisant l'unité : ($CO_{2eq}$)**. La quantité de ces gaz rejetés dans l'atmosphère dépend de la nature des sources d'électricité fournies au réseau.
- **Opérations en virgule flottante et temps d'exécution** : Pour estimer le coût relatif de l'entraînement et de l'inférence, des tâches qui prennent beaucoup moins de temps, nous utilisons le nombre d'opérations en virgule flottante (Floating Point Operations FPO), la durée d'exécution et le nombre de paramètres entraînables des modèles. Pour compter les FPO effectuées par les processeurs tout au long de la tâche, nous utilisons un outil appelé PAPI [196].
- **Pour la mise en œuvre matérielle du RC**, nous calculons la somme des puissances consommées par les différents composants de l'installation pour estimer la consommation d'énergie lors de l'exécution de l'expérience sur le matériel. L'estimation fournie est une estimation moyenne approximative de la puissance consommée lors d'exécutions répétées, mais elle donne un aperçu des ordres de grandeur des coûts.

### C.3.3 Expériences de classification

L'objectif est de diagnostiquer l'état du patient en tant que malade (AD) ou non (HC). Nous définissons cela comme un problème de classification et proposons des modèles qui prédiront l'état du patient en fonction de leur ensemble de boucles écrites.

Nous implementons trois modèles pour cette tâche : le BiLSTM, le CNN et le modèle RC. Le modèle RC est implémenté à la fois en simulation informatique et en version optoélectronique matérielle. Nous réimplémentons également l'approche des auteurs [172] basée sur le regroupement par $k$-Medoids utilisant la Dynamic Time Warping (DTW) comme mesure de dissimilarité, suivie du classifieur bayésien pour agréger la contribution des groupes et effectuer des prédictions. Pour évaluer les performances de classification dans nos expériences, nous utilisons les métriques suivantes la précision (**Pre**: pourcentage global d'individus correctement classés), la sensibilité (**Sens** : Le pourcentage de patients Alzheimer correctement classés) et la spécificité (**Spec** : Le pourcentage de personnes saines correctement classées).

### Résultats sur la précision de classification

Pour chacun des modèles étudiés, nous optimisons et sélectionnons le modèle final en utilisant la NCV décrite ci-dessus selon une recherche exhaustive (voir Table C.2). Nous constatons que les modèles récurrents traitant l'entrée comme une série temporelle, c'est-à-dire BiLSTM et RC, surpassent les méthodes statiques (CNN et k-Medoids). Le BiLSTM atteint la plus haute précision, ce qui conduit à la conclusion qu'il est le modèle le plus sensible et le plus précis parmi les modèles étudiés. Avec une précision de $88\%$, il dépasse RC et CNN respectivement de $3\%$ et $5\%$. CNN est le moins précis des trois modèles étudiés, avec une précision de $83\%$, et il offre également la sensibilité la plus faible à la pathologie.

| **Méthode** | **Pre** (%) | **Sens** (%) | **Spec** (%) |
|---|---|---|---|
| BiLSTM | 88 | 96 | 79 |
| Digital RC | 85 | 88 | 83 |
| CNN | 83 | 75 | 91 |
| k-Medoids [172] | 74 | 75.6 | 72.2 |

Table C.2: Une comparaison des métriques de performance pour tous les modèles.

### Résultats sur les coûts d'entraînement, de sélection du modèle et d'inférence

Nous estimons les coûts d'optimisation et de sélection du modèle final pour le RC numérique et le BiLSTM utilisant le framework *experiment-impact-tracker*, car ce sont les deux modèles que nous avons conçus à partir de zéro et ce sont les deux meilleures méthodes en termes de précision dans la Table C.2. Les résultats de la consommation et de l'impact environnemental sont résumés dans la Table C.3. L'énergie électrique consommée est exprimée en kilowatt-heure (*kWh*) et la masse correspondante de $CO_2$ rejetée par nos expériences d'optimisation est exprimée en kilogrammes de

$CO_2$ et d'équivalent GES. Pour explorer et sélectionner les paramètres optimaux pour l'approche BiLSTM, il a fallu utiliser $9.312$ *kWh* d'énergie électrique. L'approche RC a quant à elle nécessité seulement $1.156$ *kWh*. Par conséquent, le modèle BiLSTM coûte plus de $8$ fois plus cher en électricité que l'approche RC numérique et prend 44 *heures* (comparé à 21 *heures*) pour terminer l'optimisation sur notre processeur Intel Xeon $E5-1603$. Ces résultats sont basés sur les informations du réseau électrique de la ville de *Palaiseau* en France où l'expérience a été réalisée. Par conséquent, un total de $0.586$ *kg* de $CO_2$ et d'équivalent GES est rejeté dans l'atmosphère par nos expériences. Dont $0.521$ *kg* proviennent de l'optimisation du BiLSTM, tandis que le RC a contribué à hauteur de $0.065$ *kg* dans l'atmosphère. En conséquence, l'optimisation du RC a un impact environnemental relativement plus faible que le BiLSTM en raison des émissions moins élevées, comme le montre également la Table C.3.

| **Méthodee** | **$CO_{2eq}$** (kg) | **Énergie** $(kWh)$ | **Durée** (heures) |
|---|---|---|---|
| BiLSTM | $0.521$ | $9.312$ | $44$ |
| Digital RC | $0.065$ | $1.156$ | $21$ |

Table C.3: La consommation énergétique pour l'optimisation des hyperparamètres et la sélection du modèle final pour le RC et le BiLSTM.

Les résultats de consommation d'énergie et d'émissions présentés précédemment incluent un grand nombre de cycles d'entraînement et de tests. Cependant, nous souhaitons maintenant examiner les coûts de l'entraînement seul, de manière relative entre les modèles fonctionnant sur le même processeur. Malheureusement, *experiment-impact-tracker* n'a pas pu être utilisé pour estimer directement l'énergie et les émissions pendant l'entraînement, car les temps d'entraînement étaient courts pour les modèles considérés. Néanmoins, nous calculons le nombre de paramètres et les opérations en virgule flottante à l'aide de *PAPI* pour avoir un aperçu des coûts relatifs de l'entraînement de chaque modèle. Les résultats de cette expérience sont présentés dans la Table C.4.

Dans cette expérience, le CNN avait le plus grand nombre de paramètres et nécessitait le plus d'efforts de calcul. Par rapport au RC, le CNN a $1947$ fois plus de paramètres nécessitant une optimisation par l'entraînement. L'optimisation a nécessité plus de $110$ fois les opérations en virgule flottante requises par le RC pour atteindre l'optimalité. $k$-Medoids est la deuxième méthode la plus coûteuse malgré sa moindre précision. L'entraînement a nécessité environ $7$ fois plus d'efforts de calcul par rapport au RC, avec environ $1250$ fois plus de paramètres nécessitant une optimisation. Les deux méthodes les moins coûteuses sont également les deux plus performantes pour la tâche. Le BiLSTM, le modèle le plus performant, a dépassé le RC de $3\%$ en termes de précision de prédiction. Toutefois, cette amélioration des performances a été

obtenue en optimisant plus de $687$ fois le nombre de paramètres du RC. L'effort de calcul en termes d'opérations en virgule flottante pour l'optimisation des paramètres du RC représente seulement $63\%$ de l'effort requis par le BiLSTM, ce qui implique des économies de $37\%$.

| **Méthode** | **Paramètres entraînables** | **Milliards de FPOs** |
|---|---|---|
| CNN | 1168417 | 1313.09 |
| k-Medoids | 750016 | 83.98 |
| BiLSTM | 412289 | 18.93 |
| Digital RC | 600 | 11.92 |

Table C.4: Estimations du nombre des FPOs nécessaires pour l'entraînement.

Le modèle final entraîné est maintenant prêt à être déployé pour servir de modèle prédictif. Cela signifie que le modèle sera alimenté en continu avec des entrées provenant de nombreux nouveaux patients afin de fournir un diagnostic. Un bon modèle pour ce cas devrait avoir des coûts réduits lors d'une utilisation répétée tout en maintenant une précision raisonnable. Pour mesurer cela, nous analysons le nombre d'opérations en virgule flottante (FPOs) requises par les modèles pour l'inférence. Nous comptons les FPOs couvrant le chargement du modèle sauvegardé, le chargement des données et l'ensemble du processus aboutissant à une seule prédiction. Le résumé des résultats de cette expérience est présenté dans la Table C.5. Remarquez que, même lors de l'inférence, le RC numérique et le BiLSTM nécessitent le moins d'efforts de calcul. Par exemple, le RC ne nécessite que $15.7\%$ du nombre de FPOs nécessaires au BiLSTM pour effectuer une prédiction. Une fois de plus, pour l'inférence, le modèle CNN a nécessité le plus d'efforts de calcul, suivi par $k$-Medoids. Néanmoins, les gains pour RC par rapport aux autres modèles sont beaucoup plus importants ici, car la prédiction se réduit maintenant à une simple opération de multiplication matricielle.

| **Méthode** | **Millions de FPOs** |
|---|---|
| CNN | 79.8 |
| k-Medoids | 54 |
| BiLSTM | 8.64 |
| Digital RC | 1.36 |

Table C.5: Estimations du nombre des FPOs nécessaires pour l'inférence.

