UNIVERSITÉ DU QUÉBEC À MONTRÉAL

LEARNING AND MEMORY WITHIN CHAOTIC NEURONS

THESIS

SUBMITTED

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN COGNITIVE COMPUTING

BY

MARIO AOUN

JULY 2019

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

APPRENTISSAGE ET MÉMOIRE AU SEIN DES NEURONES CHAOTIQUES

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR

MARIO AOUN

JUILLET 2019

ACKNOWLEDGEMENTS

First of all, I am grateful to God for paving me the way in the pursuit of my goals, for driving my life and giving me the faith to overcome all obstacles. I am grateful to my country Canada for giving me the opportunity to express my best self by providing me stability, security and peace of mind. Also, I am grateful to my delightful university, its friendly people and amazing excellent personnel. I can't miss to acknowledge the backing of Mr. Sylvain Le May, head of the Handicap Students Service in our university.

On the academic level, I would like to thank Dr. Mounir Boukadoum, for embracing me initially, for his hypothesis towards modeling the human memory that matched my hypothesis, and upon which we started the journey of my Phd Thesis. From our first meeting together, I sensed his opened mindset and his strong indefinable courage. Being my director I thank him for continually believing in my potential and for his supervision all these years. As well, I thank Dr. Sylvain Chartier and Dr. Jean Philippe Thivierge for their co-direction, our department and its head: Dr. Lounis Hakim.

On the research level, I would like to offer a special thank to Prof. Nigel T. Crook from Oxford Brooks University, for his ingenuity, his original research and his insights that irrigated the bloom of my research (starting from the year 2006) and made it flourish all the years afterward. I'm honored to mention my sincere respect to Oxford Brooks University for their affluent accommodation, while I was presenting my early research at their premises upon the invitation I received from Prof. Crook. That experience opened my eyes and my mind wide large in the field of chaotic neural computing and gave me the ultimate strength that I required to proceed forward.

On a personal level, I thank Chantal K., Mitri D. and Amine S. whom without their invaluable and unconditional continuous support I could neither resist nor endure my persistent physical and moral difficulties. Without them I couldn't do this thesis...

DEDICATION

This thesis is dedicated to my father who called me an engineer when I was just seven years old, who was fascinated by perpetual motion and took the opportunity to demonstrate it to me, my father who observed that same specie of flowers have exact number of petals, thus, made me wonder what are the benefits of order in the dynamics of nature?

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

LIST OF ABREVIATIONS

ACM – Association for computing machinery

AdEx –Adaptive exponential integrate and fire

AFR – Average firing rate

ANN – Average number of neurons

AP – Approximation property

ASAP – Adjustable separation and approximation properties

CC – Chaos control

CI – Cognitive informatics

CN – Computational neuroscience

DE – Differential evolution

DEA – Differential evolution algorithm

DEFR – Differential evolution using firing rates

EEG – Electro encephalogram

EPSP – Excitatory postsynaptic potential

FR – Firing rate

FRC – Frequency response curve

ISIs – Inter spike intervals

LIF – Leaky integrate and fire

LTD – Long-term depression

LTP – Long-term potentiation

MCP – McCulloch and pitts

MIT – Massachusetts institute of technology

ML – Machine learning

MLP – Multilayer perceptron

MLP – Multilayer perceptrons

NDS – Nonlinear dynamic state neuron

NRS – Neurosciences research foundation

NTC – Nonlinear transient computation

NTCM – Nonlinear transient computing machine

ODE – Ordinary differential equation

PAC – Probably approximately correct

RAF – Resonate and fire

RC – Reservoir computing

RNN – Recurrent neural network

SDFR – Standard deviation of firing rates

SFC – Spike feedback control

SP – Separation property

SRC – Sizable reservoir computing

SRM – Spike response model

STDP – Spike time dependent plasticity

SVM – Support vector machines

UPO – Unstable periodic orbit

RÉSUMÉ

La Liquid State Machine (LSM) est une théorie selon laquelle le néo-cortex du cerveau est considéré un réservoir de neurones dont les entrées provoquent des fluctuations dynamiques non linéaires de l'activité neuronale, apprises par des neurones de lecture. Des stimulus similaires provoquent des fluctuations similaires et des stimulus différents provoquent des fluctuations différentes. En conséquence, une sortie de neurones de lecture est considéré soit comme un état mental, ou comme un énoncé ou un geste, exhorté par le cerveau par son incarnation physique vers l'extérieur. Les LSMs ont trouvé une applicabilité étendue dans le domaine du Machine Learning (ML), mais ils ont en particulier surmonté les problèmes où leurs instances d'entrée varient dans le temps. De tels problèmes sont appelés problèmes d'apprentissage, de prévision et/ou de classification de données dynamiques (c'est-à-dire de séries chronologiques).

Une critique du LSM dans son explication du cerveau est sa dépendance triviale à un mode de déclenchement classique des neurones, qui est un tir régulier. Nous savons déjà que les neurones biologiques ont une large gamme d'activités de tir: régulières, toniques, éclatantes ou irrégulières... Ainsi, réduire l'activité de tir des neurones biologiques diminue l'importance de la théorie des LSM du cerveau.
Dans le contexte de ML, la conception de LSM est généralement confrontée à: Premièrement, le réglage méticuleux du nombre de neurones constituant le réservoir qui devrait être très grand pour que la machine puisse obtenir une dynamique non linéaire adéquate. Deuxièmement, la manipulation délicate des valeurs des paramètres des neurones constituant le réservoir est toujours un travail difficile, en raison que la sensibilité de cette manipulation est toujours croissante et affecte la dynamique d'activités des neurones de la machine.

Ici, j'ai abordé ces efforts. Premièrement, j'ai contesté l'idée de considérer le neurone de base d'un LSM comme un simple modèle de neurone à tir et je l'ai transformé en modèle de neurone à tir enrichi qui engendre une diversité d'activités de déclenchement neurophysiologique. Deuxièmement, je démontre que les conditions nécessaires et suffisantes de computation, comme dans le cas du LSM, peuvent être obtenues à l'aide d'un seul modèle de neurone biologique à tirs chaotiques, ce dernier exploitant le contrôle de chaos (CC). Cela signifie que j'ai minimisé le réservoir d'un LSM à un seul neurone. Ainsi, simplifier sa conception, réduire la surcharge humaine liée à la manipulation de ses paramètres et réduire le temps et les computations excessives requis à l'exécution d'un vaste réservoir de neurones. De plus, j'ai étendu cette recherche à une nouvelle théorie des groupes neuronaux basée sur la sensibilité chaotique qui pourrait expliquer la formation de l'assemblage de neurones à

l'intérieur du cerveau.

En outre, j'ai abordé une enquête sur un phénomène neurophysiologique synaptique observé in vitro, appelé plasticité dépendant du tir de synchronisation (Spike Time Dependent Plasticity – STDP – en anglais), qui décrit la proportionnalité du poids synaptique entre les neurones et leur temps de déclenchement. L'enquête focalise au causes neuronales qui pourraient causer le STDP. Ainsi, j'ai analysé le nombre d'états qui pourraient être stabilisés dans un réservoir de LSM, composé d'un nombre variable de neurones chaotiques par rapport à des neurones réguliers intégrant STDP dans leurs connexions. J'ai trouvé que les fluctuations chaotiques d'activités neuronales permettaient à une grande diversité d'états neuronaux instables d'être stabilisées et contrôlées par rapport aux fluctuations d'activitée neurales régulières. Cette tentative et ses résultats démontrent que le STDP favorise les tirs chaotiques de neurones par rapport aux tirs réguliers et offre un aperçu du rôle que le chaos pourrait jouer pour répondre aux énigmes fondamentales du STDP.

Enfin, je déduis un nouvel outil que j'appelle le taux de déclenchement chaotique, qui peut être utilisé comme substitut plus plausible au taux de déclenchement classique simulé par les neurones de "Poisson"... De même, j'enquête sur l'utilisation des algorithmes génétiques comme classificateur d'une réservoir composé de neurones de tirs chaotiques, afin de parvenir à la classification de données statiques. La démonstration de cette recherche offre une nouvelle vision dans la conception de LSM en combinant ses propriétés éssentielles: La propriété de séparation (SP) du réservoir et la propriété d'approximation (AP) du mécanisme des neurones de lecture. Cela se fait en rendant le réservoir dimensionnable dynamiquement, de sorte que sa taille soit automatiquement ajustée par l'algorithme génétique, ce qui élimine ainsi la nécessité d'avoir des neurones de lecture et joue leur rôle dans la réalisation de l'AP. Par conséquent, SP et AP d'un LSM sont unifiés et j'appelle cette unification ASAP (c'est-à-dire propriétés de séparation et approximation ajustables).

MOTS-CLÉS: Modélisation de la mémoire, réseaux de neurones, théorie du chaos, contrôle du chaos, neurones de tirs chaotiques, plasticité synaptique, plasticité dépendant du tir de synchronisation, Spike Time Dependent Plasticity (STDP), groupes de neurones, ensemble neuronal, computation de réservoir, machine à état liquide, liquid state machine, computation de transitoire non linéaire, nonlinear transient computation (NTC), taux de décharge chaotique.

ABSTRACT

Liquid State Machine (LSM) is a theory which states that the brain's neo-cortex is a reservoir of neurons where its inputs cause nonlinear dynamic fluctuations of neural activity that are learned by readout neurons. Similar stimulus causes similar fluctuations and different stimulus causes different fluctuations. Accordingly, an outcome from readout neurons is considered as either a mental conscious state, or an utterance or gesture, exhorted by the brain through its physical embodiment to the outside. LSMs have found a widespread applicability in the Machine Learning (ML) domain, but specifically they conquered problems where the input instances vary in time (i.e. time dependent). Such problems are called dynamic data (i.e. time series data) learning, prediction and/or classification problems.

A critic of LSM in theorizing the brain is its trivial reliance on a classical firing mode of neurons, which is regular spiking. We already know that biological neurons have a wide range of firing activity, being it, to name a few, regular, tonic, bursting or irregular... Thus, narrowing down the spiking activity of biological neurons diminishes the theoretical appeal of LSMs in delineating the brain.

In the context of ML, designing LSMs is generally faced with two bottlenecks. First, the meticulous tuning of the number of neurons that constitute the reservoir that, by the way, should be very large in order for the machine to achieve adequate nonlinear dynamics. Second, the delicate manipulation of parameters values of the neurons constituting the reservoir is always a thoughtful job, due to the arising sensitivity of such manipulation that affects the whole neural dynamics of the machine.

Herein, I tackled those endeavours. First, I challenged the misconception of considering the basic core of a LSM in being a simple regular spiking neuron model and altered it to be an advanced spiking neuron model that engenders a diversity of neurophysiological firing activities. Second, I demonstrate that the necessary and sufficient conditions of computing in real-time like in LSM can be achieved using a single chaotic spiking biological neuron model whilst the latter is exploiting Chaos Control (CC). This means, I minimized the reservoir of a LSM to be one single neuron, only. Thus, simplifying its design, lessening the human overload of manipulating its parameters and reducing the time and power of the excessive computing overhead resulted in executing a large

reservoir of neurons. Furthermore, I extended this feat towards a novel theory of Neuronal Groups based on Chaotic Sensitivity that could explain the formation of neurons assembly inside the brain.
Besides, I tackled an enquiry about an in vitro observed neurophysiological synaptic phenomenon called Spike-Timing Dependent Plasticity (STDP), which describes the proportionality of the synaptic strength between neurons to their firing times. The enquiry is concerned about the neural drives that could cause STDP. Thus, I analysed the number of stabilised states of a LSM reservoir composed of a varying number of chaotic spiking neurons vs. regular spiking neurons embedding STDP through their neural connections. I found that chaotic neural activity fluctuations afford a large diversity of unstable neuronal states to be stabilized and controlled than regular neural activity fluctuations. Such attempt and its results demonstrate that STDP favours chaotic spiking of neurons over their regular spiking and give a glimpse about the role that chaos could play in answering fundamental STDP conundrums.
Finally, I deduce a novel tool that I call Chaotic-firing Rate, which can be used as a more biologically plausible substitute to the classical firing rate simulated by Poisson Neurons… As well, I investigate the use of genetic algorithms as a classifier of a reservoir composed of chaotic spiking neurons, enduring CC, in order to achieve static data classification. The demonstration of such investigation offers a new vision in LSM's design by combining its essential properties: The Separation Property (SP) of the reservoir and the Approximation Property (AP) of the readout neurons mechanism. This is done by making the reservoir dynamically sizable, such that its size is automatically adjusted by the genetic algorithm, thus the latter eliminates the requirement and necessity of having readout neurons and takes their role in achieving AP. Therefore, SP and AP of a LSM are unified and I call this unification ASAP (i.e. Adjustable Separation and Approximation Properties).

KEYWORDS: Memory Modeling, Neural Networks, Chaos Theory, Chaos Control, Chaotic Spiking Neurons, Synaptic Plasticity, Spike-Timing Dependent Plasticity, Neuronal Groups, Neural Ensemble, Reservoir Computing, Liquid State Machine, Nonlinear Transient Computation, Chaotic Firing Rate.

INTRODUCTION

This thesis falls under the umbrella of investigating the potential of pulsed neural networks commonly known as spiking neural networks in forming memories. Specifically, the thesis presents a computational approach to memory, using chaotic spiking neural networks, in terms of stabilized chaotic attractor orbits. It claims that Unstable Periodic Orbits (UPOs) of chaotic attractors, depicted by chaotic firing of neurons, can be controlled and stabilized in order to activate the formation of a diversity of neuronal groups that could model the manifestation of memory. No such attempt in the current literature has done so. Our methodology is based on Reservoir Computing (RC). In other words, we validate the thesis claim in the computer domain by studying different RC neural network architectures, specifically Liquid State Machine (LSM) and Nonlinear Transient Computation (NTC), composed of chaotic spiking neuron models whilst making them exhibit Chaos Control (CC). We provide a wide range of experimental simulations and numerical analysis to demonstrate the feasibility of the thesis claim. For instance, on one side of experimentations, we accompanied CC with synaptic plasticity inside a RC architecture composed of a small number of chaotic spiking neurons and analyzed the number of different UPOs that can be stabilized upon this accompaniment. The results that we reached in this attempt show that synaptic plasticity favors chaotic spiking of neurons over regular spiking of neurons in the means of reaching a huge repository of different controlled UPOs. This is demonstrated in chapter 2. In another attempt, we excluded synaptic plasticity and the fact of having a small number of recurrently connected chaotic spiking neurons that form a reservoir, where we considered the latter to be a single chaotic spiking neuron with one condition: That it implements CC. In this experimentation shown in detail in chapter 3, we find out that the nonlinear dynamics of a single chaotic spiking neuron, upholding CC, have the essential RC property called Separation Property (SP) that is classically achieved when using a pool of regular spiking neurons (i.e. using a recurrent

network composed of a large number of regular spiking neurons). Specifically, we show that by using a single chaotic spiking neuron, we are able to create a RC framework that is able to discriminate (i.e. separate) between similar or different inputs by stabilizing the neuron dynamics (i.e. the UPOs of the reservoir composed of a single chaotic spiking neuron) to similar or different spiking patterns relative to the inputs, respectively. The novelty and originality of this reservoir composed of a single chaotic spiking neuron is that it contrasts the classical requirement, mainly followed and agreed upon in the current literature, of having a large number of regular spiking neurons, in order to achieve RC, as is the case for a LSM. Furthermore, this single chaotic spiking neuron's reservoir is extended to work as a memory-representation system by linking its stabilized UPOs, depicted in periodic spiking patterns through CC of the chaotic spiking neuron dynamics, to a layer of resonant neurons that fire as a group when the reservoir (e.g. the chaotic spiking neuron) is faced with a specific input. This group of resonant neurons, or neural ensemble, is interpreted as a representation of the memory of the input. In addition, and most importantly, we show that the reservoir (e.g. the chaotic spiking neuron) is noise tolerant, which means that similar inputs to it will activate the same group of resonant neurons. Thus, we have created a neural network architecture that enables us to relate similar stimulus, or a class of stimuli, to a neural ensemble that could be interpreted as the memory representation of the class of similar input stimulus. Third, we extend our earliest experimentations, and their noticeable results (Aoun & Boukadoum, 2014, ibid, 2015), which target machine learning, that we came up through out the course of studying recurrent neural network architectures of chaotic spiking neurons. In fact, these earlier experimentations (Aoun & Boukadoum, 2014, ibid, 2015) relied on the control of spike coding generated by chaotic spiking neurons and incorporated synaptic plasticity in order to achieve machine learning in terms of dynamic (i.e. time series) data classification. However, our latest experimentations, herein presented in chapter 4, relied on controlling the firing rate of chaotic spiking neurons and incorporated an evolutionary algorithm in order to achieve machine learning in terms of static data classification. Thus, we have tackled dynamic data classification and static data classification using chaotic spiking neural networks. The major interpretation of the

results of this thesis yields to an acknowledgment of the potential of chaotic spiking neurons and opens the doors for further investigation of their promising impact towards synaptic plasticity, memory modeling, reservoir computing and machine learning. So, in order to grasp and understand the thesis fundamentals, its claims, approach and methodology, then let us start by introducing the thesis background, the neurophysiological scientific experiments that relayed its hypothesis and the mathematics behind it. All of this is available in Chapter 1. The investigation of STDP is presented in chapter 2. The theory of neuronal groups based on chaotic sensitivity is presented in chapter 3. Chaotic firing rate is presented in chapter 4. The last chapter (chapter 5) concludes the thesis.

CHAPTER 1

FUNDAMENTALS, BACKGROUND AND THESIS INITITATION

1.1 Scope: Cognitive informatics and its current paradigms

Prof. Yingxu Wang coined the term Cognitive Informatics at the occasion of the International Conference on Cognitive Informatics (ICCI) in 2002. According to him,

> Cognitive Informatics (CI) is a contemporary multidisciplinary field spanning across computer science, information science, cognitive science, brain science, intelligence science, knowledge science, cognitive linguistics, and cognitive philosophy. CI aims to investigate the internal information processing mechanisms and processes of the brain, the underlying abstract intelligence theories and denotational mathematics, and their engineering applications in cognitive computing and computational intelligence (Wang *et al*, 2013).

In this document I will present to you my Phd Thesis which respects the aims of CI that I just mentioned and incorporates different theories from the following fields: Computer science, information science, cognitive science, computational neuro science and Physics.

According to Eliasmith's 'mind-brain metaphors' (Eliasmith, 2003), the current CI paradigm, in the delineation of the 'mind', relies on three commonly known approaches that can intermingle: The symbolic approach, the connectionist approach and the dynamical systems approach. We will summarize these approaches next, by

referring to the 'mind-brain metaphors' in Eliasmith (2003). Symbolism considers the mind as software and the brain is its programming framework. In other words, symbolism reduces thought to elementary mental processes that are the outcomes of logical units following operations on simple rules. The connectionist approach looks at the mind as the outcome, behavior or function of a black box composed of neuron models which we can't explain the processing, or the flow of information that takes place through their interconnections. Note that the connectionists and the symbolists are both materialists because they both agree that the mind is the brain. They both rely on representations and computations, which their system should adopt in order to perform cognition. However, the dynamists say the mind is just a dynamical system and one should not rely on specific representations or mode of operation to describe its intelligent behavior. Eliasmith referring to Watt Governor and van Gelder says: "Cognitive systems are essentially dynamic and can only be properly understood by characterizing their state changes through time" (Eliasmith, 2003). In such context, we should refer to the field medalist Stevan Smale and his open problem on the limits of Intelligence (Smale, 1999). In stating the problem, Smale calls for the quest of a model that can explain intelligence. He says that such model should not be necessarily a unique one. He makes a comparison to physics in which we have classical Newtonian physics, relativity theory and quantum mechanics, each one with its own insights, understanding and limitations of physical phenomena. Smale says: "Models are idealizations with drastic simplifications which capture main truths" (Smale, 1999). Furthermore, Eliasmith makes a comparison to the contemporary paradigm of delineating "What is the Mind?" to the question of "What is the Nature of Light?" (Eliasmith, 2003). In the nineteenth century, the scientific quest that can provide the answer and the science to the nature of light was tackled by two different approaches: One considered light as a particle phenomenon, while the other considered light as a wave. Now, we know that light has a 'wave-particle duality' according to the Heisenberg uncertainty principle. Both approaches are true and provide a scientific ground to study the nature of light whereas each approach

compliments the other, and their combination leads to better understanding and comprehension of what light is!

In our work, we follow such analogies made in physics, to lay the ground of our quest in investigating the nature of intelligence under the umbrella of Cognitive Informatics (CI). Our work of delineating the mind relies on a duality between the connectionist approach and the dynamical systems approach: The former studies interconnected neuron models in order to explore their neural computing capabilities, while the latter considers the interconnected network of neurons as a dynamical system and studies its promising cognitive behavior by exploring the possibilities of its neural state changes. First let us explain what we mean by a dynamical system, define neural computing, and introduce biological neuron models and the way they communicate through their interconnections. This way of introducing those basic fundamentals facilitates the comprehension of the problem that we want to tackle and which we will describe afterwards.

1.2 Formal definition of a dynamical system

When a physical object changes its behavior through time, we say that this object has a dynamical behavior and the dynamics can be studied through a system of mathematical equations. So, lets say we have an object that is changing its behavior through time thus we can call a variable x as the state of the object and define it either over a strict period of time or over a single moment in time, depending on the behavioral timely outcome of the object. Furthermore, we can define the behavioral change of the object as a function f that depends on x. Note that the set of states that constitutes the object behavior is called its state space. We call this relation between the states of the object and its behavior a dynamical system. Referring to Weisstein, a dynamical system is "a means of describing how one state develops into another state over the course of time" (Weisstein, E. W., "Dynamical System" From *MathWorld* -- A Wolfram Web Resource,

http://mathworld.wolfram.com/DynamicalSystem.html). In the mathematical sense, this can be written as the following difference equation:

$$x_{n+1} = f(x_n)$$

Where n denotes discrete time as nonnegative integer (**Z**-*) and f a continuous function.

Lets substitute 'the previous state of x' from both sides of the equation above, doing so, the latter is now translated to a new difference equation as:

$$x_{n+1} - x_n = f(x_n) - x_n$$

Thus, we can introduce a new function g to be the description of the difference between $f(x)$ and x (i.e. their equivalence) as it follows:

$$g(x) \equiv f(x) - x$$

Which can be mathematically written using the following difference equation:

$$x_{n+1} - x_n = g(x_n) * 1$$

So, the difference between two consecutive states is equal to the difference between the current behavior *'f'* of the system and its last state *'x'*. Consequently, this defines the derivative of the current state, which can be mathematically written as:

$$x'(n) = g(x(n))$$

In this section, we introduced the notion of a dynamical system that describes an object's dynamical behavior and its state's change in mathematical terms.

1.3 Neural computing and neuron models

According to the definition given in (Aleksander and Morton, 1995), "Neural Computing is the study of networks of adaptable nodes which, through a process of learning from task examples, store experiential knowledge and make it available for use." The adaptable nodes are considered to imitate the neurons of the brain, "which acquire knowledge through changes in the function of the node by being exposed to examples" (Aleksander and Morton, 1995). Also, we refer to the definition of "Learning" given by Herbert Simon, in 1983, as: *"In a system, Learning denotes adaptive changes in the sense that they enable the system respond to the same task or tasks drawn from the same population more efficiently and more effectively the next time"*. In the next section (1.3.1), we will introduce the biological neuron and then we will describe different neurons models that can constitute the basic elements of a neural network so the latter can achieve neural computation and learning.

1.3.1 The biological neuron

From an analytical perspective, a biological neuron can be considered as an elementary processing cell as initially suggested by Ramón y. Cajal, in 1933. It is composed from a central cell body called *soma*, an output bus called *axon* and input hubs called *dendrites* (Kandel *et al*, 2000). A typical schematization of a neuron is shown in figure 1.1 below:

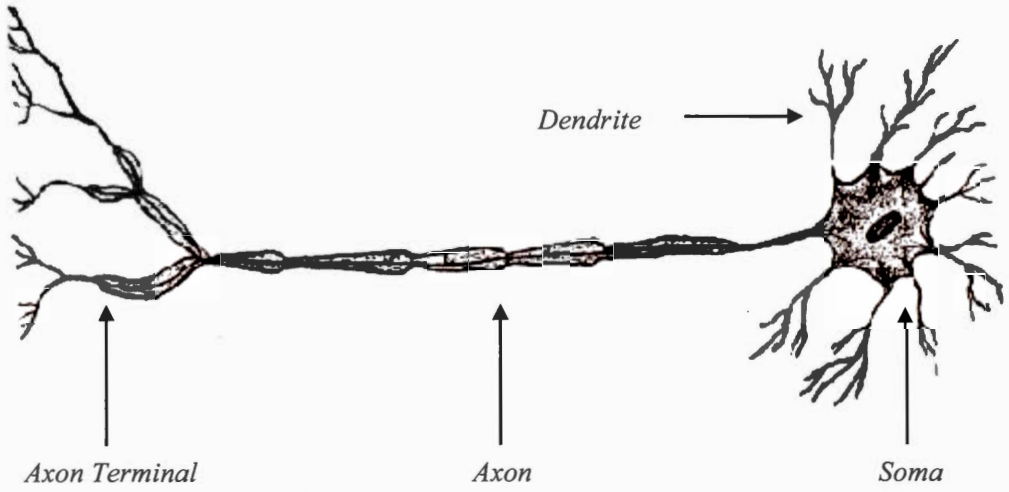


Fig. 1.1 Typical sketch of a biological Neuron

Neurons communicate with each other by streams of electrical pulses or *Action Potentials;* this happens via electrochemical conjunctions called *Synapses* (Kandel *et al*, 2000). Synaptic connections are said to be either excitatory; thus, propagating the electrical pulses of the neuron and leading to an excitation in the target neuron, or inhibitory; decaying the electrical potential of the target neuron (Kandel *et al*, 2000). The process of excitation and inhibition through an interconnected neural circuit generates its main feature of adaptability. For instance, the interconnected neural circuits will enable a living organism to perform actions in its environment, respond to stimuli and adapt to life. Biological neurons "communicate through pulses and use the timing of these pulses to transmit information and to perform computation" (Mass and Bishop, 1999, Preface, page 24). These pulses are commonly known as Spikes. But, before we proceed by the most prominent and realistic modeling of a biological neuron, which is called Spiking Neuron, we have to introduce the simplest basic and classical models of artificial neurons.

1.3.2 Basic neuron models

Inspired by the physiology of a biological neuron (Cajal, 1933), many neuron models

were envisaged afterwards like the earliest basic artificial neuron model of McCulloch and Pitts (1943) and the contemporary Spiking Neuron Model developed by Izhikevich, in 2007.

1.3.3 Threshold gates: McCulloch and Pitts neuron model

In 1943, McCulloch and Pitts proposed the first basic neuron model for neural computation. The model gained major attention in the engineering community due to its applicability in electronic circuits. The McCulloch and Pitts – MCP – model relies on the representation of a neuron as an elementary device, which works as a threshold function that sums up weighted input and fires accordingly.

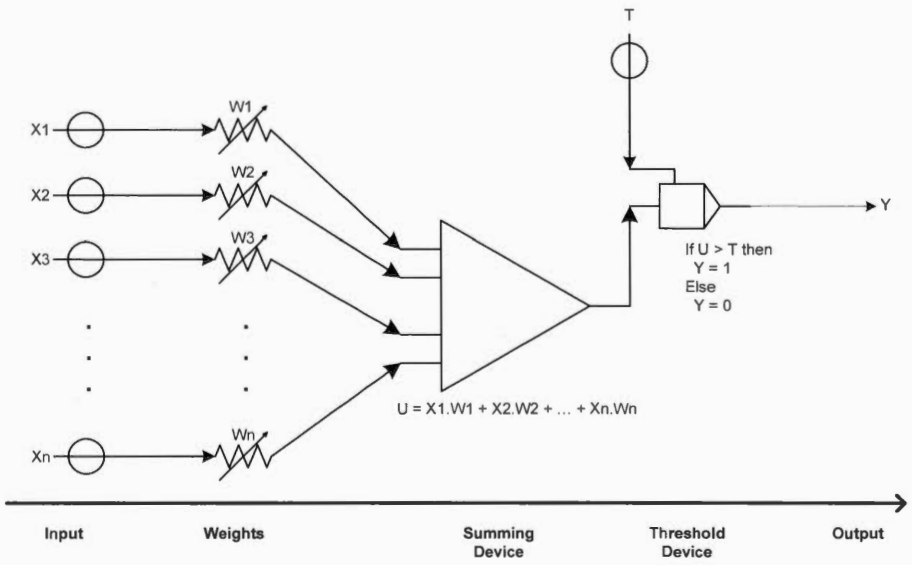


Fig. 1.2 The McCulloch and Pitts Neuron Model

As we can see in the diagram above (Figure 1.2), the synaptic effect is modeled with weight variables. The neuron is said to fire, and has an output equal to 1, this is if the presented input scaled by the connection weight is greater than the neuron threshold. In a mathematical form, when a binary input vector X is presented to the neuron that

has a predefined threshold positive real number T, then the binary output Y of the neuron is given by:

$$Y = F(\sum_{i=1}^{n} X_i.W_i)$$

Where F is a hard limiting function:

$$F(x) = 1 \text{ if } x > T \text{ otherwise } F(x) = 0$$

The basic component in classical neural computation is the McCulloch and Pitts model presented above, algorithms target weights adjustment and variations target the choice of output function F, as we will see next.

1.3.4 Sigmoidal gates: Analogue output

As we have just seen in the previous section, the output function of the McCulloch and Pitts model is a simple hard limiting function, which only communicates binary values (i.e. 0 or 1).

Another choice could be a function that will give a continuous output of the neuron; which tends to 1 when its input is very large (i.e. goes to $+\infty$) and to 0 when its input is very small (i.e. goes to $-\infty$). This can be achieved by a different choice than the Hard Limiting function, in this way the sigmoid activation function can be considered and is given next:

$$F(x) = \frac{1}{1 + e^{-x}}$$

The plot of $F(x)$ is shown in Figure 1.3:

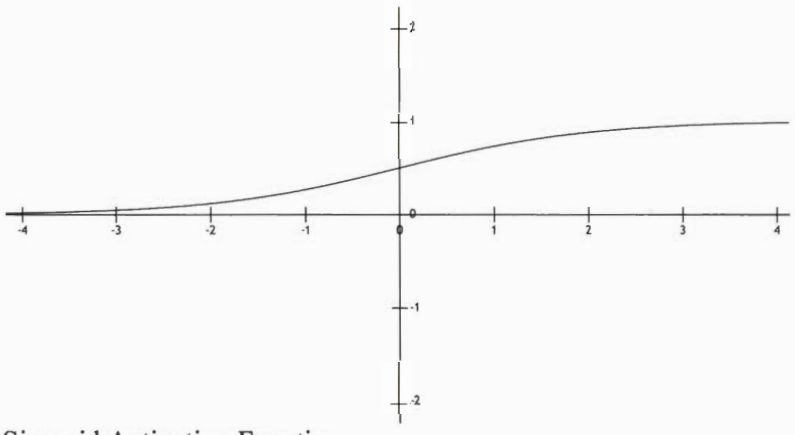


Fig. 1.3 The Sigmoid Activation Function

Note that the derivative of F is equal to:

$$F'(x) = F(x).(1 - F(x))$$

We can notice that when *F(x)* is equal to *1* then the derivative is equal to *0* and when *F(x)* is equal to *1* then the derivative is also equal to *0*. Thus, when the neuron achieves its boundary values, the rate of change of its activation settles to *0*.

By considering *F* as a continuous and differentiable function with *0* and *1* as its boundaries, we can calculate the derivative of the neuron function (i.e. its rate of change).

The synaptic weights affect the activation function of the neuron because they are part of its input; this means they can drive its performance. We can manipulate the synaptic weights based upon the modification they cause to the activation function. This means we apply the derivative of the performance based on the synaptic changes in order to update the synapses. This cannot be done if the activation function is a hard limiting function, because the derivative of a hard limiting function does not exist at *0* and its derivative is *0* elsewhere. Using a sigmoid function allows the

computation of analogue output and not only Boolean output as in the case of a threshold gate (Maass *et al.*, 1991). Furthermore, the efficiency of a neural network is translated by the means of minimizing an error function that calculates the difference between the desired output and its actual output. This is essential in order to train a network of neurons and test its performance before it processes new input. For instance, this is well elaborated in feed forward networks as Multilayer Perceptron – MLP – (Rumelhart et al, 1986) and Recurrent Networks as Hopfield Networks (Hopfield, 1982); where an error function is optimized. According to this optimization process that is tested by being minimized, a neural network will be attributed the characteristic of having a learning capability by generalizing over its inputs, thus satisfying the definitions of learning and neural computing that we introduced in section 1.3.

1.3.5 Spiking neuron model

To exemplify the idea of a spiking neuron, I will start by the definition, and illustration (Figure 1.4) - of a *spike* as given in the book "Dynamical Systems in Neuroscience" by Izhikevich, in 2007:

> A spike is an abrupt and transient change of membrane voltage that propagates to other neurons via a long protrusion called an axon. Spikes are the main means of communication between neurons. In general, neurons do not fire on their own; they fire as a result of incoming spikes from other neurons… (Izhikevich, 2007)

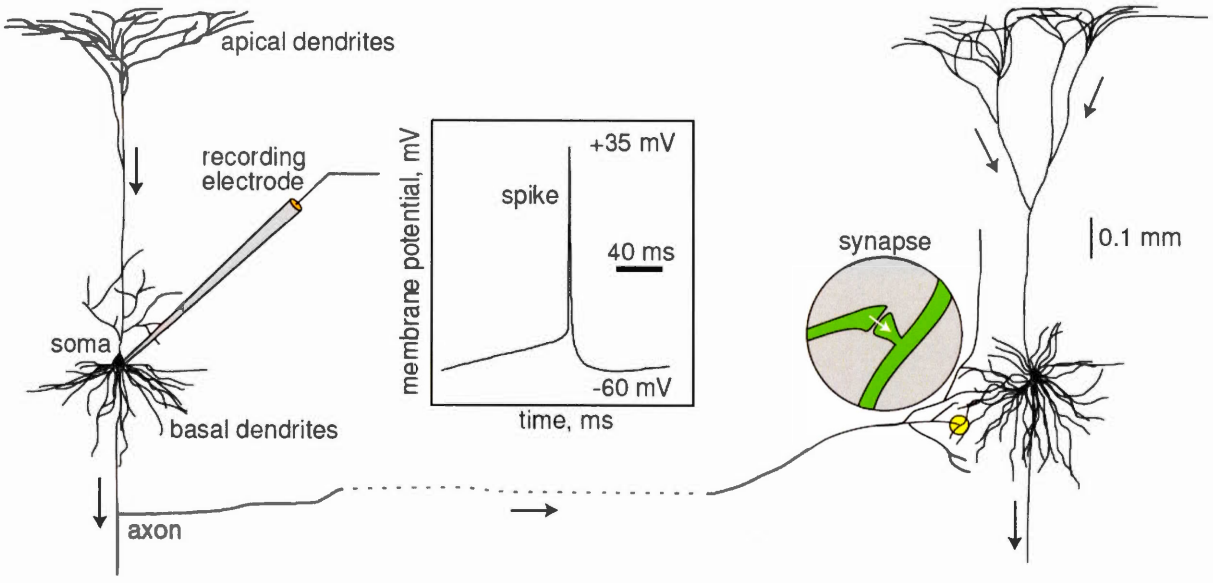


Fig. 1.4 Illustration of a *spike*, excerpt from (Izhikevich, 2007)

From an analytical perspective, "Spiking Neurons are models for the computational units in biological neural systems where information is considered to be encoded mainly in the temporal patterns of their activity" (Maass and Schmitt, 1997). Thus, a spike train F of a neuron i is the set of firing times of this neuron as described next:

$$F_i = \{t_i^{(1)}, t_i^{(2)}, \dots, t_i^{(n)}\}$$

To comprehend the concept of a spiking neuron, I will refer to the "Spike Neuron Model of Type A" of Maass (1997); this is due to its simplicity in modeling a pulse as a step function, and its mathematical analysis that was later depicted in Maass and Schmitt (1997). The model is shown in the diagram (Figure 1.5) below:

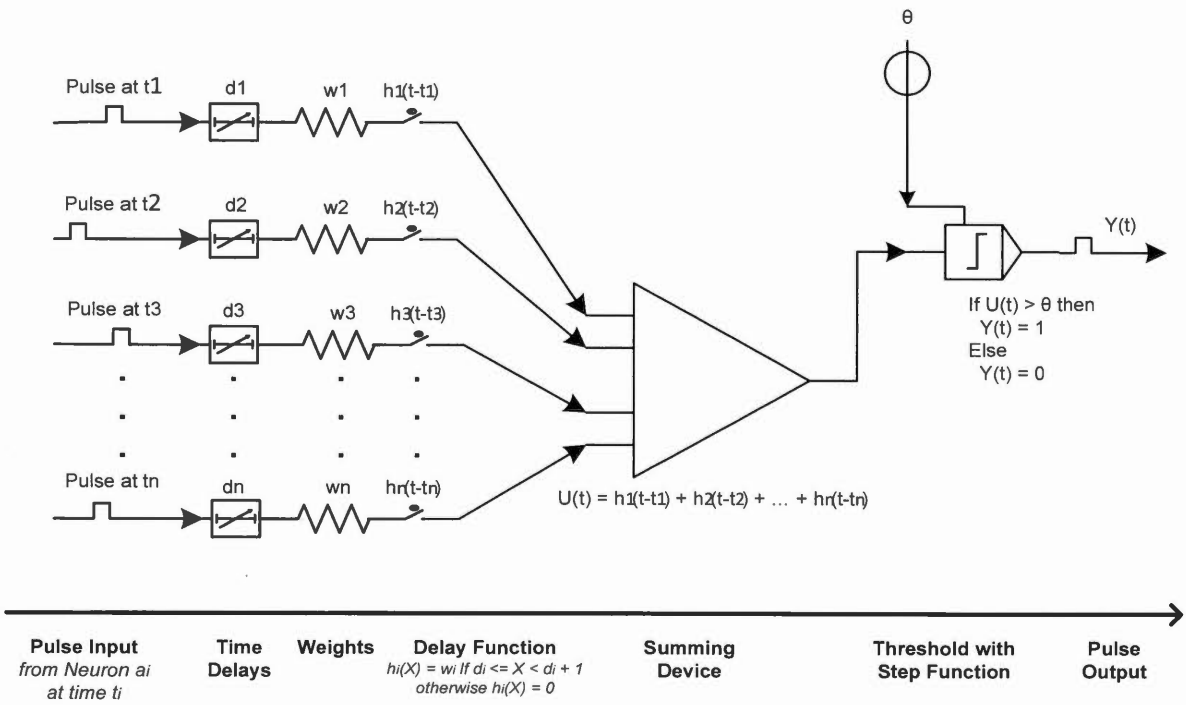


Fig. 1.5 Spike Neuron Model

The output of a spiking neuron v, that has $i = 1, \ldots, n$ input connections from $a_1, \ldots, a_n$ neurons with weight $w_i \in \Re$ and delay $d_i \in \Re^+$ each, is given by:

$$P_v(t) = \sum_{i=1}^{n} h_i(t - t_i)$$

Where,

$$h_i(x) = 0 \text{ for } x < d_i \text{ or } x \geq d_i + 1$$

And,

$$h_i(x) = w_i \text{ for } d_i \leq x < d_i + 1$$

We note that if the neuron a_i fires at time t_i this causes a pulse at time t on v of the form $h_i(t - t_i)$. The neuron v fires a pulse as soon as $P_v(t)$ becomes greater than a threshold θ_v (Maass and Schmitt, 1997).

Other spiking neuron models (Izhikevich, 2003, Brette and Gerstner, 2005) are the most prominent nowadays in the domain of computational neuroscience and neuro-computation due to their ease of implementation and fast time execution. Besides, recent studies in neuroscience as summarized in (Izhikevich, 2006) suggest that the exact firing time between neurons has much more influence on information processing rather than the firing rates of these neurons; as it was previously thought (Rieke et al, 1997, Shadlen et al, 1998). The concurrency of firing times that occurs between neurons is a major feature for information binding, memory retrieval and temporal coding inside the brain (Izhikevich, 2006). In fact, if multiple neurons fire synchronously then their pulses or spikes arrive at a target neuron with the same firing time, thus causing the latter to fire with a greater probability than when it's pulsed with multiple spikes randomly or at different times (Izhikevich, 2006). Check Annexe I for a further explanation about Spiking Neurons in modeling the biological neuron based on different levels of its abstraction. As well, you can check Annexe II for a comparison between Rate Coding and Spike Coding.

1.4 General description of the thesis problem and guidance to follow

Leslie Valiant introduced some of the foundational basics of computational complexity theory concepts based on the framework of PAC (Probably Approximately Correct) learning that he envisaged in 1984. In 2003, Valiant proposed three open problems in computer science. The third problem is entitled: "Characterizing Cortical Computation" (Valiant, 2003). In this problem, Valiant is asking "*How knowledge is represented in the brain and what the algorithms are for computing the most basic behavioral tasks*". An example of a basic behavioral task is memorization (i.e. a functional characteristic of a system, which is embedded in the system and works in a way to provide the system the ability to memorize a scene or recalls an event...). Valiant is requesting "a model of computation that describes the

essential capabilities and limitations of the brain for computing such functions" (Valiant, 2003). This request coincides with the same endeavor of Steve Smale towards his quest of the limits of intelligence (Smale, 1999). As we mentioned in Section 1.1, Steve Smale gave a list of open problems for the new century, the 18th problem in his list is called: "Limits of Intelligence" (Smale, 1999).

First, Smale in his attempt at providing guidance to the scientific quest for a model of intelligence, points to an important ingredient that the cognitive system in question should encapsulate: It is the feature of random viability. He claims "randomness in the input and in the processing itself would seem to be an important ingredient in our search for models of intelligence" (Smale, 1999). One could suggest that since a chaotic system, in its long run, performs quasi-random behavior then this system could be exploited towards a model of intelligence that fulfills the important ingredient of randomness, which Smale is requesting.

Second, Valiant proposes that the cognitive system should be based on a neural network model and should satisfy four biological criteria that are deemed essential in order to characterize its cognitive behavior. The four requirements that the model should fulfill are: "The strength of synapses, the accuracy of timing mechanisms, the existence of state not just at synapses but also globally in a cell, and the numerical parameters of cell interconnectivity" (Valiant, 2003). Conventionally, in the connectionist approach, the strength of the synapses can be tackled through models of synaptic plasticity. The timing mechanisms can be envisaged by using neuron models that exhibit spiking activity. The numerical parameters of cell interconnectivity can simply be initiated according to the boundaries of the model. What is remaining is the third property, which is the most important in our approach, and which Valiant is demanding: "The existence of state not just at synapses but also globally in a cell" (Valiant, 2003). Our attempt is to use a neuron model that can fulfill such condition: The Nonlinear Dynamic State (NDS) Neuron (Crook et al, 2005) developed by Nigel

Crook. The NDS Neuron is a chaotic spiking neuron model, which encapsulates the concept of state at the 'cell' level because its behavior can be very well ordered through chaos control; such that the latter (i.e chaos control) can lock the neuron's behavior in a single state found in its chaotic repertoire of infinitely many states. Furthermore, we want to extend the theories behind the NDS Neuron and apply them to other feasible neuron models, that are biologically plausible, like the Adaptive Exponential Integrate and Fire (AdEx) Neuron model (Brette and Gerstner, 2005) or the Izhikevich Neuron model (Izhikevich, 2003), which can also generate states when they are configured to run in chaos mode whilst applying chaos control…

In this section we highlighted the importance of **the concept of 'state' at the neuron level** and we introduced the notion of **chaos** as being an essential ingredient to be considered in attempts at building intelligent systems. Before we declare our thesis statement, we shall delineate the fundamental theories behind it: Chaos theory and the NDS Neuron theory. Also, we shall delineate the neurophysiological theory upon which our thesis builds its hypothesis and finds its support: The theory of chaotic neuro-dynamics by Walter J. Freeman (1991). Note that the NDS Neuron is included in the fundamentals because it is a neuron model that is based on the exploit of chaotic nonlinear dynamics and is solely chaotic, as we will see next, furthermore, it's the initial model that we used in our early experiments while exploring the topic of this thesis. These theories are emphasized and completed in the subsequent sections of this chapter. In the chapters afterwards, the AdEx Neuron will be used, because it has the same characteristics of the NDS Neuron, but it is, also biologically plausible. Besides, we favored the AdEx Neuron model over Izhikevich Neuron model, because we noticed through experimentation that it is more adequate and more reliable in simulating chaotic dynamics than the Izhikevich Neuron.

1.5 Chaos theory and the nonlinear dynamic state neuron

Chaos is a new Science which establishes the omnipresence of unpredictability as a fundamental feature of common experience [...] Chaos is a characteristic of dynamics, and dynamics is the time evolution of a set of states of nature (Smale, 1998).

An example from nature that embeds the characteristic of chaos in its dynamics; and is familiar to us, is the weather. The weather is defined as the state of the atmosphere at a place and time in regards to heat, dryness, sunshine, wind, rain, etc. Since the latter properties are interrelated, thus they constitute a nonlinear relationship between each other in relaying the state of the weather. This means, the weather system is nonlinear in its core. Also, it is obviously dynamic because it evolves with time. But, why is the Weather chaotic? Because it depends on numerous of such variables and because measuring all these variables is very delicate, thus predicting the behavior of the weather, using a weather forecasting system, will be highly dependent on the accuracy of measurements of its variables. In other words, slight realistic variations of the measurements (called system initial conditions in mathematical terms), which constitute the input variables of a weather forecasting system, will get enormously magnified through the internal dynamics of the system when the latter runs for too many time steps in the long future, however they slightly affect the evolution of the internal dynamics of the system when the latter runs for few time steps ahead in the near future. This is the reason that the weather is easily predictable in the short run and is highly unpredictable for the long run.

So, to start considering a nonlinear dynamical behavior as chaotic, then it should have at least such fundamental characteristic of high sensitivity on initial conditions. I will not go back neither to the origins of Chaos theory nor to the results of the famous computer experiments of Prof. Edward Lorenz, done in 1963, while he was developing models of atmospheric convection (Lorenz, 1963). For further details on the topic, I will refer the reader to the book "A Survey of Nonlinear Dynamics: Chaos Theory" by Richard Lee Ingraham (1991), and to the paper of Aubin and Dalmedico (2002).

In the common sense, the notion of 'chaos' is attributed to a nonlinear dynamical system that exhibits sensitivity on initial conditions as we just explained. However, in the formal sense, the notion of 'deterministic chaos' is attributed to such system, why?

Answer: The term "deterministic chaos" is used in mathematics because it is more accurate and tangible in designating a nonlinear dynamical system with chaotic behavior and it avoids the confusion between the terms chaos and randomness when taken in their formal sense. In deterministic chaos, randomness is considered as the unpredictability of the behavioral outcome of the system at the long run, and determinism refers to deterministic rules that are set upon the variables of the system, inscribed in mathematical equations, their simulation through time offers further analysis of the evolution of the system. For instance, in classical mechanics the behavior of a dynamical system can be described geometrically as motion on an attractor. The mathematics of classical mechanics effectively recognized three types of attractors: single points (characterizing single point steady states), closed loops (periodic cycles characterizing periodic steady states) and tori (combinations of several cycles)... In the 1960's, chaotic attractors were discovered; for which the dynamics is chaotic (https://www.britannica.com/science/chaos-theory#ref251592).

Now, we will dig deeper by examining a nonlinear dynamical system exhibiting deterministic chaos. To do that we choose the nonlinear dynamical system realized by Otto Rossler, in 1976. This choice is due to the simplicity of the dynamical equations of the system. The Rossler system is composed of three variables only which their behavior is described by three nonlinear Ordinary Differential Equations - ODEs - (an ODE is an equation containing a function of one independent variable and its derivatives). The equations that define the Rossler system are:

$$x' = -(y + z)$$

$$y' = x + ay$$

$$z' = b + z(x - c)$$

They can also be written in their difference form as:

$$x(t) = x(t-1) - y(t-1) - z(t-1)$$

$$y(t) = y(t-1) + x(t-1) + ay(t-1)$$

$$z(t) = z(t-1) + b + z(t-1)(x(t-1) - c)$$

Where,

a, b, and *c* are constants with $a = 0.2$, $b = 0.2$ *and* $c = 5.7$

t is the time step, such that the system starts at $t = 1$ with *x(0),y(0)* and *z(0)* equal to any real value in R.

If we plot any of the three variables versus time, we simply see a trajectory that represents the chaotic evolution of the variable through time. However, if we plot each variable one versus the other in a three-dimensional space where each dimension corresponds to the values of one variable, then we see a 3D trajectory that describes all possible states of the system. We call this 3D graphical representation of the three variables: The phase space diagram. In other words, a phase space diagram gives us a description of all possible states of the system. By examining the phase space of the Rossler variables we find that they are constrained (i.e. attracted) to a strict 3D geometric object and won't escape its boundaries. We call this geometric object a chaotic attractor and it is visualized in phase space like the following:

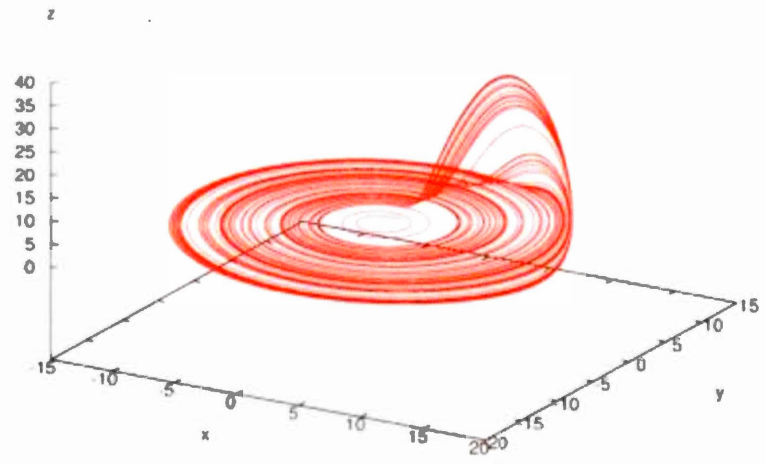


Fig. 1.6 The Chaotic Attractor of Rossler system visualized in phase space

If we examine Figure 1.6, we notice that the chaotic attractor is composed of a dense collection of points, which seem to embed a periodic pattern of slightly varying similar orbits that are partially overlapping. The reason behind this partial overlapping of orbits is due to the chaotic nature of the system which gives the chance of any point in a neighborhood of previously visited points of an orbit in the phase space to be visited again by another orbit. This is why these orbits are called: Unstable Periodic Orbits (UPOs). Thus, a chaotic attractor can simply be defined as a structure that is composed of infinitely many UPOs: It is a dense set of UPOs. Using methods of Chaos control, we can manipulate the system of differential equations and make their outcome settle to a single UPO.

Next, we will introduce the equations of the Nonlinear Dynamic State (NDS) Neuron that were derived from the equations of the Rossler system by Alhawarat, in 2007. In addition, we will present the chaos control method that makes the behavior of the NDS neuron settle to a single UPO (Crook et al, 2005).

The equations of the NDS Neuron are the following:

$$x_i(t) = x_i(t-1) + b(-y_i(t-1) - u_i(t-1))$$

$$y_i(t) = y_i(t-1) + c(x_i(t-1) + ay_i(t-1))$$

$$u_i(t) = \begin{cases} n_0, u_i(t-1) > \theta \\ u_i(t-1) + d(v + u_i(t-1)(-x_i(t-1)) + ku_i(t-1)) + I_i(t), u_i(t-1) \le \theta \end{cases}$$

where,

a=0.002, b=0.03, c=0.03, d=0.8, v=0.002 and k=-0.057 are constants (Crook et al, 2005, Alhawarat, 2007).

x_i and y_i are considered as the internal variables of a NDS Neuron *i*.

u_i is considered as the voltage output of the NDS Neuron *i*.

θ is a parameter called the Voltage threshold of u_i and is set to *0*.

n_0 is called: Voltage reset of the NDS neuron (i.e. the value that takes the neuron voltage u_i when the latter bypasses the threshold θ). It is a parameter that is set to *-0.7*.

The initial conditions of the NDS Neuron *i* are: $x_i(0) = 0$, $y_i(0) = 0$, $u_i(0) = n_0$ and the system starts at $t = 1$.

By comparing the Rossler variables to the NDS Neuron variables, we notice that the variable *z* is substituted by *u*. The reason behind this substitution is simply for interpreting *z* as a voltage that is commonly prescribed by the letter *u* in conventional spiking neuron models.

As we mentioned earlier, neurons communicate with spikes, which encode their behavioral activity. So, in order to incorporate the spike phenomenon in the NDS Neuron model, a variable γ is added to the model and is defined as it follows:

$$\gamma(t) = \begin{cases} 1, u(t) > \theta \\ \\ 0, u(t) \leq \theta \end{cases}$$

γ is interpreted as the spike output of the neuron which is equal to 1 when the neuron voltage u crosses its threshold θ and 0 elsewhere.

Next, we will visualize a trail of spikes output of the NDS Neuron based on the variable γ that alternates its value as being either 0 or 1 depending on the evolution of u.

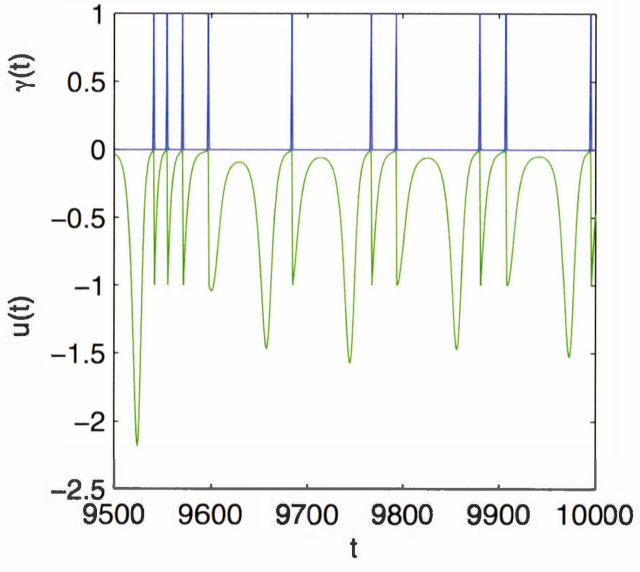


Fig. 1.7 The NDS Neuron voltage u shown in green color and its spikes output γ in blue versus time t.

By examining the output of u and γ in figure 1.7, we notice that they don't show any regularity; this is obvious because the NDS Neuron is a chaotic neuron model. Yet, in order to visualize the chaotic attractor of the NDS Neuron variables, then we should plot its phase diagram as we did for the Rossler equations previously. Note that, for the sake of simplicity, we will show the phase diagram of the NDS Neuron in two dimensions only. To do that, we choose to plot u versus x in figure 1.8, next:

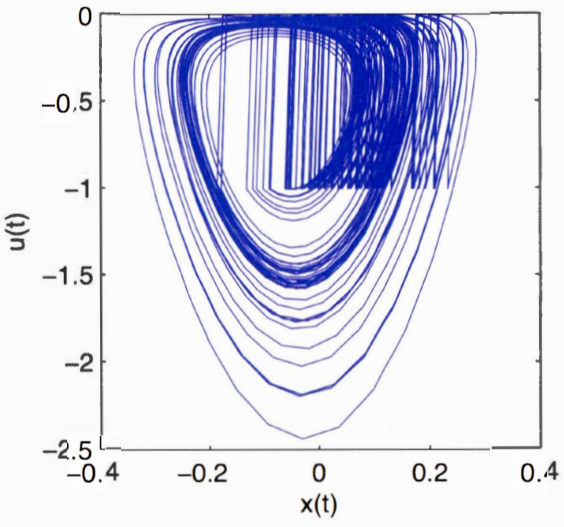


Fig. 1.8 The phase space diagram, in two dimensions, of the NDS Neuron, which shows its chaotic attractor.

If we examine the structure of the chaotic attractor of the NDS Neuron, we notice that the chaotic attractor of the NDS Neuron is composed of (UPOs). However, the methods of chaos control can be applied on the variables of a chaotic dynamical system in order to make them settle to a single UPO. For instance, to achieve chaos control in a NDS Neuron and make it settle to a single UPO, we assign its input I to its spike output γ that occurred τ time steps in the past, scaled by a factor w. This is interpreted in mathematical terms, as:

$$I(t) = w\gamma(t-\tau)$$

We can construe the formulation of I as the implementation of a self-feedback connection of the NDS Neuron on itself, which has a time delay that is defined by a parameter τ. As for w, it is considered as a degree of freedom interpreted as the weight of this self-feedback connection. This method of chaos control applied to the NDS Neuron is called: Spike Feedback Control – SFC – (Crook et al, 2005). Figure

1.9 shows an UPO cached by applying SFC and its discrete manifestation in a regular spikes output pattern of period τ. In this example, we choose τ to be equal to 100.

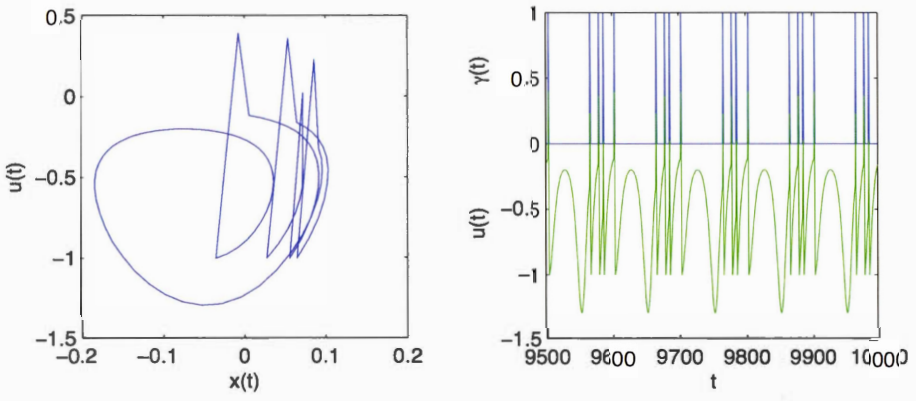


Fig. 1.9 Chaos Control: On the left, a settled UPO through chaos control in phase space. On the right, top plot, the discretized version of the UPO shown as a spikes trail of period $\tau = 100$ containing 4 spikes. On the right, bottom plot, the evolution of the neuron voltage u.

By varying the initial conditions, the degree of freedom w and/or the delay τ we will theoretically reach an infinite number of different UPOs; discretized by trails of spiking activity of the neuron.

The repertoire of UPOs that can be controlled using the method of chaos control molded in terms of SFC and discretized in spikes patterns can be considered as the vocabulary of the NDS Neuron (Crook et al, 2005). Thus, a discretized version of a UPO is an expression of a neuronal state of the NDS Neuron. Furthermore, it was shown that the number of distinct UPOs that can be achieved using chaos control is in the order of hundreds of thousands and is theoretically infinite (Alhawarat, 2007).

In this section, we introduced chaos theory and the NDS Neuron model. Furthermore, we introduced the main feature of 'states' of the NDS Neuron through chaos control, satisfying Valiant third vision (discussed previously); which suggests that we have to

extend the concept of states from the synaptic domain to the neuron domain. The NDS Neuron is a chaotic spiking neuron model and operates using time delays, which makes it plausible in the field of computational neuroscience. In addition, it satisfies Valiant suggestion because it has theoretically infinitely many states. Next, we will explain the theory of chaotic neurodynamics upon which we base our hypothesis.

1.6 Our hypothesis based on stabilization of chaotic attractors

In 1991, Freeman found that when an organism (i.e. an animal) is not smelling any odorant, it's brain Electro Encephalogram (EEG) signals activity can be represented in a phase diagram which shows a chaotic attractor, composed of dense orbits with very high degree of irregularity describing a high level of chaos. However, when an familiar odorant is offered to the organism then the attractor, still chaotic, became more regular and constrained to fine orbits, which means that the organism has either learned the odorant (i.e. cognized the odorant and generated a new memory representation for it) or remembered the odorant (i.e. recognized the odorant by slightly altering its old memory representation). The findings of (Freeman, 1991) are elucidated in the figure 1.10 next:

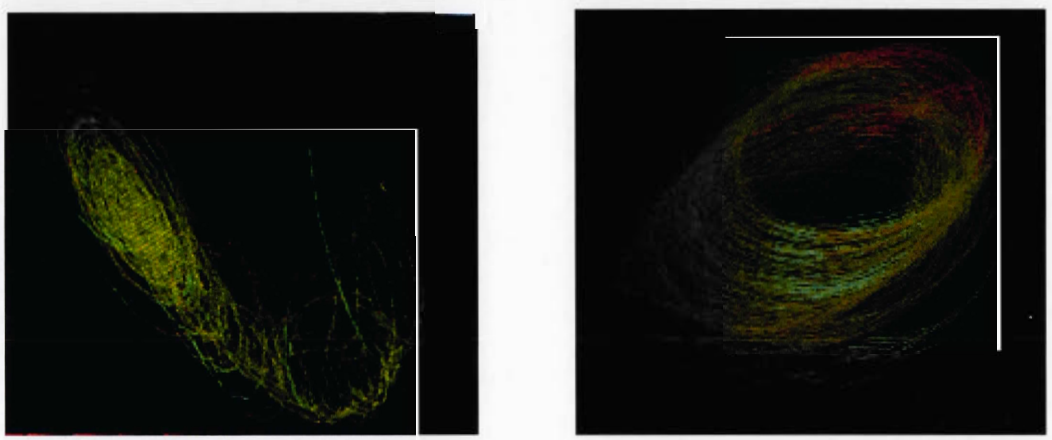


Fig. 1.10 Phase Space Diagram of EEG Bulb activity of a rabbit. Left: Phase space, when the

rabbit is not presented with an odorant, which shows Irregular portrait of neural activity. Right: Phase space, when the rabbit smells food, which seems more ordered and describes a regular portrait of neural activity.

Please check Annex III for a summary and further explanation of Freeman (1991) experiments.

Besides, we showed that an artificial spiking chaotic neural network is able to regularize its chaotic activity when trained with a general class of inputs (Aoun and Boukadoum, 2015). Furthermore, the network was able to discriminate future unknown inputs with the general class of inputs it has learned. Note that the regularization of chaotic activity of the network is manifested by the stabilization of UPOs through out the neurons activity inside the network. Thus, it seems that the regularization of the chaotic attractor, generated by an artificial chaotic spiking neural network, to a more constrained orbit depicting the memory representation of a class of well-defined inputs (Online signatures) as per the authors scope of study (Aoun and Boukadoum, 2015), could be similar to the dense orbits of a chaotic attractor that are regularized when an organism is faced with a familiar class of stimuli as per the scope of Freeman's experimental study (Freeman, 1991). This comparison between two unrestricted phenomenal representations of memory lays out our hypothesis in considering the representation of memory as a stabilized orbit of a chaotic attractor and leads us to the declaration of our thesis statement in the next section (Section 1.7).

Finally, we have to mention important research in the literature, which tackles the concept of chaos in the brain. For instance, Tsuda and Kuroda (2001) extend the work of Freeman (1991). In fact, they provide a mathematic model, based on chaotic dynamics, which they call Cantor coding, that could explain episodic memory (Episodic memory is the ability of the brain to transform short term memory to long

term memory). On the other side, Korn and Faure (2003) provide an exhaustive study with experimental evidence and variety of neuronal models that demonstrate the existence of chaos in the brain.

1.7 Thesis statement, objectives and thesis organization

We claim that 'Memories are embodied in the spiking patterns of chaotically firing neurons; their dynamical mechanisms create a chaotic attractor, such that a single memory is a controlled instance of an unstable periodic orbit (UPO) indwelled in the chaotic attractor'. Note that in our early experiments (Aoun & Boukadoum, 2014, ibid, 2015) we coded the genuine online signatures of users as spike trains and fed them onto a chaotic spiking neural network (Details in Appendix A). The network was able to achieve stabilization of UPOs that could be interpreted as the representation of memories of genuine signatures (Details in Appendix A). Yet, the weights of connections between the neurons inside the network were governed by chaos control and synaptic plasticity. So, our preliminary objective laid herein is to further study the advantage of synaptic plasticity in a network of chaotic spiking neurons over a network of regular spiking neurons (Chapter 2). In addition, our ultimate objective is to build a novel memory-processing system using the least number of chaotic spiking neurons possible. This system should embed a chaotic attractor that can be exploited through the manipulation of chaos control in order to represent a model of memory (Chapter 3). In this regard, we will consider 'a memory as an active cluster of neurons such that the latter is driven by a controlled instance of an Unstable Periodic Orbit (UPO) embedded in the chaotic attractor of a chaotic spiking neuron' that we demonstrate in details in chapter 3. Finally, we want to perform data classification and learning using chaotic spiking neurons. Exploiting the rate of chaotic spiking and remarkably finding out that it has an exponential distribution does this, as we will see in chapter 4. Afterwards, we conclude the thesis.

CHAPTER 2

INVESTIGATING SYNAPTIC PLASTICITY WITH CHAOTIC SPIKING NEURONS AND REGULAR SPIKING NEURONS

2.1 Introduction

To recapitulate, our Hypothesis is based on Freeman's experiments (Freeman, 1991) and Freeman's postulate (Freeman, 1994), which claims that memories can be modeled using chaotic attractors. Upon this Hypothesis, we came up with our thesis which states that a single memory is a controlled instance of an unstable periodic orbit (UPO) embedded in a chaotic attractor.

My early work (Aoun and Boukadoum, 2014, ibid, 2015) provided the initial endeavor towards further exploration of my thesis statement and was the ground of my Thesis Project. After presenting my thesis project, in the course DIC9411, to the jury, the jury committee and I agreed to compare the number of UPOs that can be stabilized in a network of chaotic spiking neurons versus the number of UPOs that can be stabilized in a network of regular spiking neurons. Also, the network of chaotic spiking neurons used in Aoun and Boukadoum (2014, ibid, 2015)

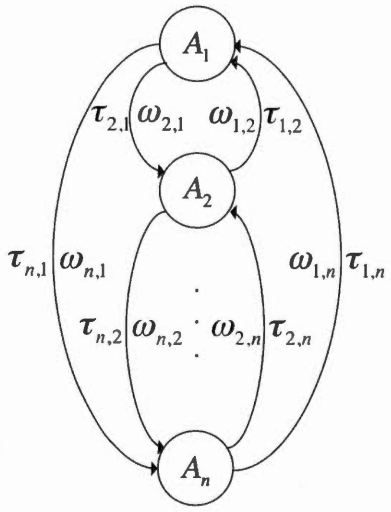
implemented synaptic plasticity on the weights of the connections inside the network. Specifically, the synaptic plasticity method was inspired from a neurophysiological phenomenon called Spike Time Dependent Plasticity – STDP – (Markram et al. 1997, Bi et Poo, 1998) but didn't follow its standard conventional model (Detailed in Section 2.3 and 2.4 of this chapter). Thus, Dr. Boukadoum suggested implementing the standard STDP equations, which are conventionally used in the literature, in a Recurrent Neural Network – RNN – of chaotic spiking neurons. This implementation is a hard task and very challenging, but it will add a substantial asset to our study. Second, Dr. Poirrier, also a member of the jury, suggested creating a layer of neurons that could read the output of chaotic neurons. So, in this chapter, we focus on Dr. Boukadoum's suggestion, which is implementing the standard STDP equations in a RNN of chaotic spiking neurons. Moreover, as we just mentioned, we will compare the number of UPOs that can be stabilized using standard STDP equations in a RNN of chaotic spiking neurons versus the number of UPOs that can be stabilized using standard STDP equations in a RNN of regular spiking neurons. Upon the findings we reach in this chapter (Chapter 2), we will tackle in Chapter 4 machine learning and data pattern classification; which constitute a basic objective of this thesis. However, in chapter 3, we will focus and proceed in fulfilling the ultimate objective of this thesis that was highlighted in the last section (Section 1.7) of Chapter 1, whilst considering Dr. Poirrier's suggestion in this regard.

Therefore, this chapter is divided as follows: In section 2.2, we sketch the Neural Network Architecture and the Neuron model that will be used in the remaining sections of this chapter. In section 2.3, we introduce Synaptic Plasticity and Spike Time Dependent Plasticity (STDP). In section 2.4, we illustrate the STDP window based on neurophysiological data, we also introduce and explain the conventional STDP equations and plot their STDP window. In section 2.5, we show our implementation of STDP in a RNN of chaotic spiking neurons while taking into consideration the conventional equations of STSP. In section 2.6, we prove the

feasibility of STDP in a RNN of chaotic spiking neurons. In section 2.7, we provide a comparable study of the number of unique UPOs that can be stabilized in RNNs of chaotic spiking neurons implementing STDP versus the number of UPOs that can be stabilized in RNNs of regular spiking neurons implementing STDP.

2.2 Neural network architecture

The neural network architecture that will be used in this chapter is composed of n (a variant number) spiking neurons that are recurrently connected to one another, hence a Recurrent Neural Network (RNN). We use the Adaptive Exponential (AdEx) Neuron model (Brette and Gerstner, 2005) as the elementary component of the RNN, because it is a biological neuron model that can simulate a wide range of physiological neuron behaviors (regular spiking, bursting, chaotic spiking…) and has fast processing speed when executed on a computer. Each connection; between any two neurons, in the network has a fixed time delay and an adjustable weight. The time delays are kept fixed because the network will synchronize its spiking activity in a period equal to the time delay. The weights are adjusted using synaptic plasticity as we will see in the next section. The network architecture is illustrated in Figure 2.1.



Recurrent Network of
n AdEx Neurons

Fig. 2.1 Neural Network Architecture: *n* AdEx Neurons are recurrently connected. Every connection has a weight "ω" and a time delay "τ"

As we can see in Figure 2.1, we have a neural network of *n* AdEx Neurons (e.g. A_1 to A_n) where the subscript *"i"* is the index of an AdEx Neuron *"A"*. Each neuron has *n-1* connections. A connection from neuron A_j to neuron A_i has a time delay *"$\tau_{i,j}$"* and a weight $\omega_{i,j}$.

We rewrite the AdEx Neuron equations (Naud et al., 2005) in their Euler form:

$$V_i(t) = V_i(t-1) + \frac{dt\left[-gL(V_i(t-1) - E_L) + gL\,\Delta_T \exp\left(\frac{V_i(t-1) - V_T}{\Delta_T}\right) + I_c - \psi\right]}{C}$$

$$\psi_i(t) = \psi_i(t-1) + dt[a(V(t-1) - E_L) - \psi]/\mathrm{K}_w$$

Where $V_i(t)$ is the neuron's voltage at time *t* and $\psi_i(t)$ is its adaptation variable. The subscript *"i"* indicates the neuron's index inside the network.

dt is the Euler time step which is set to *0.1 ms* for good precision (Naud et al, 2008).

The parameters of the AdEx Neuron equations are retrieved from (Naud et al, 2008) and are shown in table 2.1, next. For parameters units please refer to Naud et al, 2008.

TABLE 2.1 Configuration of the AdEx Neuron parameters for Regular and Chaotic firing modes (Naud et al, 2008). For parameters units please refer to Naud et al, in 2008.

Mode \ Parameters	C	gL	E_L	V_t	Δ_T	a	τ_w	b	V_r	Ic	θ
Regular	200	10	-70	-50	2	2	30	0	-58	500	0
Chaotic	100	12	-60	-50	2	-11	130	30	-48	160	0

The neuron initial conditions are:

$$V_i(0) = V_r$$

$$\psi_i(0) = 0$$

$$Ic_i = LIc + (UIc - LIc) * R_i$$

Where,

R_i is a random decimal between 0 and 1.
LIc is the lower bound of the default input current (Ic).
UIc is the upper bound of the default input current (Ic).

We note that LIc and UIc are set according to the firing mode (i.e. regular or chaotic) requested. For instance, if we want regular firing, we can set LIc to 400 and UIc to 600, which are in fact 100 units away from their mean Ic which is 500. As for chaotic firing, we can set LIc to 150 and UIc to 170, which are 10 units away from their mean Ic that is 160. Note that in the case of chaotic firing, it was observed in (Naud et al., 2008) that the neuron's chaotic firing is sensitive to the input current 160 (e.g. Ic = 160 pA) and values far away from 160 would ruin the chaotic behavior of the neuron. In fact, we need the neurons of the RNN to have slight variations in their initial conditions (i.e. initial value of the Input current Ic) so their output will be different

from one another; which will offer great opportunity to the synaptic plasticity model that depends on the firing time of the neurons, to operate properly as we will see in the next section.

When the neuron's voltage passes its threshold (i.e. $V_i >= \theta$), then the neuron fires a spike that is represented as γ_i:

$$\gamma_i(t) = \begin{cases} 1, V_i(t) \geq \theta \\ 0, V_i(t) < \theta \end{cases}$$

The neuron receives time-delayed spikes from other neurons through its incoming connections, which are scaled by each connection's weight and summed up as a total input I given by:

$$I_i(t) = \sum_{j=1, i \neq j}^{n} w_{i,j} \gamma_j (t - \tau_{i,j})$$

After calculating the input to the neuron, then the latter is added to the neuron's voltage:

$$V_i(t) = V_i(t) + I_i(t)$$

When the neuron's voltage crosses its threshold, it is reset to its reset value and its adaptation variable is set to the current value of the latter plus the adaptation reset parameter:

$$If\ V_i(t) \geq \theta\ then\ V_i(t) = V_r\ and\ \psi_i(t) = \psi_i(t) + b$$

The weights of all the connections are initially set to 0, so the neurons inside the network will run in isolation and evolve their dynamics. Afterwards, synaptic

plasticity starts to operate by updating the connection weights, thus modifying $I(t)$ of every neuron. Furthermore, as we will see in section 2.5, $\tau_{i,j}$ will vary depending on the time of occurrence of the incoming spike to the neuron and its status (i.e. Firing or quiescent). In other words, $\tau_{i,j}$ will be different if the incoming spike occurred after the neuron's firing than if the incoming spike occurred before the neuron's firing, as we will see in detail in the next section.

2.3 Synaptic plasticity based on spike time dependent plasticity (STDP)

Hebbian Learning is based on a law that was conceived by Donald Hebb, in 1949. In laymen terms, Hebb's law states that 'neurons that fire together wire together'. This means that a chemical bond is built on the synapse between neurons that are causing excitation to each other (Hebb, 1949), thus the saying Synaptic Plasticity. In retrospect, this allows the neurons on each side of the synaptic bond to be sensitive to each other, in other words their firing activity becomes mutually correlated. According to Hebb, "When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" (Hebb, 1949). This is translated mathematically as the weight change (i.e. the change in synaptic strength) between the neurons is the product of their activation. More recently, a new theory that adds to Hebbian Learning, has emerged: Spike-Timing-Dependent Plasticiy – STDP – (Markram et al. 1997, Bi et Poo, 1998). STDP is a protocol of Hebbian Learning with time constraints. Specifically, STDP Learning considers the exact timing when a presynaptic neuron fires and the exact timing a postsynaptic neuron fires, in order to build-up their synaptic strength (i.e. Synaptic Plasticity). Based on experimental evidence (Markram et al. 1997, Bi et Poo, 1998), it was observed that if a postsynaptic neuron receives spikes from a presynaptic neuron, just before the postsynaptic neuron fires, then the synapse's

strength between the two neurons is increased (This is a form of pre - post firing). In contrast, if a postsynaptic neuron receives spikes from a presynaptic neuron, after the postsynaptic neuron had fired (a form of post - pre firing), then the synapse's strength between the two neurons is decreased (Markram et al. 1997, Bi et Poo, 1998).

Donald Hebb explains the acquisition of learning as being equivalent to an organization of neuronal behavior which is manifested when neurons organize their firing activity by building chemical bonds with each other, thus forming groups or neural cell assemblies. These bonds, according to Hebb's law, are instigated and maintained based on the rate of firing of the neurons, which molds the cells assembly, and builds a mutual correlation of firing activity between the neurons. However, it was observed that the exact firing time of a single spike of a presynaptic neuron could hold much more information than the firing rate (Bi et Poo, 1998). In 1998, Bi and Poo confirmed that if a neuron's presynaptic pulse occurs just before the firing of the neuron then the synapse is strengthened, however if it occurred afterwards then the synaptic strength is depressed. In other words, the exact firing time of a presynaptic cell affects the synaptic bond between the neurons, which in return influences either a mechanism of Long Term Potentiation (LTP), or Long Term Depression (LTD), through the neuronal behavior. We have to note that such observations were observed in vitro only. These experiments where achieved on cellular tissues of neurons in order to elucidate the consequence of spike timing to the synaptic changes. From Bi and Poo, in 1998, we quote "Repetitive stimulation (60 pulses at 1 Hz) was applied to the presynaptic neuron while both cells were held in current-clamp to allow spiking... Measurements of the amplitude of EPSCs revealed a persistent increase in synaptic efficacy after the repetitive stimulation..." (Bi et Poo, 1998). However, "postsynaptic spiking was initiated by repetitive injection of depolarizing current pulses before the activation of *subthreshold* synaptic inputs... Repetitive initiation of postsynaptic action potentials that peaked at 6 msec before the onset of EPSPs

resulted in a persistent reduction in the EPSC amplitude" (Bi et Poo, 1998). This gives a glimpse that the exact firing time of a presynaptic neuron has a considerable effect to a postsynaptic neuron and plays a major role in inducing synaptic mechanisms that were simplified by only considering the firing rate. We have to note "the cellular basis that gives rise to the critical window for the induction of synaptic modifications remains to be determined" (Bi et Poo, 1998). "Such sensitivity to activity patterns is crucial for the formation of specific engrams in neuronal circuits" (Wang et al., 2005). Thus, we speculate that if a RNN encapsulating nonlinear dynamics in its core, whilst allowing STDP to manage the weights of the connections inside the RNN, could emulate self-organization of the RNN depicted in synchronous neural activity, then this phenomenon of spikes synchronization, that can be analyzed as an engram manifested by the neural circuit (e.g. RNN), would justify the grounding principles of STDP in achieving self-organization of neural circuitry and would provide additional support to the claim of crucial sensitivity of activity patterns in the formation of specific engrams in neuronal circuits (Wang et al., 2005).

2.4 STDP window

As explained in the previous section, the STDP protocol is fundamentally based on the firing times between pre-synaptic and post-synaptic neurons. Specifically, if a pre-synaptic neuron fires before a post-synaptic neuron, then the synaptic connection between the two neurons is strengthened (i.e. the weight's change should be positive, thus, the weight of their connection is increased), and if a pre-synaptic neuron fires after a post-synaptic neuron, then the synapse between the two neurons is lessened (i.e. the weight's change should be negative, thus, the weight of their connection is decreased). Mathematically, if the weight change is denoted as Δw, the pre-synaptic time is denoted as t_{pre} , the post-synaptic time is denoted as t_{post} , the difference between t_{pre} and t_{post} denoted as Δt, then the STDP protocol is translated as:

$$\Delta w = \begin{cases} A_1 * e^{\left(-\frac{\Delta t}{\tau_1}\right)} & \Delta t > 0 \\ -A_2 * e^{\left(\frac{\Delta t}{\tau_2}\right)} & \Delta t < 0 \end{cases}$$

Such that: A_1, A_2, τ_1 and τ_2 are constants.

For instance, if A_1=1, A_2 =1, τ_1 = 20 and τ_2 = 20 then the STDP window look as in Figure 2.2, next:

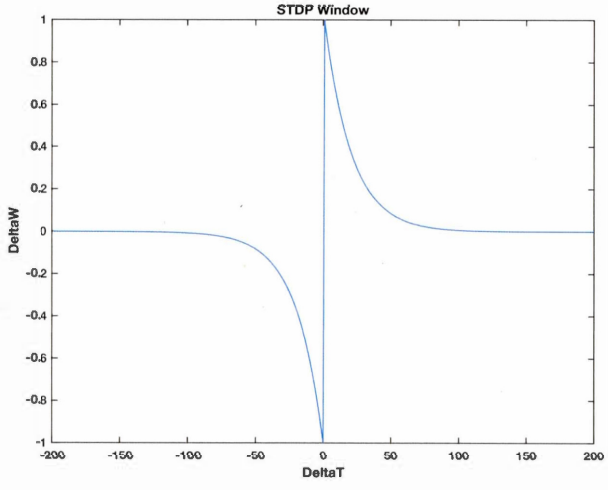


Fig. 2.2 Curve plot of the STDP protocol

The STDP window from empirical data (Yang and Poo, 2006) is shown in Figure 2.3, next. Note that the weight change is denoted as Excitatory Post Synaptic Potential – EPSP.

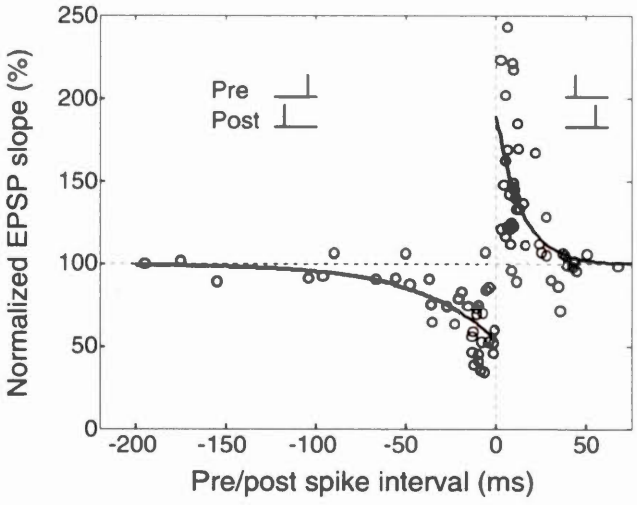


Fig. 2.3 STDP time window from empirical data. Excerpt from (Yang and Poo, 2006).