**Les résultats sur les coûts pour RC physique**

Nous avons constaté que $49\%$ des efforts de calcul sont consacrés à la projection de l'entrée, c'est-à-dire au calcul des états du réservoir tel que décrit dans l'équation 2.1.

La faisabilité matérielle du RC offre la possibilité de *réduire* ces FPOs en externalisant les opérations de projection vers une configuration réservoir optoélectronique dédiée. Nous appelons cela le RC matériel, et ladite projection sera effectuée de manière analogique plus rapide. Ce faisant, seule l'opération de régression pour calculer $\boldsymbol{W}^{out}$ est mise en œuvre sur un processeur numérique pour le RC matériel, similaire à la variante numérique. Nous estimons la consommation d'énergie moyenne par le processeur et la configuration optoélectronique pour la projection et les comparons dans la Table C.6. Pour la configuration matérielle, la puissance consommée est estimée approximativement en additionnant les énergies consommées par les composants actifs de la configuration. Pour le RC numérique, l'énergie consommée pour la projection est obtenue en exécutant plusieurs fois les projections et en calculant l'énergie moyenne consommée pour une seule exécution. Bien que les valeurs estimées soient approximatives, elles indiquent des coûts énergétiques inférieurs pour la configuration matérielle. Pourtant, il y a une réduction marginale de précision de $2\%$ pour le RC matériel par rapport au RC numérique. Cela peut être dû à la nature plus bruyante des expériences matérielles et/ou à l'ajustement limité et difficile des composants physiques.

| - | **Énergie** $(kWh)$ | **Pre** $(\%)$ | **Sens** $(\%)$ | **Spec** $(\%)$ |
|---|---|---|---|---|
| Digital RC | $1.147 \times 10^{-4}$ | 85 | 88 | 83 |
| Hardware RC | $8.3 \times 10^{-5}$ | 83 | 92 | 74 |

Table C.6: Consommation d'énergie estimée pour la couche de réservoir.

## C.4 Machine de Ising pour l'optimisation combinatoire

Nous avons présenté des systèmes physiques bio-inspirés composés d'un grand nombre d'éléments simples interconnectés qui possèdent des propriétés computationnelles intéressantes. Toutefois, nos discussions portaient uniquement sur des mises en œuvre non autonomes de systèmes neuromorphiques qui sont pilotés par un signal d'entrée et nécessitent une technique d'apprentissage (supervision) pour que leur calcul soit considéré comme utile. Explorons maintenant une variante en mode de fonctionnement autonome et non supervisée des systèmes neuromorphiques. Cette discipline manque de comparaisons avec d'autres techniques d'optimisation quasi-globale ainsi que d'évaluations de l'efficacité énergétique des systèmes proposés, et nous contribuerons à combler cette lacune. Suivant la philosophie de cette thèse, nous étudions une architecture matérielle d'une machine de Ising cohérente (Coherent Ising Machine, CIM) basée sur des composants de télécommunication disponibles sur étagère. Notre étude considère la mise en œuvre d'un tel système dans une simulation numérique, suivie d'une configuration

physique. Outre l'architecture matérielle, les principales nouveautés de notre exploration sont les suivantes : d'un point de vue applicatif, nous démontrons le potentiel de la CIM pour le débruitage statistique d'images ; d'un point de vue système, nous comparons le système mixte matériel/numérique proposé à une mise en œuvre numérique standard des réseaux de Hopfield et du recuit simulé (simulated Annealing, SAN) [144] en termes de probabilité d'atteindre l'état fondamental moyen, de complexité de calcul et de consommation d'énergie.

### C.4.1 Les problèmes d'optimisation combinatoire

Les systèmes dépendant de plusieurs paramètres pour caractériser leur fonctionnement sont nombreux dans des scénarios variés du monde réel. L'optimisation combinatoire consiste à rechercher une solution dans un ensemble discret (éventuellement multidimensionnel) de telle sorte qu'une fonction objectif du système soit optimisée (soit maximisée ou minimisée). L'espace de recherche pour les problèmes d'optimisation combinatoire se développe rapidement avec l'augmentation du nombre de paramètres, ce qui prolonge la durée de la recherche. En termes de complexité computationnelle, ces problèmes sont considérés comme des problèmes NP-difficiles. Une méthode directe pour aborder les problèmes d'optimisation combinatoire est la méthode de la force brute, capable d'obtenir des solutions exactes, mais connue pour sa difficulté à gérer l'explosion exponentielle de l'espace de recherche, ce qui entraîne une complexité accrue.

#### Optimisation MAXCUT

Ce problème consiste à trouver la division d'un graphe en deux sous-graphes de manière à maximiser le nombre (ou la somme de leurs poids) d'arêtes coupées. Supposons que $\mathbf{G} = [\mathbf{V}, \mathbf{E}]$ soit un graphe avec un ensemble de sommets $\mathbf{V}$ et un ensemble d'arêtes $\mathbf{E}$. Une coupure est une partition de $\mathbf{G}$ en $\mathbf{G_1} = [\mathbf{V_1}, \mathbf{E_1}]$ et $\mathbf{G_2} = [\mathbf{V_2}, \mathbf{E_2}]$, telle que $\mathbf{V_1} \cap \mathbf{V_2} = \emptyset$. Trouver les coupures correspondant au nombre maximal possible d'arêtes coupées, connu sous le nom de problème MAXCUT, est l'un des premiers défis identifiés comme étant des problèmes NP-complets. Heureusement, le problème peut être reformulé de telle sorte que la recherche d'une solution devienne équivalente à la minimisation de l'hamiltonien de Ising [214].

#### Débruitage d'images

L'avènement des appareils photo, des télescopes puissants et des microscopes a entraîné une multiplication des images prises pour capturer des souvenirs, étudier les galaxies

et observer le monde des micro-organismes. Malheureusement, des images subissent une dégradation due au bruit lié aux imperfections des capteurs photo, introduit par les canaux de transmission ou lors de la compression des images obtenues. Le problème de l'image bruitée peut être formulé en supposant que l'image propre $y$ est soumise à un bruit additif $\eta$ de sorte que l'image bruitée soit $\hat{y} = y + \eta$. Le débruitage d'image consiste donc à restaurer $y$ à partir de $\hat{y}$, et reste un problème difficile. De multiples solutions ont été proposées, telles que le filtrage, la régularisation de Tikhonov et les réseaux de neurones convolutionnels (CNN). Nous considérons des images en noir et blanc corrompues par un bruit de type "Salt and Pepper" et les formulons de telle sorte que chaque pixel clair (foncé) soit un spin vers le haut (vers le bas).

### C.4.2 Machine de Ising numérique et son implementation physique

En suivant [34], nous mettons en œuvre numériquement un réseau de Hopfield généralisé à l'instant discret $k$ sous la forme suivante :

$$\begin{aligned} x_s(k) &= f\left(\alpha x_s(k-1) + \beta\left(\sum_{t:(s,t)\in\mathcal{N}} J_{s,t}x_t(k-1) + b_s\right)\right) \\ \hat{\sigma}_s(k) &= \text{sign}(x_s(k)), \quad \forall s \in \Omega \end{aligned} \tag{5.4}$$

où $f(.)$ est une fonction d'activation non linéaire (par exemple, une sigmoïde, une non-linéarité périodique ou tronquée comme suggéré dans [218]), $\alpha$ et $\beta$ sont des coefficients d'échelle qui contrôlent le couplage propre et la force de rétroaction affectant la sortie du neurone $x_s(k)$, tandis que $\hat{\sigma}_s(k)$ est l'estimation de spin du pixel $s$. Cette formulation vise à optimiser l'énergie du système definie comme suit:

$$E(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{(s,t)\in\mathcal{N}} J_{s,t}\sigma_s\sigma_t - \sum_{s\in\Omega} b_s\sigma_s. \tag{5.2}$$

La machine de Ising numérique exécute une implémentation en Python de l'algorithme décrit par l'Equation 5.4 sur un processeur Intel Xeon E5-1603. La fonction non-linéaire est la fonction $\sin^2(.)$(imitant la fonction de transfert du MZM). Par rapport à une architecture Von-Neumann conventionnelle pour l'implémentation numérique, un traitement mixte analogique/numérique sous la forme d'une CIM peut permettre des économies en termes de vitesses de traitement réalisables et de consommation d'énergie. Les solutions à haute vitesse pour les problèmes d'optimisation combinatoire sont très attrayantes pour les applications nécessitant un traitement en temps réel ou une adaptation à des environnements en constante évolution.

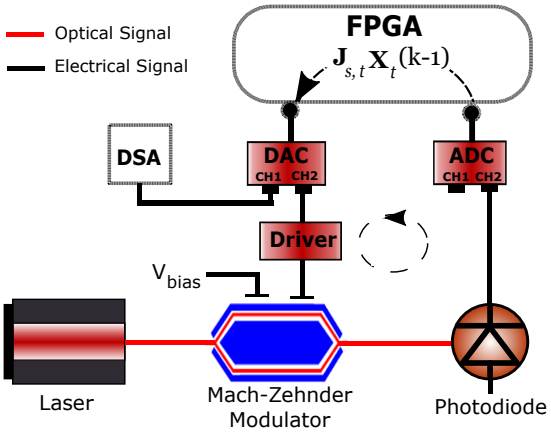



Figure C.5: Configuration expérimentale du CIM opto-électronique proposé. $\mathtt{V_{bias}}$ est le contrôle de tension de polarisation du MZM.