2.5 Implementing STDP in RNN of spiking neurons with time delay

As we highlighted in section 2.2 of this chapter, a connection between two neurons *i* and *j,* inside a RNN explained in section 2.2 and illustrated in Figure 2.1, has a time delay denoted by $\tau_{i,j}$. Furthermore, a neuron *i* calculates its Voltage at time *t* (i.e. $V_i(t)$) and then integrates the incoming spikes available through its input connections in a summation denoted by $I_i(t)$ which will be added to $V_i(t)$ and the result is evaluated by the voltage threshold (i.e. if $V_i(t) >= \theta$ then the neuron emits a spike). But, before doing so (i.e. before evaluating $I_i(t)$ and adding it to $V_i(t)$), the weights of the connections should be updated (i.e. $\Delta w_{i,j}$ should be calculated and added to $w_{i,j}$). So, in order to update the weights of the connections of the RNN then we implement a competitive scenario of STDP that is similar to the approach made by Song et al, in 2000, in their work on Competitive Hebbian learning based on STDP (Song et al, 2000). Furthermore, this implementation ensures synchronization of neural states, as we will demonstrate through numerical simulations in the remaining sections of this chapter. The competitive STDP protocol is described as:

$$If \quad (t-\tau_{i,j}) - t_{pre(j)} < t_{post(j)} - (t-\tau_{i,j}) \;\&\; V_i(t) > \theta \quad then \quad \Delta w_{i,j} = \Delta w^+$$
$$If \quad (t-\tau_{i,j}) - t_{pre(j)} > t_{post(j)} - (t-\tau_{i,j}) \quad then \quad \Delta w_{i,j} = \Delta w^-$$

Where,

$t_{pre(j)}$ is the spike time of the spike, of input neuron j, that occurred just before $t - \tau_{i,j}$

$t_{post(j)}$ is the spike time of the spike, of input j, that occurred right after $t - \tau_{i,j}$

Δw^+ is the positive increase of the weight change, equal to:

$$\Delta w^+ = A^+ * e^{\left(-\frac{t-\tau_{i,j}-t_{pre(j)}}{\tau_{i,j}}\right)}$$

Δw^- is the negative decrease of the weight change and equal to:

$$\Delta w^- = A^- * e^{\left(\frac{t-\tau_{i,j}-t_{post(j)}}{\tau_{i,j}}\right)}$$

A^+, A^- are the upper and lower boundaries, of the positive and negative weight change, respectively, which are defined as the following:

$$A^+ = \left((\theta - V_r) - w_{i,j}\right) * \mu^+$$

And,

$$A^- = V_r * \mu^-$$

Where,

μ^+, μ^- constants that are experimentally set to 0.1

The reason behind this choice in defining A^+ and A^- is supported by the fact that these parameters should be voltage dependent as suggested by Clopath and Gerstner, in

2010. Furthermore, by setting $A^- = V_r * \mu^-$ then we are sure that this parameter is always negative, which is a must for LTD to take place in the STDP protocol. Second, by setting $A^+ = \left((\theta - V_r) - w_{i,j}\right) * \mu^+$ then we are sure that LTP will settle down once the weight of a connection has reached a maximal value equal to $\theta - V_r$.

Last but not least, the weight of every connection is updated according to the following:

$$w_{i,j}(t) = w_{i,j}(t-1) + \Delta w_{i,j}$$

Afterwards, the Input to neuron i is calculated according to the following:

$$I_i(t) = \sum_{j=1, i \neq j}^{n} w_{i,j}(t) * \gamma_j(t_{(pre,post)(j)})$$

Where,

$$t_{(pre,post)(j)} = \begin{cases} t_{pre(j)} \; in \; case \; of \; \Delta w^+ \\ t_{post(j)} \; in \; case \; of \; \Delta w^- \end{cases}$$

And, $I_i(t)$ is added to the neuron's voltage $V_i(t)$:

$$V_i(t) = V_i(t) + I_i(t)$$

$V_i(t)$ is then evaluated by the threshold θ in order to check if the neuron i will emit a spike or not.

2.6 Behavior of the RNN using STDP

In this section, we experiment the STDP protocol discussed in the previous section in

a RNN of chaotic spiking neurons. The number of neurons inside the network is set to 5. The simulation time is 20000 time steps. In the first 5000 steps, STDP is not activated so the neurons have enough time to evolve their dynamics. At time step 5001, STDP is turned on and the connections start updating their weights, which in return will affect the dynamics of the neurons. The time delay of every connection is set to 200 and the weights start with initial value equal to 0. The choice of a time delay equal to 200 is for experimental purposes only and is supported by neurophysiological data (Yang and Poo, 2006), of course other choices are possible. As we will see in the graph of Figure 2.4, when the neurons are in isolation (e.g. Time Step $\leq$ 5000), a neuron's activity inside the network is different from the activity of other neurons. However, when STDP starts operating at time step 5001 then the neurons start stabilizing their dynamics, which results in a synchronous neural activity all over the neural network.

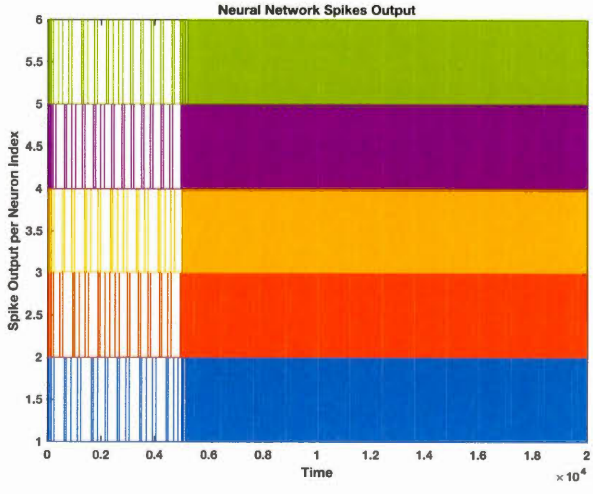


Fig. 2.4 Neural Network Spikes Output: Neurons run in a simulation of 20000 time steps. At time step 5001 STDP starts operating on the connections weights. When STDP is on, the neurons end up synchronizing their spikes activity.

To show that every neuron inside the neural network is synchronizing its spiking activity over a period equal to "τ"; which is the time delay of any connection between two neurons in the network and is the same for all connections, we calculate

the difference between the absolute value of a neuron's spike at time *"t"* and its delayed spike at *"t - τ"*, where *"τ"* indicates the time delay of any connection in the neural network; which is in fact the same for all the connections as we just mentioned. If the difference between a neuron's spike absolute value at time *"t"* and its spike absolute value at time *"t - τ"* is equal to 0 all over *"τ"*, then this means that the neuron is synchronizing its spikes activity over the period *"τ"*. The plot of these differences is given in Figure 2.5, next.

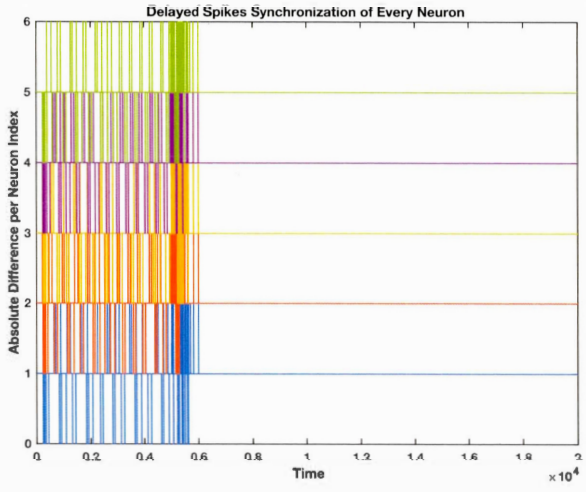


Fig. 2.5 Periodic Spikes Synchronization of every neuron inside the Neural Network: Calculating the difference between the absolute value of a neuron's spike output at time *"t"* and the absolute value of its spike output at time *"t - τ"* shows that every neuron is synchronizing its activity in a time period equal to *"τ"* because the difference settles to 0 in the long run.

In figure 2.5, we showed that every neuron is synchronizing its output spikes activity to itself over a period that is equal to the time delay of the connections inside the network. But, to show that the neurons are synchronizing their system dynamics; their voltage *"V"*, Adaptation variable *"ψ"* and Spike output *"γ"*, between each other, then we will resort to the standard deviation of the data of each of these variables at time *"t"*. In fact, the standard deviation, at time *"t"*, of a set of data values that a variable *"x"* takes at time *"t"*, illustrates the dispersion of these data values, at time *"t"*, to the mean value of *"x"* at time *"t"*. Thus, if the data is highly dispersed around

the mean then the standard deviation is maximal and if the data is closely dispersed around the mean then the standard deviation is minimal. In our case, if all the neurons are truly synchronizing their activity between each other at time *"t"* then the standard deviation of their dynamic variables should be minimal and the better case equal to *0*. The standard deviation of data variables *"γ"*, *"V"* and *"ψ"* of all the neurons is shown in figures 2.6, 2.7 and 2.8, respectively, next.

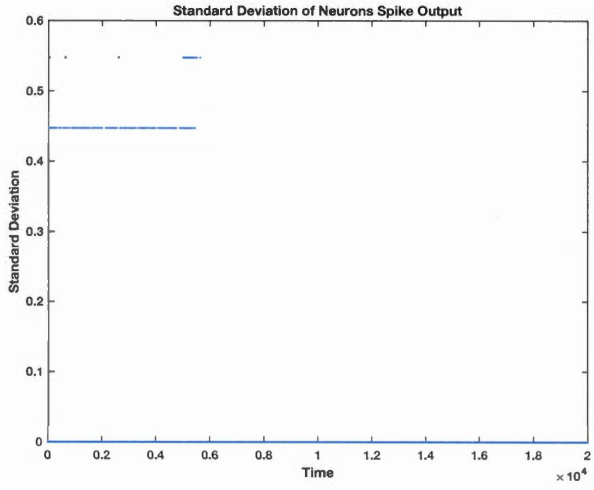


Fig. 2.6 Proof of Synchronization of all Neurons Output Spikes using Standard Deviation

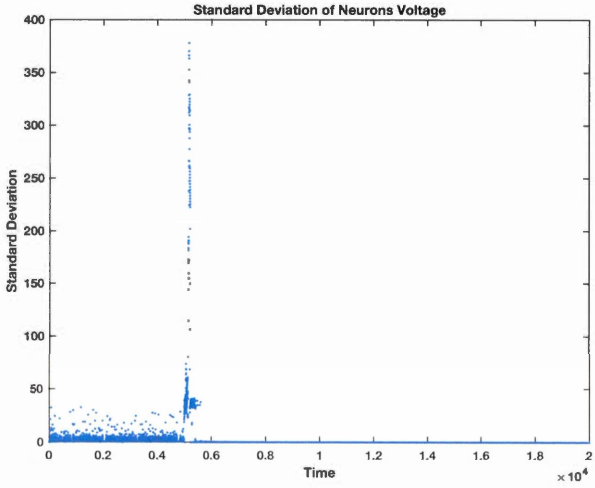


Fig. 2.7 Proof of Synchronization of all Neurons Voltages using Standard Deviation

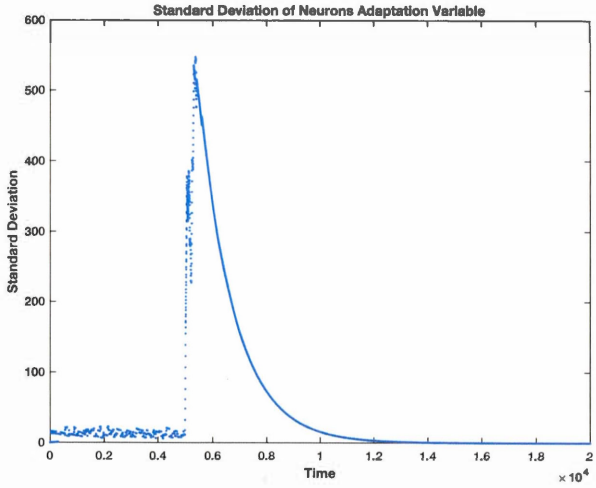


Fig. 2.8 Proof of Synchronization of all Neurons Adaptation Variables using Standard Deviation

As we can see in Figures 2.6, 2.7 and 2.8 the standard deviation between the dynamic variables of every neuron in the network converges to 0, which proves that all the neurons are synchronizing their dynamics. Furthermore, we show the average weight of the connections inside the network, in figure 2.9 next.

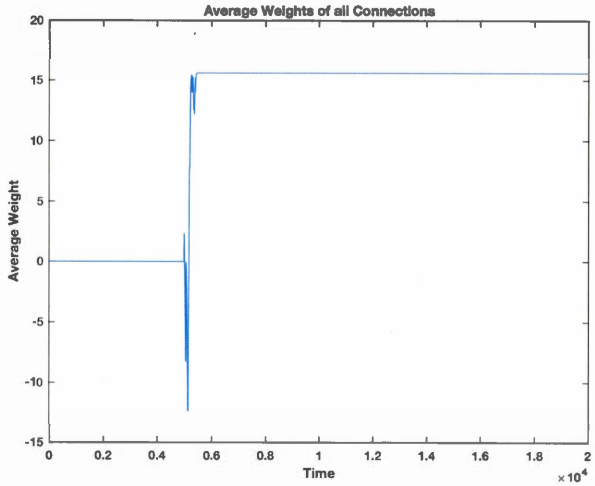


Fig. 2.9 Average Weight of the connections inside the RNN

As we can see in Figure 2.9, when STDP starts operating at time step 5001, the weights alternate between inhibitions and excitations, thus apply STDP rules by

undergoing depression and potentiation respectively, before they fully stabilize around time step 6000 and remain unchanged afterwards.

Finally, we show the STDP window of the weights changes in figure 2.10, next.

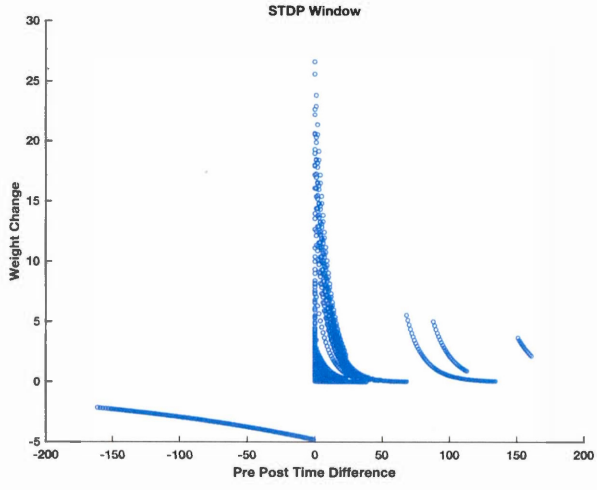


Fig. 2.10 STDP window of the RNN that is composed from chaotic spiking neurons

As we can see in figure 2.10, the STDP window shows more activity on the LTP side than on the LTD side and this shows that causal interactions inside the neural network are indeed taking place, which eventually lead to synchronous activity of the neurons. We note that if LTD dominated then the neurons wouldn't synchronize and their behavior would be different from one another (i.e. asynchronous), which is not the case.

2.7 Unstable periodic orbits (UPOs) in RNN using STDP

In this section, we study the number of different Unstable Periodic Orbits (UPOs) that can be stabilized in a RNN composed of P chaotic spiking neurons, such that P is an arbitrary positive integer that dictates the size of the network, when STDP is

managing the connections weights between the neurons. Furthermore, we compare this number with the number of stabilized UPOs, using STDP, of a P size RNN composed of P regular spiking neurons. For instance, the number of neurons that constitute both the RNN of chaotic spiking neurons and the RNN of regular spiking neurons will be between 10 and 50 (e.g. $P \in \{10, 20, 30, 40, 50\}$). We aim to find out that the memory capacity of the chaotic RNN is larger than the memory capacity of a regular RNN (i.e. The number of stabilized UPOs inside RNN of chaotic spiking neurons is superior than the number of stabilized UPOs inside RNN of regular spiking neurons). To do this, we run twice, five sets of one hundred random experiments (i.e. random in the sense that an experiment starts with random initial conditions) where each experiment is composed of a RNN of P Neurons while increasing P by 10 for each set, starting from $P = 10$ to $P = 50$. The first five sets of experiments will be executed on RNN composed of chaotic spiking neurons, while the second five sets of experiments will be executed on RNN composed of regular spiking neurons. The results of the two 'five sets' of experiments are illustrated in a bar graph in figure 2.11, next.

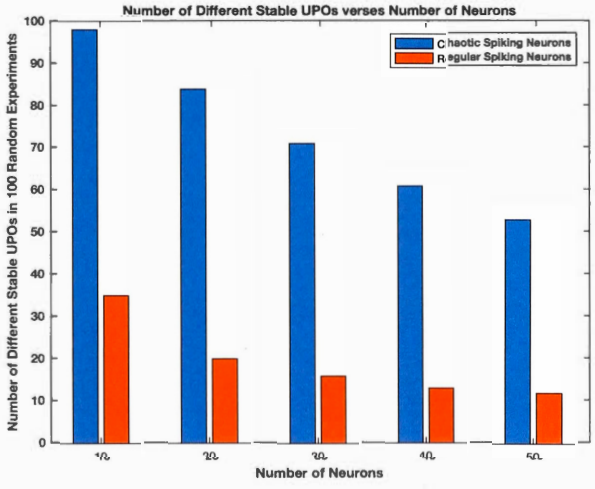


Fig. 2.11 Comparison of the number of different stabilized UPOs of chaotic spiking RNN and regular spiking RNN.

By analyzing the graph of figure 2.11, we observe that the number of different

stabilized UPOs of RNN composed of chaotic spiking neurons is always superior to the number of different stabilized UPOs of RNN composed of regular spiking neurons for any number of neurons.

2.8 Conclusion

In the introductory section of this chapter, we raised two important investigations: The first investigation challenged **the feasibility of implementing the standard equations that describe Spike-Timing Synaptic Plasticity (STDP) in a network of chaotic spiking neurons** and the second questioned the advantage of using chaotic spiking neurons that implement synaptic plasticity compared to the advantage of using regular spiking neurons implementing STDP. We fulfilled the first challenge by realizing a competitive STDP protocol within a recurrent network of chaotic spiking neurons protocol. As for the second challenge, experimental simulations were conducted in the aim of comparing the number of different stabilized UPOs through a network of chaotic spiking neurons vs. a network of regular spiking neurons whilst the connections between the neurons inside the networks followed a competitive STDP protocol. The results of the experiments confirmed that STDP favors chaotic spiking over regular spiking of neurons. In fact, the number of different UPOs stabilized within RNN of chaotic spiking neurons implementing STDP was shown to be way larger than the number of different UPOs stabilized within RNN of regular spiking neurons implementing STDP.

2.9 Concluding Remarks

It should be noted that the effect of chaos control over a chaotic spiking neuron, or a recurrent network of chaotic spiking neurons, is not to be considered as an external

regulation. This is because chaos control is solely achieved either through a feedback connection from a neuron to itself (i.e. a self-feedback connection) as we have seen in the previous chapter or a feedback connection from a neuron to another neuron through synchronization as we have seen in this chapter.

Another important remark: Chaos control could work as guidance for learning since it helps to achieve fast stabilization of the weights of the connections between neurons as shown in (Aoun, 2010, Aoun et Boukadoum, 2014, Aoun et Boukadoum, 2015).

Furthermore, we notice in figure 2.11 that the number of different stabilized UPOs is inversely proportional to the size of the RNN for both types of neurons that constitute the RNN (e.g. Chaotic and Regular). Our analysis in interpreting such result is the following: The increased number of neurons will increase the probability of their concurrent and nearby activity, which results in maximal causal interaction between the neurons that leads to an increased saturation of different possible outcomes which is translated in a decreased number of different possible stable states (i.e. decreased number of different stabilized UPOs). A work around to overcome this downward limitation is explained next.

Through simulations (Chapter 4), we will demonstrate that by increasing the time delay of the connections inside large networks of recurrently connected chaotic spiking neurons, then we are able to maintain a large a number of different stabilized UPOs despite the increase of the network size. This means that a huge memory capacity, in terms of different stabilized UPOs, can be sustained for big neural networks.

Last but not least, we have to mention that in these experiments the time delay was always fixed and set to a value equal to 300. Fixing the time delay to a constant value all over the experiments is just in the sake of easing the proof of concept in regards to the sole purpose of this chapter towards the investigation of the effect of STDP in

recurrent networks of chaotic spiking neurons vs. regular spiking neurons. We assure the reader that any value of the time delay could have generated the same consequences in regards to the number of stabilized UPOs of regular spiking neurons vs. chaotic spiking neurons. In other words, the choice of any time delay has no importance in interpreting the results that we reached in this chapter. However, an exhaustive study that solely tackles the variation of the time delay and its potential is elaborated in chapter 4.

CHAPTER 3

THEORY OF NEURONAL GROUPS BASED ON CHAOTIC SENSITIVITY

3.1 Introduction

In our thesis statement, we claimed that: 'Memories are embodied in the irregular spiking patterns of chaotically firing neurons which through dynamical mechanisms generate a chaotic attractor, such that a single memory is a controlled instance of an unstable periodic orbit (UPO) in the chaotic attractor'. We aim to build a novel memory-processing system using chaotic spiking neurons, which should embed a chaotic attractor that can be exploited through the manipulation of chaos control in order to achieve memorization. In such endeavor, we challenge our thesis to consider the least number of chaotic spiking neurons as one single chaotic spiking neuron. The reason of a single neuron will be explained shortly. Consequently, we will slightly alter our thesis statement and rephrase its claim by declaring 'a memory as an active cluster of resonant neurons driven by a controlled instance of an Unstable Periodic Orbit (UPO) embedded in the chaotic attractor of a single chaotic spiking neuron'. In this chapter we create a Neural Network Architecture that elucidates our thesis statement through experimental and numerical simulations. Our approach in building the neural network architecture is based on Reservoir Computing (RC) (Natschlaeger

et al., 2002) and our choice of one neuron is to simplify two models of RC, the first is Nonlinear Transient Computing Machine (NTCM) (Crook, 2007) and the second is called Liquid State Machine (LSM) (Maass et al., 2002). We simplify NTCM, which embeds two chaotic neurons. In fact, we demonstrate that the job of a NTCM can be done with a single chaotic neuron. On the other side, the reservoir of neurons inside a LSM is composed of a large number (in the range of hundreds or even thousands) of regular spiking neurons while the reservoir in our case is composed of a single chaotic neuron. We demonstrate that the effect of nonlinear dynamics generated by a single chaotic spiking neuron is a substitute to the effect of nonlinear dynamics that could be generated with a recurrent neural network (i.e. a reservoir) composed of a large number of regular spiking neurons. Specifically, we demonstrate the separation property of a reservoir of regular spiking neurons holds true when using a single chaotic spiking neuron. Showing that the separation property holds for a chaotic spiking neuron is the first challenge and stepping stone that we reach in this chapter. Second, to show that the controlled instances of UPOs can depict memory then we propagate these UPOs to a layer of Resonant Neurons. We explain further:

The Neural Network Architecture, presented in this chapter, is composed of a chaotic spiking neuron and a layer of Resonant Neurons. The chaotic spiking neuron projects its spiking output to the layer of Resonant Neurons. The Resonant Neurons catch Inter Spike Intervals (ISIs) coming from the output spikes of the chaotic spiking neuron, such that each Resonant Neuron is configured to resonate (i.e. emit a spike) for a specific ISI.

The spiking output of the chaotic spiking neuron is a reflection of its Unstable Periodic Orbits (UPOs). The latter are unstable, thus the spiking output of the chaotic spiking neuron is irregular. Chaos control is applied over the chaotic spiking neuron, leading its dynamics to stabilize towards a single UPO. To achieve chaos control, a self-feedback connection with time delay is implemented on the chaotic spiking neuron, which in return renders its output activity into a repetitive spikes pattern having a time window equal to the time delay of the self-feedback connection. The

repetitive spike pattern has distinct ISIs inside of it, because it originated from a delayed instance of irregular spikes; that were held in the self-feedback connection. So, the repetitive spikes pattern is composed of a 'set' of ISIs. A layer of Resonant Neurons is always receiving output spikes coming from the chaotic spiking neuron, but after the initiation of chaos control, these output spikes are now trapped in a repetitive spikes pattern composed of a set of ISIs. Since every Resonant Neuron resonates for a specific ISI, and since we have a repetitive set of ISIs, then we have a group of Resonant Neurons that are resonating for a set of ISIs during a time window equal to the time delay connection of the chaotic spiking neuron. This group of neurons models a memory and is considered as a memory representation of an external input. Furthermore, we have a large number of UPOs that can be stabilized using chaos control, so we have a large number of unique spike patterns that can be generated by the chaotic spiking neuron when it is under the influence of chaos control. Besides, these unique spike patterns are mirrored as groups of Resonant Neurons. Our goal is to feed the chaotic spiking neuron with an external input, control its dynamics towards a regular behavior - depicted by a repetitive spikes output pattern of distinct ISIs - and associate the group of Resonant Neurons that mirror its regular behavior to the external input it received. In this way, our neural network architecture can be considered as an infinite state machine because it can associate (or create) an output to every input (or new input) it receives. Does it have memory? And, what is its memory capacity? The answer, to the first question, is yes if we consider the group of Resonant Neurons as a cell assembly that is active whenever a specific input is provided to the network. Furthermore, we can say that is has memory of all past and future inputs: It has memory of all past inputs if whenever a past input is fed to the network then the same cell assembly that was previously activated for that input, is activated. It has memory of all future inputs if whenever a new input is fed to the network then a new cell assembly is created for it. To answer the second question, its memory capacity is theoretically infinite due to the chaotic nature of the system dynamics, which allows the control of an infinite number of

UPOs that are reflected onto neuronal groups considered as cell assemblies. In this chapter, we provide experimental results to prove these assumptions. First, we introduce a model of a simple spiking neuron (Section 3.2). In section 3.3, we introduce the chaotic spiking neuron. In section 3.4, we apply chaos control on the chaotic spiking neuron. In Section 3.5, we introduce the Resonant Neuron upon which a layer of Resonant Neurons is built and configured. In section 3.6, we sketch the neural network architecture that will use the neuron models introduced in section 3.2, 3.3 and 3.5 and the chaos control method explained in section 3.4. In section 3.7, we give an example that facilitates the understanding of Neuronal Groups by visualizing them in a 3 dimensional space. In section 3.8, we study the separation property of the neural network. In section 3.9, we study the number of neuronal groups that can be reached using this neural network. In Section 3.10, we discuss the resemblance of our work to important theories in the field of computational neuroscience and neuro-computing. Section 3.10 concludes the chapter.

3.2 The leaky integrate and fire neuron

The leaky Integrate and Fire (LIF) Neuron is a basic spiking neuron model which can simulate the basic activity of a biological neuron. The LIF is well known in the Computational Neuroscience domain and is commonly used to illustrate basic neuron behavior (i.e. Regular spiking activity). It is depicted in the following difference equation:

$$u(t+1) = u(t) + [-\left(\frac{u(t)}{R * C}\right) + \left(\frac{I(t)}{C}\right)]$$

Where,

u(t) is considered as the neuron voltage at time t.

$u(0) = n_0$ and n_0 is a random number between – 1 and 0, which is considered as the reset value of the neuron voltage when the latter hits a threshold defined as θ.

R is the resistance and C is the capacitance.

I(t) is the input current to the neuron.

We define γ(t) as the spike output of the neuron at time t, as it follows:

$$\gamma(t) = \begin{cases} 1, u(t) > \theta \\ 0, u(t) \leq \theta \end{cases}$$

The behavior of the LIF Neuron for $n_0 = -0.7$, $R = 6$, $C = 10$, $I = 0.05$ and $\theta = 0$ is illustrated in Figure 3.1 Next.

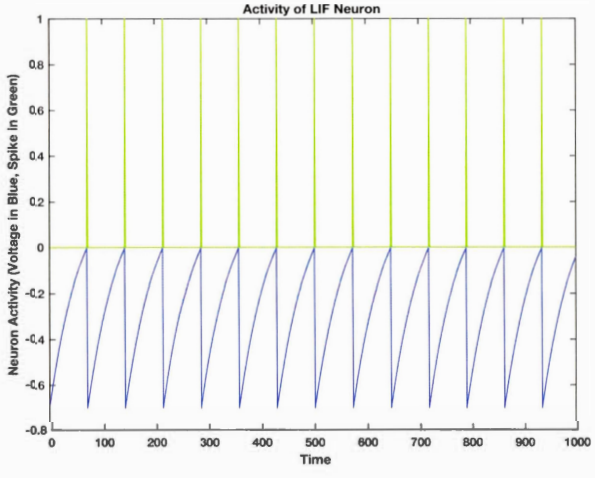


Fig. 3.1 Plot of the activity of LIF Neuron. Simulation Time = 1000 Time steps. Voltage in blue. Spike Output in green.

3.3 The adaptive exponential integrate and fire (AdEx) neuron

The Adaptive Exponential Integrate and Fire (AdEx) Neuron was invented by Brette and Gerstner, in 2005. The AdEx Neuron is a very powerful neuron model because it has a very high computational processing speed, is biologically plausible and has different configurations that make it simulate a wide range of neuron behavior

(Regular spiking, irregular spiking, tonic spiking, bursting…). Here we are interested in two of these behaviors: Regular Spiking and Irregular Spiking. The equations of the AdEx Neuron model are the following:

$$C\frac{dV}{dt} = -gL(V - E_L) + gL\,\Delta_T \exp\left(\frac{V - V_T}{\Delta_T}\right) + I_c - \psi + I$$

$$\tau_w \frac{d\psi}{dt} = a(V - E_L) - \psi$$

If $V > \theta$ *then* $V = V_r$ *and* $\psi = \psi + b$

Where V is considered as the action potential of the neuron, ψ is the adaptation variable, Ic is a constant current, I is the incoming current, C is the membrane conductance, gL is the leak conductance, E_L is the leak reversal, V_T is the threshold potential (not to be confused with θ which is the firing threshold), V_r is the reset potential, Δ_T is the rise slope factor, τ_w is the adaptation time constant, a is the sub-threshold adaptation conductance and b is a current increment. Different values of these parameters lead to different firing activity of the neuron. Table 3.1 next describes two configurations of these parameters, which are regular mode and irregular mode.

TABLE 3.1 Parameters configurations of the AdEx Neuron in Two modes

Mode \ Parameters	C	gL	E_L	V_t	Δ_T	a	τ_w	b	V_r	Ic	θ
Regular Spiking:	200	10	-70	-50	2	2	30	0	-58	500	0
Irregular Spiking:	100	12	-60	-50	2	-11	130	30	-48	160	0

When $V > \theta$, the neuron emits a spike which is represented as γ (i.e. γ=*1 if* $V > \theta$ *and 0 otherwise*).

Next, in figures 3.2 and 3.3, we will plot the activity of the AdEx Neuron using configurations of Table 3.1

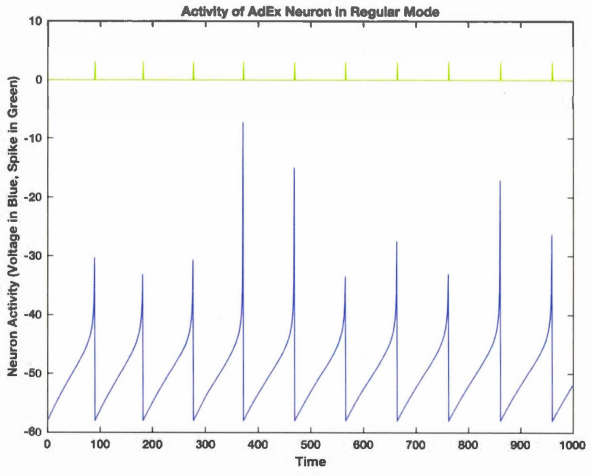


Fig. 3.2 Plot of the regular activity of AdEx Neuron when it is in regular configuration mode. Simulation Time = 1000 Time steps. Voltage in blue. Spike Output in green.

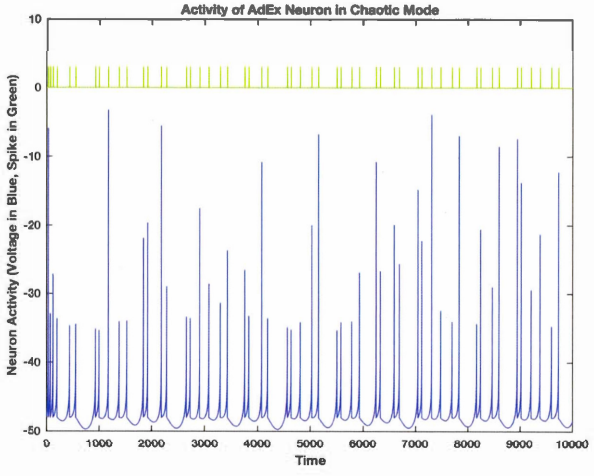


Fig. 3.3 Plot of the irregular activity of AdEx Neuron when it is in chaotic configuration mode. Simulation Time = 10000 Time steps. Voltage in blue. Spike output in green.

3.4 Controlling the chaotic behavior of the AdEx neuron

To control the chaotic activity of the AdEx Neuron when the latter is running in chaotic mode, we use the SFC method discussed in chapter 1. The SFC method applies a feedback connection on the voltage equation of the neuron. Note that in the remainder of this chapter, we refer to this method simply as Time Delay feedback control.

First, let us write the equations of the AdEx Neuron in their Euler form:

$$V(t) = V(t-1) + dt\left[-gL(V(t-1) - E_L) + gL\,\Delta_T \exp\left(\frac{V(t-1) - V_T}{\Delta_T}\right) + I_c - \psi + I\right]/C$$

$$\psi(t) = \psi(t-1) + dt[a(V(t-1) - E_L) - \psi]/\tau_w$$

Where *dt* is the Euler time step which is set to *0.1* for good precision.
The system starts with $V(0) = V_r$ and $\psi(0)=0;$
If $V(t) > \theta$ *then* $V(t) = V_r$ *and* $\psi(t) = \psi(t) + b$

Also,

$$\gamma(t) = \begin{cases} 1, V(t) > \theta \\ 0, V(t) \leq \theta \end{cases}$$

The SFC method requires the definition of a new parameter called τ_{DFC}, which is a time delay that is applied on the spike output γ of the neuron. Furthermore, we define $t_{control}$ as the time step when the chaos control should start taking effect. Of course, $t_{control}$ should be greater or equal to τ_{DFC}. So, as soon as γ has evolved past the time delay τ_{DFC}, then $\gamma(t - \tau_{DFC})$ becomes valid and applicable to the voltage of the neuron as the following:

If $t >= t_{control}$ *then*

$$V(t) = V(t) + \big(\theta - V(t)\big)(1+\epsilon)\gamma(t - \tau_{DFC})$$

Where, ϵ is a small positive infinitesimal quantity that is experimentally set to 0.0001

This means that if $\gamma(t - \tau_{DFC})$ is equal to 1 then the neuron voltage that is equal to $V(t) + \big(\theta - V(t)\big) * (1+\epsilon) * \gamma(t - \tau_{DFC})$ becomes $V(t) + \big(\theta - V(t)\big) * (1+\epsilon) * 1$ which results to $\theta + \epsilon$. Otherwise, when no incoming spike is available at $t - \tau_{DFC}$, thus $\gamma(t - \tau_{DFC}) = 0$, then $V(t)$ remains the same because $V(t) + \big(\theta - V(t)\big) * (1+\epsilon) * 0$ results to $V(t)$. In this mode of operation, a single delayed spike, that is exciting the neuron, is enough to drive the neuron's voltage above its threshold.

Next, we will show the evolution of an AdEx Neuron running in chaotic mode, where chaos control is applied at time step 1500 ($t_{control} = 1500$) for a self-feedback connection time delay of 200 ($\tau_{DFC} = 200$).

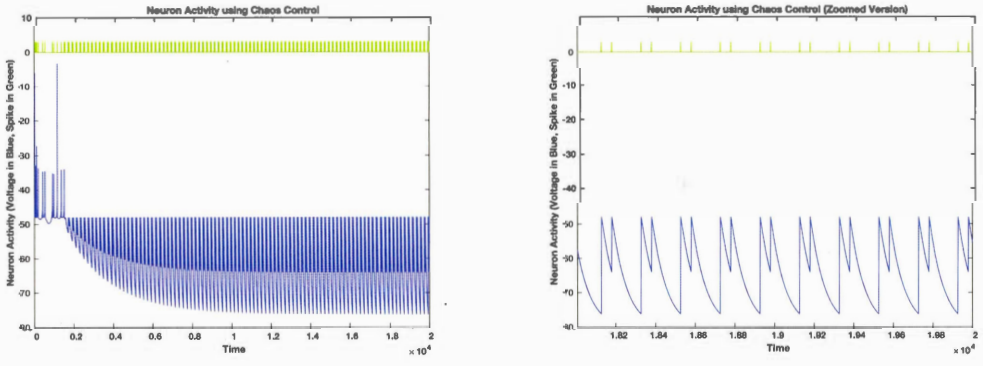


Fig. 3.4 Chaos Control: Plot of the AdEx Neuron activity using Chaos Control. The AdEx Neuron is configured to run in chaos mode. The Neuron runs for 20000 time steps. Chaos control is applied at time step 1500. As we can see in the graph on the left, before time step 1500, the output of the neuron is irregular, but when chaos control is achieved the output of the neuron becomes regular and periodic, in period containing two spikes; occurring in a time window period of length 200 for this example. Left: Simulation for 20000 time steps. Right: Zoomed version of the simulation from time step 18000 to time step 20000.

In the same settings, we plot (Figure 3.5) the evolution of the adaptation variable "ψ".

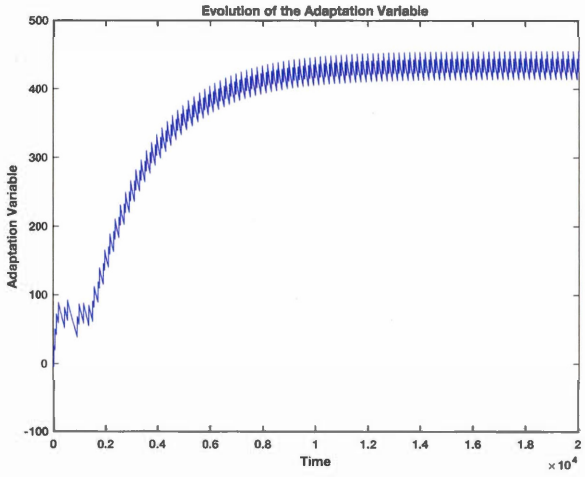


Fig. 3.5 Chaos Control: Plot of the evolution of the AdEx Neuron Adaptation Variable "ψ" using Chaos Control. As we can see in the plot, when chaos control is applied at time step 1500, the adaptation variable "ψ" expands until it stabilizes in the range of 400 and 430 (i.e. it stabilizes between 400 and 400 'plus (+)' its adaptation reset value b; which is equal to 30).

Now, we demonstrate the synchronization of the AdEx Neuron variables when chaos control is applied. In order to do so, we plot the difference between the absolute value of the neuron voltage at time t and the absolute value of the neuron voltage at time $t - \tau_{DFC}$. Also, we plot the difference between the absolute value of the neuron adaptation variable at time t and the absolute value of the neuron adaptation variable at time $t - \tau_{DFC}$. Furthermore, we plot the difference between the absolute value of the neuron spike output at time t and the absolute value of the neuron spike output at time $t - \tau_{DFC}$. If the neuron is synchronizing then these absolute differences should converge to 0. This is shown in the plot of figure 3.6, next.

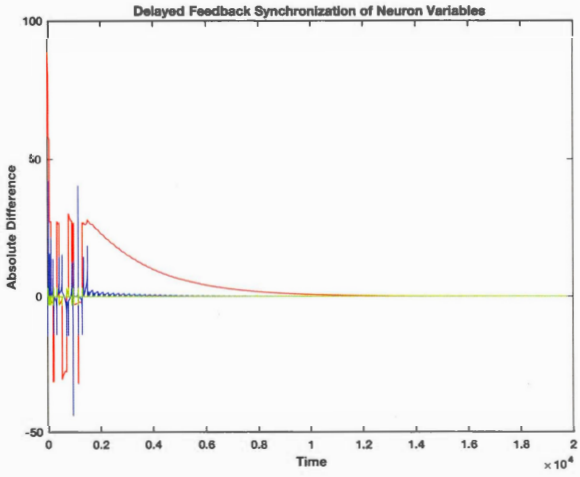


Fig. 3.6 Chaos Control: Synchronization of Neuron Variables using Time Delay feedback control. As we can see in the plot, when chaos control is applied at time step 1500, the difference between the absolute values of the neuron variables and their time-delayed (e.g. Time delay equal to 200) absolute values converges to 0 confirming their synchronization. In blue: Difference between the voltage and its delayed value. In red: Difference between the adaptation variable and its delayed value. In green: Difference between the spike output and its delayed value.

In the plot of Figure 3.7 next, we show the stabilized UPO of the AdEx Neuron after chaos control.

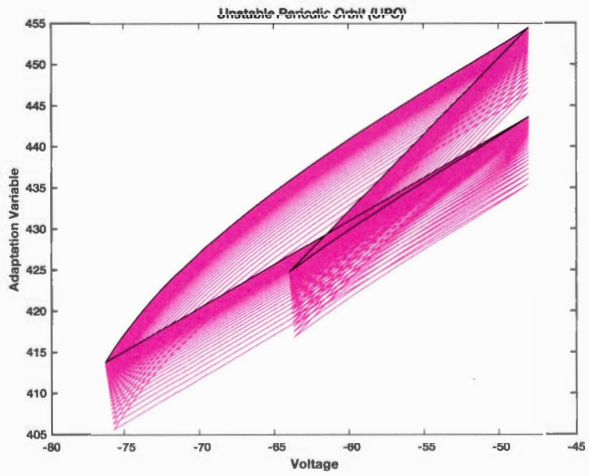


Fig. 3.7 Phase Space Plot of the AdEx Neuron after chaos control: We see the stabilization of the Unstable Periodic Orbit (UPO); depicting the Neuron's state, by plotting the Adaptation variable of the AdEx Neuron versus its voltage. In magenta: The evolution of the UPO

towards its stabilization. In black: **The stabilized UPO**.

3.5 The resonate and fire neuron

The Resonate and Fire Neuron (RAF) is a neuron model invented by Izhikevich, in 2001. The RAF Neuron has an important characteristic, which is: It is sensitive to the spike timing frequency of the input stimulus. The state variable of the RAF is complex, this means its equations has a real and imaginary part which are the following:

$$\frac{dx}{dt} = bx - wy$$

$$\frac{dy}{dt} = wx + by$$

That can be written in complex form as:

$$\frac{dz}{dt} = (b + iw)z$$

Here, $z = x+iy$ is "complex variable that describes the oscillatory activity of the neuron" (Izhikevich, 2001). The imaginary part, y, describes the voltage. The parameter b is the rate at which the neuron goes back to its reset value (i.e. $z=0$). The parameter w describes the frequency of oscillation. In general, b is negative and is set to -1 (Izhikevich, 2001), $w = 10$ (Izhikevich, 2001).

The threshold upon which the neuron fires a spike is applied on the voltage variable y and is equal to 1 (Izhikevich, 2001). In other words, if $y > 1$ then $z = 0$ and the neuron emits a spike.

Let us write the equations of the RAF Neuron in Euler form:

$$x(t) = x(t-1) + bx(t-1) - wy(t-1)$$

$$y(t) = y(t-1) + wx(t-1) + by(t-1)$$

The spike output of the neuron is defined as:

$$\gamma(t) = \begin{cases} 1, y(t) > \theta \\ 0, y(t) \leq \theta \end{cases}$$

Where $\theta=0$ is the threshold of the neuron. As we mentioned earlier, whenever the neuron voltage crosses its threshold then it is reset to 0:

$$If\ y(t) > \theta\ then\ x(t)=0\ and\ y(t)=0$$

The neuron receives input in the form of spikes; from other neurons, that are integrated through the voltage variable. For instance, consider a spiking neuron *j* that is connected to a RAF neuron *i* via a feedforward connection; from *j* to *i*, that has weight c_{ij}, then its voltage equation is rewritten as:

$$y_i(t) = y(t-1) + wx_i(t-1) + by_i(t-1) + c_{ij}\gamma_j(t)$$

The RAF Neuron fires when the Interval between two consecutive spikes, that it receives, is near its period, called eigenperiod (Izhikevich, 2001) (Remark: note that in the remainder of this chapter we sometimes refer to the eigenperiod of a RAF Neuron, simply, as its resonance). This is clarified in the following sketch (Figure 3.8) where *V(t)* corresponds to the voltage *y(t)* of the RAF Neuron. For instance, as we can see in Figure 3.8, if input pulse 2 occurs near the eigenperiod of the neuron then the neuron fires, otherwise it won't.

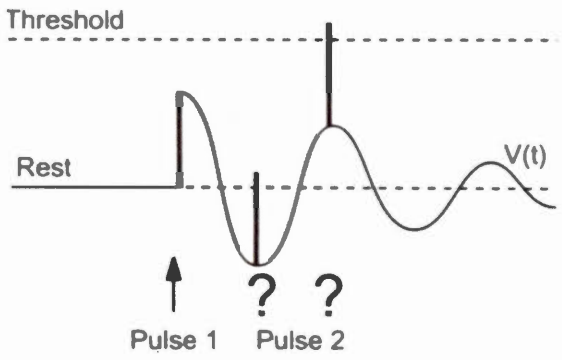


Fig. 3.8 Voltage activity and firing of the RAF Neuron: Input Pulse 1 activates the RAF Neuron. If input pulse 2 occurs near the period of the voltage then the neuron fires, if pulse 2 is not near the period then the neuron won't fire. Excerpt from (Izhikevich, 2001).

3.6 The neural network architecture

In this section, we explain the neural network architecture which is composed of an LIF Neuron, an AdEx Neuron and a layer of RAF Neurons. These neurons are connected via feedforward connections and the AdEx Neuron has a self-feedback connection to itself as we can see in Figure 3.9 next.

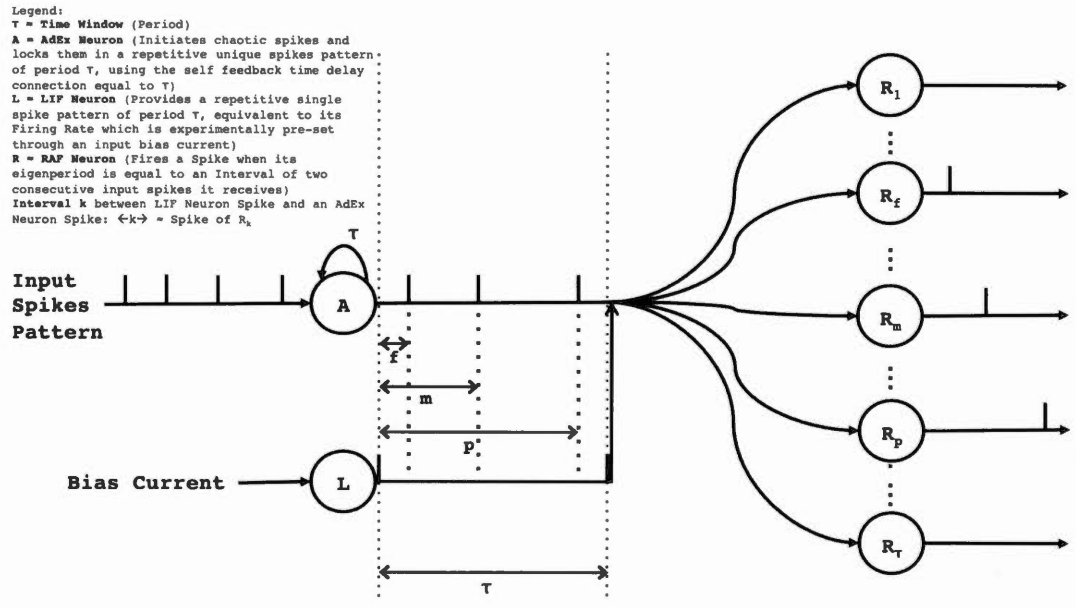


Fig. 3.9 Neural Network Architecture: Spikes are represented with vertical bars. "L" is a LIF Neuron, which repetitively emits a single spike according to the value of its bias current. The bias current is preset to a value that allows the firing rate of the LIF Neuron to be equal to $1/\tau$. "A" is an AdEx Neuron configured to run in chaotic mode and has a self-feedback connection that achieves chaos control. The self-feedback connection of the AdEx Neuron has a time delay equal to "τ". The spikes emitted by the LIF Neuron "L" and the AdEx Neuron "A" are received in repetitive manner by RAF Neurons ("R_1" to "R_τ"). The subscript next to "R" indicates its resonance. A RAF Neuron that has resonance equal to the Inter Spike Interval between the LIF Neuron Spike and the AdEx Neuron Spike will fire; in this illustration "R_f", "R_m" and "R_p" will fire and constitute a single neuronal group that characterizes the Input Spikes Pattern fed to the AdEx Neuron.

The goal of this Neural Network Architecture is to discriminate different Input Spike Patterns that the AdEx Neuron receives. The discrimination is manifested through the activation of different groups of RAF Neurons. In other words, similar input patterns to the AdEx Neuron should activate the same group of RAF Neurons, and different input patterns should activate different groups of RAF Neurons. To achieve this goal, an AdEx Neuron; configured to fire chaotically, is controlled by embedding a self-feedback connection that relays the spikes it receives to itself forcing it to fire in a period equal to the time delay of its self-feedback connection. Since the AdEx

Neuron is chaotic, then it is sensitive to the timing of an input spike it receives. This means, the short-term behavior of the AdEx Neuron is not drastically affected for similar timings of input spike instances; however, the long-term behavior is drastically affected. In other words, according to the first principal of Chaos theory (i.e. sensitivity on initial conditions; discussed in the first chapter), nearby initial inputs should slightly affect the short-term behavior of the dynamics of the system (e.g. The AdEx Neuron) but they would drastically affect its long-term behavior. However, distant initial inputs, also slightly affect the short-term behavior of the dynamics of the system (e.g. The AdEx Neuron), and of course drastically affect its long-term behavior. Besides, according to the theory of Nonlinear Transient Computing (Crook, 2007), nearby timings should produce the same transient in the dynamics of a chaotic neuron and distant timings should produce different transients (Crook, 2007). We exploit these theories in order to achieve our goal of discriminating input spikes patterns fed to the AdEx Neuron: An input is applied during the early evolution of the dynamics of the AdEx Neuron, so similar inputs won't affect much its dynamics but different (i.e. distant) inputs would. Furthermore, we apply chaos control on the AdEx Neuron so the transient it produces, when it receives an input spikes pattern; will remain alive through its controlled dynamics. This means the transient becomes incorporated in the Unstable Periodic Orbit (UPO) that the AdEx Neuron will be converging onto. Moreover, we have to note that we have a LIF Neuron that is emitting a single spike in period equivalent to the period of the AdEx Neuron's UPO, this acts as a time clock. Finally, we have RAF Neurons that are mirroring the manifested repetitive spikes pattern of the stable UPO of the AdEx Neuron per time clock provided by the LIF Neuron.

In order to achieve chaos control over "A" and make it fire periodically, we implement a connection from "A" to itself, as we can see in Figure 3.9. This self-feedback connection has a time delay "τ" which constitutes the length of the unstable periodic orbit (UPO) of the AdEx Neuron. The UPO is represented by a spike pattern

that repeats itself every "τ" time steps and this repetition is due to the self-feedback connection, which is of length "τ". This repetitive spikes pattern is considered as the output of the AdEx Neuron. The LIF Neuron "L" receives a bias input current which is preset to a value that insures the firing frequency of the LIF to be equal one over the length (i.e. $1/\tau$) of the periodic spike pattern emitted by the AdEx Neuron. The spike emitted by the LIF Neuron and the spikes emitted by the AdEx Neuron "A" are received by "τ" RAF Neurons. Each RAF Neuron has an eigenperiod (i.e. a resonance). The eigenperiod of a RAF Neuron is the time gap between any two consecutive input spikes the neuron receives and upon which it fires a spike. In other words, a RAF Neuron fires a spike when it receives two consecutive spikes separated by a time gap equal to its eigenperiod. As we said, we have "τ" RAF Neurons, so "τ" eigenperiods ranging from 1 to "τ" that we set for RAF Neuron with index 1 to "τ" respectively. So, every "τ" time steps, the LIF Neuron fires a spike that is propagated to all RAF Neurons, and during "τ", the AdEx Neuron fires some other spikes that are also propagated to all RAF Neurons. The RAF Neuron with an eigenperiod equal to the Inter Spike Interval (ISI) between the LIF Neuron "L" Spike and the AdEx Neuron "A" Spike will fire. Note that the maximum number of RAF Neurons should be equal to "τ" so we ensure that even if the repetitive output spike pattern of the AdEx Neuron "A" contains a single spike occurring at "τ", then the RAF Neuron that has an eigenperiod equal to "τ" caches it.

In retrospect, we have RAF Neurons that keep on firing according to a repetitive spike pattern coming from an LIF Neuron; which is firing according to a bias current, and from an AdEx Neuron which is firing according to an input spike pattern that it is receiving from outside. So, in this context, we declare the following assertion:

"The subset of RAF Neurons that are firing will constitute a single neuronal group that characterizes the input spikes pattern of the AdEx Neuron".

In the next section, we will attempt to visualize active RAF Neuronal groups through

a set of 100 experiments. Then, in section 3.8, we will study the separation property of these groups for similar and different, single and multiple spikes input patterns. In section 3.9, we will show the large number of unique RAF Neuronal Groups that can be reached with this architecture. In section 3.10, we discuss the relation between our work and important theories in our field of study.

Remarks:

We should note that an external input occurring at time t, to the AdEx neuron is denoted as $\gamma_{External}(t)$ and is added to the adaptation variable of the AdEx Neuron like the following:

$$\psi(t) = \psi(t-1) + \frac{dt[a(V(t-1) - E_L) - \psi]}{\tau_w} + b\gamma_{External}(t)$$

We decided to integrate the external input of the AdEx Neuron (e.g. "A" in Figure 3.9) into its adaptation variable because we follow the approach of (Crook, 2007) in this regard. In fact, we want an input to the neuron to have a slight influence on the neuron's evolving dynamics; which can be achieved by perturbing its adaptation variable, and not a direct impact on the neuron's evolving dynamics; which can be achieved by perturbing its action potential.

Also, the equations of a RAF Neuron "R_i", *where* $i \in \{1, \dots, \tau\}$ are evaluated as the following:

$$x_i(t) = x_i(t-1) - (\frac{\pi}{i+1})y_i(t-1)$$
$$y_i(t) = y_i(t-1) + \left(\frac{\pi}{i+1}\right)x_i(t) - c_{RAF}f_i(t)$$

Such that,

C_{RAF} is a constant that is experimentally set to 1.

And,

$$f_i(t) = \gamma_L(t) D_i(y_i(t-1))$$

Where, $\gamma_L(t)$ is the spike output of the LIF neuron "L" and D is the Dirac delta function, which is defined as:

$$D(a) = \begin{cases} 0 & if\ a < 0\ or\ a > 0 \\ 1 & if\ a = 0 \end{cases}$$

In this way of evaluating the equations of a RAF Neuron, we are sure that a RAF Neuron only responds to an input spike that is occurring at its eigenperiod which is equal to its index i and neglects any input spike that occurs while the RAF Neuron is evolving its dynamics.

To conclude these remarks, we note that the time delay of the AdEx Neuron "A" is experimentally set depending on the length of the input spikes pattern that is fed to "A" as we will see later on in the remainder of this chapter. And, the bias current of the LIF Neuron "L" is experimentally set in order to make its firing rate equal to one spike per time delay of "A".

3.7 Visualizing RAF neuronal groups

In this section, we show the results of approximately 100 experiments that we performed using the neural network architecture explained in section 3.6 but instead of considering an input as a pattern of multiple spikes, we embed a LIF Neuron (e.g. "L_1"); in front of the AdEx Neuron, that takes an arbitrary Input Current and emits a single spike accordingly. This is illustrated in the figure 3.10 next:

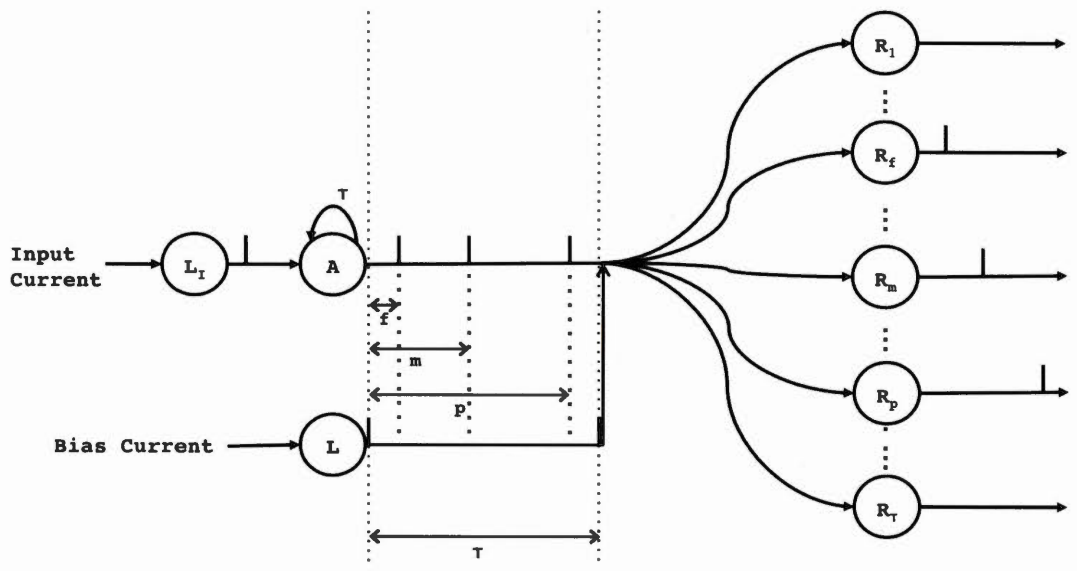


Fig. 3.10 Network Architecture with a variable input current to a LIF Neuron "L_I" embedded before the AdEx Neuron.

We have to note that as soon as the LIF Neuron "L_I" emits a spike; it should remain in its rest state. This is because if we let the LIF Neuron "L_I" keep on firing then this will ruin the behavior of the AdEx Neuron. Thus, a condition is exerted on the equations of the LIF Neuron "L_I" to force it to shut down once it has fired its first spike. In fact, a LIF Neuron fires in regular mode by default, so its repetitive spikes are just redundant and meaningless for our purpose in triggering the AdEx Neuron, besides they have an undesired influence on the evolution of the AdEx Neuron dynamics. In fact, we are only interested in the timing of the first spike; the AdEx Neuron is receiving, which is significantly enough in order to characterize an input current and offers the sufficient and necessary effect of that input with regards to its influence on the AdEx Neuron behavior.

In each experiment, the neural network is fed with an input current to the LIF Neuron (i.e. L_I), which increases by 1 pico Ampere (pA) from an experiment to the next. The input currents range from 51 pA to 162 pA. The lower bound of 51 and the upper bound of 162 were chosen for experimental purposes only, other values can also be

considered.

We note that in these experiments, the Time Delay "τ" of the self-feedback connection of the AdEx Neuron is set to 112 ms. We chose this value because: First, we noticed that a time period of 112 ms is enough to capture the first spike time emitted by the LIF Neuron (i.e. L_I) which ranges between 71ms for Input current equal to 51 pA and 33 ms for an Input current equal to 162 pA. Second, we want each experiment to activate 3 RAF Neurons in order to be able to visualize them in a 3D plot. So, we want a time window that contains 3 spikes (shown in Table 3.2 column 4), which in return will activate 3 RAF Neurons (shown in Table 3.2 column 5). By experimentation, we noticed that a time period of 112 could afford the existence of 3 spikes in it, thus the choice of "τ" equal 112 is convincing.

In the following table, we show the results of our experiments:

Table 3.2 Results of Experiments where Input Current is varied between 51pA and 162pA, AdEx Connection Time Delay is equal to 112 and AdEx Adaptation reset parameter is equal to 20.

Experiment Nbr	Input Current in pA	LIF_I Output (Time of First Spike in mS)	AdEx Output (Spike Times relative to Period 112)	Indexes of Active RAF Neurons	RAF Neuronal Group
1	51	71	11, 34, 71	11, 34, 71	1
2	52	71	11, 34, 71	11, 34, 71	1
3	53	70	11, 34, 71	11, 34, 71	1
4	54	69	11, 34, 71	11, 34, 71	1
5	55	68	11, 34, 71	11, 34, 71	1
6	56	67	11, 34, 71	11, 34, 71	1
7	57	67	11, 34, 71	11, 34, 71	1
8	58	66	11, 34, 71	11, 34, 71	1
9	59	65	11, 34, 71	11, 34, 71	1
10	60	65	11, 34, 71	11, 34, 71	1
11	61	64	11, 34, 71	11, 34, 71	1
12	62	63	11, 34, 71	11, 34, 71	1
13	63	63	11, 34, 71	11, 34, 71	1

14	64	62	11, 34, 71	11, 34, 71	1
15	65	62	11, 34, 71	11, 34, 71	1
16	66	61	12, 34, 72	12, 34, 72	2
17	67	61	12, 34, 72	12, 34, 72	2
18	68	60	12, 34, 72	12, 34, 72	2
19	69	59	12, 34, 72	12, 34, 72	2
20	70	59	12, 34, 72	12, 34, 72	2
21	71	58	12, 34, 72	12, 34, 72	2
22	72	58	12, 34, 72	12, 34, 72	2
23	73	57	12, 34, 72	12, 34, 72	2
24	74	57	12, 34, 72	12, 34, 72	2
25	75	56	12, 34, 72	12, 34, 72	2
26	76	56	12, 34, 72	12, 34, 72	2
27	77	55	12, 34, 72	12, 34, 72	2
28	78	55	12, 34, 72	12, 34, 72	2
29	79	54	12, 34, 72	12, 34, 72	2
30	80	54	12, 34, 72	12, 34, 72	2
31	81	54	12, 34, 72	12, 34, 72	2
32	82	53	13, 34, 73	13, 34, 73	3
33	83	53	13, 34, 73	13, 34, 73	3
34	84	52	13, 34, 73	13, 34, 73	3
35	85	52	13, 34, 73	13, 34, 73	3
36	86	52	13, 34, 73	13, 34, 73	3
37	87	51	13, 34, 73	13, 34, 73	3
38	88	51	13, 34, 73	13, 34, 73	3
39	89	50	13, 34, 73	13, 34, 73	3
40	90	50	13, 34, 73	13, 34, 73	3
41	91	50	13, 34, 73	13, 34, 73	3
42	92	49	13, 34, 73	13, 34, 73	3
43	93	49	13, 34, 73	13, 34, 73	3
44	94	49	13, 34, 73	13, 34, 73	3
45	95	48	14, 34, 74	14, 34, 74	4
46	96	48	14, 34, 74	14, 34, 74	4
47	97	47	14, 34, 74	14, 34, 74	4
48	98	47	14, 34, 74	14, 34, 74	4
49	99	47	14, 34, 74	14, 34, 74	4
50	100	47	14, 34, 74	14, 34, 74	4
51	101	46	14, 34, 74	14, 34, 74	4

52	102	46	14, 34, 74	14, 34, 74	4
53	103	46	14, 34, 74	14, 34, 74	4
54	104	45	14, 34, 74	14, 34, 74	4
55	105	45	14, 34, 74	14, 34, 74	4
56	106	45	14, 34, 74	14, 34, 74	4
57	107	44	15, 34, 75	15, 34, 75	5
58	108	44	15, 34, 75	15, 34, 75	5
59	109	44	15, 34, 75	15, 34, 75	5
60	110	44	15, 34, 75	15, 34, 75	5
61	111	43	15, 34, 75	15, 34, 75	5
62	112	43	15, 34, 75	15, 34, 75	5
63	113	43	15, 34, 75	15, 34, 75	5
64	114	42	15, 34, 75	15, 34, 75	5
65	115	42	15, 34, 75	15, 34, 75	5
66	116	42	15, 34, 75	15, 34, 75	5
67	117	42	15, 34, 75	15, 34, 75	5
68	118	41	15, 34, 76	15, 34, 76	6
69	119	41	15, 34, 76	15, 34, 76	6
70	120	41	15, 34, 76	15, 34, 76	6
71	121	41	15, 34, 76	15, 34, 76	6
72	122	40	16, 34, 76	16, 34, 76	7
73	123	40	16, 34, 76	16, 34, 76	7
74	124	40	16, 34, 76	16, 34, 76	7
75	125	40	16, 34, 76	16, 34, 76	7
76	126	39	16, 34, 76	16, 34, 76	7
77	127	39	16, 34, 76	16, 34, 76	7
78	128	39	16, 34, 76	16, 34, 76	7
79	129	39	16, 34, 76	16, 34, 76	7
80	130	39	16, 34, 76	16, 34, 76	7
81	131	38	16, 34, 77	16, 34, 77	8
82	132	38	16, 34, 77	16, 34, 77	8
83	133	38	16, 34, 77	16, 34, 77	8
84	134	38	16, 34, 77	16, 34, 77	8
85	135	38	16, 34, 77	16, 34, 77	8
86	136	37	16, 34, 77	16, 34, 77	8
87	137	37	16, 34, 77	16, 34, 77	8
88	138	37	16, 34, 77	16, 34, 77	8
89	139	37	16, 34, 77	16, 34, 77	8

90	140	37	16, 34, 77	16, 34, 77	8
91	141	36	16, 34, 77	16, 34, 77	8
92	142	36	16, 34, 77	16, 34, 77	8
93	143	36	16, 34, 77	16, 34, 77	8
94	144	36	16, 34, 77	16, 34, 77	8
95	145	36	16, 34, 77	16, 34, 77	8
96	146	35	17, 34, 78	17, 34, 78	9
97	147	35	17, 34, 78	17, 34, 78	9
98	148	35	17, 34, 78	17, 34, 78	9
99	149	35	17, 34, 78	17, 34, 78	9
100	150	35	17, 34, 78	17, 34, 78	9
101	151	35	17, 34, 78	17, 34, 78	9
102	152	34	17, 34, 78	17, 34, 78	9
103	153	34	17, 34, 78	17, 34, 78	9
104	154	34	17, 34, 78	17, 34, 78	9
105	155	34	17, 34, 78	17, 34, 78	9
106	156	34	17, 34, 78	17, 34, 78	9
107	157	34	17, 34, 78	17, 34, 78	9
108	158	33	17, 34, 78	17, 34, 78	9
109	159	33	17, 34, 78	17, 34, 78	9
110	160	33	17, 34, 78	17, 34, 78	9
111	161	33	17, 34, 78	17, 34, 78	9
112	162	33	17, 34, 78	17, 34, 78	9

As we can see in Table 3.2, we have 9 different RAF Neuronal Groups for 112 different values of Input currents. Very close input currents generate the same response from the LIF Neuron (i.e. L_I); which is visible through its spike output (Column 3 in the table). Furthermore, nearby occurrences of the first spike (i.e. LIF L_I Spike output) received by the AdEx Neuron make it generates same responses (Column 4 in the table) that are mirrored by RAF Neurons (Column 5 in the table). This is the reason why we have the same sets of RAF Neurons firing accordingly, which will constitute the neuronal groups 1 to 9 (Column 6 in the table).

In order to visualize the RAF neuronal groups, we plot in 3D space the indexes of

RAF Neurons that fire depending on the input current of each experiment. Same sets of active neurons constitute the same group. We have 9 different sets, so 9 groups. This is shown in figure 3.11 next.

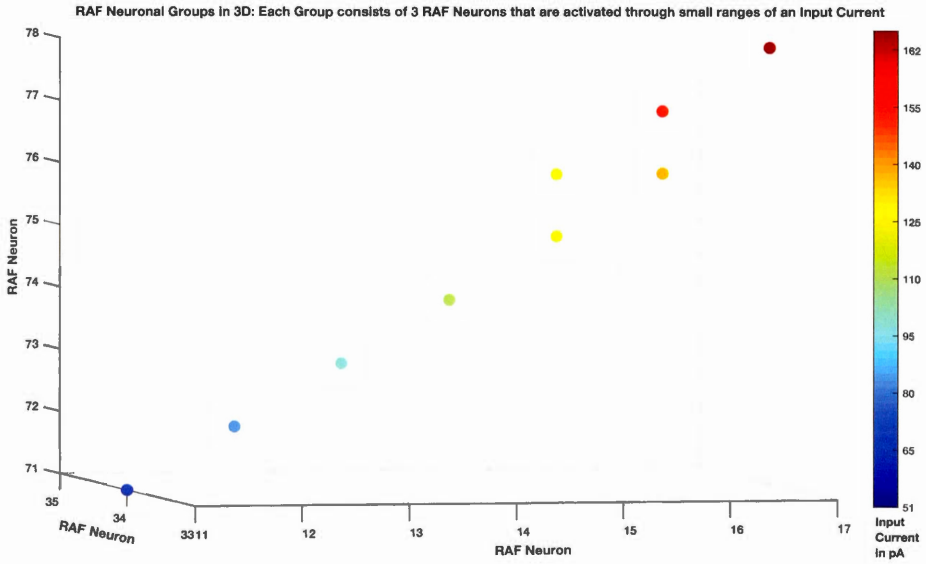


Fig. 3.11 Visualization of 9 neuronal groups of RAF Neurons showed in 3D plot on the left, by varying the input current of LIF Neuron (i.e. L_I) ranging from 51 pA to 162 pA in increments of 1pA (color bar on the right). In the 3D plot, each axe represents the index of a RAF Neuron. We plot the indexes of the RAF Neurons that were active for a particular input current. Very close input currents activate the same RAF Neurons, thus the same neuronal group. The colors indicate the match between very close input currents and their corresponding neuronal group.

In this section, we demonstrated RAF Neuronal groups characterizing an input current. Next, we will study the separation property of these groups depending on different varieties of input spike patterns.

3.8 Separation property of neuronal groups

In order to study the separation property of RAF Neuronal Groups, we will refer back

to the original neural network architecture illustrated in Figure 3.9.

Crook (2007) studied the Separation property of his Nonlinear Transient Computing Machine (NTCM) using Euclidean Distance measure between input patterns to the machine and its response. This measure will be discussed later, but first as a flash back, the core of the NTCM is composed of two chaotic spiking neurons, namely the Nonlinear Dynamic State (NDS) Neurons (Crook et al., 2005), which are in chaos control mode. The chaos control is removed whenever an input is presented to the network. But, when a chaotic spiking neuron that is functioning in control mode is altered to chaotic mode, then its Unstable Periodic Orbit (UPO) starts to slightly diverge from its course. This slight divergence is depicted in the spikes output pattern of the neuron and is called a Nonlinear Transient. In our work, the same phenomenon happens, but instead of diverging from a stable UPO which happens on the long run after chaos control, our AdEx Neuron diverges from the first UPO, which is also stable since it is the first one. Then, we apply chaos control just to lock it in a periodic output.

In this section we use the Euclidean Distance to study the separation of RAF Neurons to inputs fed to the network. Euclidean distance was also used in (Crook, 2007) and (Maass et al., 2002) to study the separation property of their neural network architectures.

If our Neural network architecture has a separation property then:

1. The network should provide similar responses for similar inputs fed to it.
2. The network should provide different responses for different inputs fed to it.
3. The Euclidean Distance between any two inputs should be proportional to the Euclidean Distance of their corresponding responses.

The Response here is the Group of RAF Neurons that is activated for a specific input. The Euclidean Distance between two groups, lets say G_a and G_b of equal number of

RAF Neurons, is calculated as the following:

$$D(G_a, G_b) = \sqrt{\sum_k^n (RAF_{k,a} - RAF_{k,b})^2}$$

Where G_v is a group of n RAF Neurons, k indicates the k^{th} RAF Neuron inside group v, and $RAF_{k,v}$ is the index of a RAF Neuron.

Separation of Single spike Input patterns:

To study the separation property of the network when the latter is fed with an input pattern consisting of a single spike then we conduct a sequence of experiments where the position of a single spike, is shifted between 1 and the length of the input pattern. In each experiment, the input pattern and its corresponding RAF Neuronal group are recorded.

We use these recordings in order to compare, using Euclidean Distance measure, the similarity of inputs and their corresponding outputs, the differences of the inputs and their corresponding outputs and finally the proportionality of the Euclidean distances between the inputs and their corresponding outputs. We follow three scenarios in order to fulfill such comparison:

First scenario: We pick an input pattern that has its single spike occurring at the **first position** in the pattern. We calculate the Euclidean Distances between the prototype and all the remaining input patterns. Then we calculate the Euclidean Distances between the RAF Neurons Group that was activated for this input prototype and the RAF Neurons Groups that were activated for all the remaining input patterns.

Second Scenario: We pick an input pattern that has the position of its single spike at the **middle** of the pattern. We calculate the Euclidean Distances between the prototype and all the remaining input patterns. Then we calculate the Euclidean

Distances between the RAF Neurons Group that was activated for this input prototype and the RAF Neurons Groups that were activated for all the remaining input patterns.

Third Scenario: We pick an input pattern that has the position of its single spike at the **end** of the pattern. We calculate the Euclidean Distances between the prototype and all the remaining input patterns. Then we calculate the Euclidean Distances between the RAF Neurons Group that was activated for this input prototype and the RAF Neurons Groups that were activated for all the remaining input patterns.

While executing the first scenario, we noticed that the responses of the network flattens out over a very wide range (Figure 3.12 next) of the time of occurrence of the single spike in the input pattern (i.e. from 120 to 160 as seen in Figure 3.12), which badly ruins the separation of the inputs because we'll have the same RAF Neurons firing for a very large portion of nearby inputs. This is seen in the plot of Figure 3.12 next.

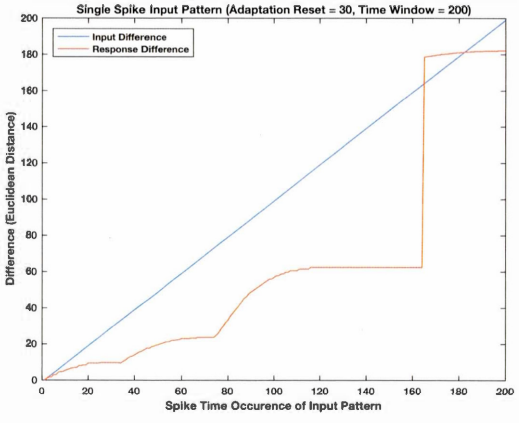


Figure 3.12 Separation of Single Spike Input Patterns when the spike time occurrence is ranging from 1 to 200.

By doing further experiments, we noticed that this is a drawback of the adaptation variable ψ of the AdEx Neuron, which is accumulating by b at each reset after a spike

emission. In other words, for large adaptation reset (i.e. for large b), we have large gaps between the spikes emitted by the AdEx Neuron. But, b is a parameter that can be arranged, and which is set to 30 by default. Also, remember that active RAF Neurons are a mirror of the spikes output pattern of the AdEx Neuron. So, if we reduce the value of b then we will reduce the gap between consecutive AdEx spikes, which in return offers the opportunity of additional spikes to be emitted by the AdEx Neuron, thus enriching its dynamic response to input spikes and making it not quiescent for big time laps. Furthermore, this will solve the redundancy of having same RAF Neurons being activated for a large number of nearby input spikes. In Figure 3.13 we plot the activity of the AdEx Neuron for three different values of b, which as we said affects its adaption and therefore the time lapse between spikes.

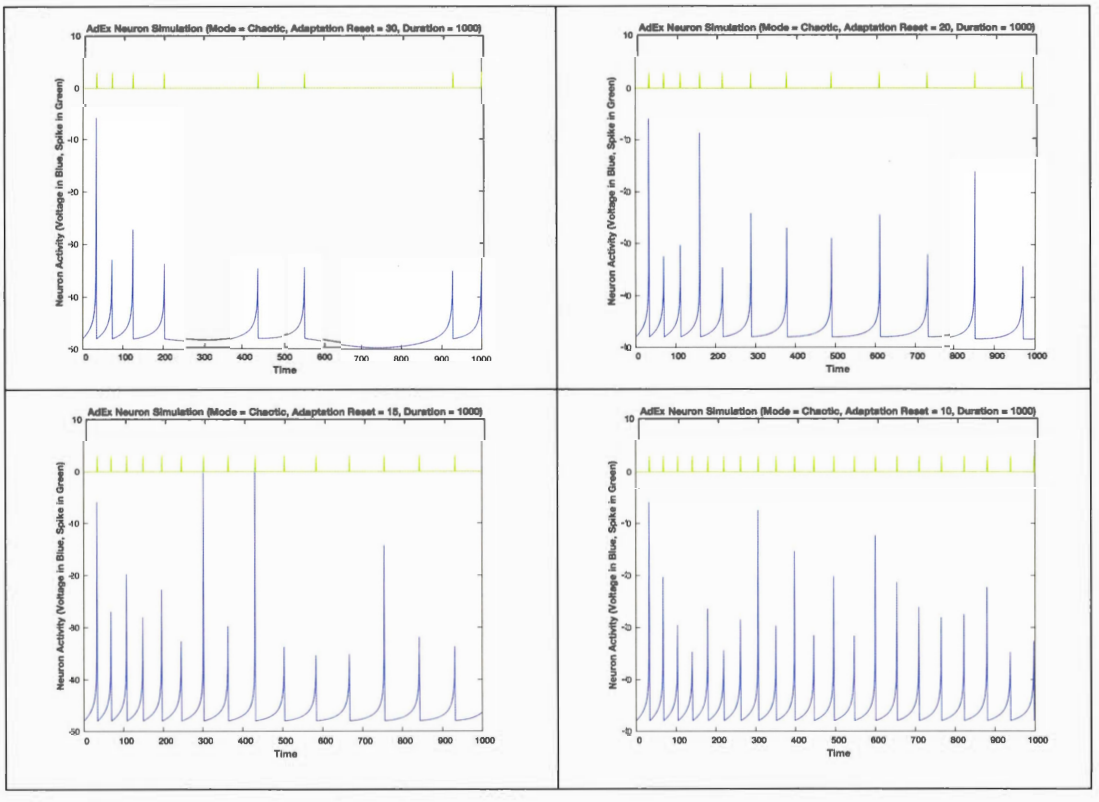


Figure 3.13 Behavior of AdEx Neurons for different values of the reset parameter of its adaptation variable. By decreasing the adaptation reset, the gap between spikes decreases.

In this regard, we chose the value of b to be equal to 20, this way, the AdEx Neuron became weakly chaotic but didn't lose its property of chaotic behavior, also the gap between early spikes won't expand fast as it was for b = 30 and is approximately in the range of 40. Next, in figure 3.14, we will show the improvement of the separation in terms of the Euclidean Distances between RAF Neurons groups responding to single spikes ranging from 1 to the length of the input pattern.

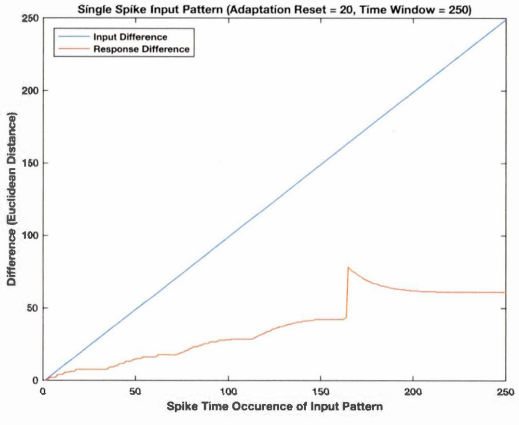


Figure 3.14 Amelioration of the Separation of single spike input patterns

After solving the problem of the flattening of the responses, we faced another problem: As the difference between the responses is increasing for a large part of the input patterns (2 third of the input length in figure 3.14), which is very good, then it starts decreasing at a small part in the end (last third of the input length in figure 3.14). This behavior was also observed in (Crook, 2007) and was suggested that it might be the artifact of the distance measure (e.g. The Euclidean Distance) being used. We tried other distance measures and we dig deeper in the problem, we find out that the decrease in the separation responses for spike inputs occurring at the end of the input is not the outcome of the norm being used. In fact, the last spike that the AdEx Neuron is emitting within its time window and the feedback control that is being exerted on that spike causes this phenomenon (i.e. the decrease in the separation responses for a spike occurring at the end part of the input pattern). In

other words, the feedback control is not giving time for the neuron dynamics to properly evolve after the occurrence of the last spike being received. A work around that I came up with and which functioned is the following:

1- Force the input pattern to have a length equal to the time of the delay feedback connection of the AdEx Neuron.
2- Choose the time window of the delay feedback connection equal to the time reached to arrive at one of the time occurrences of the spikes that the AdEx Neuron emits by default (i.e. when it is running in isolation). For instance, the spikes emitted by the AdEx Neuron as shown in figure 3.13 when it is running in isolation in chaotic mode with different values of its adaptation reset parameter are summarized in the table 3.3 below.

Table 3.3 Spikes Occurrences for different values of the adaptation reset parameter and suitable Time Windows for the self-feedback connection

Adaptation Reset Parameter	**Spike Time Occurrences**	**Suitable Time Windows**
30	34, 74, 126, 204, 440…	33, 73, 125, 203, 339…
20	34, 71, 113, 162, 219, 290, 380, 492…	33, 70, 112, 161, 218, 289, 379, 491…
15	34, 70, 109, 151, 197, 247, 303, 364, 492…	33, 69, 108, 150, 196, 246, 302, 363, 491…
10	34, 69, 105, 142, 181, 221, 263, 307, 352, 399, 447, 497…	33, 68, 104, 141, 180, 220, 262, 306, 351, 398, 446, 496…

So, depending on the user choice of a maximum length desired for the input pattern, then any of the values of the Spike Time Occurrences shown in table 3.3 minus 1; with respect to the adaptation reset value, is a suitable time

window for the delay feedback connection of the AdEx Neuron.

Next, we plot (Figure 3.15) the separation of the inputs by considering the work around that I just explained above. In this example, we proceed with the same value of the adaptation reset parameter (e.g. b = 20), but we set the length of the input pattern to be equal to 289 and we choose a proper time delay for the AdEx Neuron self feedback connection equal to 289.

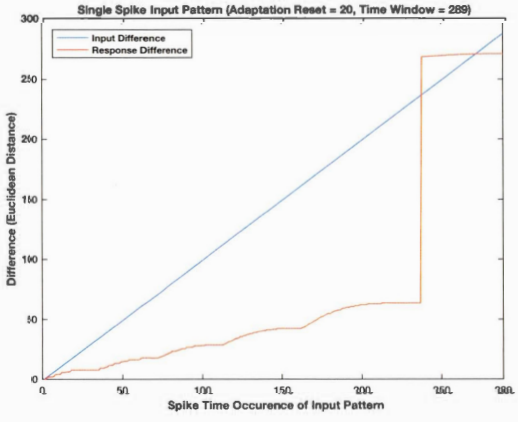


Figure 3.15 Further Amelioration of the Separation of single spike input patterns by choosing a suitable time window of the connection delay.