### C.4.3 Experiences, résultats et les discussions

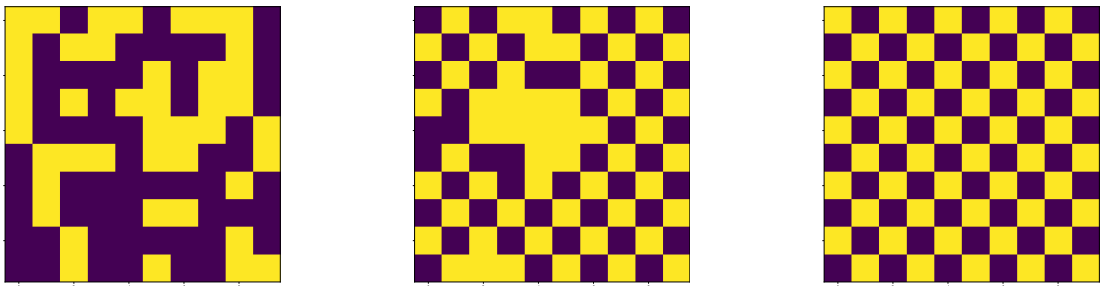
Dans cette section, nous évaluons et comparons l'architecture CIM optoélectronique à la Figure C.5 selon les métriques de performance suivantes:

- La probabilité de réussite d'atteindre (ou d'approximer) l'état fondamental.
- La complexité mesurée par le nombre d'opérations en virgule flottante (FPOs)
- La consommation moyenne d'énergie des éléments constitutifs.

Nous comparons ces métriques avec celles du CIM implémenté numériquement, et nous utilisons SAN basée sur l'échantillonneur de Gibbs [224] comme méthode de référence. Sauf indication contraire, $\mathcal{N}$ est choisi comme l'ensemble des couples de plus proches voisins à 4 points dans le réseau avec des conditions limites périodiques. Pour le CIM numérique et optoélectronique, la configuration initiale des spins est choisie de manière uniforme et indépendante au hasard, la fonction d'activation $f(.)$ est celle définie par [34] les hyperparamètres sont fixés à $(\alpha = 0.25, \beta = 0.29)$, tandis que le nombre d'itérations est fixé à $N_{it} = 100$. Aussi, nous utilisons SAN avec une température initiale de $2$, un paramètre de recuit géométrique égal à $0.99$, et $200$ itérations [224]. Aussi, la légende adoptée pour toutes les images est la suivante : jaune vif pour spin-up $(+1)$ et violet foncé pour spin-down $(-1)$. Nous étudions deux problèmes qui peuvent être formulés comme la minimisation d'un hamiltonien ayant la forme donnée par l'Equation 5.2, tels que le modèle de Ising antiferromagnétique et le débruitage d'image MAP en utilisant la machine de Ising cohérente (CIM) optoélectronique, à la fois dans les simulations et les implémentations matérielles.

**Le modèle de Ising antiferromagnétique.**

Nous commençons par un exemple de faible dimension où $n = 10$ avec des interactions antiferromagnétiques, c'est-à-dire $J_{s,t} = -1 \; \forall (s,t) \in \mathcal{N}$ et $b_s = 0 \; \forall s \in \Omega$. Il est bien connu que l'état fondamental correspond à un motif de damier (configuration alternée de spins hauts et bas avec une énergie minimale égale à $-n^2$). La solution exacte avec ce motif est illustrée à la Figure C.6c pour un réseau de $10 \times 10$. Dans nos



(a) Itération = 0 (b) Itération = 25 (c) Itération = 50 à 200

Figure C.6: Les spins initiaux (Itération $= 0$) sont choisis de manière aléatoire et indépendante pour le CIM numérique, tandis que pour le CIM optoélectronique, le bruit du système initialise les spins. Un motif de damier apparaît (Itération $= 25$) et se stabilise lorsque le système converge (Itération = $50$ à $200$).

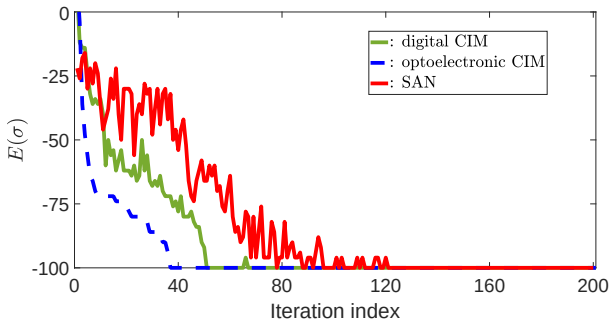



Figure C.7: L'évolution de l'énergie pour une seule exécution de la grille carrée de spins avec des interactions antiferromagnétiques.

expériences, la Figure C.6 représente la configuration initiale et finale estimée des spins pour une seule exécution du CIM optoélectronique proposé, qui converge avec succès vers l'état fondamental. Au cours d'une seule exécution réussie, la Figure C.7 montre que le CIM optoélectronique proposé et le CIM numérique ont des dynamiques similaires, la convergence étant généralement atteinte après quelques dizaines d'itérations. la Tableau C.7 résume nos mesures de performance en répétant toutes les expériences mentionnées précédemment indépendamment $1000$ fois.

| Metric | Digital CIM | Optoeletronic CIM | SAN |
|---|---|---|---|
| Success probability (%) | 91.4 | 90 | 98.9 |
| Time ($ms$) | 10 | 2.1 | 79 |
| FPOs ($\times 10^6$) | 0.76 | 0.32 | 5.68 |
| Énergie ($\times 10^{-6}\ kWh$) | 0.6 | $3 \times 10^{-4}$ | 14 |
| $CO_{eq}$ ($\times 10^{-6} kg$) | 0.044 | $2.2 \times 10^{-5}$ | 1 |

Table C.7: Performance moyennes pour le modèle antiferromagnétique.

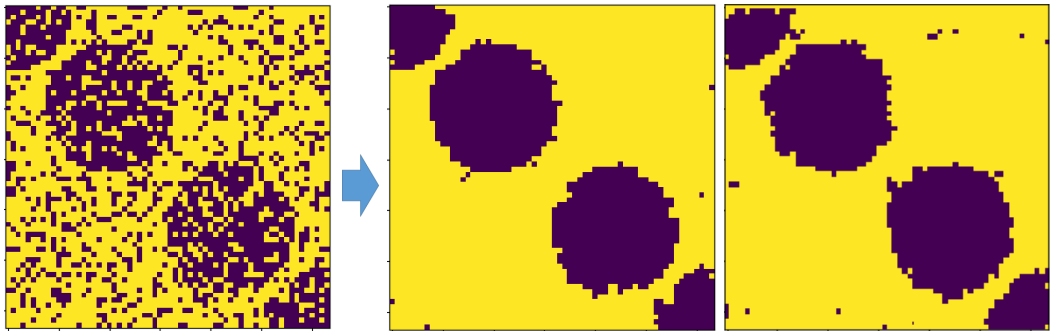
**Débruitage d'images au sens du maximum a posteriori (MAP)**

Nous considérons ici une image cachée en noir et blanc $(\sigma_s)_{s\in\Omega}$ devant être restaurée à partir d'une image observée $(y_s)_{s\in\Omega}$ selon un modèle de bruit "salt and pepper", c'est-à-dire $y_s = -\sigma_s$ avec une probabilité $p$ et $y_s = \sigma_s$ avec une probabilité $1-p$, indépendamment pour chaque pixel $s \in \Omega$. Dans notre configuration, $n = 64$ et la distribution de probabilité a priori des spins est choisie comme le champ aléatoire de Markov (MRF) avec le paramètre de couplage entre les pixels voisins fixé à 1 (interactions ferromagnétiques). Il est facile de montrer que le débruitage d'image au sens de maximum a posteriori (MAP) correspond à la sélection de $J_{s,t} = 1$ pour tous $(s,t) \in \mathcal{N}$ et $b_s = -\frac{1}{2}\log\left(\frac{p}{1-p}\right) y_s$ pour tous les $s \in \Omega$ (comme dérivé dans l'annexe B).

La Figure C.8a présente l'image initiale bruitée (pour $p = 0.2$ ), Figure C.8b et Figure C.8c après l'exécution, du CIM numérique et du CIM optoélectronique respectivement. Au cours d'une seule exécution réussie, la Figure C.9 montre que le CIM optoélectronique proposé et le CIM numérique ont des dynamiques similaires, avec une convergence généralement atteinte après 15 itérations. la Table C.8 résume nos métriques de performance - en ajoutant le taux d'erreur de classification pixel par pixel (Pixel-wise classification error rate: PCER %) - en répétant toutes les expériences mentionnées précédemment indépendamment 1000 fois.

| Métrique | Digital CIM | Optoeletronic CIM | SAN |
|---|---|---|---|
| Probabilité de réussite (%) | 89 | 86.75 | 100 |
| PCER (%) | 2.3 | 3 | 1.7 |
| Temps ($ms$) | 4110 | 86 | 39130 |
| FPOs ($\times 10^9$) | 0.7 | 0.0131 | 7.1 |
| Énergie ($\times 10^{-6}\ kWh$) | 90 | $1.45 \times 10^{-2}$ | 1100 |
| $CO_{eq}$ ($\times 10^{-6} kg$) | 5 | $8 \times 10^{-4}$ | 62 |

Table C.8: Performance pour la débruitage d'images au sens MAP



(a) (b) (c)

Figure C.8: (a) Une image bruitée initiale (itération$=1$). Après $100$ itérations nous obtenons respectivement (b) et (c) pour les CIM numérique et opto-électronique.

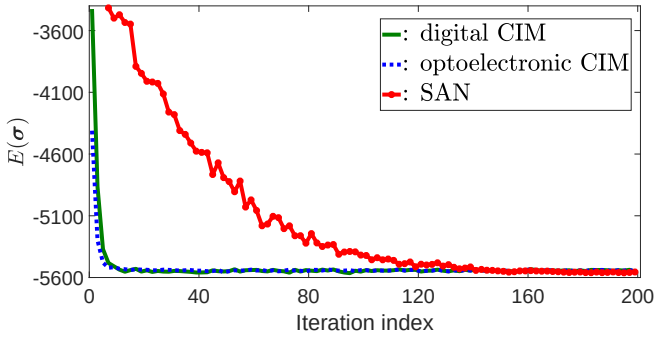



Figure C.9: Évolution de l'énergie de débruitage d'image au sens de MAP.

Pour cette application, nous constatons que SAN atteint l'état fondamental pour toutes les exécutions, tandis que le CIM numérique et le CIM matériel atteignent l'état fondamental dans $89\%$ et $86,75\%$ des exécutions respectivement. Les critères de réussite sont définis par le système qui se situe dans les trois écarts-types de l'énergie moyenne de l'état fondamental, qui est d'environ $E(\boldsymbol{\sigma}) = -5550$, correspondant ainsi à un intervalle de confiance de $98\%$. Les images propres générées par SAN ont un taux d'erreur de classification pixel par pixel (PCER) de $1,7\%$, tandis que les CIM numérique et matériel convergent vers un PCER de $2,3\%$ et $3\%$ respectivement. Pourtant, une analyse plus approfondie révèle que les performances de SAN sont obtenues à un temps d'exécution environ 9,5 fois plus élevé, 10 fois le nombre des FPOs et avec une consommation d'énergie 12,22 fois plus élevée (facteur identique pour les émissions de $CO_{eq}$) par rapport au CIM numérique. Les émissions de $CO_{eq}$ sont rapportées en tenant compte de la nature des réseaux électriques dans la ville de Palaiseau en France. Les résultats de consommation d'énergie pour le CIM matériel montrent un gain de facteur $6216$ et $75970$ en termes d'énergie et d'impact environnemental.

# Bibliography

[1] Alan Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, Oct. 1950.

[2] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12, Dec. 2006.

[3] Jon Agar. What is science for? the lighthill report on artificial intelligence reinterpreted. *The British Journal for the History of Science*, 53(3):1–22, Jul. 2020.

[4] Michael Haenlein and Andreas Kaplan. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, 61(4):5–14, Jul. 2019.

[5] William van Melle. Mycin: A knowledge-based computer program applied to infectious diseases. *International Journal of Man-Machine Studies*, 10(3):313–322, Oct. 1977.

[6] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[7] Greg L. Shaw. The organization of behavior: A neuropsychological theory. In *Brain Theory*, page 231–233. Lawrence Erlbaum Associates, 1949. ISBN 978-3-642-70911-1.

[8] Allen Newell. A step toward the understanding of information processes: Perceptrons by marvin minsky and seymour papert. *Science*, 165(3895):780–782, Aug. 1969.

[9] George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[10] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. Learning functions: When is deep better than shallow. *Proceedings of the AAAI Conference on Artificial Intelligence,*, 31(1):1–868, Feb. 2016.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, Nevada, United States, 2012. Curran Associates, Inc.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*, page 1–14, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1(1):770–778, Jun. 2016.

[15] Xiao-Dong Li, J.K.L. Ho, and T.W.S. Chow. Approximation of dynamical time-variant systems by continuous-time recurrent neural networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 52(10):656–660, 2005.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–80, Dec. 1997.

[17] Pierre Baldi, Søren Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics (Oxford, England)*, 15(11):937–46, Dec. 1999.

[18] Alex Graves and Jürgen Schmidhuber. Frame-wise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052, Montreal, Canada, Aug. 2005.

[19] Mike Schuster and Kuldip Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov. 1997.

[20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, Apr. 2020.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, volume 17, page 6000–6010, Red Hook, NY, USA, 2017.

[22] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM (CACM)*, 63(12):54–63, Nov. 2020.

[23] Tien-Ju Yang, Yu hsin Chen, J. Emer, and V. Sze. A method to estimate the energy consumption of deep neural networks. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 1916–1920, Pacific Grove, CA, USA, Oct. 2017.

[24] Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, Dec. 2020.

[25] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, ICPP 2018, New York, NY, USA, 2018.

[26] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *Computing Research Repository*, abs/1706.02677, Jun. 2017.

[27] Whole brain emulation: A roadmap, technical report, 2008.

[28] Tingxing Dong, Veselin Dobrev, Tzanio Kolev, Robert Rieben, Stanimire Tomov, and Jack Dongarra. A step towards energy efficient computing: Redesigning a hydrodynamic application on cpu-gpu. In *2014 IEEE* $28^{th}$ *International Parallel and Distributed Processing Symposium*, pages 972–981, Phoenix, AZ, USA, May. 2014.

[29] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware. *Computing Research Repository*, abs/1705.06963, May. 2017.

[30] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78 (10):1629–1636, Oct. 1990.

[31] Dario Amodei. Ai and compute, Jun. 2021.

[32] Bo Marr, Brian Degnan, Paul Hasler, and David Anderson. Scaling energy per operation via an asynchronous pipeline. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(1):147–151, 2013.

[33] John Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554–8, May. 1982.

[34] Fabian Böhm, Guy Verschaffelt, and Guy Van der Sande. A poor man's coherent ising machine based on optoelectronic feedback systems for solving optimization problems. *Nature Communications*, 10(3538):3538, Aug. 2019.

[35] Snehashish Chakraverty, Deepti Moyi Sahoo, and Nisha Rani Mahato. *Concepts of Soft Computing: Fuzzy and ANN with Programming*, pages 175–182. Springer Singapore, Singapore, Jan. 2019.

[36] Hinton Geoffrey E. Rumelhart David E. and Williams Ronald J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Jul. 1986.

[37] Kenji Doya. Bifurcations in the learning of recurrent neural networks. *1992 IEEE International Symposium on Circuits and Systems*, 6:2777–2780, Jan. 1992.

[38] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Bellevue Washington USA, Jun. 2011.

[39] Hinton Geoffrey E. Rumelhart David E. and Williams Ronald J. Learning internal representations by error propagation. In David E. Rumelhart and James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, volume 1, pages 318–362. MIT Press, Cambridge, Jan. 1986.

[40] Amir Atiya and Alexander G. Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, 11(3):697–709, May. 2000.

[41] Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct. 1990.

[42] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, Aug. 2009.

[43] Ulf D. Schiller and Jochen Jakob Steill. Analyzing the weight dynamics of recurrent learning algorithms. *Neurocomputing*, 63:5–23, Jan. 2005.

[44] Jochen Jakob Steil. Backpropagation-decorrelation: online recurrent learning with o(n) complexity. In *2004 IEEE International Joint Conference on Neural Networks*, volume 2, pages 843–848, Budapest, Hungary, Jul. 2004.

[45] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14:2531–60, Dec. 2002.

[46] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. *GMD-Report 148, German National Research Institute for Computer Science*, Jan. 2001.

[47] Miguel Soriano, S Ortín, Daniel Brunner, Laurent Larger, Claudio Mirasso, Ingo Fischer, and L Pesquera. Optoelectronic reservoir computing: Tackling noise-induced performance degradation. *Optics express*, 21:12–20, Jan. 2013.

[48] Lennert Appeltant, Guy Van der Sande, Jan Danckaert, and Ingo Fischer. Constructing optimized binary masks for reservoir computing with delay systems. *Scientific Reports*, 4(3629), Jan. 2014.

[49] Michiel Hermans, Miguel C. Soriano, Joni Dambre, Peter Bienstman, and Ingo Fischer. Photonic delay systems as machine learning implementations. *The Journal of Machine Learning Research*, 16, Jan. 2015.

[50] Izzet B. Yildiz, Herbert Jaeger, and Stefan J. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, Nov. 2012.

[51] Herbert Jaeger and Herbert Jaeger. Short-term memory in echo state networks. gmd-report 152. In *GMD Report 152*, Sankt Augustin, Germany, Jan. 2002. GMD Forschungszentrum Informationstechnik.

[52] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[53] Olivia L. White, Daniel D. Lee, and Haim Sompolinsky. Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92(14):148102, Apr. 2004.

[54] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 105(48):18970–18975, Dec. 2008.

[55] Michiel Hermans and Benjamin Schrauwen. Memory in linear recurrent neural networks in continuous time. *Neural Networks*, 23(3):341–355, Apr. 2010.

[56] Michiel Hermans and Benjamin Schrauwen. Memory in reservoirs for high dimensional input. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Barcelona, Spain, 2010.