As we can see in Figure 3.15, the Euclidean Distance of the RAF Neurons Groups won't decrease at the end in contrast of what was happening before (Figure 3.14).

Now, we continue with our experimental proof towards Scenario 2 and 3 by applying the appropriate conditions we discussed above. The results of Scenario 2 are shown in figure 3.16, here we fixed the occurrence of the single spike at the middle of the input pattern (i.e. at 289/2 = 144) and we compared the Euclidean Distances accordingly.

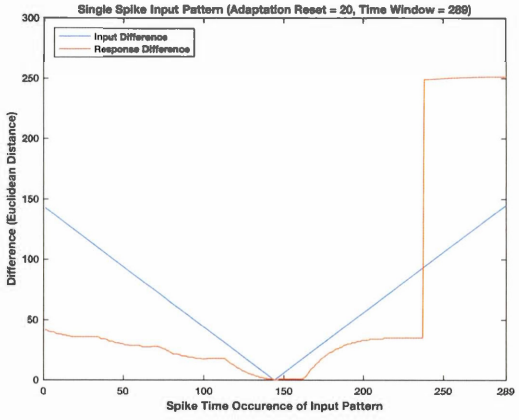


Figure 3.16 Separation of Single Spike Input Patterns when the spike time occurrence is ranging from 143 to 1 and from 145 to 289.

The results of Scenario 3 are shown in figure 3.17, here we fixed the occurrence of the single spike at the end of the input pattern (i.e. at 289) and we compared the Euclidean Distances accordingly.

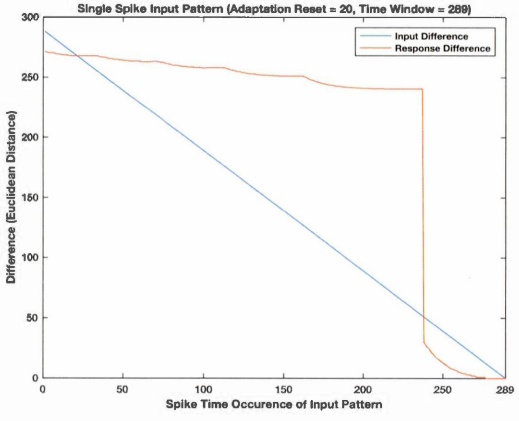


Figure 3.17 Separation of Single Spike Input Patterns when the spike time occurrence is decreasing from 289 to 1.

As we can see in Figures 3.15, 3.16 and 3.17 the responses of the neural network satisfy the separation property for different essential positions of a single spike in the input pattern. For the sake of clarity, we will show the same process of single spike input pattern separation for an adaptation reset value equal to 10 and a time window equal to 262. The results of these experiments are shown in figure 3.18 next.

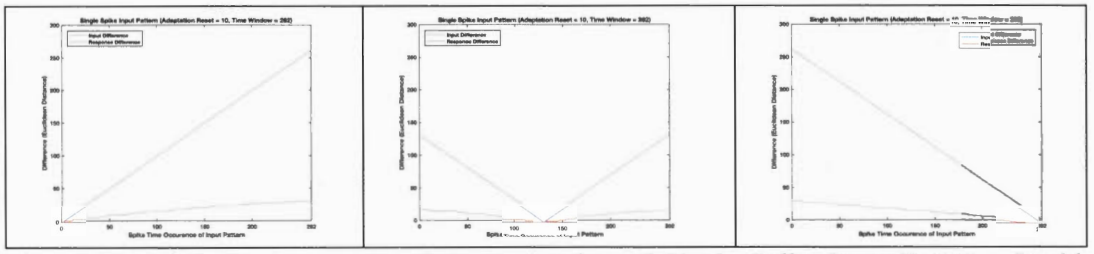


Figure 3.18 Additional example of the Separation of Single Spike Input Patterns. In this example the adaptation reset is equal to 10 and the time window is equal to 262. The spike time occurrence is increasing from 1 to 262, in the first graph, decreasing from 130 to 1 and increasing from 132 to 262 in the second graph, and decreasing from 262 to 1 in the third graph.

Until now, we have shown that the neural network architecture is able to separate single spike input patterns. Next, we will show that it is also able to separate multiple spike input patterns.

Separation of Multiple Spikes Input patterns:

To test the separation property of the network when the latter is fed with an input pattern consisting of multiple spikes then we conduct different experiments where noise is applied on the input. This means, a jitter is applied on the spikes of an input pattern such that the position of each spike in the input is jittered by a random value. We choose a jitter equal to 10, so every spike in the input is randomly shifted between - 10 and +10 from its initial position in the pattern. Also, we choose the adaptation reset parameter of the AdEx Neuron to be equal to 10, the input pattern length equal to 220 and the self-feedback connection time delay of the AdEx Neuron equal to 220.

We tested input patterns consisting of 2, 3, 4 and 5 spikes. In each test, a set of 50 experiments is conducted. In the first experiment of every test, a random spikes pattern, considered as a prototype pattern, is picked and fed to the network, and its RAF Group is recorded. In each of the remaining 49 experiments, a noisy version of the prototype pattern is fed to the network; by applying the jitter discussed above, and its corresponding RAF Group is also recorded. Then, we calculate the Euclidean

Distance between each noisy version of the prototype and the prototype recorded in the first experiment, also, we calculate the Euclidean Distance between each RAF Group activated in response to the noisy input and the RAF Group initially recorded in the first experiment. The Euclidean Distances of the input differences and the RAF Group differences are plotted (Figure 3.19) in a surface area type of plot so we can easily visualize their proportions. Remember that if the Separation Property holds then the differences of inputs and responses should be proportional. The results of 4 sets of 50 experiments are shown in the figure 3.19, next. The prototype input patterns are randomly picked in each set of experiments. As we can see in the plots of figure 3.19, the input difference and the RAF Group difference are always proportional; in other words, whenever the input difference is large then the response difference is large and vice versa. This confirms the separation property of the network for multiple spikes input patterns.

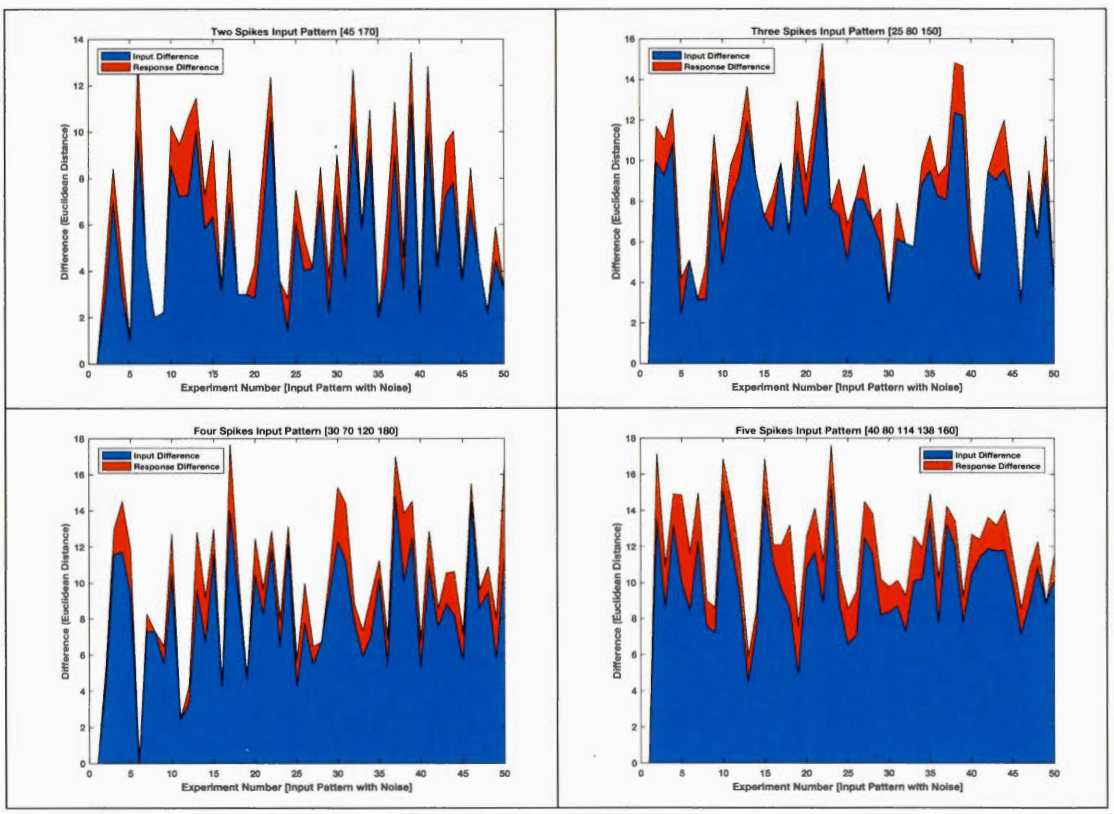


Fig. 3.19 Separation of multiple spikes input patterns.

3.9 Large number of RAF neuronal groups

A large number of RAF neuronal groups can be reached by varying the time delay "τ" of the self-feedback connection of the AdEx Neuron, the adaptation reset parameter "b" of the adaptation variable of the AdEx Neuron and the time of the first spike that the AdEx Neuron is receiving. To demonstrate this, we conduct eighty thousand experiments approximately, which are divided into subsets of experiments by following this protocol: First, for the sake of simplicity, we fix the adaptation-reset parameter to 10 (e.g. b = 10). Then we proceed as follows: The time delay "τ" is increased by time increments depending on the default spike time occurrence of the AdEx Neuorn with adaptation reset equal to 10. Each value of the time delay defines a subset of experiments. In each subset, the time of the first spike that the AdEx neuron is receiving will be varied from 1 to the value of "τ-1" of that subset. Remember that "τ-1" represents a suitable time window for separation as we discussed in the previous section. Table 3.4 summarizes the protocol of our experimentation.

TABLE 3.4 Experiments Protocol by varying the Time Delay and the Time of first spike

Default Spike Occurence	Suitable Time Delay	Time(s) of first spike
34	33	1, 2, 3, ..., 33
69	68	1, 2, 3, ..., 68
105	104	1, 2, 3, ..., 104
142	141	1, 2, 3, ..., 141
.	.	...
.	.	...
.	.	...

399	398	1, 2, 3, …, 398

As we can see in Table 3.4, the maximum value of the time delay that we want to reach is 398. We chose the value of 398 as an upper limit because it is enough to give us a good approximation of the rate by which the number of groups is increasing, as we will see later.

To find the exact total number of experiments that we'll be conducting, we can sum up the number of experiments (say k) that is conducted for each value of the time delay "τ". This can be written as the following series:

$$S = \sum_{k=1}^{\tau} k$$

Where its τ^{th} partial sum is the triangular number and is equal to:

$$\sum_{k=1}^{\tau} k = \frac{\tau}{2}(\tau + 1)$$

Substituting τ by 33 to 398, we get: (33/2) * (33 +1) + (68/2) * (68+1) + (104/2) * (104 +1) + (141/2) * (141 +1) + … + (398/2) * (398 +1) = 79401. So, the total number of experiments that we conduct by varying "τ" with its suitable time increments and the time of the first spike between 1 and "τ" is 79401.

In these experiments the AdEx Neuron is directly fed with an input spike in an automatic fashion through each experiment. A routine is created that will feed the AdEx Neuron with an input spike depending on the value of "τ"; which will define a subset of "τ" experiments to be executed. For example, if "τ" equal 33, then a subset of 33 experiments are executed where the time position of the input spike to the AdEx neuron is incremented by 1 in each experiment of the subset. This means the input spike to the AdEx neuron occurs at t=1 for the first experiment in the subset of 33 experiments, at t = 2 for the second experiment in the subset of 33 experiments, t =

3 for the third, up to t = 33 for the 33rd experiment. Afterwards, "τ" is set to the next suitable time window (e.g. τ =68) and the input spike to the AdEx neuron will occur at t = 1 for the first experiment in the subset of 68 experiments, at t = 2 for the second experiment in the subset of 68 experiments, t = 3 for the third, up to t = 68 for the 68th experiment. The same process will go on until τ = 398. Of course, in each experiment, the network is reinitialized to its initial conditions.

Table 3.5 contains the results of our experiments, where we plot important data that we gather from it, in the following figures next.

Table 3.5 Results of the Experiments

Time Delay	Minimum Nbr of Active RAF Neurons	Maximum Nbr of Active RAF Neurons	Number of RAF Groups per Time Delay	Number of Unique RAF Groups by varying Input Spike in Time Delay	Running Total of Unique RAF Groups
33	1	1	33	3	3
68	2	2	68	5	8
104	3	3	104	8	16
141	4	4	141	11	27
180	5	5	180	15	42
220	6	6	220	21	63
262	7	7	262	26	89
306	8	8	306	32	121
351	9	9	351	41	162
398	10	10	398	54	216

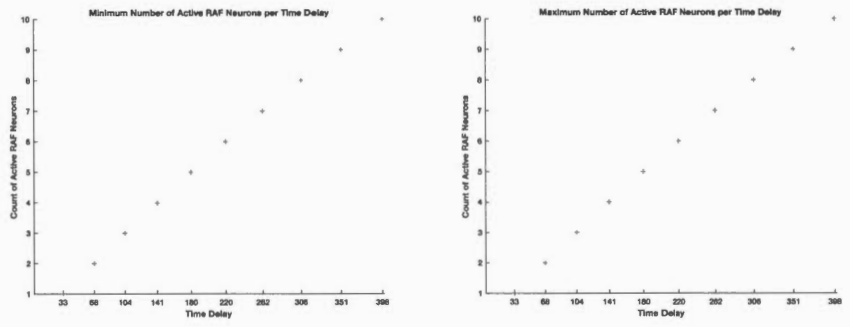


Fig. 3.20 Plot of the Minimum and Maximum number of Activated RAF Neurons per Time Delay of the self-feedback connection of the AdEx Neuron of each subset of experiments. Each subset of experiments has a fixed Time Delay. In each subset, the input spike time of the AdEx Neuron was varied between 1 and the Time Delay, and the Minimum and Maximum number of Activated RAF Neurons were recorded.

As we can see in Figure 3.20, the minimum and maximum number of activated RAF Neurons are always increasing and they are equal. This increase in the number of Activated RAF Neurons is due to the increase of the self-feedback connection time delay of the AdEx Neuron, which gives the opportunity to the connection to capture more spikes emitted by the AdEx Neuron. The latter is chaotic by nature and will continue emitting irregular spikes until the application of chaos control through the self-feedback connection. The equality between the minimum and maximum number of activated RAF Neurons is a proof of the stability of the AdEx Neuron controlled behavior, which is based on a suitable choice of a reliable time delay for the self-feedback connection that was envisaged through hundreds of trials and conceived in well-founded assertions meticulously explained in the previous section.

Next, we plot the number of RAF Neuronal groups reached in every subset of experiments.

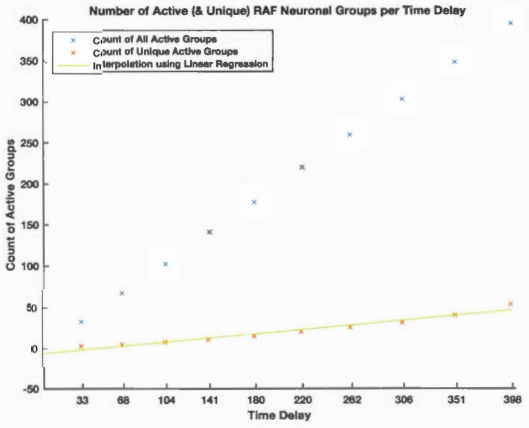


Fig. 3.21 Plot of RAF Neuronal groups per Time Delay of the self-feedback connection of the AdEx Neuron of each subset of experiments. Each subset of experiments has a fixed Time Delay. In each subset, the input spike time of the AdEx Neuron was varied between 1 and the Time Delay, and the active RAF Neuronal groups were recorded. In blue, count of active groups per Time Delay, in red, count of unique RAF Groups per Time Delay, and in Green linear regression curve fitting of unique RAF Groups.

As we can see in Figure 3.21, the number of active RAF Neuronal groups converges to the diagonal which shows that for every input there is an output response from the network. The red marks shows the unique RAF Neuronal Groups per Time Delay. As we can see the interpolation of a linear regression, the number of RAF Neuronal groups is around 13% of the value of the time delay. The Linear Regression has a coefficient of Determination (i.e $R^2 = 0.96$) and is equal to: $-5.91 + 0.13 * \tau$ plotted as a green line in figure 3.21.

After each subset of experiments, the active neuronal groups of the subset are compared to all previous active neuronal groups in all the previous subsets of experiments. The duplicates are removed and the unique ones are summed up. We plot in Figure 3.22 all the unique neuron groups that we reached in all the 80,000 experiments that we conducted. As we can see in the last record in table 3.5 (also shown in Figure 3.22, in the subset of experiments that had time delay equal 398), we reach 216 unique RAF Neuronal groups.

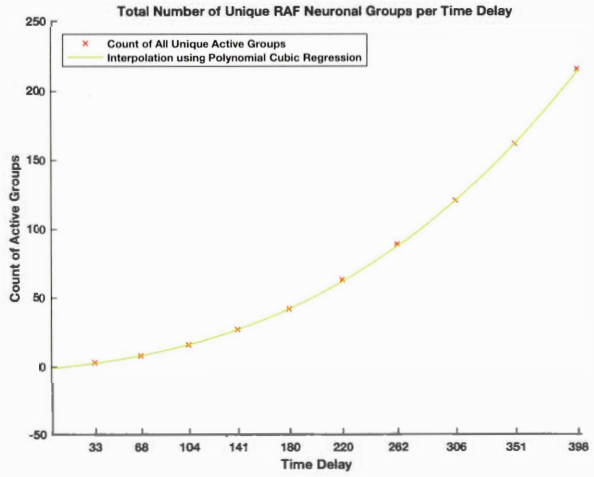


Fig. 3.22 Plot (in red) of the total number of unique RAF Neuronal Groups that is reached through all the experiments. In green, interpolation using polynomial regression. As we can see in the last subset of experiments; which has a time delay of 398, the number reached is nearly 225 unique neuronal groups.

Finally, the graph in Figure 3.22 tells us that the number of coexisting unique RAF neuronal groups that we can reach in this architecture grows nonlinearly with the value of the time delay that we set on the self-feedback connection of the AdEx Neuron. Theoretically, it fits a third degree polynomial with coefficient of Determination (i.e $R^2 = 0.9999$) given by cubic polynomial regression and equal to: -0.911 + 0.086 τ + 0.000616 τ^2 + 0.00000134 τ^3 plotted in an extended form up to τ=1000 in figure 3.23 next.

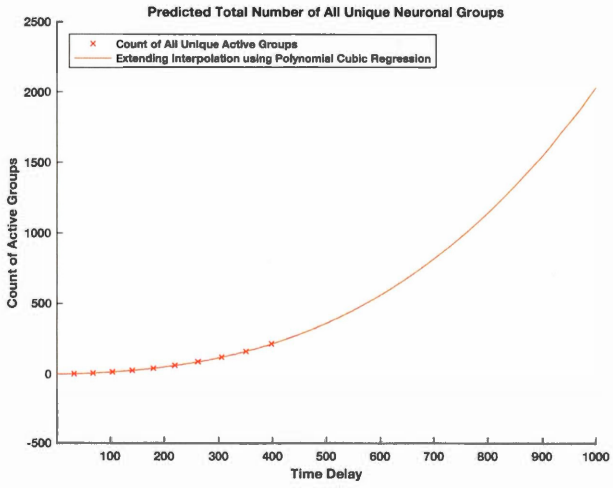


Fig. 3.23 Polynomial regression of the total number of RAF Neuronal Groups. This graph tells us that the number of coexisting RAF neuronal groups that we can reach in this architecture theoretically grows in a nonlinear and power expansion with respect to the value of the time delay that we set on the self-feedback connection of the AdEx Neuron.

A php Online Regression tool from http://polynomialregression.drque.net/online.php was used for curve fitting.

3.10 Discussion

- Resemblance to the Theory of Polychronous Groups

  The theory of Polychronization, namely Polychronous Groups illustrated in Figure 3.24, states that memories inside the brain could be represented as groups of many (hence the word poly in Greek) neurons that are activated through coordinated timing (hence the word chronous in Greek). It is based on large pool of Izhikevich neurons (i.e. a biologically plausible Neuron Model also invented by Izhikevich himself, in 2004) that are connected via random time delays. The connections between neurons are strengthened using hebbian learning (e.g. Spike Timing Dependent Plasticity). The main idea of

polychronization is that when an input spike (coming from the exterior or from a neuron inside the network) is delivered to a bunch of neurons then the time delays upon which it arrives to the neurons will activate a single neuronal group. Furthermore, Inter Spikes Intervals between spikes of a multiple spikes input pattern would have a great influence on the activity of the neurons since the latter are affected by time delays, which will create a concurrency of timings that will boost the activation of a Neuron (i.e. its action potential). So, the order of spikes timings is very important and essential in defining a single unique polychromous group. In our case, groups of RAF Neurons could also be considered as polychrounous because they are time locked to the firing pattern emitted by the controlled chaotic Neuron (i.e. The AdEx Neuron) and furthermore the order of their firing constitute their identity. In addition, in the theory of Polychronous Groups, the same neuron can belong to many groups depending on the input pattern being executed, which is also the case with RAF Neuronal Groups.

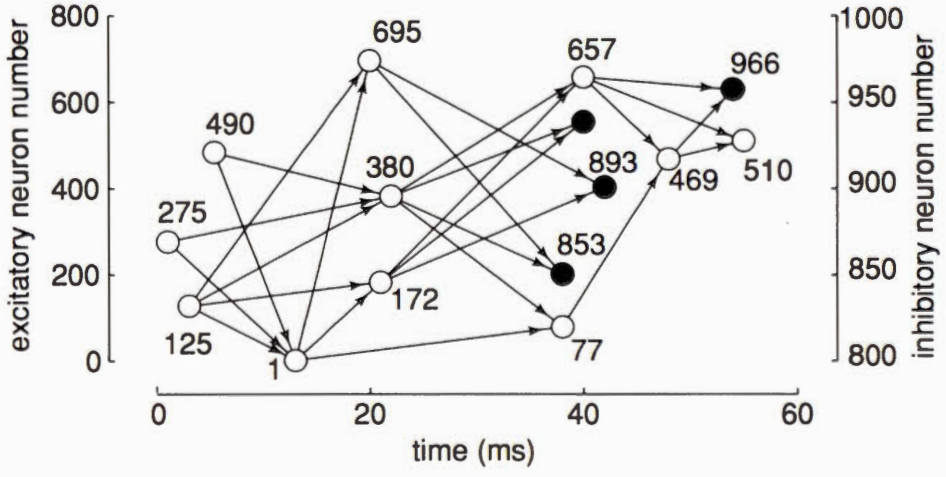


Fig. 3.24 Example of a Polychronous Group. When Neurons 125, 275 and 490 fire with an input pattern arriving at 0, 3 and 7 ms then their activity propagates to other neurons through specific time delays to finally generate a polychromous group which its last neuron is 510. The concurrency of spikes arrivals combined with the paths of conduction time delays are crucial in the generation of a polychromous group. Excerpt from (Izhikevich, 2006).

We have to note that polychronous groups depend on the current state of the network, which depends on the current input but is also affected by previous inputs experiences. This is not the case with RAF Neuronal Groups because the AdEx Neuron, which projects its output to RAF Neurons, is not affected by previous input experiences since it is reinitialized to its initial conditions each time an input is presented to it. However, this will not omit the possibility of creating another system architecture where traces of previous inputs remain alive for a short term in the dynamics of the AdEx Neuron, which could in return affect its current response, thus the activity of RAF Neurons becomes dependent on the current state of the system and not its initial state. This attempt would add a short-term memory property to the system where new input is treated in the context of previous inputs (Izhikevich, 2006) through the ongoing dynamics of the AdEx Neuron, which creates a new memory -represented by RAF Neurons - as an alteration of previous memories.

- Resemblance to the Theory of neural computation based on perturbations

  In 2002, Wolfgang Maass envisaged the theory of neural computation based on perturbations. The main idea behind this theory is that neural computation could be achieved by perturbing a dynamical system with an input. This perturbation is reflected by the response of the dynamical system upon that input such that similar inputs will induce similar responses from the system and different inputs will produce different responses from the system. The response here is depicted by the activity of the dynamical system and the latter could be a pool of recurrently connected spiking neurons (Maass et al., 2002). This is very similar to the core of our neural network architecture, but instead of defining the dynamical system as a pool of recurrently connected spiking

neurons, it is defined as a chaotic neuron (e.g. The AdEx Neuron), which is in fact a nonlinear dynamical system. Furthermore, we proved that our neural network has a separation property, which is the main property of neural computing based on perturbations (Maass et al., 2002). Note that we exploited the major principle of chaos theory, which is chaotic sensitivity upon initial conditions, in order to achieve a separation property for our system.

- Resemblance to the Theory of Selective Communication via Bursting

  It is generally believed that the role of bursting activity of a neuron is to boost synaptic transmission towards a target neuron. A different point of view was considered in (Izhikevich et al., 2003) which says that Bursting could act as a medium of communication between neurons. In fact, specific Inter Spike Intervals (ISI) inside a burst (i.e. inside bursting activity) of a neuron, that are transmitted to another neuron, were proven to be effective in initiating postsynaptic potentiation, while other ISIs were not (Izhikevich et al., 2003). We have to mention that the same principle of Resonance that we implemented in our study using Resonate and Fire (RAF) Neurons was applied in (Izhikevich et al., 2003). For instance, a burst from neuron A, having a burst inter spike frequency say X, would be resonant for a synapse between A and neuron B, and not resonant for a synapse between A and neuron C. While another burst from neuron A, having a different burst inter spike frequency say Y, would be resonant for the synapse between A and C but not resonant for the synapse between A and B (Izhikevich et al., 2003).

  The output activity of a chaotic spiking neuron, like an AdEx Neuron configured to run in chaotic mode, when controlled, resembles bursting activity (Figure 3.25). This is because a delayed spike pattern emitted by a chaotic spiking neuron when fed back to the neuron via a self-feedbak

connection becomes locked in a repetitive time window equal to the delay of the feedback connection. This repetitive time window would contain different ISIs depending on the time of occurrence of chaos control and on the input that was affecting the neuron dynamics, as we explained in the previous sections. Such diversity of ISIs achieved through chaos control that becomes repetitive in a bursting manner within time locked bursts, and which target a pool of RAF Neurons and selectively resonates in a group depending on their resonant inter spike frequencies (i.e. a RAF Neuron becomes active when it catches a specific ISI), supports the theory of (Izhikevich et al., 2003) of selective neural spike communication via bursts.

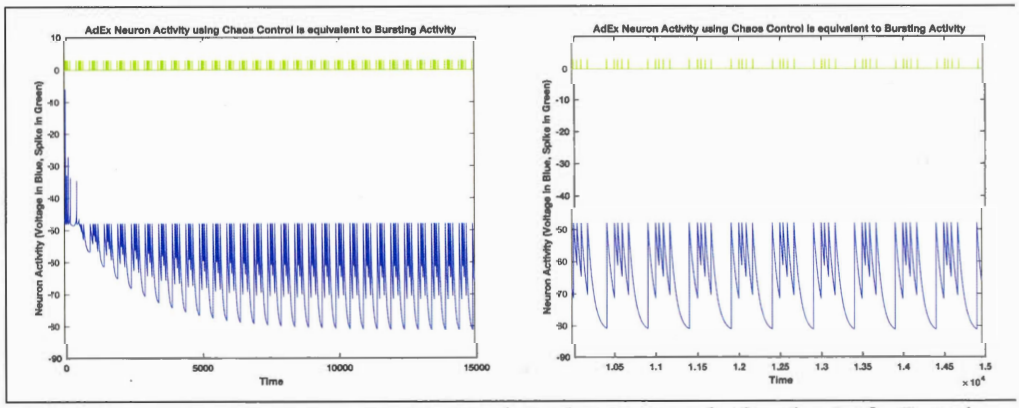


Fig. 3.25 Bursting Activity simulated using chaos control. On the Left, Bursting Activity of a chaotic AdEx Neuron that was subjected to chaos control occurring at Time Step 500 via a self-feedback connection that has time delay equal to 500. On the Right, zoomed version of the activity between Time Step = 10000 and Time Step = 15000.

3.11 Conclusion

In this chapter we provided experimental and theoretical proof of our thesis statement, which claimed that Unstable Periodic Orbits of a controlled chaotic spiking neuron can be considered as neuron states and can be represented as memories. In this regard, a memory is depicted as the activation of a neuronal group

(i.e. cells assembly that is periodically active for a specific external input). We showed that the activation of neuronal groups is consistent in discriminating external inputs fed to the neural network; by studying the separation property of the network. The idea of neuronal groups activated in coordinated time locks and representing memories is inspired from the work of Izhikevich in 2006. The idea of exploiting chaotic sensitivity on initial conditions is inspired from Crook in 2007. The approach that we followed in analyzing the separation property of the neural network is similar to the work of (Maass et al., 2002) and (Crook, 2007). We have to note that our neural network architecture has similarity with the Nonlinear Transient Computing Machine (NTCM) and got its inspiration from it (Crook, 2007). However, we use only one chaotic spiking neuron whereas in the NTCM two chaotic spiking neurons are used (Details were discussed in section 3.8). Furthermore, we used a biological model of chaotic spiking neuron in contrast to the non-biological model used in the NTCM. This could improve the biological plausibility of the theory of Nonlinear Transient Computation – NTC – (Crook, 2007). Also, an unfavorable performance of the separation property of the NTCM when executing the end part of an input spikes pattern was observed and was not resolved (Crook, 2007). In our experiments, we faced the same problem but we solved it. In fact, we showed that the separation property of our neural network was consistent for the whole portions of the input spikes pattern (Details in Section 3.8). Our solution offered favorable performance of the separation property compared to NTCM. Another point that we highlighted in section 3.10 and which makes our work different from the work of Izhikevich (2006) is that our neural network has no short-term memory, which means that its output is affected by current input only and not affected by past inputs, however the neural network of Izhikevich has short-term memory as the output of the network depends on the input presented to it and on the network dynamics which hold traces from past instances because they are always evolving. Finally, by demonstrating the separation property of our neural network based on the exploit of chaotic sensitivity upon initial conditions using neural computing in terms of a reservoir computing approach and by

showing evidence of the large number of neuronal groups that can coexist in the network due to the fact that a resonant neuron can belong to many neuronal groups, thus representing memory as a group of neurons, then this puts our work in the category of important theories on modeling memory using neuronal groups (e.g. Polychronous groups – (Izhikevich, 2006)) and neural computing using reservoir models (e.g. Real time computing based on perturbations – (Maass et al., 2002)).

CHAPTER 4

DATA CLASSIFICATION AND LEARNING BASED ON FIRING RATES OF CHAOTIC SPIKING NEURONS AND DIFFERENTIAL EVOLUTION

4.1 Introduction

In order to increase the firing rate of a regular spiking neuron like the Leaky Integrate and Fire (LIF) Neuron, we increase the input current it receives. But, what can we do if we want to increase the firing rate of a chaotic spiking neuron? Increasing its input current in order to increase its firing rate will not work properly for many reasons. First, a chaotic spiking neuron is very sensitive to small changes of its variables settings and this is due to its chaotic nature. So changing the input current will lead the neuron model to either lose or narrow down its chaotic spiking (Naud et al, 2008), thus this is not a good solution. Furthermore, there are some types of chaotic spiking neurons that don't even give the option of an input current variable like the NDS Neuron (Crook et al., 2005) where the input is induced in a discrete manner as a form of spikes (i.e. not continuous). One solution is to create recurrent neural network architecture composed of chaotic spiking neurons that can synchronize their chaotic activity and then increase the number of neurons in the network. By increasing the

number of neurons, the firing rate of chaotic spiking would increase and the recurrent neural network composed of chaotic spiking neurons would be then considered as a single entity (i.e. a chaotic spiking neuron) firing chaotic spikes.

We tested this hypothesis and it worked! As we will see in this chapter, we will use this feature of firing rate of synchronous chaotic spiking to achieve data classification and machine learning. In this regard, an engineering application is built and then tested on two data sets retrieved from the well-known, and globally accredited, 'machine learning database' of the University of California, Irvine (UCI Machine Learning Repository, [http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science). The two datasets are: IRIS Data Set and Breast Cancer Dataset. The details of these two datasets and the reason behind their choice (i.e. being both nonlinearly separable, one providing few attributes vs. few number of samples and the other many attributes vs. a large number of samples) are explained in the experimental results in Section 4.5 of this chapter. The performance of our system is compared to other machine learning algorithms (e.g. Support Vector Machines (SVM) and Multilayer Perceptrons (MLP)).

In section 4.2, next, we illustrate a neural network architecture composed of chaotic spiking neurons (e.g. AdEx Neurons in chaotic mode (Naud et al., 2008)) that are recurrently connected one to the other, and we study their synchronous activity. In section 4.3, we analyze the firing rate of the neural synchronous activity for different number of neurons that constitute the network. In fact, we show that the number of neurons of the network and the firing rate; of their neural activity, are proportional. We will deduce a Frequency Response Curve (FRC), using curve fitting, that will relate the firing rate and the number of neurons in a quadratic function. In section 4.4, we will use the average firing rate and the standard deviation of firing rate to create a cost function upon which data samples can be classified. This cost function will be used in an evolutionary algorithm called Differential Evolution – DE – (Price et al.,

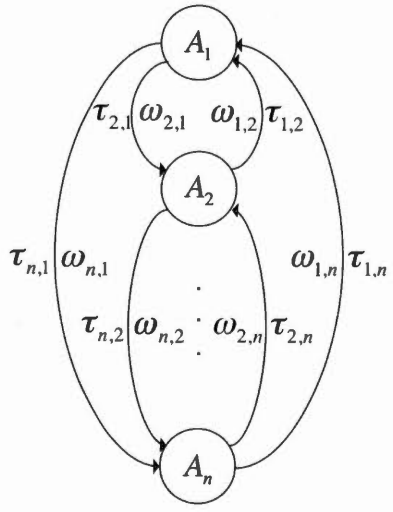
2005) so we can detect the number of neurons required for a specific data set, in order to perform machine learning upon that dataset... In section 4.5, we show experimental results of data training, testing and classification upon two datasets and compare the accuracy of the system with other machine learning methods. Section 4.6 is a discussion and section 4.7 concludes the chapter.

4.2 Neural network architecture and its chaotic spikes synchronization

Since a recurrent neural network composed of chaotic spiking neurons can synchronize its activity as we have shown in chapter 2, then we choose this architecture in order to analyse the firing rate relatively to any number of chaotic spiking neurons that compose the network. We show through experimentation that the firing rate can be managed (increased or decreased) depending on the number of neurons inside the network. Furthermore, we fix the number of neurons inside the network and we study its behaviour by varying the time that governs the connections between the neurons inside the network. We find out that both the time delay and the number of neurons are key factors in expanding the memory capacity of the network, in terms of firing rate, as we will discuss later on. For now, we justify our choice of a recurrent neural network architecture composed of chaotic spiking neurons as in chapter 2 because it provides the opportunity of implementing static data classification as we will demonstrate in this chapter, in contrast to the architecture presented in chapter 3 which targets time series data classification.

The neural network architecture that will be used in this chapter is composed of n (a variant number) chaotic spiking neurons (e.g. AdEx Neurons configured to run in chaotic mode) that are recurrently connected one to another as in chapter 2. The weights between the connections have no need to be managed through synaptic plasticity since the objective in this experiment is to induce causal interactions between the neurons that can be translated by just assigning positive real values to the

weights of the connections between the neurons. We use the AdEx Neuron model discussed in the previous chapters, because it is a biological neuron model that can simulate a wide range of physiological neuron behaviors (including chaotic spiking) and has fast processing speed when executed on a computer. Each connection; between any two neurons, in the network has a fixed time delay and a fixed weight. The network architecture is illustrated in Figure 4.1, next.



Recurrent Network of
n AdEx Neurons

Fig. 4.1 Neural Network Architecture: *n* AdEx Neurons are recurrently connected. Every connection has a weight "ω" and a time delay "τ"

As we can see in Figure 4.1, we have a neural network of *n* AdEx Neurons (e.g. A_1 to A_n) where the subscript *"i"* is the index of an AdEx Neuron *"A"*. Each neuron has *n-1* connections because there are no self-feedback connections. A connection from neuron A_j to neuron A_i has a time delay *"$\tau_{i,j}$"* and a weight $\omega_{i,j}$.

We rewrite the AdEx Neuron equations (Naud et al., 2008) in their Euler form:

$$V_i(t) = V_i(t-1) + \frac{dt\left[-gL(V_i(t-1) - E_L) + gL\,\Delta_T \exp\left(\frac{V_i(t-1) - V_T}{\Delta_T}\right) + I_c - \psi\right]}{C}$$

$$\psi_i(t) = \psi_i(t-1) + dt[a(V(t-1) - E_L) - \psi]/\mathrm{K}_w$$

Where $V_i(t)$ is the neuron's voltage at time t and $\psi_i(t)$ is its adaptation variable. The subscript *"i"* indicates the neuron's index when it is embedded in a neural network of AdEx Neurons.

dt is the Euler time step which is set to *0.1* for good precision.
The parameters from (Naud et al., 2008) are given in table 4.1, next:

Table 4.1 Parameters configurations of the AdEx Neuron in Chaotic mode (Naud et al., 2008)

Mode \ Parameters	C	gL	E_L	V_t	Δ_T	a	K_w	b	V_r	Ic	θ
Chaotic	100	12	-60	-50	2	-11	130	30	-48	160	0

The neuron initial conditions are $V_i(0) = V_{ri}$ and $\psi_i(0) = 0$;

Where,

$$V_{ri} = V_r - R_i$$

R_i is an infinitesimal random decimal between 0 and 0.01. We need the neurons voltages to have slight variation in their initial values so their output will be different at the long run due to chaotic sensitivity on initial conditions. Remember that the system is chaotic thus it is sensitive to its initial conditions, so slight variation in the initial conditions will lead to different long-term behavior.

When the neuron's voltage bypasses its threshold (i.e. $V_i >= \theta$), then the neuron emits a spike which is represented as γ_i:

$$\gamma_i(t) = \begin{cases} 1, V_i(t) \geq \theta \\ 0, V_i(t) < \theta \end{cases}$$

Also, when the neuron's voltage bypasses its threshold then it is reset to its reset value and its adaptation variable is set to its current value plus the adaptation reset parameter:

$$If\ V_i(t) \geq \theta\ then\ V_i(t) = V_{ri}\ and\ \psi_i(t) = \psi_i(t) + b$$

The neuron receives time-delayed spikes from other neurons through its incoming connections, which are scaled by each connection's weight and summed up as a total input I given by:

$$I_i(t) = \sum_{j=1\ \&\ j<>i}^{n} w_{i,j}\gamma_j(t - \tau_{i,j})$$

We note that the weights of all the connections are set to 1. To insure that a single input spike to the neuron is enough to drive the neuron's voltage to its voltage threshold, then we update the voltage of the neuron according to:

$$V_i(t) = V_i(t) + \big(\theta - V_i(t)\big)H(I_i(t) - 1)$$

Where H(x) is the Heaviside step function defined as:

$$H(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

This means that if $I_i(t)$ of neuron i is equal to at least 1 (i.e. a single incoming spike is available in the neuron's input) then $H(I_i(t) - 1)$ is equal to 1 and the neuron

voltage that is equal to $V_i(t) + \big(\theta - V_i(t)\big)H(I_i(t) - 1)$ becomes $V_i(t) + \big(\theta - V_i(t)\big) * 1$ which results to θ. Otherwise, when no incoming spike is available in the neuron's input (i.e. $I_i(t) = 0$, thus $H(I_i(t) - 1) = 0$), then $V_i(t)$ remains the same because $V_i(t) + \big(\theta - V_i(t)\big) * 0$ results to $V_i(t)$. In this mode of operation, a single spike, that is exciting the neuron, is enough to drive the neuron's voltage to its threshold.

In Figure 4.2, we show the behavior of the recurrent neural network illustrated in Figure 4.1 using AdEx Neurons that are configured to run in chaotic mode (Table 4.1), following the equations that describe their dynamics and evolution through time as described above (i.e. using random initial conditions and input via time delayed connections). In this example, we choose the number of neurons in the network to be equal to 50 (i.e. n = 50), the time delay of the connection between any two neurons to be equal to 1000. The network runs for 80000 time steps, which is the simulation time of this experiment. In the first 20000 time steps, the weight of every connection is set to 0, thus the neurons run in isolation in order to evolve their chaotic dynamics independently. At time step 20001, all the weights are set to 1; this means the neurons are connected to each other and start synchronizing their dynamics. As we will see in the graph of Figure 4.2, when the neurons are in isolation (e.g. Time Step $\leq$ 20000), a neuron's activity inside the network is different than any activity of other neuron. However, when they are connected at time step 20001 then they start stabilizing their dynamics, which results in a synchronous neural activity all over the neural network.