[57] Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. Information processing capacity of dynamical systems. *Scientific Reports*, 2(1): 514, Jul. 2012.

[58] Chris G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1):12–37, Jun. 1990.

[59] Nils Bertschinger, Thomas Natschläger, and Robert Legenstein. At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 17, Vancouver, Canada, Dec. 2004. MIT Press.

[60] Klaus-Robert Müller Grégoire Montavon, Geneviève B. Orr. Adaptation toward the edge of chaos. In *Dynamic Patterns in Complex Systems*, pages 293–301. University of Illinois at Urbana-Champaign, 2012.

[61] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, Jul. 2019.

[62] Mantas Lukosevicius. *Reservoir Computing and Self-Organized Neural Hierarchies*. PhD thesis, Jacobs University, Nov. 2012.

[63] Kristof Vandoorne, Wouter Dierckx, Benjamin Schrauwen, David Verstraeten, Roel Baets, Peter Bienstman, and Jan Van Campenhout. Toward optical signal processing using photonic reservoir computing. *Opt. Express*, 16(15):11182–11192, Jul. 2008.

[64] Lennert Appeltant, Miguel Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, J. Dambre, Benjamin Schrauwen, Claudio Mirasso, and Ingo Fischer. Information processing using a single dynamical node as complex system. *Nature communications*, 2:468, Sep. 2011.

[65] Miguel C. Soriano, Daniel Brunner, Miguel Escalona-Morán, Claudio R. Mirasso, and Ingo Fischer. Minimal approach to neuro-inspired information processing. *Frontiers in Computational Neuroscience*, 9(68), Jun. 2015.

[66] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In *European Conference on Artificial Life*, page 588–597, Sapporo, Japan, Sep. 2003.

[67] Jean Coulombe, Mark York, and Julien Sylvestre. Computing with networks of nonlinear mechanical oscillators. *PLoS ONE*, 12(6):1–13, Jun. 2017.

[68] Johannes H. Jensen and Gunnar Tufte. Reservoir computing with a chaotic circuit. In *European Conference on Artificial Life*, pages 222–229, Lyon, France, Sep. 2017.

[69] Jialing Li, Kangjun Bai, Lingjia Liu, and Yang Yi. A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system. In *2018 19th International Symposium on Quality Electronic Design (ISQED)*, pages 308–313, Santa Clara, CA, USA, May. 2018.

[70] Hong Zhang, Xue Feng, Boxun Li, Yu Wang, Kaiyu Cui, Fang Liu, Weibei Dou, and Yidong Huang. Integrated photonic reservoir computing based on hierarchical time-multiplexing structure. In *2015 Conference on Lasers and Electro-Optics (CLEO)*, pages 31356–31370, San Jose Convention Center, California, USA, Dec. 2015.

[71] Piotr Antonik. *Application of FPGA to Real-Time Machine Learning: Hardware Reservoir Computers and Software Image Processing*. Springer, Jan. 2018.

[72] Nicholas D. Haynes, Miguel C. Soriano, David P. Rosin, Ingo Fischer, and Daniel J. Gauthier. Reservoir computing with a single time-delay autonomous boolean node. *Physical Review E*, 91(2):20801, Feb. 2015.

[73] David Verstraeten, Benjamin Schrauwen, and Dirk Stroobandt. Reservoir computing with stochastic bitstream neurons. In *Proceedings of the 16th Annual ProRISC Workshop*, pages 454–459, Veldhoven, Netherlands, 2005.

[74] M. L. Alomar, V. Canals, V. Martínez-Moll, and J. L. Rosselló. Low-cost hardware implementation of reservoir computers. In *24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–5, Palma de Mallorca, Spain, Nov. 2014.

[75] Benjamin Schrauwen, Michiel D'Haene, David Verstraeten, and Jan Van Campenhout. Compact hardware liquid state machines on fpga for real-time speech recognition. *Neural Networks*, 21(2):511–523, 2008.

[76] Qian Wang, Yingyezhe Jin, and Peng Li. General-purpose lsm learning processor architecture and theoretically guided design space exploration. In *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4, Atlanta, GA, USA, 2015.

[77] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[78] L.O. Chua and Sung Mo Kang. Memristive devices and systems. *Proceedings of the IEEE*, 64(2):209–223, 1976.

[79] Giacomo Indiveri, Bernabé Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38): 384010, Sep. 2013.

[80] Xiao Yang, Wanlong Chen, and Frank Z. Wang. Investigations of the staircase memristor model and applications of memristor-based local connections. *Analog Integrated Circuits and Signal Processing*, 87:263–273, 2016.

[81] Colin Donahue, Cory Merkel, Qutaiba Saleh, Levs Dolgovs, Yu Kee Ooi, Dhireesha Kudithipudi, and Bryant Wysocki. Design and analysis of neuromemristive echo state networks with limited-precision synapses. In *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–6, Verona, NY, USA, 2015.

[82] Nicholas Soures, Lydia Hays, and Dhireesha Kudithipudi. Robustness of a memristor based liquid state machine. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2414–2420, 2017.

[83] Manjari S. Kulkarni and Christof Teuscher. Memristor-based reservoir computing. In *2012 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 226–232, Amsterdam, Netherlands, Jan. 2012.

[84] Jens Bürger and Christof Teuscher. Variation-tolerant computing with memristive reservoirs. In *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 1–6, Brooklyn, NY, USA, 2013.

[85] Yanan Zhong, Jianshi Tang, Xinyi Li, Bin Gao, He Qian, and Huaqiang Wu. Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing. *Nature Communications*, 12:408, Jan. 2021.

[86] S. J. Dat Tran and Christof Teuscher. Memcapacitive reservoir computing. In *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 115–116, Newport, RI, USA, 2017.

[87] Felix Schürmann, Karlheinz Meier, and Johannes Schemmel. Edge of chaos computation in mixed-mode vlsi - a hard liquid. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, Vancouver, Canada, 2004. MIT Press.

[88] Anvesh Polepalli, Nicholas Soures, and Dhireesha Kudithipudi. Digital neuromorphic design of a liquid state machine for real-time processing. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, San Diego, CA, USA, 2016.

[89] K. Vandoorne, Pauline Mechet, Thomas Van Vaerenbergh, Martin Fiers, G. Morthier, David Verstraeten, Benjamin Schrauwen, J. Dambre, and Peter Bienstman. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature communications*, 5(3541):3541, Mar. 2014.

[90] Floris Laporte, Andrew Katumba, Joni Dambre, and Peter Bienstman. Numerical demonstration of neuromorphic computing with photonic crystal cavities. *Opt. Express*, 26(7):7955–7964, Apr. 2018.

[91] Charis Mesaritakis. Micro ring resonators as building blocks for an all-optical high-speed reservoir computing bit-pattern recognition system. *Journal of the Optical Society of America B*, 30(11):3048–3055, Nov. 2013.

[92] Daniel Brunner and Ingo Fischer. Reconfigurable semiconductor laser networks based on diffractive coupling. *Optics letters*, 40(16):3854–7, 2015.

[93] Jonathan Dong, Sylvain Gigan, Florent Krzakala, and Gilles Wainrib. Scaling up echo-state networks with multiple light scattering. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 448–452, Freiburg, Germany, Jun. 2018. IEEE.

[94] Francois Duport, Bendix Schneider, Anteo Smerieri, Marc Haelterman, and Serge Massar. All-optical reservoir computing. *Optics Express*, 20(20):22783, Sep. 2012.

[95] Daniel Brunner, Miguel C. Soriano, Claudio R. Mirasso, and Ingo Fischer. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature Communications*, 4(1364), Jan. 2013.

[96] Laurent Larger, M. C. Soriano, Daniel Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Optics Express*, 20(3):3241–3249, Jan. 2012.

[97] Yvan Paquot, François Duport, Anteo Smerieri, J. Dambre, Benjamin Schrauwen, M. Haelterman, and Serge Massar. Optoelectronic reservoir computing. *Scientific Reports*, 2, Nov. 2011.

[98] François Duport, Anteo Smerieri, Akram Akrout, Marc Haelterman, and Serge Massar. Virtualization of a photonic reservoir computer. *Journal of Lightwave Technology*, 34(9):2085–2091, 2016.

[99] Anteo Smerieri, François Duport, Yvan Paquot, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Analog readout for optical reservoir computers. *Advances in Neural Information Processing Systems*, 25:1–9, Sep. 2012.

[100] Duport François, Smerieri Anteo, Akrout Akram, Haelterman M., and Massar Serge. Fully analogue photonic reservoir computer. *Scientific Reports*, 6:22381, Mar. 2016.

[101] Laurent Larger, Antonio Baylón-Fuentes, Romain Martinenghi, Vladimir S. Udaltsov, Yanne K. Chembo, and Maxime Jacquot. High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification. *Phys. Rev. X*, 7:011015, Feb. 2017.

[102] Liberman Mark. Ti 46-word ldc93s9, 1993.

[103] iXblue Photonics. *10 GHz Analog RF Amplifier*. iXblue Photonics, 2022. URL https://www.ixblue.com/wp-content/uploads/2022/02/DR-AN-10-MO.pdf.

[104] Exail Photonics. *MX-LN series 1550 nm band intensity Modulators*. Exail Photonics, 2022. URL https://www.ixblue.com/wp-content/uploads/2022/02/MX-LN_SERIES.pdf.

[105] Thorlabs Inc. *RXM25 Series: 25 GHz Amplified Photoreceivers*. Thorlabs Inc, 2020. URL https://www.thorlabs.com/drawings/837128602a2ebcbf-28D2FBF7-DFF4-2FDE-65D7E8EE3ECA5EE6/RXM25AF-SpecSheet.pdf.

[106] Piotr Antonik, Marc Haelterman, and Serge Massar. Brain-inspired photonic signal processor for generating periodic patterns and emulating chaotic systems. *Physical Review Applied*, 7(5), May. 2017.

[107] Piotr Antonik, Michiel Hermans, Marc Haelterman, and Serge Massar. Random pattern and frequency generation using a photonic reservoir computer with output feedback. *Neural Processing Letters*, 47(3):1041–1054, Apr. 2017.

[108] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78 – 80, 2004.

[109] Wei-Jia Wang, Yong Tang, Jason Xiong, and Yi-Cheng Zhang. Stock market index prediction based on reservoir computing models. *Expert Systems with Applications*, 178:115022, 2021.

[110] Rajat Budhiraja, Manish Kumar, Mrinal K. Das, Anil Singh Bafila, and Sanjeev Singh. A reservoir computing approach for forecasting and regenerating both dynamical and time-delay controlled financial system behavior. *PLOS ONE*, 16: 1–24, Feb. 2021.

[111] Forecasting competition for neural networks & computational intelligence, 2007. URL http://www.neural-forecasting.com/nn3-competition.htm.

[112] Florian Denis-Le Coarer, Marc Sciamanna, Andrew Katumba, Matthias Freiberger, Joni Dambre, Peter Bienstman, and Damien Rontani. All-optical reservoir computing on a photonic chip using silicon-based ring resonators. *IEEE Journal of Selected Topics in Quantum Electronics*, 24(6):1–8, 2018.

[113] Piotr Antonik, François Duport, Michiel Hermans, Anteo Smerieri, Marc Haelterman, and Serge Massar. Online training of an opto-electronic reservoir computer applied to real-time channel equalization. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2686–2698, 2017.

[114] Jérémy Vatin, Damien Rontani, and Marc Sciamanna. Experimental realization of dual task processing with a photonic reservoir computer. *APL Photonics*, 5: 086105, Aug. 2020.

[115] Adonis Bogris, Charis Mesaritakis, Stavros Deligiannidis, and Pu Li. Fabry-perot lasers as enablers for parallel reservoir computing. *IEEE Journal of Selected Topics in Quantum Electronics*, 27(2):1–7, Mar. 2021.

[116] Apostolos Argyris, Julián Bueno, and Ingo Fischer. Pam-4 transmission at 1550 nm using photonic reservoir computing post-processing. *IEEE Access*, 7:37017–37025, 2019.

[117] Stenio M. Ranzini, Roman Dischler, Francesco Da Ros, Henning Bülow, and Darko Zibar. Experimental investigation of optoelectronic receiver with reservoir computing in short reach optical fiber communications. *Journal of Lightwave Technology*, 39(8):2460–2467, 2021.

[118] Francesco Da Ros, Stenio M. Ranzini, Henning Bülow, and Darko Zibar. Reservoir-computing based equalization with optical pre-processing for short-reach optical transmission. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(5):1–12, 2020.

[119] Azarakhsh Jalalvand, Glenn Van Wallendael, and Rik Van De Walle. Real-time reservoir computing network-based systems for detection tasks on visual contents. In *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*, pages 146–151, Riga, Latvia, 2015.

[120] Dillon Graham, Seyed Hamed Fatemi Langroudi, Christopher Kanan, and Dhireesha Kudithipudi. Convolutional drift networks for video classification, 2017.

[121] Piotr Antonik, Nicolas Marsal, Daniel Brunner, and Damien Rontani. Human action recognition with a large-scale brain-inspired photonic computer. *Nature Machine Intelligence*, 1, Nov. 2019.

[122] Claudio Gallicchio and Alessio Micheli. Deep echo state network (deepesn): A brief survey, 2017.

[123] Pieter Buteneers, David Verstraeten, Pieter van Mierlo, Tine Wyckhuys, Dirk Stroobandt, Robrecht Raedt, Hans Hallez, and Benjamin Schrauwen. Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing. *Artificial intelligence in medicine*, 53 3:215–223, 2011.

[124] Nickson Mwamsojo, Kamel Merghem, Mounim A. El-Yacoubi, Yann Frignac, Badr-Eddine Benkelfat, Anne-Sophie Rigaud, and Frederic Lehmann. Opto-electronic reservoir computer for early stage alzheimer's disease detection. In *Conference on Lasers and Electro-Optics*, page ATh4I.5, California, USA, 2022. Optica Publishing Group.

[125] Nickson Mwamsojo, Frederic Lehmann, Mounim A. El-Yacoubi, Kamel Merghem, Yann Frignac, Badr-Eddine Benkelfat, and Anne-Sophie Rigaud. Reservoir computing for early stage alzheimer's disease detection. *IEEE Access*, 10:59821–59831, 2022.

[126] Eric A. Antonelo, Benjamin Schrauwen, and Dirk Stroobandt. Event detection and localization for small mobile robots using reservoir computing. *Neural networks : the official journal of the International Neural Network Society*, 21 6: 862–71, 2008.

[127] Piotr Antonik, Marvyn Gulina, Jaël Pauwels, Damien Rontani, Marc Haelterman, and Serge Massar. Spying on chaos-based cryptosystems with reservoir computing. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Rio de Janeiro, Brazil, 2018.

[128] Jie Qin, Qingchun Zhao, Hongxi Yin, Yu Jin, and Chang Liu. Numerical simulation and experiment on optical packet header recognition utilizing reservoir computing based on optoelectronic feedback. *IEEE Photonics Journal*, 9(1): 1–11, 2017.

[129] Qingchun Zhao, Hongxi Yin, and Hegui Zhu. Simultaneous recognition of two channels of optical packet headers utilizing reservoir computing subject to mutual-coupling optoelectronic feedback. *Optik*, 157:951–956, 2018.

[130] Luca Anthony Thiede and Ulrich Parlitz. Gradient based hyperparameter optimization in echo state networks. *Neural Networks*, 115:23–29, Jul. 2019.

[131] Michiel Hermans and Benjamin Schrauwen. One step back-propagation through time for learning input mapping in reservoir computing applied to speech recog-

nition. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 521–524, Paris, France, 2010.

[132] Nathan Trouvain, Luca Pedrelli, Thanh Trung Dinh, and Xavier Hinaut. ReservoirPy: an Efficient and User-Friendly Library to Design Echo State Networks. In *ICANN 2020 - 29th International Conference on Artificial Neural Networks*, pages 1–19, Bratislava, Slovakia, Sep. 2020.

[133] Aida A. Ferreira and Teresa B. Ludermir. Genetic algorithm for reservoir computing optimization. In *2009 International Joint Conference on Neural Networks*, pages 811–815, Atlanta, USA, 2009.

[134] Anderson Tenório Sergio and Teresa B. Ludermir. Pso for reservoir computing optimization. In Alessandro E. P. Villa, Włodzisław Duch, Péter Érdi, Francesco Masulli, and Günther Palm, editors, *Artificial Neural Networks and Machine Learning – ICANN 2012*, Lausanne, Switzerland, 2012. Springer Berlin Heidelberg.

[135] Anderson T. Sergio and Teresa B. Ludermir. Reservoir computing optimization with a hybrid method. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2653–2660, Beijing, China, 2014.

[136] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, Sep. 1951.

[137] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, Sep. 1952.

[138] Harold J. Kushner and Dean S. Clark. *Stochastic Approximation for Constrained and Unconstrained Systems*, volume 26 of *Applied Mathematical Sciences*, chapter Introduction. Springer, New York, 1978.

[139] D.C. Chin. Comparative study of stochastic algorithms for system optimization based on gradient approximations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):244–249, Apr. 1997.

[140] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 3 edition, 1996.

[141] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[142] Helmut Lütkepohl. *Handbook of matrices*. John Wiley and Sons, Berlin, Germany, 1996.

[143] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge; New York, 1994.

[144] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, chapter Stochastic Search and Optimization: Motivation and Supporting Results, pages 1–33. John Wiley & Sons, Ltd, 2003.

[145] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer, New York, 2005.

[146] R. Gary Leonard and George Doddington. Tidigits ldc93s10. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[147] Roberto Togneri and Daniel Pullella. An overview of speaker identification: Accuracy and robustness issues. *IEEE Circuits and Systems Magazine*, 11(2): 23–61, Sep. 2011.

[148] Hui-Ping Cheng and Chuen-Sheng Cheng. Control chart pattern recognition using wavelet analysis and neural networks. *Journal of Quality*, 16, Jan. 2009.

[149] R. Alcock and Yannis Manolopoulos. Time-series similarity queries employing a feature-based approach. In *7th Hellenic Conference on Informatics*, University of Ioannina, Greece, Jan. 1999.

[150] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: The collective of transformation-based ensembles. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1548–1549, Helsinki, Finlan, 2016.