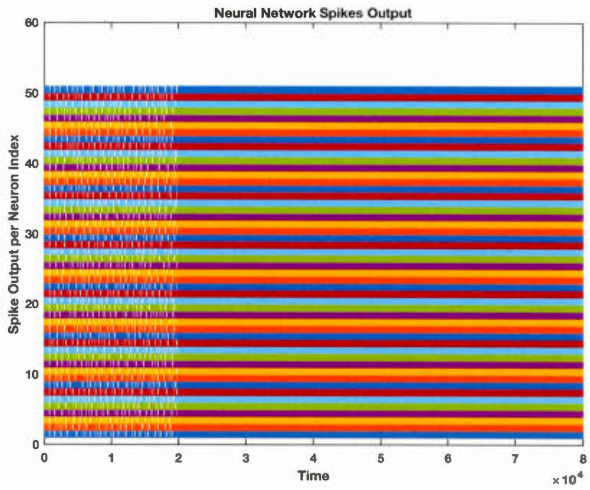


Fig. 4.2 Neural Network Spike Output: Neurons run in isolation for 20000 time steps and they are connected afterwards. When the neurons are connected they synchronize their activity.

To show that every neuron inside the neural network is synchronizing its spiking activity over a period equal to *"τ"*; which is the time delay of any connection between two neurons in the network and is the same for all the connections, then we calculate the difference between the absolute value of a neuron's spike at time *"t"* and its delayed spike at *"t - τ"*, where *"τ"* indicates the time delay of any connection in the neural network; which is in fact the same for all the connections as we just mentioned. If the difference between a neuron's spike absolute value at time *"t"* and its spike absolute value at time *"t - τ"* is equal to 0 al over *"τ"*, then this means that the neuron is synchronizing its spikes activity over the period *"τ"*. The plot of these differences is given in Figure 4.3, next.

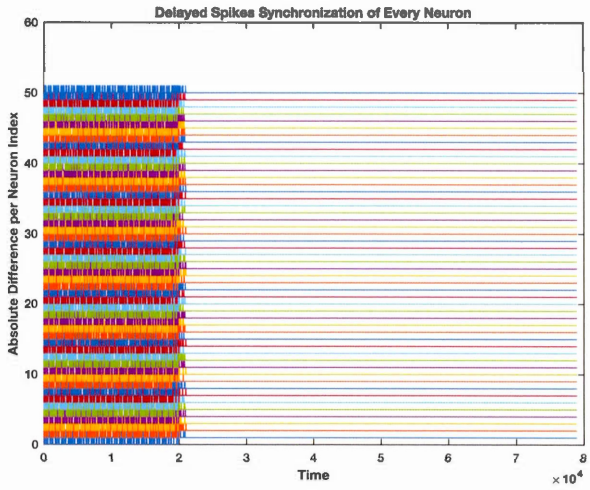


Fig. 4.3 Periodic Spikes Synchronization of every neuron inside the Neural Network: Calculating the difference between the absolute value of a neuron's spike output at time *"t"* and the absolute value of its spike output at time *"t - τ"* shows that every neuron is synchronizing its activity in a time period equal to *"τ"* because the difference settles to 0 at the long run (e.g. after time step 20000).

In figure 4.3, we showed that every neuron is synchronizing its output spikes activity to itself over a period that is equal to the time delay of the connections inside the network. But, to show that the neurons are synchronizing their system dynamics; their voltage *"V"*, Adaptation variable *"ψ"* and Spike output *"γ"*, between each other, then we will resort to the standard deviation of the data of each of these variables at time *"t"*. In fact, the standard deviation, at time *"t"*, of a set of data values that a variable *"x"* takes at time *"t"*, illustrates the dispersion of these data values, at time *"t"*, to the mean value of *"x"* at time *"t"*. Thus, if the data is highly dispersed to the mean then the standard deviation is maximal and if the data is closely dispersed to the mean then the standard deviation is minimal. In our case, if all the neurons are truly synchronizing their activity between each other at time *"t"* then the standard deviation of their dynamic variables should be minimal and the better case equal to *0*. The standard deviation of data variables *"γ"*, *"V"* and *"ψ"* of all the neurons is shown in figures 4.4, 4.5 and 4.6, respectively, next.

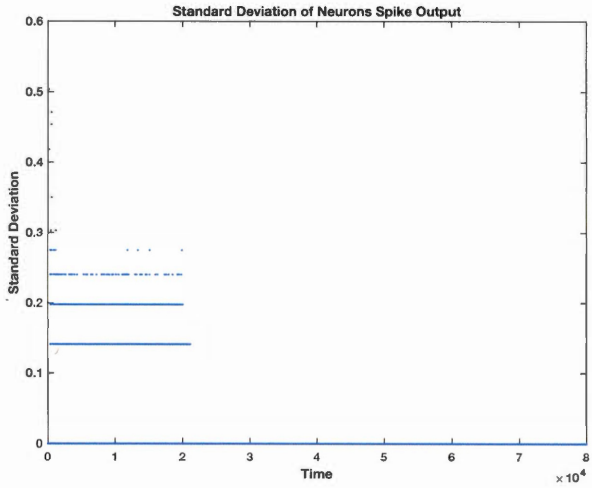


Fig. 4.4 Proof of Synchronization of all Neurons Output Spikes using Standard Deviation

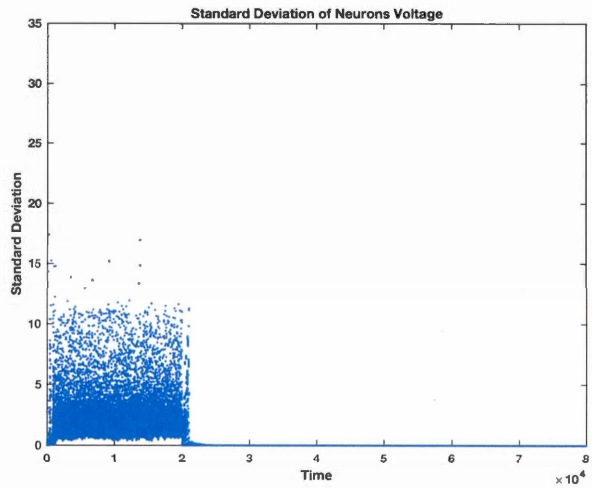


Fig. 4.5 Proof of Synchronization of all Neurons Voltages using Standard Deviation

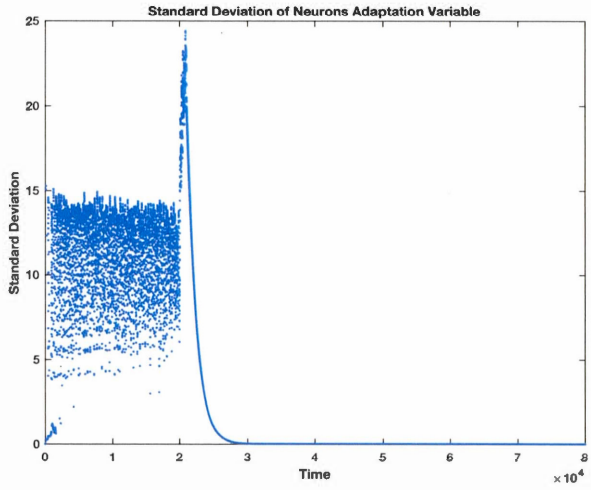


Fig. 4.6 Proof of Synchronization of all Neurons Adaptation Variables using Standard Deviation

As we can see in Figures 4.4, 4.5 and 4.6 the standard deviation between the dynamic variables of every neuron in the network converges to 0, which proves that all the neurons are synchronizing their dynamics.

Next, we will show that the distribution of the spike events, during synchronization, demonstrates an exponential distribution: At time step 20001 (i.e. when the neurons are connected and start synchronizing), we pick a neuron from the network and we record its Inter Spikes Intervals (ISIs), then we sort them in ISI bins from minimum ISI to maximum ISI, we count the ISIs in each bin and we plot the count in a histogram as seen in Figure 4.7.

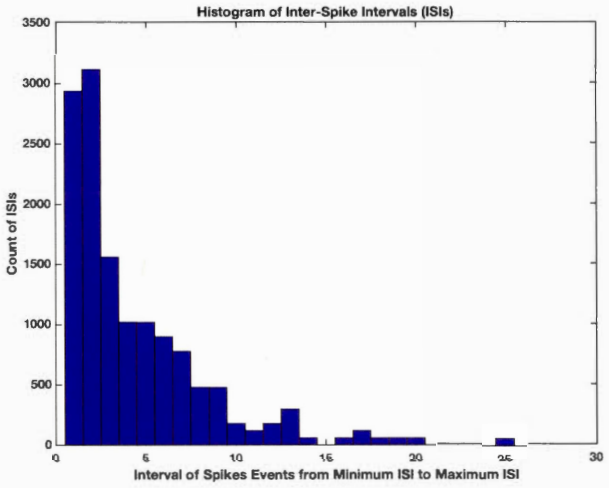


Fig. 4.7 Histogram of Inter-Spike Intervals shows an Exponential Distribution

The exponential distribution of spikes events through the course of synchronization confirms that these spikes are occurring continuously and independently at a constant average firing rate. Thus, the synchronous neuron's spikes output induces a Poissonian process in its course, which means the spikes are not regular.

In this section, we demonstrated the synchronization in a recurrent network of 50 AdEx Neurons running in chaotic mode. In the next section, we show that the firing rate of such networks increases with the number of neurons that constitute the network.

4.3 Relationship between firing rates and number of neurons

Now, we will show that the synchronous firing of AdEx Neurons that are configured to run in chaotic mode and which are recurrently connected one to another (Figure 4.1), has a firing rate that can be manipulated by the choice of the number of neurons which constitute the network. To prove this, we conduct 9 experiments by varying the number of neurons in each experiment and we calculate the number of spikes per

time period equal to the time delay of the connections inside the network. Remember that all connections have the same time delay as we explained in the previous section. The results are presented in Table 4.2, which contains three columns, the first is the experiment number, the second is the number of neurons that constitute the neural network of an experiment and the third column shows the resulting firing rate.

Table 4.2 Results of 9 Experiments by varying the number of neurons of the neural network

Experiment Number	Number of Neurons	Firing Rate
1	10	50
2	20	93
3	30	148
4	40	188
5	50	234
6	75	321
7	100	393
8	150	528
9	200	644

Next, we plot the results in Figure 4.8 and we fit them to a quadratic curve using polynomial regression. The coefficient of Determination is equal to 0.9987 and the quadratic function is $f(x) = 10.716008378670088 + 4.632685586027018x - 0.007467944346502461x^2$ where x is the number of neurons. A php Online Regression tool from http://polynomialregression.drque.net/online.php was used for curve fitting.

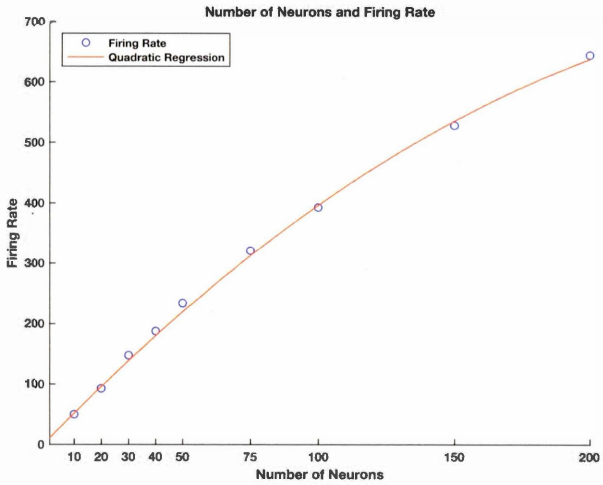


Fig. 4.8 Firing Rate versus Number of Neurons with Quadratic regression. This graph tells us that the firing rate (Blue circles) is proportional to the number of neurons of the neural network and increases in a quadratic fashion (Red plot) with the increase of the number of neurons.

As we can see in Figure 4.8 the firing rate increases almost linearly until a network of 75 neurons and it shows a quadratic increase afterwards.

In the 9 experiments presented above, we followed the same settings as the previous section: The time delay of every connection inside the network was set to 1000, the neurons runs in isolation for 20000 time steps, connect to each other at time step 20001 and the whole simulation time of an experiment endures for 80000 times steps. So, to prove that the results are neither an artifact of the choice of the connection time delay (e.g. 1000) nor dependent on the value of a specific isolation time (e.g. 20000), and to find the best choice of these settings, we conduct another set of experiments where we randomly vary the Isolation Time (Figure 4.9) and the Time Delay (Figure 4.10).

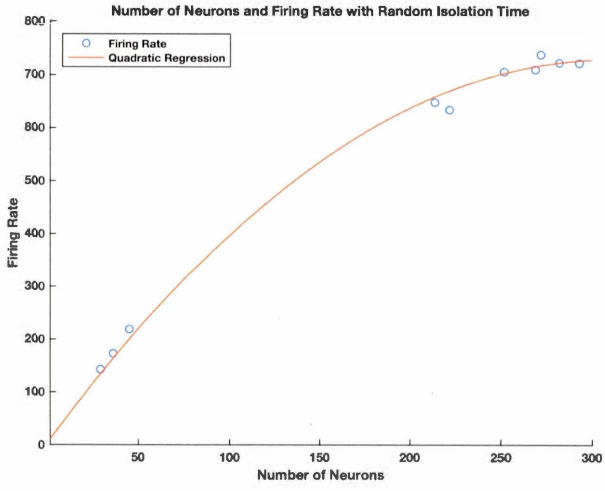


Fig. 4.9 Firing Rate versus Number of Neurons with Quadratic regression using Random Isolation Time. This graph tells us that the firing rate (Blue circles) is proportional to the number of neurons of the neural network and increases in a quadratic fashion (Red plot) with the increase of the number of neurons independently of the Isolation Time that could be set on the neurons while they are evolving their dynamics.

Since the Isolation Time has no effect on the Firing Rate of the Neural Network, then we fix the Isolation time to a value of 20000 and we vary the Time Delay of the Connections. The results are shown in Figure 4.10, next:

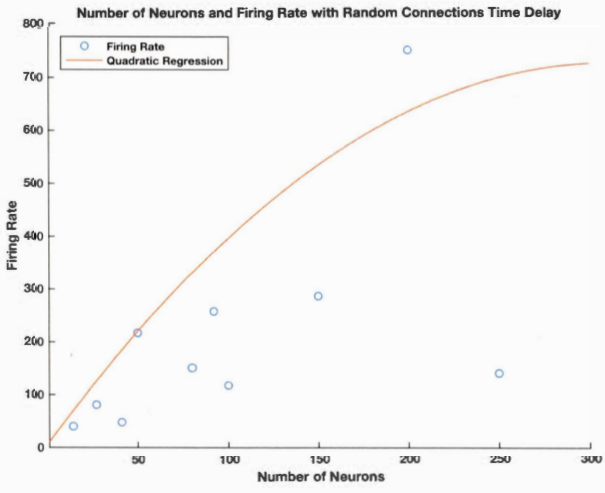


Fig. 4.10 Firing Rate versus Number of Neurons using Random Connection's Time Delay. Original Quadratic regression, from previous experiment, is plotted, too. This graph tells us that the Time Delay has an effect on the firing rate (Blue circles) which is shown as not proportional to the number of neurons of the neural network and does not increase in a quadratic fashion (Red plot) as it should do; with the increase of the number of neurons.

The graph in Figure 4.10 tells us that the Time Delay has an effect on the Firing Rate. In fact, if we increase the Time Delay, the Firing Rate should also increase which is very probable because the Firing Rate is the number of spikes per Time Delay. So, if the Time Delay is increased then the number of spikes that it contains should also increase. We put this assumption under test and we conduct 100 experiments such that in each experiment the Time Delay takes a random value between 200 and 2000. In these experiments, the number of Neurons in the network is set to 30 Neurons. The results are shown in Figure 4.11, next:

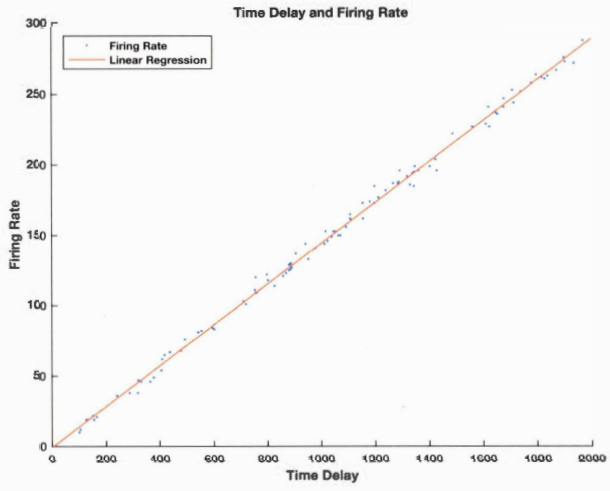


Fig. 4.11 Firing Rate versus Time Delay. Linear Fit is plotted in red using linear regression. The Firing Rate per Time Delay is dotted in blue color. This graph tells us that the firing rate is proportional to the Time Delay that is set on the connections in the neural network and it increases linearly (Red plot). In other words, it does not saturate for large Time Delays.

The curve fitting shown in red color in Figure 4.11 is derived from a linear regression with coefficient of determination equal to 0.9967 which means it's a very good fit. The Linear Function is: f(x) = -0.5872843296282478 + 0.14501031760169575x where x is the Time Delay and f(x) the Firing Rate.

After proving that the Firing Rate is linearly related to the Time Delay of the connections in the neural network, we conduct 100 experiments such that in each experiment, the Number of Neurons takes a random value between 3 and 50 Neurons, and the Time Delay of each connection takes a random value between 100 and 2000. We plot the Firing Rate achieved in each experiment as a colored circle that varies in diameter depending on the value of the Firing Rate. The Firing Rate is each experiment is plotted per Time Delay and Number of Neurons of the experiment. This is shown in Figure 4.12.

Fig. 4.12 Firing Rate versus Time Delay and Number of Neurons: Firing Rate of each experiment which has a specific Time Delay and a specific Number Of Neurons is plotted as a colored circle which varies its diameter and color depending on the value of the Firing Rate. As we can see for large neural network (i.e. large number of neurons) and large Time Delay of the neural network connections we have large Firing Rate and vice versa.

So, to increase the Firing Rate of a Recurrent Neural Network as described in Section 4.2 and illustrated in Figure 4.1, we can either fix the Time Delay of the Connections inside the network and increase the Number of Neurons, or we can fix the Number of Neurons of the network and increase the Time Delay of their Connections.

We conclude this section with the following discussion:

The time delay is considered as the dimension of the input vector (its size in terms of length of bits). Increasing the time delay will allow an increase in the dispersion of the elements inside the input vector. For instance, an increase in the time delay will allow a high Standard Deviation of the input. On the other side, increasing the cardinality of the network (i.e. the number of neurons that compose the network) will allow condensation of spikes in response to a neural network of a fixed length of an input vector. This could be used to generate a frequency curve that could relate the firing rate of neurons in terms of the number of spikes per time delay (required when

we have a fixed length of input) to the number of neurons that compose the network.

4.4 Data classification based on firing rates of chaotic spiking neurons

In the previous section (Section 4.3), we concluded that the Firing Rate of the Recurrent Neural Network described in Section 4.2 and illustrated in Figure 4.1 can be either proportional to the Number of Neurons inside the network when we fix the Time Delay of the neural network connections, or proportional to the Time Delay of the neural network connections when we fix the Number of Neurons inside the network. In this section, and afterwards, we always choose a varied number of neurons and a fixed connections Time Delay scenario. Thus, the number of neurons is the variable upon which the firing rate is determined. In other words, the dependent variable of the firing rate is the number of neurons. Note that choosing a varying Time Delay scenario is also feasible but due to lack of time it was not additionally considered herein.

Another point that we have to mention: The methodology followed in this section is inspired from the work of Vazquez (2010), and some equations used here were originally proposed by Valadez-Godinez et al., in 2017. However, our work is different than the work of (Vazquez, 2010, Valadez-Godinez et al., 2017), in that their approach is based on the firing rate of a single spiking neuron that fires in regular mode. The originality of our approach is to use chaotic spiking neurons and is supported by the relationship between the firing rate and the number of neurons inside a recurrent chaotic spiking neural network, as we have seen in the previous section. Now, we will benefit from these findings to implement machine learning (i.e. data pattern classification).

Our goal is to use the Firing Rate of a recurrent network of chaotic spiking neurons (described in the previous sections, section 4.2 and 4.3 of this chapter) to achieve data

pattern classification. To do so, imagine we have a dataset D of dimension d that contains m real patterns where a pattern is denoted as x (i.e. $x^i \in \mathbb{R}^d: 1 \leq i \leq m$), the patterns are divided into C classes where each pattern belongs to a class c (i.e. $1 \leq c \leq C$). Thus, D can be written as:

$$D = \{x^i, c : x^i \in \mathbb{R}^d \,\&\, 1 \leq i \leq m \,\&\, 1 \leq c \leq C\}$$

Then, an input pattern is transformed to a neural network's cardinality. We will see how this transformation is achieved, later on. Also, remember that the cardinality of a set is the number of elements inside the set; therefore neural network cardinality is the number of neurons that constitute the neural network. So, after an input pattern is transformed to a cardinality (say n), then a recurrent neural network, as illustrated in Figure 4.1 of this chapter, is built and simulated using n AdEx Neurons configured to run in chaotic mode as described in section 4.2. Then the Firing Rate (FR) is recorded.

Since the Firing Rate of the neural network is proportional to its cardinality (i.e. *FR* increases when n increases), we expect that input patterns that belong to the same class will have very close Firing Rates and input patterns that belong to different classes will have different Firing Rates (i.e. not very close). First, if Firing Rates are very close then they should have small Standard Deviation. We will use the information of Standard Deviation, later on. Second, the Average Firing Rate – AFR – of input patterns that belong to one class, say C1, should be different than the Average Firing Rate of input patterns that belong to another class, say C2. Furthermore, the Firing Rate of an input pattern should be close to the Average Firing Rate of the class it belongs to and distant from the Average Firing Rate of other classes. This means that the class of an input pattern can be determined as:

$$Class(x) = argmin_{c=1}^{C}(|AFR_c - FR(x)|)$$

Where, *FR(x)* is the Firing Rate of an input pattern x, AFR_c is the Average Firing Rate of input patterns that belong to class c and C is the total number of Classes.

The Average Firing Rate of a class c is the sum of the Firing Rates generated by input samples that belong to this class divided by the total number of samples in that class. It is written as:

$$AFR_c = \frac{\sum_{i \in c} FR(i)}{\overline{\overline{c}}}$$

Where the double bars on c denote its number of elements.

To transform an input pattern to neural network cardinality n, we write n as a dot product as follows:

$$n = x \cdot p$$

Where, $x \in \mathbb{R}^d$ is the input pattern that can consist of d real attributes, and $p \in \mathbb{R}^d$ is a vector of free parameters with dimension d, too.

The free parameters of vector p are optimized using the Differential Evolution Algorithm – DEA – (Price et al., 2005). DEA is an optimization algorithm similar to genetic algorithms (Check ANNEXE IV for a detailed description of DEA). It is very fast and rarely gets stuck in a local optimum. Same as any evolutionary algorithm (e.g. Genetic algorithm), DEA requires a fitness function to evaluate candidate solutions in a population. Since, as we previously mentioned, input patterns that generate very close firing rates should belong to the same class and their firing rates

would have a small standard deviation, and since every class has an Average Firing Rate – AFR – that should be distant to the Average Firing Rate of any other class, then, we use the following fitness function that was originally proposed by Vazquez, in 2010:

$$fitness = \frac{1}{S} + \sum_{c=1}^{C} SDFR_c$$

Where, $SDFR_c$ is the standard deviation of Firing Rates of input patterns that belong to class c, and S is the summation of Euclidean Distances between the Average Firing Rates of all the classes, given as:

$$S = \sum_{i,j \in C : j > i} |AFR_i - AFR_j|$$

Note that the Euclidean Distance is written as an Absolute Difference because it is one-dimensional, it is also know as Manhattan Distance.

Interpreting the fitness function, we notice that small standard deviations combined with large AFR distances lead to minimum fitness, which is the aim of the an Evolutionary Algorithm – EA – since its objective is to minimize the fitness function in order to find the best candidate solution.

4.5 Experimental results

Data classification can be implemented using Firing Rates of chaotic spiking neurons combined with DEA. To demonstrate this implementation, we choose two datasets from the Machine Learning Database (UCI Machine Learning Repository,

[http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science). The Datasets names are: IRIS and Breast Cancer. The details of these datasets are the following: The IRIS dataset has 4 attributes, contains 150 data samples that are classified in 3 classes; which are nonlinearly separable. The Breast Cancer dataset has 30 attributes, contains 569 data samples that are classified in 2 classes and which are nonlinearly separable, too. The reason behind the choice of these two datasets is that one class has few attributes and few number of samples while the other dataset has two many attributes and a large number of samples.

The settings of the neural network that is used for both datasets are the following (Table 4.3):

Table 4.3 Neural Network Settings

Neuron Model	Adex Neuron
Mode	Chaotic
Network Architecture	Recurrent
Number of Neurons	Integer Linear Variable
Connection Time Delay	1000
Connection Weight	1
Initial Voltage	Uniform Distribution (between -48 and -48.01)
Voltage Reset	Initial Voltage
Voltage Threshold	0
Initial Adaptation	0

We have to note that the settings provided in table 4.3 are applied on the neural network architecture illustrated in section 4.2 of this chapter (i.e. Figure 4.1) and were conceived in section 4.3. In fact, this neural network architecture with these settings

fulfills the analysis provided in section 4.3 (i.e. the relationship between the number of neurons and the firing rate shown in Figure 4.8). Furthermore, the Firing Rate is dependent on the number of neurons (i.e. the cardinality of the network) and could be calculated using a quadratic function as we proved in the previous section (Section 4.3). The quadratic function of the Firing Rate (FR) that depends on a recurrent network of cardinality n, is of the form:

$$FR = a0 + a1 * n + a2 * n^2$$

Such that *a0*, *a1* and *a2* are coefficients derived from curve fitting of many experiments using polynomial (e.g. quadratic) regression. For instance, a0 =10.716008378670088, a1 = 4.632685586027018 and a2 = - 0.007467944346502461 were derived by varying the cardinality from 10 to 200 for a neural network that had a Time Delay of its connections equal to 1000 (Details in Section 4.3 of this chapter).

In other words, we can avoid running the neural network for each input sample x by transforming the input sample to n and applying the quadratic function on n. This transformation, explained in the previous section (Section 4.3), is done using the dot product:

$$n = x \cdot p$$

Where, $x \in \mathbb{R}^d$ is the input pattern that can consist of d real attributes, and $p \in \mathbb{R}^d$ is a vector of d parameters determined by the DEA.

The settings of the DEA are experimentally tuned depending on the Dataset in hand. As for the IRIS dataset, the DEA settings are given in table 4.4, next:

Table 4.4 DEA Settings for the IRIS Dataset

Populations	60
Cross Over Probability	0.5
Mutation	0.5
Number of Iterations	30
Decision Variables	Between 1 and 10

We use 10 Fold Cross Validation on the IRIS dataset to test the accuracy of our neural network architecture that embeds Differential Evolution (DE); which detects the required number of neurons (i.e. neural network cardinality) in order to find the suitable Average Firing Rate (AFR) upon which data samples are classified. By running the DE algorithm provided in ANNEXE IV, using the settings of table 4.3 and 4.4, we show the AFR and the Average Number of Neurons (ANN) in table 4.5, next:

Table 4.5 Average Firing Rates (AFR) and Average Number of Neurons (ANN)

Dataset Class	AFR	ANN
IRIS Class 1	56.928	10.142
IRIS Class 2	75.3887	14.292
IRIS Class 3	88.0838	17.18

Using the AFR, the algorithm was able to classify the testing data accordingly. This is shown in Figures 4.13 and 4.14. In fact, we would like to illustrate the class distribution of the testing data samples that were used in the experiments. This is shown in figure 4.13, next:

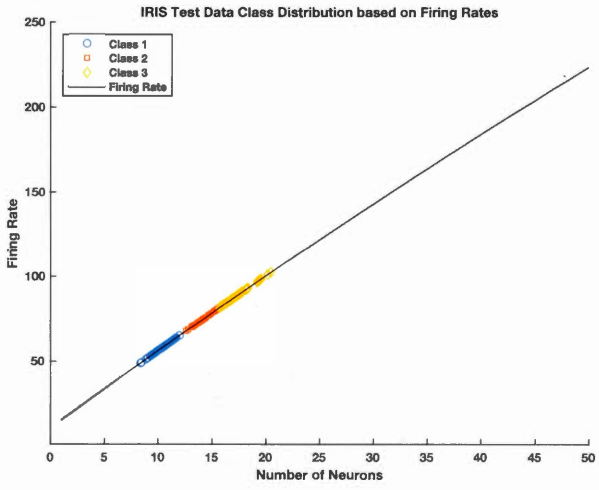


Fig. 4.13 IRIS Testing data samples class distribution. Each class of a data sample is retrieved using the absolute difference between the Average Firing Rate (AFR) of the class that was learned using training, and the firing rate resulting from the dot product of the data sample attributes and the optimized coefficients of the DEFR algorithm.

The distribution of the classes of testing samples fits the quadratic curve of the firing rate generated by our recurrent neural network architecture of varying cardinality; which was conceived in Section 4.3 of this chapter and illustrated in Figure 4.8. Remember, the cardinality of the neural network is a result of the dot product between the Differential Evolution using Firing Rates – DEFR – coefficients and the attributes of a data sample presented upon testing. It is remarkable that DEFR separates the classes on a 2D Curve as seen in Figure 4.14.

In figure 4.14, we plot the 10 fold cross validation test samples and their classes.

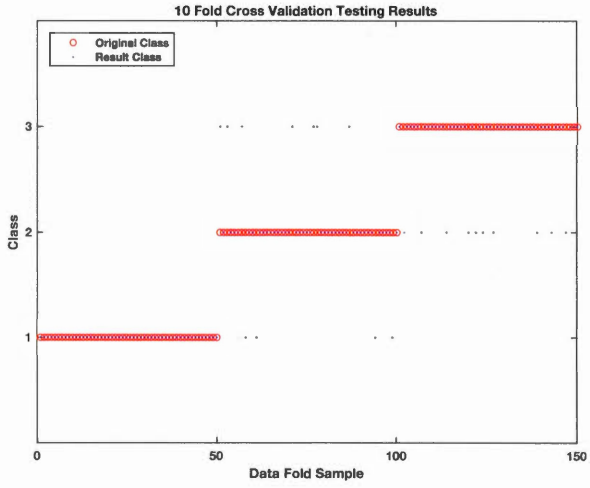


Fig. 4.14 10 Fold Cross Validation performed on the IRIS Dataset. The original class of a testing sample is shown in red circles. The result class of a testing sample when the latter is recognized using DEFR is shown in blue dots. When a data sample, of a class, is correctly recognized then a blue dot is inside a red circle and when a data sample, of a class, is not correctly recognized then a blue dot is outside a red circle.

As we can see in Figure 4.14, original and resulting classes of data samples from the IRIS Dataset 10 fold cross validation experiments are well matched on a large number of testing samples. For instance, all data samples from class 1 are correctly classified, however, intermingling between class 2 and 3 is noticed. The overall classification accuracy of the 10 fold cross validation experiments is presented in table 4.8 at the end of this section.

Next, we use the Breast Cancer Dataset for further experimentation. In this regard, its DEA settings are given in table 4.6, next:

Table 4.6 DEA Settings for the Breast Cancer Dataset

Populations	60
Cross Over Probability	1
Mutation	0.05
Number of Iterations	30
Decision Variables	Between 0.05 and 0.055

Here also, we use 10 Fold Cross Validation on the Breast Cancer dataset to test the accuracy of our neural network architecture that embeds differential evolution; which detects the required number of neurons (i.e. neural network cardinality) in order to find the suitable Average Firing Rate upon which data samples are classified. By running the DEA provided in ANNEXE IV, using the settings of table 4.6, we show the AFR and the Average Number of Neurons (ANN) in table 4.7, next:

Table 4.7 Average Firing Rates (AFR) and Average Number of Neurons (ANN)

Dataset Class	AFR	ANN
Breast Cancer Class 1	499.2752	142.8979
Breast Cancer Class 2	278.4103	65.0426

Using the AFR, the algorithm was able to classify the testing data accordingly. This is shown in Figures 4.15 and 4.16. In fact, we would like to illustrate the class distribution of the testing data samples that were used in the experiments. This is shown in figure 4.15, next:

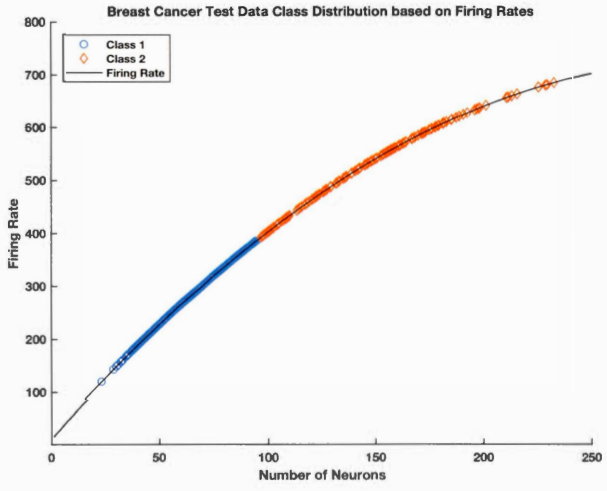


Fig. 4.15 Breast Cancer Testing data samples class distribution. Each class of a data sample is retrieved using the absolute difference between the Average Firing Rate (AFR) of the class that was learned using training, and the firing rate resulting from the dot product of the data sample attributes and the optimized coefficients of the DEFR algorithm.

The distribution of the classes of testing samples fits the quadratic curve of the firing rate generated by our recurrent neural network architecture of varying cardinality; which was conceived in Section 4.3 of this chapter and illustrated in Figure 4.8. Remember, the cardinality of the neural network is a result of the dot product between the DEFR coefficients and the attributes of a data sample presented upon testing. It is remarkable that DEFR separates the classes on a 2D Curve as seen in Figure 4.15.

In figure 4.16, we plot the 10 fold cross validation test samples and their classes. Note that in this plot we sort the data samples according to their test result class so we can clearly identify the samples that are misclassified.

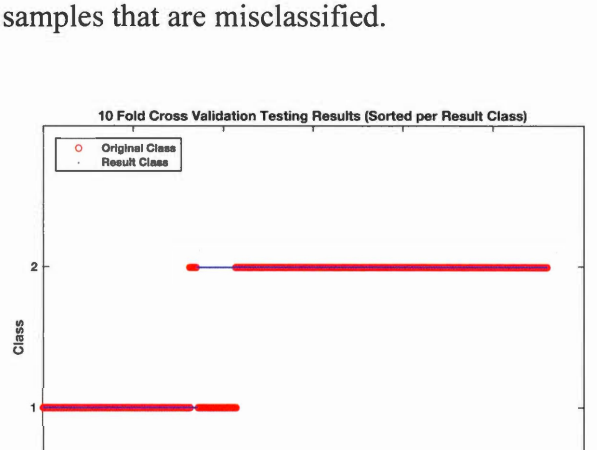


Fig. 4.16 10 Fold Cross Validation performed on the Breast Cancer Dataset. The original class of a testing sample is shown in red circles. The result class of a testing sample when the latter is recognized using DEFR is shown in blue dots. When a data sample, of a class, is correctly recognized then a blue dot is inside a red circle and when a data sample, of a class, is not correctly recognized then a blue dot is outside a red circle.

As we can see in Figure 4.16, original and resulting classes of data samples from the Breast Cancer Dataset 10 fold cross validation experiments are well matched on a large number of testing samples. For instance, less than 100 data samples of class 1 and 2 are intermingled.

Finally, we show the overall accuracy of 10 Folds Cross Validation testing results for both the IRIS and Breast Cancer Datasets using the proposed algorithm and we compare them to a Multilayer Perceptron (MLP) and Support Vector Machines (SVM), simulated on weka software, that apply a 10 Folds Cross Validation scenario, too. This is presented in table 4.8 next:

Table 4.8 Overall Accuracy of the proposed algorithm compared with MLP and SVM through 10 Folds Cross Validation

Dataset	Proposed Algorithm	MLP	SVM
IRIS	0.86	0.96	0.966667
Breast Cancer	0.90893	0.57469	0.952548

As we can see in table 4.8, the proposed algorithm, when applied on the IRIS dataset, shows less accuracy (i.e. 86%) compared to MLP that showed 96% accuracy and SVM that showed approximately 97% accuracy. However, the proposed algorithm shows better accuracy (i.e. approximately 91%) on the Breast Cancer dataset than the MLP that showed approximately 57% accuracy and it competes with SVM that showed approximately 95% accuracy. Even though our algorithm didn't achieve great overall accuracy, however the accuracy (e.g. ~91%) it provided when applied on a dataset with large number of attributes (e.g. Breast Cancer) is still very promising and is worth looking into. These results encourage further improvement of the proposed algorithm since the latter challenged one of the best machine learning algorithms (e.g. SVM).

4.6 Discussion

The idea of using the firing rate of chaotic spiking neurons in the context of computational neuroscience (CN) and machine learning (ML) is innovative and was never tackled in the literature. With regards to CN, when we want to analyze the response of neurons that is generally depicted in their irregular spiking activity, we

refer to Firing Rates generated by Poisson neurons (Heeger, 2000). In fact, Poisson neurons provide a manageable irregular spiking activity that could be set by assigning the firing rate (Heeger, 2000). The drawback of this approach is that a Poisson neuron withholds the refractory window observed in real neuronal activity. The fact that we found a way to provide a Poissonian spiking distribution using chaotic spiking neurons is extremely important because it maintains refractoriness in the spiking output, which is biologically more plausible than Poisson neurons. Second, with regards to ML, there was one single attempt in the literature combining the firing rate of a regularly spiking neuron with an evolutionary algorithm in order to implement ML (Vazquez, 2010, Valadez-Godinez et al., 2017). Our approach makes another attempt in this regard, by using irregular spiking neurons; while controlling their firing rate, and embeds an evolutionary algorithm in order to achieve ML. This sheds light on the overlooked potential of irregular spiking neurons and encourages further investigation in their use with evolutionary algorithms. We note, that evolutionary algorithms are starting to be considered in the context of Deep Neural Networks to enhance their performance (David and Greental, 2017).

One potential application of using chaotic spiking and firing rates could be the generation of Poissonian spikes with refractoriness for the Mixed National Institute of Standards and Technology – MNIST – handwritten digits database (Lecun and Cortes, 2010). This is because MNIST is extensively used, in the literature, to test visual object recognition algorithms using spiking neurons. The visual objects in the database are transformed to spikes using Poisson neurons, but such approach lacks refractoriness in spiking, however our approach maintains refractoriness, so it can be used for such transformation, instead of Poisson neurons.

In the context of cognitive and biological neuro-computing, two important questions arise: First, does the brain use an optimization algorithm to manipulate the number of neurons that could act as a cluster to perform classification of a stimulus? Second,

does the size of an assembly of cells affect its cumulative (i.e. synchronized) firing rate? Due to time constraints, we are unable to fully answer these questions. Thus, we leave them as open questions to be tackled in the future.

In this chapter, we provided a radical approach towards learning. This is because learning, in general, is achieved by managing the connections weights inside a neural network, thus creating new connections, strengthening or inhibiting existing connections between the neurons, however we claim here that learning can be achieved by managing the number of neurons of a neural network. In other words, the conventional approach fixes the number of the neurons inside a neural network and works only on tuning the connections between the neurons of the neural network, however in our case, we say the number of neurons should not be static, it should be dynamic. For instance, a Perceptron optimizes the weights that scale the inputs, in order to separate the input. In our vision, the Perceptron becomes a model of clusters of neurons, and its goal is to optimize the number of neurons, which scale the inputs through their firing rates.

4.7 Conclusion

In this chapter, we reached machine learning using firing rates of chaotic spiking of neurons. To do this, we envisaged a recurrent neural network composed of spiking neurons that fire chaotically, and then we controlled their activity over a fixed time period, that could be set experimentally without restrictions. The number of spikes divided by the time period constitutes the firing rate of the neural network. Furthermore, we applied an optimization algorithm to find the suitable number of neurons inside the neural network in order for the latter to perform supervised machine learning and data classification. The system accuracy in learning and classifying data patterns was compared to other machine learning methods and

showed promising results. Further work should tackle trying different optimization algorithms to be embedded in the classification task.

Last but not least, an important revelation towards the design of novel LSMs could be brought forward in this regard: As you may have noticed, in this chapter we created a sizable neural network that eventually combined both the SP and AP of a traditional Reservoir Computer like a LSM. Having an evolutionary algorithm to manipulate the size of the neural network did in fact work as a classifier in terms of a LSM readout mechanism. Thus, such work around highlights the possibility of a new theory that I call Sizable Reservoir Computing (SRC) by upgrading the necessary and sufficient conditions (i.e. SP and AP) deemed obligatory for RC, and this is done by making these properties interrelated in an automatic design fashion. I call such interrelation 'Adjustable Separation and Approximation Properties' (ASAP).

Finally, my discovery of chaotic firing rate that I laid out its demonstration here, is also a new adventure to explore for future work since it provides a novel approach of considering the use of Chaotic Spike coding in computational neuroscience, very similarly to the application of Homogeneous Poisson Spike coding (Heeger, 2000) in this regard and which is more biologically plausible.

CONCLUSION

Advantages, disadvantages, potentials and limitations of the mains themes elaborated in the thesis

As we have seen in chapter 3, one of the main advantages of using chaotic spiking neurons is the ability to reduce the computational burden of a Reservoir Computer, which requires large number of neurons in order to achieve nonlinear dynamical outputs that reflect its inputs. The nonlinear dynamics generated by a large number of recurrently connected regular spiking neurons of a Reservoir Computer can be obtained by a single chaotic spiking neuron as we have demonstrated in chapter 3. This reduces both design complexity and computation time of Reservoir Computing. We noticed a single computational disadvantage while using chaotic spiking neurons, which is the precision and allocation of large decimal points while initializing the variables and parameters of the equations of the dynamical system of the chaotic neuron model. This is due to the sensitivity of the chaotic neuron model to initial conditions, which is the main property of chaotic dynamical systems. However, this is not a significant disadvantage knowing that a Reservoir Computer as any other type of Recurrent Neural Network requires a delicate observation and manipulation of its parameters before it can function properly.

The neuroscientific advantage of exploring chaotic neuron models is their ability to reproduce nonlinear dynamics depicted in their irregular firing patterns as observed in biological neurons. For instance, a chaotic neuron model can qualitatively reproduce

the irregular activity in nerve membranes observed in squid giant axons and the Hodgkin-Huxley equations (Aihara et al., 1990). One of the main neuroscientific disadvantages of using chaotic neurons is the lack of ultimate evidence in proving that the irregular activity observed through neurophysiological readings of biological neurons is indeed chaotic and not just stochastic.

However, studying chaotic spiking neurons finds some advantages in the cognitive science domain due to the fact that such study would fall under the umbrella of dynamical systems, which are one of the fundamental and challenging approaches in studying cognition. For instance, Diane Larsen-Freeman (1997) theory suggests that second language acquisition has its roots in chaos theory and can be modeled using chaotic dynamical systems (Larsen-Freeman D., 1997). Other supporting arguments are the following: First, the theory of chaotic neurodynamics by Walter J. Freeman (1991), which claims that memories are manifestations of quasi-attractor chaotic activity of neurons, known as Chaotic Itinerancy. Second, the episodic memory model of Tsuda (2001), which describes the transfer from short-term memory to long-term memory, based on the approach of chaotic nonlinear dynamics. One disadvantage of considering chaotic neural dynamics in modeling cognition is facing the controversial question in interpreting the representation of a mental process. Some consider a mental process as the manipulation of a set of symbols that their outcome can be merely assessed by a Turing Machine while others think of it as a dynamic process transcending the manipulation of elementary symbols towards unpredictable transitions governed by chaotic nonlinear dynamics (a phenomenon called Chaotic Itinerancy).

On another side, the potential of considering UPOs as basis of memory lies in the fact of having an immense number of UPOs inside chaotic attractors that can be stabilized using methods of chaos control. If we model the UPOs of a chaotic attractor as a memory repository, then such model could theoretically encapsulate an infinite memory capacity. Considering UPOs as memory blocks has some limitations due to their exclusive belonging inside chaotic attractors, which will require a detailed

dynamical system analysis of the chaotic attractor in order to demonstrate their diversity such that they could convey a wide range of memory instances.

As for Resonant and Fire (RAF) Neuron used in this thesis, we have to note that a RAF is a simplification of the Generalized Integrate and Fire Neuron model called Spike Response Model (SRM) (Gerstner W. et al., 1996). In fact, the equations of the SRM can be modified in order to provide resonant neural activity that is dependent on previous spikes received by the neuron, like the RAF Neuron does, which is called cumulative SRM (Gerstner W. and van Hemmen J. L., 1992, Gerstner W. et al., 1996 and Gerstner W., 2000). However, a cumulative SRM requires an adequate choice and examination of the equations of its kernel filters. Thus, using RAF neurons overcome the computational burden of transforming a SRM to a cumulative SRM. The computational disadvantage of using Resonant and Fire Neurons is just the fact of having a neuron model defined in the Complex Domain ($\mathbb{C}$). The neuron model is defined with a complex variable in $\mathbb{C}$ that can be transformed to two real variables (the action potential and the adaptation) and simulated in the real domain ($\mathbb{R}$). The Neuroscientific advantage of using Resonant and Fire Neuronal groups is their ease of assimilation to Polychrounous Neuronal Groups. As we have seen in chapter 3, the parameters of every RAF Neuron were easily assigned in order to make a neuron get activated when it grasps a specific delay between two consecutive spikes. The use of RAF Neurons is very recent in the neuroscientific literature, which drives researchers in the scientific community to underrate its potential and make them repel its inclusion in their research. The lack of its popularity is due to the presentation of its dynamical equations as a complex variable defined in complex domain. The use of RAF neuronal groups introduces a new approach in the Cognitive Science domain given the fact that it provides an alternative to the Polychrounous groups approach. However, RAF neuronal groups rely on the controlled activity of a chaotic spiking neuron but this has no ground or scientific evidence in the cognitive science domain.

We mentioned in the conclusion of chapter 3 that RAF Neuronal groups are not

aware of their preceding states. We discussed such observation at the end of chapter 3 and provided potential remarks by comparing the behaviour of RAF Neuronal groups to the behaviour of Polychrounous neuronal groups. Hereafter, we would like to highlight an additional important remark regarding this observation from a cognitive science point of view. That remark relies on the fact that the neural network composed of a chaotic spiking neuron and RAF Neurons does not need to keep trace of its recent states in order to either evolve its dynamics or train them in the favour of learning or relating assigned outcomes or targeted representations of memories towards their corresponding inputs. This is since the chaotic spiking neuron, that is driving RAF Neurons, has the ability through its controlled nonlinear chaotic dynamics to always generate a wide range of stabilized instances of UPOs that are noise tolerant to similar inputs. Furthermore, it can always confine a stabilized instance of a UPO to any input (the input could be new, already seen or unseen). These stabilized instances of UPOs are represented as memories in the form of neuronal groups through the activation of RAF Neurons. A neuronal group (depicting a memory representation) could be considered new, recent or old, because it will be always the manifestation of a controlled chaotic nonlinear dynamic transformation of an input, where the latter could be new (unseen), recent (recently seen) or old (already seen or even unseen).

Finally, by expanding the time delay of the connections inside a recurrent neural network composed of chaotic spiking neurons, then we can increase the diversity of the possible neural states the network would stabilize onto. Furthermore, by increasing the number of neurons that compose the recurrent neural network then we can increase the chance of spikes occurrence which can be used as either a coincidence detection or an indicator of a high rate of chaotic spiking. We explored the latter case, which was very innovative from a neuro-scientific point of view since there is no study on the firing rate of chaotic spiking neurons in the literature.

Final remarks

In this thesis, I presented an exhaustive study that tackled chaotic spiking neurons. In chapter I, we introduced our thesis statement and the theories behind it. In chapter II, Synaptic Plasticity (e.g. STDP) was investigated in recurrent neural networks of regular spiking neurons vs. chaotic spiking neurons. We found out that STDP favored chaotic spiking over regular spiking. In Chapter III, I presented my theory of Neuronal Groups based on Chaotic Sensitivity, where memories were presented as active clusters of neurons; called Neuronal groups that were driven by a chaotic spiking neuron. This proved the thesis statement, which considered looking at memories from the perception of chaotic spiking, chaotic attractors and their Unstable Periodic Orbits (UPOs). Major contributions of this chapter (details in the discussion and conclusion section of chapter 3), include: I enhanced the Separation Property (SP) of a Nonlinear Transient Computing Machine (NTCM) (Crook, 2007) and built it using a biologically plausible neuron model called Adaptive Exponential Integrate and Fire (AdEx) Neuron (Brette, R., and Gerstner, W., 2005). Furthermore, I created a theory of Neuronal Groups, based on Resonant and Fire (RAF) neurons, that is similar to the theory of Polychronization: Computation with Spikes (Izhikevich, 2006). In chapter 4, I implemented machine learning (static data classification) using the firing rate of chaotic neurons. In fact, the major contribution of chapter 4 consisted in finding a way to increase the rate of chaotic spiking in a neural network. On the other side, in my published works (Aoun M. and Boukadoum, 2014 and 2015), done through the time course of this thesis, I implemented machine learning that tackled time series data classification (dynamic data classification) using a recurrent network of chaotic spiking neurons with synaptic plasticity. Thus, both static data classification and dynamics data classification were tackled using chaotic neurons. Also, in chapter 4, I laid down my discovery of chaotic firing rate that could be also used as a substitute for the classical firing rate of Homogeneous Poisson

Spike coding (Heeger, 2000) because it is more biologically plausible. As well, using chaotic firing rate and an optimization algorithm, I highlighted, in section 4.7, on another invention that I came up with, which I called Sizable Reservoir Computing (SRC) - (Check section 4.7 for details).

My journey in this Phd Program was very beneficial to me, and this thesis was the fruit of many years of exhaustive research, experimentations and studies. I am looking forward to expand the journey further more and continue my work in order to take the results of this thesis to a next level…

ANNEXE I

REPORT ON PULSED NEURONS (I.E. SPIKING NEURONS) BASED ON THEIR ABSTRACT MODELING LEVELS (E.G. BIOLOGICAL, PHYSIOCHEMICAL AND FUNCTIONAL) OF THE BIOLOGICAL NEURON

1. Introduction

In this report, we introduce different types of spiking neurons, which elucidate the real biological neuron on different levels of abstraction (biological, physiochemical and functional). On the lowest level of abstraction (biological, physiochemical), which is the microscopic level, we have the Conductance-based models that consider the spatial structure of a neuron and the ionic channels that the neuron has over its synapses and their mechanisms (Section 3). On an intermediate level of abstraction (functional), we simply disregard the heterogeneity of the neuron structure and consider the neuron as a "homogeneous unit that generates spikes if the total excitation is sufficiently large" [1, page 16]. In the latter, the neuron models falls under the category of Threshold fire models (Section 2). For a very high level of abstraction (mostly aiming towards functional), we have the classical rate models of neurons where the pulse structure of the neuron output is neglected [1, page 16] (Section 4). Also, we introduce the Izhikevich Neuron, which falls under the biological level and the functional level of modeling a neuron (Section 5). In section 6, we conclude the report.

2. Threshold Fire models *(Functional Level)*

2.1. Spike Response Model

When a spiking neuron i fires, its internal state described by a variable u_i is reset to an after spike value called v.
u is considered as the membrane potential of the neuron. The neuron is said

to fire when u reaches a threshold v. The firing time of neuron i is denoted as t_i. Since we have many firing times then they are designated by the superscript f as $t_i^{(f)}$ where f is the spike number.
So, the set of firing times having a cardinality n of a neuron i is defined as it follows:

$$F_i = \left\{t_i^{(f)}; 1 \leq f \leq n\right\} = \{t | u_i(t) = v\}$$

To denote the most recent spike we write $t_i^{(n)}$ or just $\hat{t}$.
To lower the internal variable u_i we can add a negative contribution to it. The negative contribution can be interpreted as $n_i(t - t_i^{(f)})$. This happens when the neuron i had just fired and we want to reset (lower) its internal state variable u_i.
n_i can be considered as a refractory function which works as a kernel. For instance, a good choice of $n_i(s)$ is a function that vanishes when s is negative and that decays to zero when s goes to infinity.
The neuron i receives input from other neurons via its synapse. The set of presynaptic neurons is denoted Γ_i.

$$\Gamma_i = \{j \mid j \textit{ denotes the index of the presynaptic neuron to } i\}$$

$t_j^{(f)}$ collaborates in modifying the internal state of neuron i. This stimulation to the variable u_i (which denotes the internal state of the neuron i) is depicted by a collaboration that is stipulated by either an increase or a decrease to u_i. The amount of change caused by this collaboration is described as a kernel; denoted as ϵ_{ij}, and is scaled by the synaptic strength between neuron i and neuron j; which is the weight of the synapse; denoted by w_{ij}.
Thus, the state of a neuron i at time t is "the linear superposition of all contributions" [1, page 17] that affects its behaviour; which is merely caused by the neuron's response to its own spikes denoted as n_i and the neuron's response to its presynaptic spikes depicted by the kernel ϵ_{ij}. This can be modeled by the mathematical formula next; which is known as the Spike Response Model (SRM):

$$u_i(t) = \sum_{t_i^{(f)} \in F_i} n_i\left(t - t_i^{(f)}\right) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in F_j} w_{ij}\epsilon_{ij}(t - t_j^{(f)})$$

Note that u is biologically interpreted as the membrane potential, ϵ is the presynaptic potential and n is the refractoriness of the neuron.
A typical example of the refractoriness is:

$$n_i(s) = -v \ exp\left(-\frac{s}{\tau}\right) H(s)$$

Where τ is a time constant, v is the threshold and $H(s)$ is the Heaviside step function defined as:

$$H(s) = 0 \textit{ for } s \leq 0 \textit{ and } H(s) = 1 \textit{ for } s > 0.$$

$n_i(s)$ insures that the neuron is reset to zero after it fires. The threshold can be set as $n_0 \neq v$ and then the rest value would be $v - n_0 \neq 0$.
As for ϵ, its formulation is given next:

$$\epsilon_{ij}(s) = \left[\exp\left(-\frac{s-\Delta^{ax}}{\tau_m}\right) - \exp\left(-\frac{s-\Delta^{ax}}{\tau_s}\right)\right] H(s-\Delta^{ax})$$

τ_s, τ_m are time constants and Δ^{ax} is the axonal transmission delay of the neuron. Note that if w_{ij} is positive then the synapse is excitatory and of w_{ij} is negative then the synapse is inhibitory. Respectively, the Kernel acts as an excitatory postsynaptic potential (EPSP) or inhibitory postsynaptic potential (IPSP) according to the sign of w_{ij} which is either positive or negative.
Variations on the threshold formulation will allow a dynamic model of the threshold. This is achieved by setting the threshold to be equal to $v - \sum_{t_i^{(f)} \in F_i} n_i\left(t - t_i^{(f)}\right)$.
Also, note that if we let $\sum_{t_i^{(f)} \in F_i} n_i\left(t - t_i^{(f)}\right) \rightarrow n(t - \hat{t}_i)$ then the last spike is to be considered as the sole contributor to the neuron refractoriness. This means that the neuron's state is solely influenced by the contribution of its latest spike (i.e. most recent). In this case, the neuron is considered as having short-term memory [1, page 20].
In addition, if an external input is considered then it can be depicted as:

$$h^{ext}(t) = \int_0^{\infty} \tilde{\epsilon}(s) \mathrm{I}^{ext}(t-s) ds$$

Finally, If I^{ext} is a analogue input current and $\tilde{\epsilon}$ is another kernel to such input, then the contributions from other neurons and from external input that affect the neuron would be:

$$h(t) = \sum_{j \in \Gamma_i} w_{ij} \sum_{t_j^{(f)} \in F_j} \epsilon_{ij}\left(t - t_j^{(f)}\right) + h^{ext}(t)$$

Thus, the SRM of a neuron i depicted by its membrane potential u_i having short-term memory and an input (from other neurons and from external sources) would be:

$$u_i(t) = n(t - \hat{t}_i) + h(t)$$

One more thing, if we want to reach the threshold from below, then the set of spikes of a neuron i should be denoted using a derivative of the variable u as it follows:

$$F_i = \{t | u_i(t) = v \wedge u_i'(t) > 0\}$$

2.2. Integrate and Fire Neuron Model

The Integrate and Fire Neuron model is a special case of the SRM where the refractoriness function n and the presynaptic potential ε are defined as it

follows:

$$n(s) = -(v - u_r)\exp\left(-\frac{s}{\tau_m}\right)H(s)$$

$$\epsilon(s) = \frac{1}{1-(\tau_s/\tau_m)}\left[\exp\left(-\frac{s}{\tau_m}\right) - \exp\left(-\frac{s}{\tau_s}\right)\right]H(s)$$

u_r is the after spike reset value of the neuron, τ_m is the membrane time constant of the neuron and τ_s is a synaptic time constant.

Here is a MATLab code snippet (Reference: Public Domain) that illustrates a Basic Integrate and Fire Neuron Model:

```
% Basic Integrate and Fire neuron Model
% Based on the work of R. Rao 1999
clear
%Input Current:
I = 0.3 % in nA
% Capacitance:
C = 1 % in nF
%Leak Resistance:
R = 40 % in M ohms
% I & F implementation dV/dt = - V/RC + I/C
% Using h = 1 ms step size, Euler method
V = 0;
tstop = 300;
abs_ref = 5;%9;%5; % absolute refractory period
ref = 0; % absolute refractory period counter
V_trace = []; % voltage trace for plotting
V_th = 10; % spike threshold
for t = 1:tstop
   if ~ref
     V = V + [-(V/(R*C)) + (I/C)]
   else
     ref = ref - 1;
     V = 0.2*V_th; % reset voltage
   end
   if (V > V_th)
     V = 20;  % emit spike
     ref = abs_ref; % set refractory counter
   end
   V_trace = [V_trace V];
end
  plot(V_trace)
```

We run the above code on MATLab to illustrate the result. The plot of the

code is given in figure 2.2.1 next:

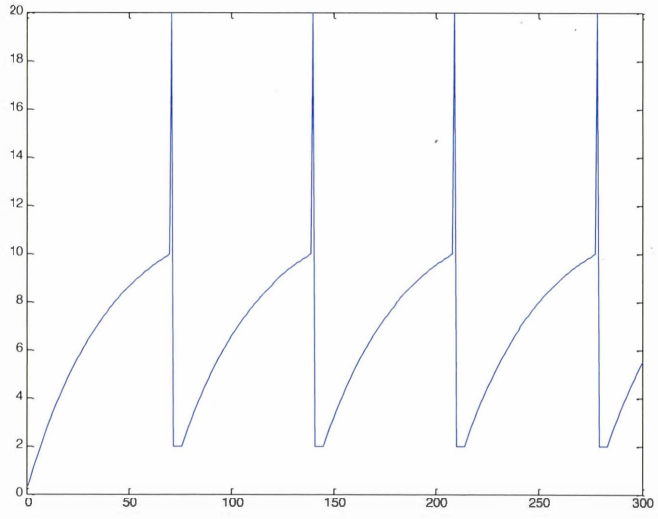


Fig. 2.2.1 Plot of a basic Integrate and Fire Neuron model when run on Matlab showing voltage versus time

3. Conductance Based Models

By applying variations on the SRM, then Conductance based models can be targeted under the umbrella of SRM. These variations consider the refractoriness function n and the presynaptic potential ε. In this era, SRM provides useful neuronal dynamics that can be indeed depicted by the Hodgkin-Huxley equations. Furthermore, other variations target what is called compartment models in which the spatial structure of the dendritic tree of a neuron and its synaptic transmission are considered.

3.1. Hodgkin–Huxley Neuron Model *(Biological Level)*

In 1952, Hodgkin and Huxley derived the action potential of a neuron and won a Nobel Prize according to their invention. Their experiments were based on the squid giant axon. In their work, they provided a mathematical model that can explain the mechanisms of the ionic channels that occur over the neuron semipermeable cell membrane. These mechanisms allow the neuron membrane to charge and discharge according to the flow of ionic currents that occurs through the ionic channels. The result of such process is the generation of an action potential, which is considered as a pulse (i.e. Spike) and propagated to other neurons.

Now, we will describe the Hodgkin and Huxley neuron model. The Hodgkin and Huxley neuron model considers the neuron as an excitable cell medium. The model is sketched in figure 3.1.1.

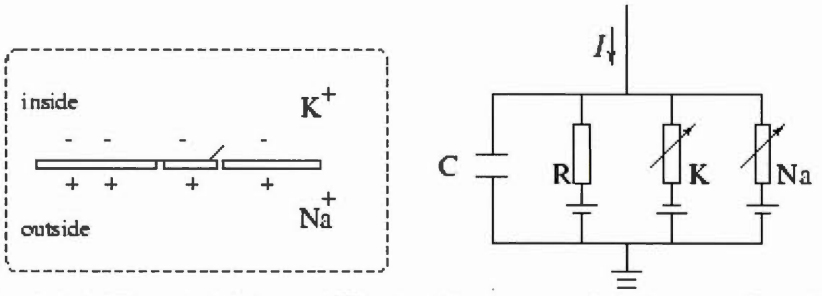


Fig. 3.1.1. The Hodgkin and Huxley Neuron model. Excerpt from [2]

The membrane of the cell, which is the excitable medium, is a surface that can be charged either positively or negatively according to the flow of ion currents as seen in figure 3.1.1.
Since the membrane can be electrically charged then it has a charge capacitance, called C. The difference between the electric potentials inside and outside the cell membrane is a voltage; called u. The conservation of electrical charges through the influence of many ionic currents I_k (where k defines the ion) and an external current I is given by:

$$C\,\frac{du}{dt} = -\sum_k I_k + I(t)$$

In the Hodgkin and Huxley model there are three ions: Na, K and L. Na is Sodium, K is Potassium and L describes the combination of Chloride and other ions. The total of the ionic currents is given by:

$$\sum_k I_k = g_{Na}\, m^3\, h\, (u - V_{Na}) + g_K\, n^4\, (u - V_K) + g_L\, (u - V_L)$$

where,
$V_{Na} = 115$ mV , $V_K = -12$ mV and $V_L = 10.6$ mV are voltage constants which describe the resting potential or the equilibrium potential of the ions.
$g_{Na} = 120$, $g_K = 36$ and $g_L = 0.3$ are constants describing the maximum conductance.
m, h and n are called gate variables and their function is to control the conductance of the ion channels. These variables vary between 0 and 1. m and h describe the dynamics of the Na channel while n describes the

dynamics of K channels.

m, n and h evolve in time and their rate of change is given according to the following equation:

$$\dot{x} = -\frac{1}{\tau_x(u)}\,[x - x_0(u)]$$

such that:

$$\tau_x(u) = \frac{1}{[\alpha_x(u) + \beta_x(u)]}$$

and

$$x_0(u) = \frac{\alpha_x(u)}{[\alpha_x(u) + \beta_x(u)]}$$

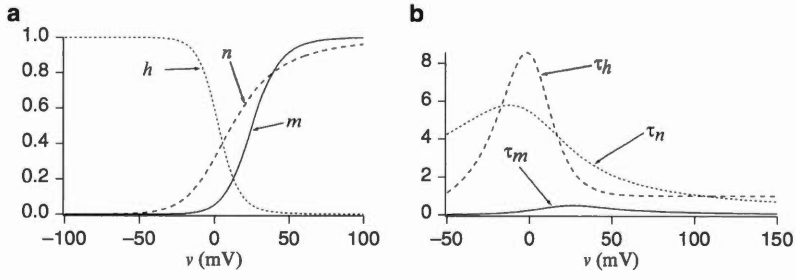


Fig. 3.1.2. Gate variables evolution of the Hodgkin and Huxley Neuron model. a) is the evolution of $x_0(v)$ in respect to v where x is h, n or m. b) The behavior of $\tau_x(v)$ in respect to v where x is h, n or m. Excerpt from [3].

where x is either m, n or h. Note that the time constant τ and x_0 are expressed by empirical functions α and β, which depends on u and are derived according to experimental data.

As we can see in figure 3.1.2, x_0 has a sigmoidal shape. Between -50 and +50 both τ and x_0 have significant behaviour. In this range, a threshold θ_x is considered which constitutes the maximum value of the sigmoid function.

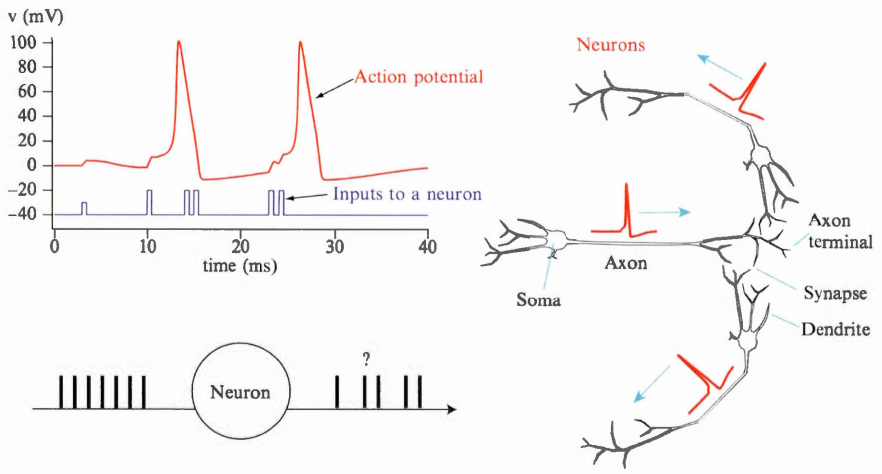


Fig. 3.1.3. Action Potential propagation between neurons. Excerpt from [3]. The refractory period of the neuron is the downfall of the curve of the action potential; where the third and fourth input spikes (shown in blue in the top left of the figure) are inadequate to cause a firing impulse in the neuron.

According to the equations above, the membrane potential is increased when an external input is provided. This is because Na conductance increases due to the increase of m and the flow of K to the neuron (Fig. 3.1.2). This causes an action potential which results a spike generation. When u reaches it maximum, the Na conductance is turned off due to h. K settles down and reaches equilibrium, which in returns lowers the potential (i.e. This is called the refractoriness period and is shown in figure 3.1.3).
A spike train is generated when a constant external input that is larger than a critical value I of theta is provided to the neuron as shown in figure 3.1.4 A. If the total number of spikes in an interval T is calculated and divided by T, then a firing rate can be generated as shown in figure 3.1.4. B. The latter is called the gain function of the HH model.

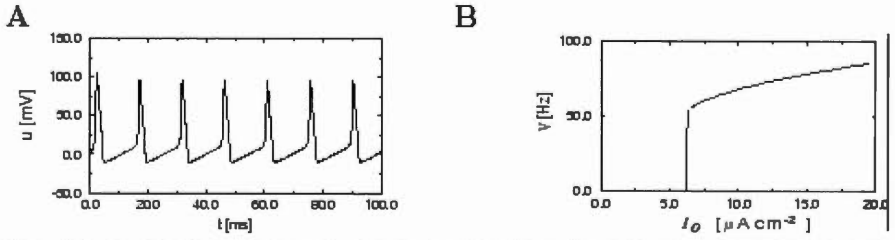


Fig. 3.1.4. A) Shows a spike train caused by an external constant input current. B) Shows the firing rate as a gain function averaged over an interval of spikes. Excerpt from [2].

Simplifications over the Hodgkin-Huxley model results in a two dimensional model called the Morris LeCar model or the FitzHugh-Nagumo Model. More generally, they are called the Bonhoeffer/Van der Pol oscillator. This simplification allows a phase plane analysis. These models where reduced to an Integrate and Fire model, too [1, page 38]. The SRM can also simulate the behaviour of the Hodgkin-Huxley model by studying the refractoriness function n and the presynaptic potential ε of the SRM in regards to HH dynamics.

3.2. Compartmental Models *(Physiochemical Level)*

In the previous spiking neuron models, we considered the neuron as a homogeneous entity and we neglected its special physical structure. To model the biological spiking neuron in its physic-chemical abstraction allows us to consider the influence of the spatial structure of the dendritic tree and its synaptic transmission in more details. Such modeling approach is called the Compartmental modeling of a neuron. Furthermore, the compartmental model includes additional ionic currents than the HH model, which operates on the synaptic cleft, like Ca^{2+} and Ca-mediated K currents. Note that these slow currents are related to neuronal adaptation as proved in [4]. In the case of a compartmental model, the dendritic tree is divided to segments where each segment constitutes an electric circuit in the sense of the HH model with additional ionic conductance variables (Fig. 3.2.1). To elucidate the idea, we rely on the compartment model as introduced in [1, page 43] which refers to [5] where the segmentation of the dendritic tree is elucidated by a chain of compartments. Let us consider a single chain to exemplify the model. Each compartment will have a capacitance C and a Resistance R. Each compartment is connected to another compartment by a variable r, which is the resistor. The neuron has many compartments that are to be denoted by an index n. The soma of the neuron is considered as the initial the compartment. The index of the soma is equal to 1.

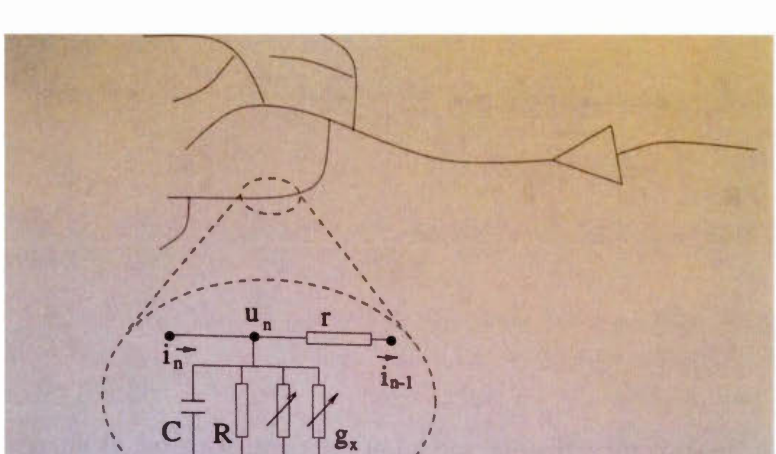


Fig. 3.2.1 A simplified diagram of the compartmental neuron model, each segment of the dendrite is an electrical circuit. Excerpt from [1]

In a more general form than the HH model that we have seen above, the conservation of current through a chain of N neurons is given at each interior compartments 2<= n<= N-1 by the following equation indexed by *n*:

$$\frac{u_{n+1} - u_n}{r} - \frac{u_n - u_{n-1}}{r} = C\,\frac{du_n}{dt} + \frac{u_n}{R}$$

for n =1 and n = N, we have similar equations. Note that the left term of the equation describes the longitudinal currents i_n and i_{n-1}. Since the membrane has a resistor R and a Capacitance C, then $\tau_m = RC$ is introduced as the membrane time constant. Thus, we have:

$$\tau_m\,\frac{du_n}{dt} = -\left(1 + 2\,\frac{R}{r}\right)u_n + \frac{R}{r}\,(u_{n+1} + u_{n-1})$$

for 2<= n<= N-1.

For n = 1 we have:

$$\tau_m\,\frac{du_1}{dt} = -\left(1 + \frac{R}{r}\right)u_1 + \frac{R}{r}\,u_2$$

For n = N, we have:

$$\tau_m\,\frac{du_N}{dt} = -\left(1 + \frac{R}{r}\right)u_N + \frac{R}{r}\,u_{N-1}$$

Note that the difference between two compartments' currents either leaks through R or charges C.

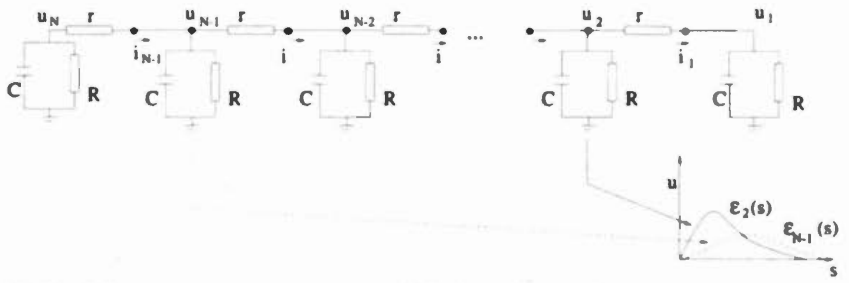


Fig. 3.2.2. A linear compartment model showing voltage responses according to injected currents to the soma and to the N-1 compartment, respectively. Excerpt from [1]

In the regard of a physic-chemical level of abstraction, an interesting question could be asked: "What is the voltage response of compartment number n =1 to a short input current pulse into compartment number n?" [1, page 44]. The answer is: If the spike is close to n = 1 (i.e. n=2) then the response is a fast rising function. If the input is far away from n = 1 (i.e. the soma) then the voltage at the soma rises slowly [1, page 44]. This concludes that the location of the input affects the postsynaptic potential measured at the soma (Fig. 3.2.2).

4. Spiking Neurons based on Rate Models *(Very high level of abstraction)*

A spiking neuron that falls under the category of classical rate models is described as it follows:

$$v_i = g(u_i)$$

Where g is a sigmoidal function that tends to 0 when u goes to – infinity and tends to 1 when u goes to + infinity.

$$u_i = \sum_{j \in \Gamma_i} w_{ij} v_j$$

Where j denotes the index of the presynaptic neuron, i denotes the index of the postsynaptic neuron and v, u are their action potentials respectively. w_{ij} denotes the synaptic strength between the neurons j and i.
If we consider v_i as a differential equation over time then we get:

$$\tau \frac{dv_i}{dt} = -v_i + g(\sum_{j \in \Gamma_i} w_{ij} v_j)$$

Where τ is a time constant. Furthermore, this can be extended to a population of neurons.

5. Izhikevich Spiking Neuron Model *(Combined levels of abstraction)*

The Izhikevich Neuron falls under the biological level and the functional level of modeling a neuron since it combines the biological plausibility of Hodgkin–Huxley-type dynamics and the computational efficiency of integrate-and-fire neurons. It is described by the following system of ordinary differential equations:

$$\frac{dv}{dt} = 0.04\, v^2 + 5v + 140 - u + I$$

$$\frac{du}{dt} = a\,(bv - u)$$

Where $a = 0.02$, $b = 0.2$, $c = -65$ *and* $d = 2$. When the neuron emits a spike (i.e. when $v >= 30mV$) then

$$v = c$$
$$u = u + d$$

The biological spiking neuron characteristics' of Izhikevich [6] simple neuron model are summarized in the thumbnail representation (Fig. 5.1), next.

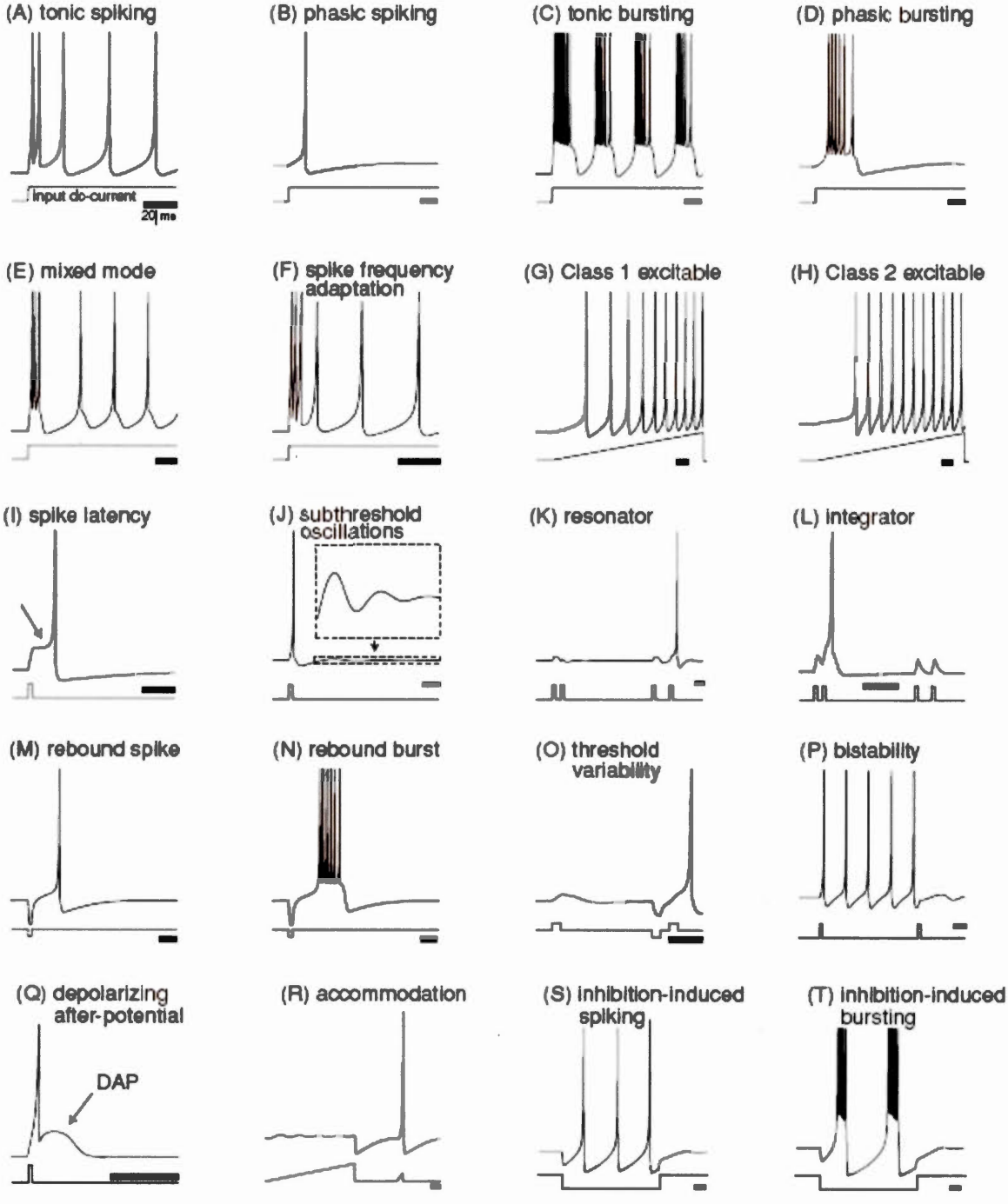


Fig. 5.1: Neurocomputational features of the biological spiking neurons of Izhikevich model [6]. Available online from http://www.izhikevich.com

6. Conclusion

In this report, we introduced the Threshold Fire models (i.e. Spike Response Model and the Integrate and Fire Neurons Model), which depict the functional level of a neuron. Also, we explained the Conductance Based Models: The Hodgkin–Huxley Neuron Model and the Compartment Models that elucidates the biological level and the physiochemical level, respectively. Furthermore, Spiking Neurons based on classical Rate Models and Izhikevich Model were exemplified.

References

1. Mass W. & Bishop C.M., *Pulsed Neural Networks*, MIT Press, 1999.
2. Gerstner and Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity.* Cambridge University Press, 2002
3. S. Doi et al., *Computational Electrophysiology*, 37 DOI 10.1007/978-4-431-53862-2 2, Springer 2010
4. W M Yamada, C Koch, P R Adams. *Multiple channels and calcium dynamics*. In Methods in neural modeling (pp. 137– 170) C. Koch & I. Segev (Eds.),MIT Press, Cambridge, 1998
5. Rall, W. (1989). *Cable theory for dendritic neurons*. In Koch, C. and Segev, I., editors, *Methods in Neuronal Modeling*, pages 9-62, Cambridge. MIT Press.
6. Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*, 1569–1572.

ANNEXE II

RATE CODING VS. SPIKE CODING

1. Introduction

In this report, we explain rate coding (Section 2) and spike coding (section 3) from a biologically plausible analysis perspective. In section 4, we conclude the report, where we highlight on some advantageous aspects of spike coding over its counterpart that is rate coding.

2. Rate Coding

In a rate coding scenario, the activation function of a neuron can be considered as the rate of firing times over a time window in which this neuron fires a spike. This means that the number of spikes averaged over the time window holds the information that is to be gathered from this input. This is called mean firing rate (Fig. 2.1).

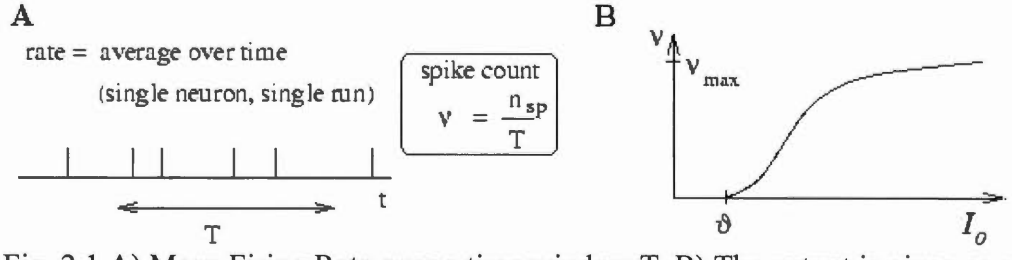


Fig. 2.1 A) Mean Firing Rate over a time window T. B) The output is given as a total input as a function of the total input I0. Excerpt from [1, page 8]

One interesting point of firing rate is that it can provide the strength of an input, and this is over a respective long duration. So, this can be a convenient method to communicate analog output depicted over a long range in time. However, this is not the case in reality. For instance, a fly can detect a stimulus and behave accordingly in a time window (e.g. 30 to 40 ms) [2] that is much smaller than the time required to average all the spikes input of this stimulus [3]. Which means that the fly is reacting to single spikes. Thus, the exact timing of the spikes holds much more information than the mean firing time of these spikes. Averaging the spikes input can work for static input or for an input stimulus that is moving slowly, which in return does not require a fast reaction. Likewise, Gerstner in [1, page 8] gives the example of viewing an image; where we perform instant recognition of the details while changing the direction of gaze.

Another method in rate coding considers the average over several runs. This is called the Spike Density and it is applied in the Eri-Stimulus-Time Histogram (PSTH) model [1, page 9], where different runs of the same stimulus are averaged over the time window of the stimulus and the number of runs (Fig. 2.2).