[151] Robert Alcock. Synthetic control, 1999. URL `https://archive.ics.uci.edu/ml/datasets/Synthetic+Control+Chart+Time+Series`.

[152] Robert T. Olszewski, Roy A. Maxion, and Daniel P. Siewiorek. *Generalized feature extraction for structural pattern recognition in time-series data*. PhD thesis, Carnegie Mellon University, 2001.

[153] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun. 2002.

[154] Jorge Haddock and John Mittenthal. Simulation optimization using simulated annealing. *Computers and Industrial Engineering*, 22(4):387–395, 1992.

[155] Harold Szu and Ralph Hartley. Fast simulated annealing. *Physics Letters A*, 122 (3):157–162, 1987.

[156] Moon-Won Park and Yeong-Dae Kim. A systematic procedure for setting parameters in simulated annealing algorithms. *Computers and Operations Research*, 25(3):207–217, 1998.

[157] Keysight Technologies. *M3302A/M3300A PXIe Arbitrary Waveform Generator and Digitizer Combos with Optional Real-Time Sequencing and FPGA Programming*. Keysight Technologies, 2019.

[158] Alzheimers Dement. 2021 alzheimer's disease facts and figures. Technical report, Alzheimer's Association, 2021.

[159] Anthony Sabine, Pradier Christian, Chevrier Roland, Festraëts Julie, Karim Tifratene, and Philippe Robert. A fast growing database for researchers and clinicians. Technical report, The French National Alzheimer Database, 2015.

[160] Marilyn Albert, Steven DeKosky, Dennis Dickson, Bruno Dubois, Howard Feldman, Nick Fox, Anthony Gamst, David Holtzman, William Jagust, Ronald Petersen, Peter Snyder, Maria Carrillo, Bill Thies, and Creighton Phelps. The diagnosis of mild cognitive impairment due to alzheimer's disease: recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. alzheimers dement. *Alzheimer's and dementia : the journal of the Alzheimer's Association*, 7(3):270–9, May. 2011.

[161] Christian Kahindo Senge Muvingi. *Analyse automatique de l'écriture manuscrite sur tablette pour la détection et le suivi thérapeutique de personnes présentant des pathologies*. Theses, Université Paris-Saclay, Nov. 2019.

[162] G. McKhann, D. Drachman, M. Folstein, R. Katzman, D. Price, and E.M. Stadlan. Clinical diagnosis of alzheimer's disease: Report of the nincds-adrda work group under the auspices of department of health and human services task force on alzheimer's disease. *Neurology*, 34(7):939–944, 1984.

[163] Maria Angelillo, Fabrizio Balducci, Donato Impedovo, Giuseppe Pirlo, and Gennaro Vessio. Attentional pattern classification for automatic dementia detection. *IEEE Access*, 7:57706–57716, Apr. 2019.

[164] Carl Olson and Carol Colby. *Fundamental Neuroscience*, chapter Spatial Cognition, pages 969–988. Psychology Press, 4 edition, Dec. 2013.

[165] Jin H. Yan, Susan Rountree, Paul Massman, Rachelle Smith Doody, and Hong Li. Alzheimer's disease and mild cognitive impairment deteriorate fine movement control. *Journal of Psychiatric Research*, 42(14):1203–1212, 2008.

[166] A. Schröter, R. Mergl, K. Bürger, H. Hampel, H.-J. Möller, and U. Hegerl. Kinematic analysis of handwriting movements in patients with Alzheimer's disease, mild cognitive impairment, depression and healthy subjects. *Dementia and Geriatric Cognitive Disorders*, 15(3):132–142, 2003.

[167] Nan-Ying Yu and Shao-Hsia Chang. Kinematic analyses of graphomotor functions in individuals with alzheimer's disease and amnesic mild cognitive impairment. *Journal of Medical and Biological Engineering*, 36(3):334–343, Jun. 2016.

[168] Josep Garre-Olmo, Marcos Faundez-Zanuy, Karmele Lopez-de Ipiña, Laia Calvó-Perxas, and Oriol Turró-Garriga. Kinematic and pressure features of handwriting and drawing: Preliminary results between patients with mild cognitive impairment, alzheimer disease and healthy controls. *Current Alzheimer research*, 14, May. 2016.

[169] Perla Werner, Sara Rosenblum, Gady Bar-On, Jeremia Heinik, and Amos Korczyn. Handwriting process variables discriminating mild Alzheimer's disease and mild cognitive impairment. *The Journals of Gerontology. Series B, Psychological Sciences and Social Sciences*, 61(4):228–236, Jul. 2006.

[170] Jacek Kawa, Adam Bednorz, Paula Stępień, Jarosław Derejczyk, and Monika Bugdol. Spatial and dynamical handwriting analysis in mild cognitive impairment. *Computers in Biology and Medicine*, 82:21–28, Jan. 2017.

[171] Mounîm A. El-Yacoubi, Sonia Garcia-Salicetti, Christian Kahindo, Anne-Sophie Rigaud, and Victoria Cristancho-Lacroix. From aging to early-stage Alzheimer's: Uncovering handwriting multimodal behaviors by semi-supervised learning and sequential representation learning. *Pattern Recognition*, 86:112–133, 2019.

[172] Christian Kahindo, Mounim A. El-Yacoubi, Sonia Garcia-Salicetti, Anne-Sophie Rigaud, and Victoria Cristancho-Lacroix. Characterizing early-stage alzheimer through spatiotemporal dynamics of handwriting. *IEEE Signal Processing Letters*, 25(8):1136–1140, 2018.

[173] Hans-Leo Teulings and George E. Stelmach. Control of stroke size, peak acceleration, and stroke duration in parkinsonian handwriting. *Human Movement Science*, 10(2):315–334, 1991.

[174] Catherine Taleb, Maha Khachab, Chafic Mokbel, and Laurence Likforman-Sulem. Visual representation of online handwriting time series for deep learning parkinson's disease detection. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, pages 25–30, 2019.

[175] Clayton R. Pereira, Danilo R. Pereira, Gustavo H. Rosa, Victor H.C. Albuquerque, Silke A.T. Weber, Christian Hook, and João P. Papa. Handwritten dynamics assessment through convolutional neural networks: An application to parkinson's disease identification. *Artificial Intelligence in Medicine*, 87:67–77, 2018.

[176] M.J. Slavin, J.G. Phillips, J.L. Bradshaw, K.A. Hall, and I. Presnell. Consistency of handwriting movements in dementia of the Alzheimer's type: A comparison with Huntington's and Parkinson's diseases. *Journal of the International Neuropsychological Society*, 5(1):20–25, 1999.

[177] Marianne J U Novak and Sarah J Tabrizi. Huntington's disease. *British Medical Journal*, 340, 2010.

[178] Naiqian Zhi, Beverly Jaeger, Andrew Gouldstone, Rifat Sipahi, and Samuel Frank. Toward Monitoring Parkinson's Through Analysis of Static Handwriting Samples: A Quantitative Analytical Framework. *IEEE Journal of Biomedical and Health Informatics*, 21(2):488–495, Jan. 2016.

[179] Donato Impedovo and Giuseppe Pirlo. Dynamic handwriting analysis for the assessment of neurodegenerative diseases: A pattern recognition perspective. *IEEE Reviews in Biomedical Engineering*, 12:209–220, May. 2018.

[180] V. Kumar and Ezio Giacobini. Use of agraphia in subtyping of alzheimer's disease. *Archives of gerontology and geriatrics*, 11 2:155–159, 1990.

[181] Herve Platel, Lambert Jany, Francis Eustache, Bernard Cadet, Martine Dary, Fausto Viader, and Bernard Lechevalier. Characteristics and evolution of writing impairment in alzheimer's disease. *Neuropsychologia*, 31:1147–58, Dec. 1993.

[182] Oliver Schabos, Knut Hoffmann, Björn Enzi, Georg Juckel, and Paraskevi Mavrogiorgou. Kinematic analysis of handwriting movements in individuals

with intellectual disabilities with and without obsessive compulsive symptoms. *Psychopathology*, 52:1–12, Jan. 2020.

[183] Nan-Ying Yu and Shao-Hsia Chang. Kinematic analyses of graphomotor functions in individuals with alzheimer's disease and amnestic mild cognitive impairment. *Journal of Medical and Biological Engineering*, 36(3):334–343, Jun. 2016.

[184] Kawa Jacek, Bednorz Adam, Stepien Paula, Derejczyk Jaroslaw, and Bugdol Monika. Spatial and dynamical handwriting analysis in mild cognitive impairment. *Computers in Biology and Medicine*, 82:21–28, 2017.

[185] Olusola O. Abayomi-Alli, Robertas Damaševičius, Rytis Maskeliūnas, and Adebayo Abayomi-Alli. Bilstm with data augmentation using interpolation methods to improve early detection of parkinson disease. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 371–380, Sofia, Bulgaria, 2020.

[186] Catherine Taleb, Laurence Likforman-Sulem, and Chafic Mokbel. Improving deep learning parkinson's disease detection through data augmentation training. In *Mediterranean Conference on Pattern Recognition and Artificial Intelligence (MedPRAI)*, volume 1144, page 79–93, Istanbul, Turkey, 2019.

[187] Gabriel Marzinotto, José C. Rosales, Mounîm A. El-Yacoubi, Sonia Garcia-Salicetti, Christian Kahindo, Hélène Kerhervé, Victoria Cristancho-Lacroix, and Anne-Sophie Rigaud. Age-related evolution patterns in online handwriting. *Comput. Math. Methods Medicine*, 2016:1–5, 2016.

[188] Ronald Petersen, Glenn Smith, Steve Waring, Robert Ivnik, Eric Tangalos, and Emre Kokmen. Mild cognitive impairment: Clinical characterization and outcome. *Archives of neurology*, 56:303–8, Apr. 1999.

[189] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, page 3939–3945, Buenos Aires, Argentina, 2015.

[190] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv*, abs/1605.07678, 2017.

[191] Mahmoud Assran, Joshua Romoff, Nicolas Ballas, Joelle Pineau, and Michael G. Rabbat. Gossip-based actor-learner architectures for deep reinforcement learning. *CoRR*, abs/1704.04861, 2019.

[192] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 5955–5965, 2018.

[193] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[194] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248): 10039–10081, 2020.

[195] Katharine Ricke, Laurent Drouet, Ken Caldeira, and Massimo Tavoni. Country-level social cost of carbon. *Nature Climate Change*, 8(10):895, 2018.

[196] Dan Terpstra, Heike Jagode, Haihang You, and Jack Dongarra. Collecting performance data with papi-c. In *Tools for High-Performance Computing 2009*, pages 157–173, Berlin, Heidelberg, 2009.

[197] Jacques Wainer and Gavin Cawley. Nested cross-validation when selecting classifiers is overzealous for most practical applications. *Expert Systems with Applications*, 182:115222, 2018.

[198] M. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the royal statistical society series b-methodological*, 38: 102–102, 1976.

[199] Michael Armbrust, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM*, 53 (4):50–58, Apr. 2010.

[200] John Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *National Academy of Sciences of the United States of America*, 81(10):3088–92, Jun. 1984.

[201] John Hopfield and D Tank. Neural computation of decisions in optimization problems. *Biological cybernetics*, 52:141–52, Feb. 1985.

[202] S.V.B. Aiyer, M. Niranjan, and F. Fallside. A theoretical investigation into the performance of the hopfield model. *IEEE Transactions on Neural Networks*, 1 (2):204–215, 1990.

[203] Jean-Yves Potvin. State-of-the-art survey - the traveling salesman problem: A neural network perspective. *INFORMS Journal of Computing*, 5:328–348, 1993.

[204] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9:147–169, 1985.

[205] Yutaka Akiyama, Atsushi Yamashita, Masahiro Kajiura, and Hiroshi Aiso. Combinatorial optimization with gaussian machines. In *International 1989 Joint Conference on Neural Networks*, volume 1, pages 533–540, Washington, DC, USA, 1989.

[206] Luonan Chen and Kazuyuki Aihara. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, 8(6):915–930, 1995.

[207] Dmitry Krotov. Hierarchical associative memory. *arXiv*, pages 1–13, 2021.

[208] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, Apr. 2001.

[209] Sergio Boixo, Vadim Smelyanskiy, Alireza Shabani, Sergei Isakov, Mark Dykman, Vasil Denchev, Mohammad Amin, Anatoly Smirnov, M. Mohseni, and Hartmut Neven. Computational multiqubit tunnelling in programmable quantum annealers. *Nature Communications*, 7:10327, Jan. 2016.

[210] Brian Sutton, Kerem Yunus Camsari, Behtash Behin-Aein, and Supriyo Datta. Intrinsic optimization using stochastic nanomagnets. *Scientific Reports*, 7(1): 1–15, Mar. 2017.

[211] Z. Fahimi, M. R. Mahmoodi, H. Nili, Valentin Polishchuk, and D. B. Strukov. Combinatorial optimization by weight annealing in memristive hopfield networks. *Scientific Reports*, 11(1):16383, Aug. 2021.

[212] Shoko Utsunomiya, Kenta Takata, and Yoshihisa Yamamoto. Mapping of ising models onto injection-locked laser systems. *Opt. Express*, 19(19):18091–18108, Sep. 2011.

[213] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2(5):1–5, 2014.

[214] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, May. 1988.

[215] Pantelis Bouboulis, Konstantinos Slavakis, and Sergios Theodoridis. Adaptive kernel-based image denoising employing semi-parametric regularization. *IEEE Transactions on Image Processing*, 19(6):1465–1479, 2010.

[216] Ralph A. Willoughby. Solutions of ill-posed problems (a. n. tikhonov and v. y. arsenin). *SIAM Review*, 21(2):266–267, 1979.

[217] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2017.

[218] Guy Verschaffelt & Guy Van der Sande Fabian Böhm, Thomas Van Vaerenbergh. Order-of-magnitude differences in computational performance of analog ising machines induced by the choice of nonlinearity. *Commun Phys*, 4(149), Jul. 2021.

[219] Mircea Dabacan. *Zmod Scope Reference Manual*. Digilent Inc, 2022.

[220] Digilent. *Eclypse Z7 Hardware Reference Manual*. Digilent Inc, 2022.

[221] Aadit N. A, Grimaldi A., Carpentieri M., Theogarajan L., Martinis J. M., Finocchio G., and Camsari K. Y. Massively parallel probabilistic computing with sparse ising machines. *Nature Electron*, 5(149):460–468, Jul. 2022.

[222] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[223] Naeimeh Mohseni, Peter L. McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Review Physics*, 4:363–379, 2022.

[224] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the royal statistical society series b-methodological*, 48(3):259–279, 1986.

[225] Qin Zou, Kamel Merghem, and Nickson Mwamsojo. First hopf bifurcation of semiconductor lasers with delayed optoelectronic feedback produced by weak external optical feedback. In *Optique Dijon*, Dijon, France, 2021.

[226] Qin Zou, Kamel Merghem, and Nickson Mwamsojo. Optical-injection-induced period-one oscillation in semiconductor lasers with coherent optical feedback. In *Optique Dijon*, Nice, France, 2022.

[227] Qin Zou, Kamel Merghem, and Nickson Mwamsojo. Period-doubling bifurcation and intermittency route to chaos in semiconductor lasers with dual feedback. In *Optique Dijon*, Nice, France, 2022.

[228] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

INSTITUT POLYTECHNIQUE DE PARIS

**Titre:** Systèmes photoniques neuro-morphiques pour le traitement et le transport de l'information.

**Mots clés:** Reservoir Computing, Machine d'Ising, Photonique, Neuromorphique

**Résumé:** Par une utilisation performante de nombreux algorithmes dont les réseaux neuronaux, l'intelligence artificielle révolutionne le développement de la société numérique. Néanmoins, la tendance actuelle dépasse les limites prédites par la loi de Moore et celle de Koomey, ce qui implique des limitations éventuelles des implémentations numériques de ces systèmes. Pour répondre plus efficacement aux besoins calculatoires spécifiques de cette révolution, des systèmes physiques innovants tentent en amont d'apporter des solutions, nommés "neuro-morphiques" puisqu'ils imitent le fonctionnement des cerveaux biologiques. Les systèmes existants sont basés sur des techniques dites de "Reservoir Computing" ou "coherent Ising Machine." Leurs versions photoniques, ont permis de démontrer l'intérêt de ces techniques notamment pour la reconnaissance vocale avec un état de l'art en 2017 attestant de bonnes performances en termes de reconnaissance à un rythme d'1 million de mots par seconde. Nous proposons dans un premier temps une technique d'ajustement automatique des hyperparamètres pour le "Reservoir Computing", accompagnée d'une étude théorique de convergence. Nous proposons ensuite une solution au problème de la détection précoce de la maladie d'Alzheimer de type "Reservoir Computing" optoélectronique. En plus des taux de classifications obtenus meilleurs que l'état de l'art, une étude complète du compromis coût énergétique performance démontre la validité de cette approche. Enfin, le problème de la restauration d'image par maximum de vraisemblance est abordé à l'aide d'une implémentation optoélectronique appropriée de type "coherent Ising Machine".

**Title:** Neuromorphic photonic systems for information processing and transport.

**Keywords:** Reservoir Computing, Ising Machine, Photonic, Neuromorphic

**Abstract:**
Artificial Intelligence has revolutionized the scientific community thanks to the advent of a robust computation workforce and Artificial Neural Neural Networks. However, the current implementation trends introduce a rapidly growing demand for computational power surpassing the rates and limitations of Moore's and Koomey's Laws, which implies an eventual efficiency barricade. To respond to these demands, bio-inspired techniques, known as 'neuro-morphic' systems, are proposed using physical devices. Of these systems, we focus on 'Reservoir Computing' and 'Coherent Ising Machines' in our works. Reservoir Computing, for instance, demonstrated its computation power such as the state-of-the-art performance of up to 1 million words per second using photonic hardware in 2017. We propose an automatic hyperparameter tuning algorithm for Reservoir Computing and give a theoretical study of its convergence. Moreover, we propose Reservoir Computing for early-stage Alzheimer's disease detection with a thorough assessment of the energy costs versus performance compromise. Finally, we confront the noisy image restoration problem by maximum a posteriori using an optoelectronic implementation of Coherent Ising Machine.

**Institut Polytechnique de Paris**
Palaiseau 91120 Palaiseau, France