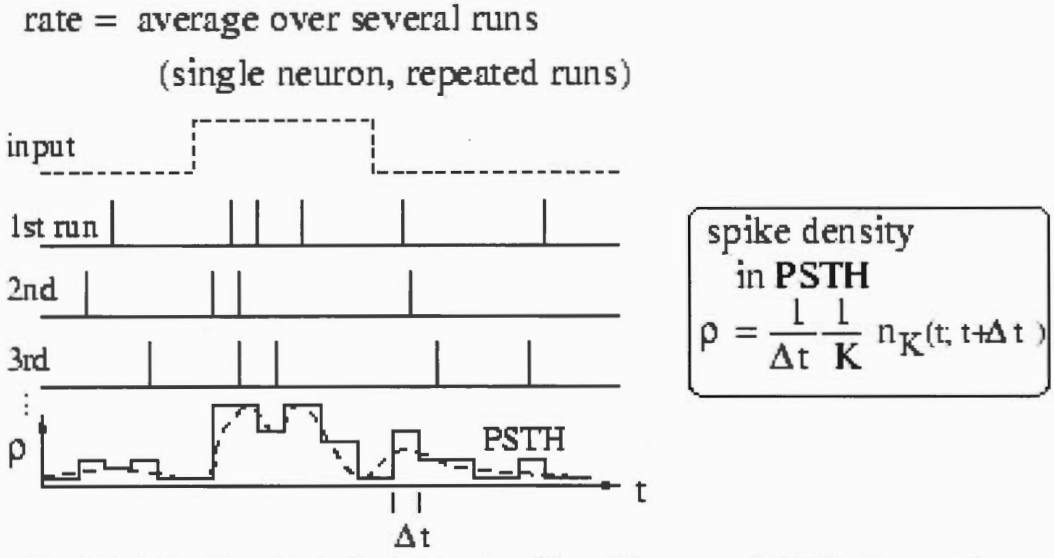


Fig. 2.2 Spike Density in Peri-Stimulus-Time Histogram (PSTH). Excerpt from [1, page 9].

Still, this method doesn't achieve a realistic sense. For instance, Gerstner in [1, page 10] gives the example of a frog, which wants to capture a fly. The frog will not wait for repeated flying trajectories of the fly in order to decide how to catch it. The frog shall act on a single flying trajectory to perform its action. Furthermore, the flies and their flying trajectories are much different even though they constitute

similar variations of the same stimulus.

The last scenario of rate coding considers the averaging of activity over a population of neurons in a small time interval (Fig. 2.3). In this situation, we escape the problems discussed previously as averaging on a single unit. However, this requires that the population should have homogeneous neuronal structure. In other words, all the neurons in the population should be homogeneous in terms of the internal parameters that drive their behaviour and should have identical connectivity; which is not usually the case in real biological neuron populations.

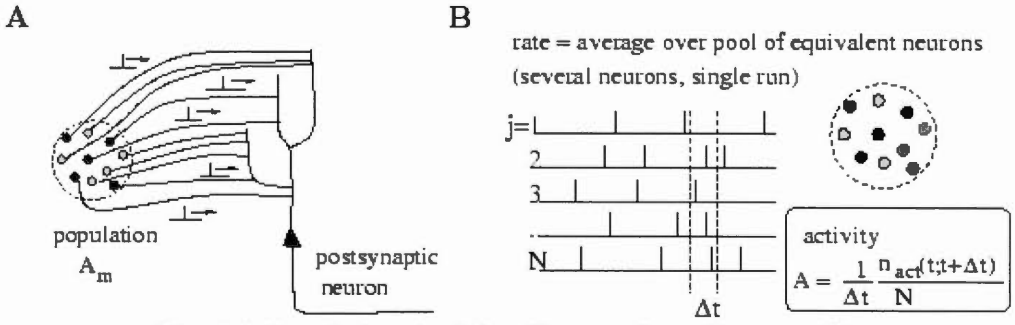


Fig. 2.3 Population Activity. Excerpt from [1, page 10]

In this regard, rate coding simplifies brain activity and real biological realism. Thus, spike coding should have an influential role in depicting a stimulus where a response is treated based on the exact firing time of each spike in the stimulus signal.

3. Spike Coding

As we have seen in the previous section, abrupt changes in the input signal of a neuron are quite often in the realistic world of biological neural information processing. Different Spike coding strategies have been tackled in this regard.

An idealization strategy is called the Time to First Spike Strategy. For instance, "when we look at a picture, our gaze jumps from one point to the next" [1, page 11]. In this scenario, the spikes timings that occur after each jump (lets say the jump occurred at time t0) provide significant information about the stimulus of this new jump. When a neuron fires slightly after t0, is to be considered highly stimulated by this stimulus. However, a neuron that fires later signifies a weaker response to this stimulus. In this strategy, the first spike is the most relevant in order to analyse the information that is to be grasped by this stimulus [4, 5].

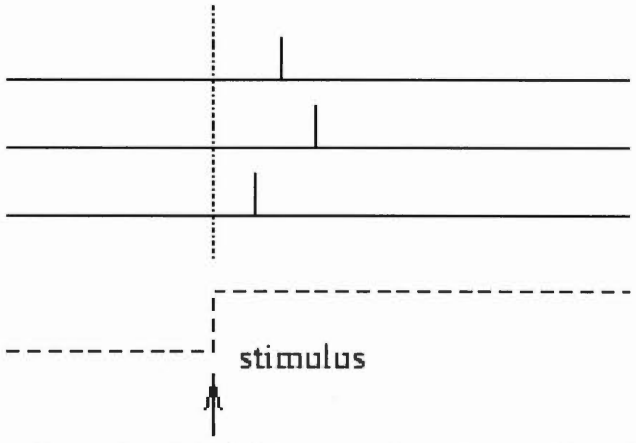


Fig. 3.1 Time to first spike. The third neuron is to fire the first after the onset of the stimulus. Excerpt from [1, page 12]

Also, the first spiking strategy can be extended to a phase signalling strategy, where the first spikes that occur in a phase period are to be considered relevant in order to analyse a new stimulus. This assumes that "the reference signal is not a single event, but a periodic signal" [1, page 12]. Such periodic signalling and oscillations serve as an reference signal and are found in the hippocampus and the olfactory cortex where the oscillations repeat periodically while there is no change in the input stimulus. These periodic patterns of spikes are driven by a background oscillation (Fig. 3.2).

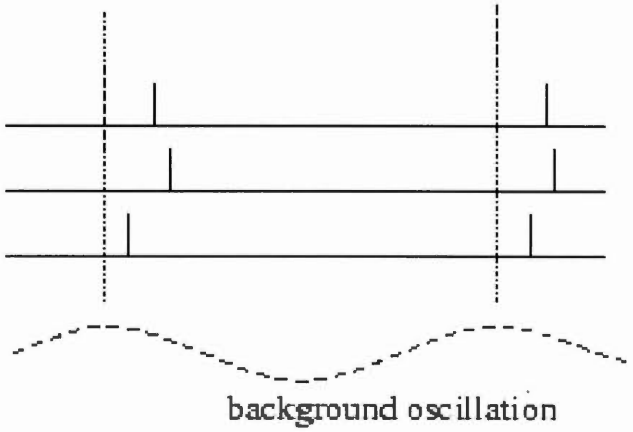


Fig. 3.2 Phase Coding. Three neurons fire at different phases according to the background oscillation. Excerpt from [1, page 12]

In [6], it was shown that the spatial location of a rat is correlated with phase spikes during oscillations that occur in the hippocampus.

Another mode of spike coding is called Correlations and Synchrony where spikes generated by a group of neurons could convey significant information (Fig. 3.3).

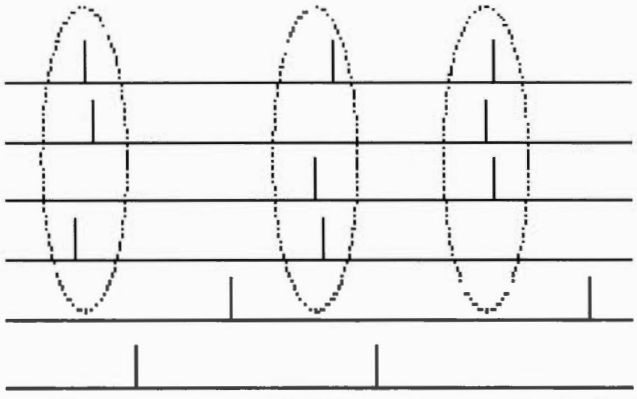
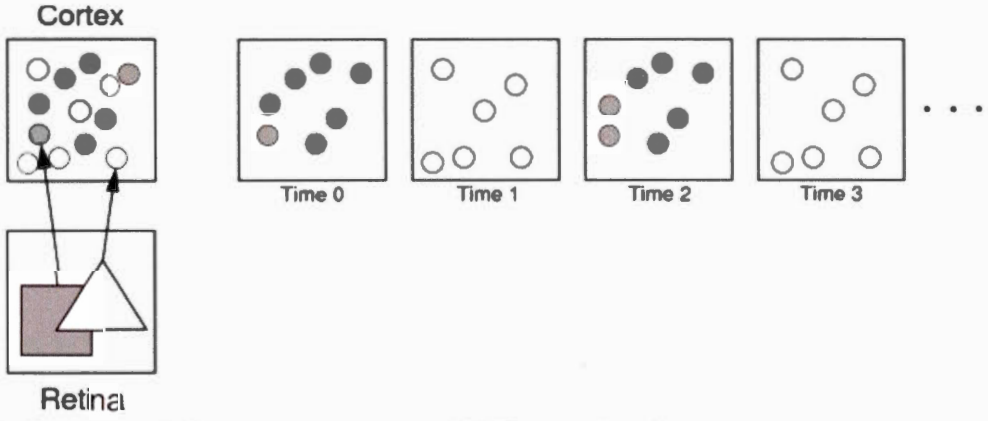


Fig. 3.3 Synchrony. Four neurons are firing synchronously depicting the response of the same object. Excerpt from [1, page 12]

Furthermore, this is well tolerated in Malsburg theory of Superposition Catastrophe [7, 8]. For instance, consider a group of neurons that fire in response to a triangle in the scene, and another group of neurons fires in response to a rectangle. To differentiate between each object (a triangle or a rectangle), we consider each group of neurons that is to be firing in a different synchronous pattern (Fig. 3.4).



(*a*) Static representation (*b*) Temporal coding

Fig. 3.4 Malsburg Superposition Catastrophe. If static presentation (i.e. firing rates) is used (a) to represent objects then this leads to confusion because both populations of neurons fire together. However, if temporal coding is used (b), as suggested by Malsburg, then the superposition catastrophe is solved because each

population will fire at different time step. Excerpt from [9].

Last but not least, using spike coding, Rieke et al [2] where able to reconstruct a stimulus according to its neural code (i.e. its spikes firing times) (Fig. 3.5). This means that a spike can indeed provide the time evolution of its preceding stimulus.

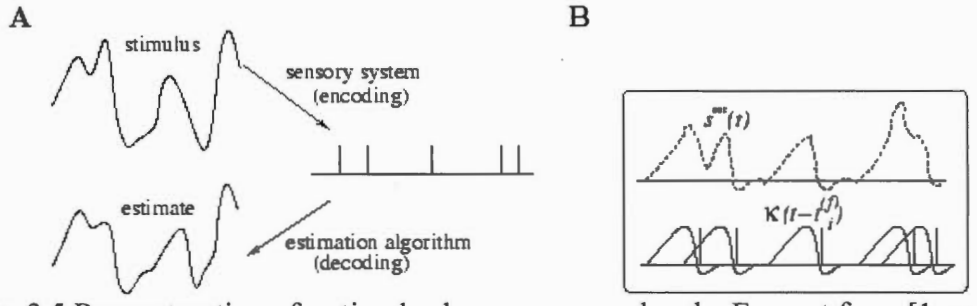


Fig. 3.5 Reconstruction of a stimulus base on a neural code. Excerpt from [1, page 15]

Note that the firing rate of a neuron is consistent and interrelated with the time-to-first-spike coding strategy, because if the rate is high then it is highly probable that the first spike occurred earlier and vice versa.

4. Conclusion

Rate or spike coding can be both considered in describing neural coding because they can be interrelated (as in the example of time-to-first-spike). However, one important coding scheme like spike coding takes over rate coding when we want to build a neural architecture that is aimed to model a response to a given stimulus conveniently and very quickly (like near instantaneous). For instance, a spiking neural network (SNN) based on spike coding is very feasible in the simulation of biological neural circuits like in the case of auditory processing of an owl this is because an owl can detect the localisation of sound with a single degree of precision. Such task can be simulated using spiking coding of SNNs with adaptable neuronal synapses. However, we should never neglect a major drawback of spike coding which is the computational power consumption that it requires when implemented on large scales of SNNs.

Last but not least, we have to note that in a network of spiking neurons, the inputs and outputs of neurons are vectors that encode time series, which is different than the traditional neural network architectures where information is encoded as vectors of numbers. However, we can simulate traditional neural network architectures like a Multi-Layer Perceptron (MLP) with SNN but the solution

requires synchronized output of spiking neurons. In fact, this leads the SNN to lose valuable information while it is attempting to simulate a MLP. Thus, an asynchronized mode is considered which shows promising results in terms of computational power of consumption of SNN architecture over a traditional MLP architecture. The importance of an asynchronous mode of spiking in a network of spiking neurons, with analogue values encoded as temporal patterns, shows that a spiking neuron has much more computational power than the classical sigmoidal gate neuron models used in MLP. For instance, a spiking neuron can act as a coincidence detector for a set of train pulses, which are considered as input vectors. An example is exhibited as the following: Let the numbers in a set of an input consist of the timing of arrival pulses. The spiking neuron will be able to give an approximate measurement of how much these timing instances that constitute the input set coincide. Such operation is really expensive in terms of computational steps when executed on an MLP with sigmoidal gates. For instance, any feed forward network (e.g. MLP) of sigmoidal gates needs to have at least (n-4)/2 gates in order to compute an element distinctness of a set of input variables of order n [1, page 61]. Furthermore, an additional advantage of using a spiking neuron in this regard is that it can compute this elementary distinctness even if the input is jittered with noise. Besides, an important property of SNN based on the proof of Gerstner [1, page 66] that is worth mentioning is the following: "Any given continuous function F: $F: [0,1]^n \rightarrow [0,1]^m$ can be approximated arbitrarily closely by a network of spiking neurons with input and outputs encoded by temporal delays". This means that Spiking Neurons can simulate universal approximation of continuous functions.

Finally, one novel possibility of information coding and processing that is plenteous and that can be realized by the use of spike coding through SNNs is called a Liquid State Machine (LSM) [10]. This framework allows computing with time perturbations without stable states. The state of the machine is like a liquid that is continuously changing with the presence of an input, and the output of the machine depends on a readout mechanism that can read the state of the liquid and transform it to an output. We have to note that the output of the readout does not depend on an instance of the input but on the current state of the liquid that is affected by an input; which would produce perturbations of the liquid. In this framework of computation on time varying signals, the first function (i.e. the liquid) describes temporal integration of the input and the second function (i.e. the readout) describes the special integration and learning of such information. Thus, the input is being transformed to higher dimension, which will help the readout mechanism to process it and will ease its learning and classification especially if the inputs are nonlinearly separable. Two necessary and sufficient conditions are required for computing with perturbations [11] as in Liquid State Machines (LSM) [10]. The first is called SP (Separation Property) and the second is called

AP (Approximation Property). SP mentions that different inputs to a pool of neurons (The Liquid) should cause different transients (perturbations) in the pool, and similar inputs should cause similar transients. The AP mentions that a reliable readout mechanism should be able to learn and map these transients to specific target outputs. In this framework, one can describe how an outcome behavior can be generated by a complex input stimulus by just analyzing the different neural activity caused by this stimulus to the pool. In analysing the corresponding neural activity of stimuli brings a new approach to the concept of "intelligence without representation, that has previously been proposed in the context of robotics [12] in order to overcome known deficiencies of traditional approaches from artificial intelligence in coping with the need to deliver adequate output-behaviors in real-time for realistically complex sensory input streams" [13].

References

1. Mass W. & Bishop C.M., *Pulsed Neural Networks*, MIT Press, 1999.
2. Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek, W. (1996). *Spikes - Exploring the neural code*. MIT Press, Cambridge, MA.
3. Thorpe S, Fize D, Marlot C. *Speed of processing in the human visual system.* Nature. 1996 Jun 6; 381(6582): 520-2.
4. Tovee, M. J. and Rolls, E. T. (1995). Information encoding in short firing rate epochs by single neurons in the primate temporal visual cortex. *Vis. Cognit.*, 2(1):35-58.
5. Tovee, M. J., Rolls, E. T., Treves, A., and Belles, R. P. (1993). Information encoding and the responses of single neurons in the primate visual cortex. *J. Neurophysiol.*, 70:640-654.
6. John O'Keefe and Michael L. Recce. Phase Relationship Between Hippocampal Place Units and the EEG Theta Rhythm. Hippocampus, 3:317-330. 1993
7. Von der Malsburg, C. (1981). The correlation theory of brain function. Internal Report 81-2, MPI für Biophysikalische Chemie, Göttingen. Reprinted in Models of Neural Networks II, Domany et al. (Eds.), Springer, 1994.
8. Von der Malsburg, C. and Buhmann, J. (1992). Sensory segmentation with coupled neural oscillators. *Biol. Cybern.*, 67:233-242.
9. Miikkulainen, Bednar, Choe, and Sirosh. *Computational Maps in the Visual Cortex* by (New York: Springer 2005)
10. W. Maass, T. Natschlager, H. Markram. *Real-time computing without stable states: a new framework for neural computation based on perturbations*. Neural Computation. 14 (11) (2002) 2531–2560.
11. Jaeger, H., Maass, W., & Principe, J. (2007). Editorial: Special issue on echo state networks and liquid state machines. *Neural Networks*, vol. 20, no. 3, pp. 287-289.
12. R.A. Brooks. *Intelligence without representation.* Artificial Intelligence 47 (1991) 139–159.
13. W. Maass, H. Markram. *On the computational power of recurrent circuits of spiking neurons*. Journal of Computer and System Sciences, 69 (4) (2004), pp. 593–616.

ANNEXE III

IMPORTANT POINTS GATHERED FROM WALTER J. FREEMAN'S ARTICLE ENTITLED "THE PHYSIOLOGY OF PERCEPTION"

Freeman (1991) explains that the intensity of a scent stimulus; the smell of an odorant, is depicted by the number of the receptors activated by this scent stimulus. The location of the activated receptors describes the nature of the scent. Thus, "each scent is expressed by a spatial pattern of receptor activity" (Freeman, 1991). Note that in the nose, receptors that are sensitive to an odorant are the only ones that are activated when such an odorant is sniffed through inhalation. The information is gathered from the sensory receptors, synthesized in the olfactory bulb and then transmitted to the olfactory cortex. From there, signals (that come from different sensory organs) are combined through the entorhinal cortex and finally a gestalt is formed in the brain (Freeman, 1991). This gestalt is unique to each individual because it is the meaning of the perceived scent: "A meaning-laden perception" (Freeman, 1991). For instance, the scent of a fox can have the meaning of prey for a dog while it has the meaning of an escape, like the feeling of fear arising memories of chase, for a rabbit. Furthermore, Freeman asks:

When a person glimpses the face of a famous actor, sniffs a favorite food or hears the

> voice of a friend, recognition is instant. Within a fraction of a second after the eyes, nose, ears, tongue or skin is stimulated, one knows the object is familiar and whether it is desirable or dangerous. How does such recognition, which psychologists call preattentive perception, happen so accurately and quickly, even when the stimuli are complex and the context in which they arise varies? Freeman (1991).

So, disregarding the subject's category or its identity, and disregarding the subjective meaning of perception, Freeman focuses on studying the neural dynamics beneath the recognition process of the scent perception itself. Freeman suggests that organisms would share a common framework of chaotic neural dynamics that lies beneath the memory process, which could take place when a scent is initially learned by an organism, and the same framework would explain the recognition process upon which a scent is perceived by the organism. Freeman supports his suggestion by delineating his analysis of the learning and perception if smell in rabbits as we outline next:

First, he notices that instantaneous, and drastic state changes occur in the neural activity of the bulb and cortex (shown in Figure 1) even when a minimal stimulus is presented to the organism (Freeman, 1991).

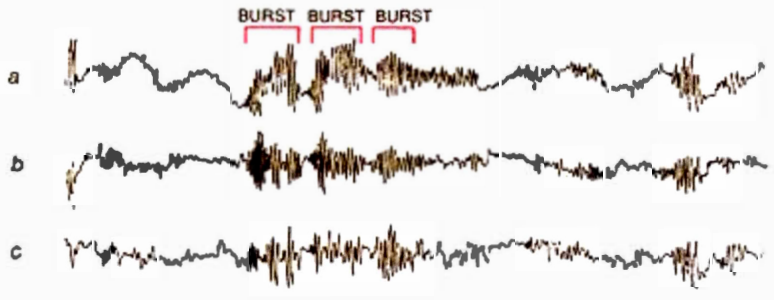


Fig. 1. Electrical Signals Activity in "the olfactory bulb (a) and front (b) and rear (c) parts of a cat's olfactory cortex" that is "generated when odors are perceived. The average amplitude of a burst is some 100 microvolts. Each lasts a fraction of a second, for the interval between inhalation and exhalation." Excerpted from (Freeman, 1991).

His intuition is that such changes - from low frequency waves to high frequency oscillations (called Bursts) – that occur in response to weak input "form another feature of chaotic systems" (Freeman, 1991). He proceeds: These set of bursts consisting of high amplitude and high frequency waves (called Gamma Waves) seen when an odorant is smelled, share a common waveform. This waveform is called the carrier wave: "A shared pattern of rises and falls that is embedded in each tracing" (Freeman, 1991). If we plot (Figure 2) the average amplitude of every carrier wave on a grid representing the surface of the olfactory bulb, then we'll have a contour map that is unique when the animal sniffs a particular odorant, however the carrier wave is always different for every odorant (Freeman, 1991)

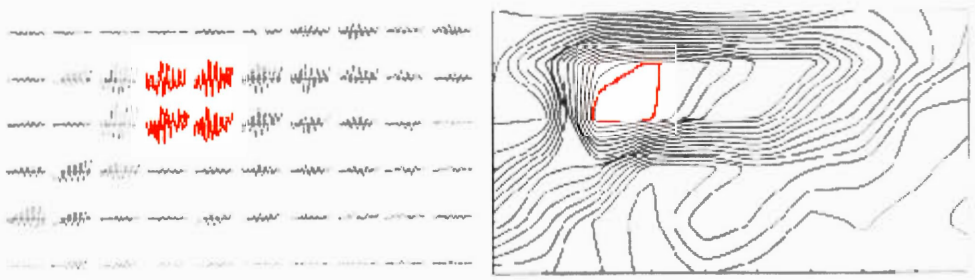


Fig. 2 Carrier waves *(Left)* and corresponding Contour Map *(Right)*. Excerpted from (Freeman, 1991).

Furthermore, he notices that bulb activity always bring new changes to the embedded memory, for instance, when the training of a particular odorant is altered, then the amplitude map representing this odorant is also altered and a new map representing this odorant would be newly perceived (Figure 3). This means that the bulb shapes the saved memories after it has been confronted with new stimuli (Freeman, 1991).

Fig. 3 Three Maps depicting the alternation of learning. The Contour map representing the perception of the Sawdust *(Left)* is altered when a new odorant, which is the smell of a banana, is learned *(Middle)*. A new contour map representing the perception of the Sawdust is generated *(Right)*.

By analyzing these maps, Freeman provides insights towards the design of new models of artificial neural computing systems that can exhibit cognitive characteristics like Attention, Learning and Perception:

On Attention:

"Chaos provides the system with a ready state so that it is unnecessary for the system to `wake up' from or return to a `dormant' equlibrium state every time that an input is given" (Freeman, 1991).

On Learning:

> Chaos constitutes the basic form of collective neural activity for all perceptual processes and functions as a controlled source of noise, as a means to ensure continual access to previously learned sensory patterns, and as the means for learning new sensory patterns... (Freeman, 1991).

On Instant Recognition and Perception:

"The brain transforms sensory messages into conscious perceptions almost instantly.

Chaotic, collective activity involving millions of neurons seems essential for such rapid recognition" (Freeman, 1991).

Finally, according to Freeman, memory binding and meaning inside the brain should happen through and within chaotic attractors, where he claims:

> *We think the olfactory bulb and cortex maintain many chaotic attractors, one for each odorant an animal or human being can discriminate. Whenever an odorant becomes meaningful in some way, another attractor is added, and all the others undergo slight modification* (Freeman, 1991)

ANNEXE IV

DIFFERENTIAL EVOLUTION ALGORITHM AS DESCRIBED BY VAZQUEZ IN 2010

Differential evolution begins by generating a random population of candidate solutions in the form of numerical vectors. The first of these vectors is selected as the target vector. Next, differential evolution builds a trial vector by executing the following sequence of steps:

1. Randomly select two vectors from the current generation.

2. Use these two vectors to compute a difference vector.

3. Multiply the difference vector by weighting factor.

4. Form the new trial vector by adding the weighted difference vector to a third vector randomly selected from the current population.

The trial vector replaces the target vector in the next generation if and only if the trial vector represents a better solution, as indicated by its measured cost value computed with a fitness function. Differential evolution repeats this process for each of the remaining vectors in the current generation. Differential evolution then replaces the current generation with the next generation, and continues the evolutionary process over many generations.

This description is excerpted from (Vazquez, 2010).

APPENDIX A

PUBLISHED PAPERS

Aoun, Mario Antoine, and Boukadoum, Mounir. "Learning algorithm and neurocomputing architecture for NDS Neurons." *IEEE 13th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), 2014.*

And, its extended version:

Aoun, Mario Antoine and Boukadoum, Mounir. 2015. **Chaotic Liquid State Machine**. International Journal of Cognitive Informatics and Natural Intelligence. 9, 4 (October 2015), 1-20. DOI=http://dx.doi.org/10.4018/IJCINI.2015100101

Chaotic Liquid State Machine

Mario Antoine Aoun[1] and Mounir Boukadoum[2]

Department of the Phd Program in Cognitive Informatics, Faculty of Science
Université du Québec à Montréal, UQAM
Montréal, QC, Canada
[1] e-mail: aoun.mario@courrier.uqam.ca; mario@live.ca
[2] e-mail: boukadoum.mounir@uqam.ca

Abstract— **We implement a Liquid State Machine composed from a pool of chaotic spiking neurons. Furthermore, a synaptic plasticity mechanism operates on the connection weights between the neurons inside the pool. A special feature of the system's classification capability is that it can learn the class of a set of time varying inputs when trained from positive examples only, thus, it is a one class classifier. To demonstrate the applicability of this novel neurocomputing architecture, we apply it for Online Signature Verification.**

A Grant from The Natural Sciences and Engineering Research Council of Canada (NSERC) supports this work.

Keywords— NDS Neuron; Chaotic Spiking Neuron; Chaotic Spiking Neural Network; Chaos Control; Hebbian Learning; Synaptic Plasticity; STDP; Reservoir Computing; Nonlinear Transient Computation; Liquid State Machines; Signature Coding; Online Signature Verification, One Class Classification.

I. Introduction

Reservoir Computing (Maass, Natschläger & Markram, 2002; Jaeger, Maass & Principe, 2007) design methodology consists of building a neural network that is divided into three parts. The first part is an input layer. The second part is the reservoir, namely a pool, consisting of randomly and recurrently connected nodes with fixed connections' weights. These nodes can be any type of spiking neurons. The third part is an output layer consisting of a read out mechanism that is able to read the transient activity of the pool (i.e. the activation of every node in the pool) and perform a classification task accordingly. In the context of Reservoir Computing, an input should be a time varying signal. Furthermore, the connections from the input layer to the pool, and the connections between the neurons that constitute the pool, have weights that are randomly set. All these connections maintain fixed weights during training (i.e. their weights are kept fixed and won't undergo any changes). The connections from the pool to the output layer are also randomly set, but they have flexible weights; this means, they are the only connections of the network that undergo training (i.e. their weights are updated during training). Fundamental models of Reservoir Computing are the Liquid State Machines (LSM) (Maass, Natschläger & Markram, 2002; Maass & Markram, 2004), Echo State Networks (ESN) (Jaeger, Maass & Principe, 2007) and Nonlinear Transient Computation (NTC) (Crook, 2007). Inspired by the work of Maass and Crook, we implement the chaotic Liquid State Machine model that is presented herein. First, the Chaotic LSM incorporates a synaptic plasticity mechanism inside the liquid layer, which is an approach suggested by Mass et al. (2002) to enhance the 'Separation Property' of the machine. Second, the Chaotic LSM uses a minimal number of chaotic spiking neurons inside the liquid layer; an approach suggest by Crook (2007), which can offer a substitute to the nonlinear dynamics offered by a large number of simple Leaky Integrate and Fire (LIF) neurons that are commonly used in the traditional design of a LSM. Third, the Chaotic LSM implements the theory of Delay Feedback Control (DFC) (Pyragas, 2003); on the neurons connections, to stabilize the chaotic dynamics of the liquid when the latter is fed with external input. In such *chaos control* scheme, the chaotic LSM operates on the critical region between chaos and order (i.e. operates on the *edge of chaos* (Langton, 1990; Natschläger, Bertschinger & Legenstein, 2004)); which can contribute to its generalization capability, especially since it is combined with synaptic plasticity, synaptic scaling and the use of one class similar training inputs (Maass, Legenstein & Bertschinger, 2005; Legenstein & Maass, 2006; Legenstein & Maass, 2007).

II. Liquid State Machines and Nonlinear Transient Computation

Two necessary and sufficient conditions are required for a system to perform Transient Computation on time varying signals as in Reservoir Computing and specifically Liquid State Machines (LSM) (Maass, Natschläger & Markram, 2002). The first is called SP (Separation Property) and the second is called AP (Approximation Property). SP mentions that different

inputs to a pool of neurons; that constitute the Reservoir or the Liquid in LSM terms, should cause different transients in the pool, and similar inputs should cause similar transients. The AP mentions that a reliable readout mechanism should be able to learn and map these transients to specific target outputs. Both properties were confirmed in the Nonlinear Transient Computing Machine (NTCM) model (Crook, 2007). To perform transient computation on time varying signals, the NTCM uses two Nonlinear Dynamic State (NDS) (Crook, Goh & Hawarat, 2005) Neurons only; in contrast to the huge number of Leaky Integrate and Fire Neurons hypothetically required by a LSM. NDS Neurons are spiking neurons that fire chaotically (Crook, Goh & Hawarat, 2005). We recreate the NTCM model, specifically, we exploit the effect of the neural connections' Delay Feedback Control (DFC) mechanism, by imposing a neural connections' Synaptic Plasticity tuning strategy, inspired by the biological phenomenon of Spike Timing Dependent Plasticity. The outcome is a new version of a Reservoir Computer called Chaotic Liquid State Machine and it works as a one-class classifier. Particularly, the machine can be trained on a set of time varying inputs. Each input, in the set, is a sequence of time intervals that are considered as its building blocks. These building blocks constitute the elementary characteristic features of every input, their arrangement in each input, and their slight length variations in the set, characterizes a one-class object and thus identities its unique identity. By way of an application, we choose Online Signature Verification (Jain, Griess & Connell, 2002) for two reasons: First, an online signature is a time varying input. Second, a set of instances of an online signature of a same person is in fact a repository of a set of time varying input examples that share alike kinetics. Note that the kinetics of a signature can be encoded as a sequence of timing intervals. Furthermore, the generalization of these sequences to a unique output by the means of a chaotic LSM is emphasized through the synaptic plasticity mechanism that operates on their connections weights with respect to the timing intervals and connections' time delay. The Chaotic Liquid State Machine is illustrated in Fig.1 next.

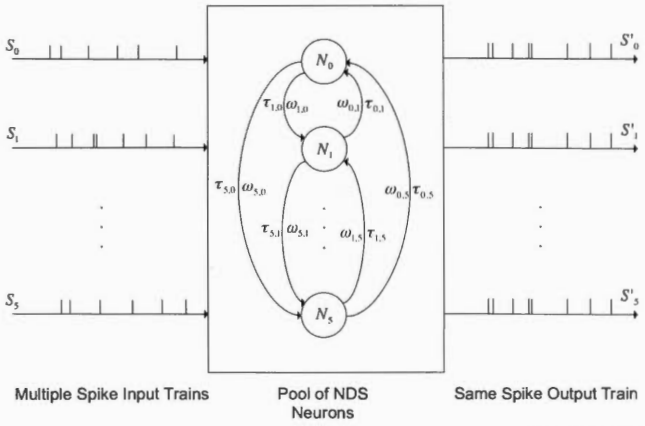


Fig. 1. Illustration of the Chaotic Liquid State Machine. It shows a pool (i.e. a network) of six chaotic spiking neurons called NDS Neurons (N_0, N_1, ... N_5) that are recurrently connected. Each connection has a time delay τ and a synaptic weight ω. The input consists of a set of six spikes patterns (S_0 to S_5) that are fed to the network. Using Chaos control and STDP, all NDS Neurons converge to the same spikes pattern S' considered as the output of the network.

III. Nds Neuron and STDP Learning

The NDS Neuron (Crook, Goh & Hawarat, 2005) is fundamentally based on the Rössler

Attractor (Rössler, 1976), which is a simpler version of the Lorenz Attractor (Lorenz, 1963). The NDS Neuron is a chaotic spiking neuron model, it is described by three Ordinary Differential Equations (ODEs), represented in their difference form based on Euler method as it follows:

$$u_i(t) = u_i(t_{pre}) + I_i(t)$$

$$x_i(t) = x_i(t-1) + b(-y_i(t-1) - u_i(t-1)) \quad (1)$$

$$y_i(t) = y_i(t-1) + c(x_i(t-1) + a y_i(t-1))$$

With initial conditions at time t = 0:

$$x_i(0) = 0, y_i(0) = 0 \; and \; u_i(0) = \eta_0$$

n_0 is the neuron after spike reset value, it is set to a random number between -1 and 0.
$x_i(t)$ and $y_i(t)$ are internal variables of the NDS Neuron.

$$u_i(t_{pre}) = u_i(t-1) + d(v + u_i(t-1)(-x_i(t-1)) + k u_i(t-1)) \quad (2)$$

a, b, c, d, v and *k* are constant parameters *(a=0.002, b=0.03, c=0.03, d=0.8, v=0.002, k=-0.057)*
$u_i(t)$ is the action potential of the NDS Neuron *i* at time *t*,
$u_i(t_{pre})$ is the internal voltage of the neuron during potentiation time (t_{pre})
$I_i(t)$ is the total neuron input and is equal to:

$$I_i(t) = \sum_{j=1}^{n} w_{ij}(t)\gamma_j(t - \tau_{ij}) \quad (3)$$

Where, *n* is the number of connections of the *ith* neuron, $w_{ij}(t)$ denotes the weight from node *j* to neuron *i* (Note that *j* can be a neuron or an index of an external input), τ_{ij} is the connection time delay (which is called Delay Feedback Control – DFC – (Pyragas, 2003)) from node *j* to neuron *i*.
$\gamma_j(t - \tau_{ij})$ is the spike output of neuron *j* at time step $t - \tau_{ij}$.
$\forall r \in (N = Number\ of\ Neurons\ in\ the\ network)$

$$\gamma_r(t) = \begin{cases} 1, & u_r(t) > \theta \\ 0, & u_r(t) \leq \theta \end{cases} \quad (4)$$

θ is the threshold and equal to zero. When the NDS neuron *i* fires then the neuron is reset to its after-spike reset value (i.e. when $u_i(t) > \theta$ then $u_i(t) = \eta_0$). Thus, the output of a NDS Neuron at time *t* is either 0 or 1 (the number 1 is interpreted as a Spike or a Pulse). The spikes output pattern of a NDS Neuron is a train of 0's and 1's in a time window of length τ. The state of a NDS Neuron, when it is in isolation, is chaotic (Fig. 2). Rich mathematical and dynamical system analysis of the NDS Neuron model; with comparison to the Rössler

dynamics, is provided in Alhawarat work (2007, 2015). Further studies, analysis and investigations are provided in (Alhawarat, Nazih & Eldesouki, 2013a, 2013b; Alhawarat, Scheper & Crook, 2014). In comparison to the Rössler system, Alhawarat results ensures the rich dynamics of the NDS Neuron model in terms of its extremely large number of UPOs, the importance of the reset mechanism and feedback mechanism which indulge the system between its two fixed points and stabilize its UPOs.

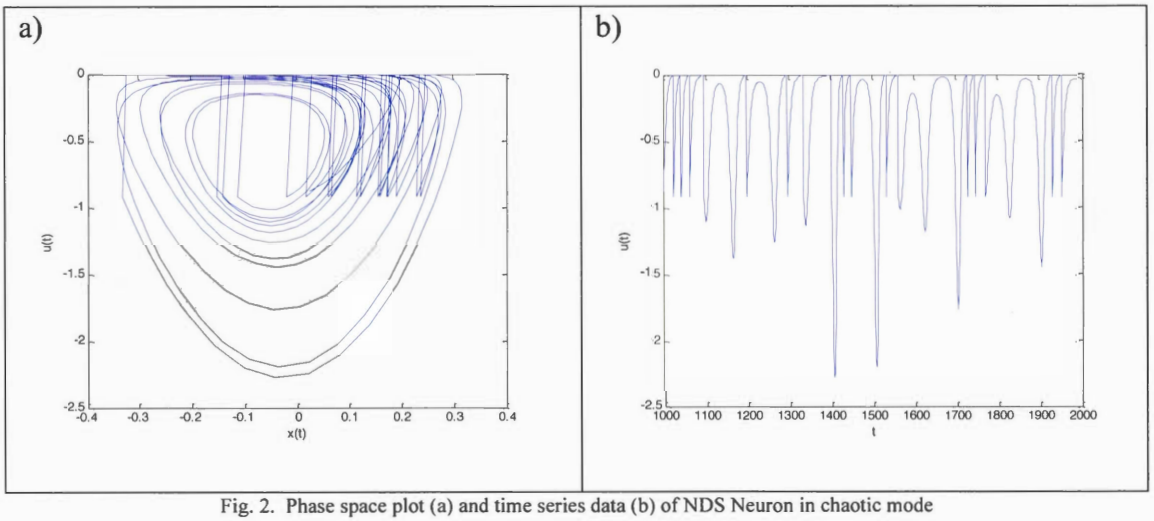


Fig. 2. Phase space plot (a) and time series data (b) of NDS Neuron in chaotic mode

Spike Timing Dependent Plasticity (STDP) learning rule considers the timing a presynaptic neuron fires and the timing a postsynaptic neuron fires, in order to build-up their synaptic strength. STDP can be stated as *"synapses that are activated slightly before the cell fires are strengthened whereas those that are activated slightly after are weakened"* (Rao & Sejnowski, 2001). Detailed analysis of the stability of a new form of STDP learning; implemented in a network of NDS Neurons, is given in our previous work (Aoun, 2010). Here, we will present enhancements to this learning rule and give further explanation on its functionality.

The goal of the learning algorithm depicted by the STDP learning rule and presented next, is to stabilize a network of NDS Neurons. So, NDS Neurons synchronize their dynamics and converge to the same neural state (e.g. Fig. 3a and Fig. 3c). In other words, all the neurons will generate the same spikes output pattern, while the network is fed with a stream of spikes inputs as illustrated in Fig. 1. After that, the network is considered as having a generalization over the inputs. Then the weights are saved and the network is ready to process new Input in a form of Nonlinear Transient Computation, as we will see in the next sections.

At time t = 0, the weights are set to random numbers between 0.05 and 0.3 (Crook, Goh & Hawarat, 2007):

$$w_{ij}(0) = \textit{Random number between 0.05 and 0.3}$$

After 3000 time steps ($t \geq 3000$) which is the time given for the neurons to run in isolation and evolve their dynamics, the synaptic changes start to take place and the weights are

updated according to the following:

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t_{pre}) \tag{5}$$

The weight update $\Delta w_{ij}(t_{pre})$ is driven by the following synaptic plasticity equation:

$$\Delta w_{ij}(t_{pre}) = \left(-u_i(t_{pre}) - I_i(t_{pre})\right) P(u_i(t_{pre}), I_i(t_{pre}))\rho_{ij}(t - \tau_{ij}) \tag{6}$$

Where, $-u_i(t_{pre})$ being the membrane voltage, taken as the inverse of the internal voltage $u_i(t_{pre})$.
$I_i(t_{pre})$ is the total input of presynaptic activity normalized over the number of input connections n:

$$I_i(t_{pre}) = \left[\sum_{j=1}^{n} w_{ij}(t-1)\rho_j(t - \tau_{ij})\right]/n \tag{7}$$

$\rho_j(t - \tau_{ij})$ represents the duration of potentiation of neuron j by calculating the number of trailing zeros between two consecutive spikes (i.e. the number of deactivation time steps that occurred before the activation at $t - \tau_{ij}$). In other words, $\rho_j(t - \tau_{ij})$ is a condenser that handles the potentiation time of the presynaptic neuron:

$$\rho_j(t - \tau_{ij}) = (\alpha/\tau_{ij}) \sum_{m=r}^{t-\tau_{ij}} (1 - \gamma_j(m)) \tag{8}$$

r is the time step of the last spike that occurred before the spike at $t - \tau_{ij}$. This defines h the inter spike interval (ISI):
$r < h < t - \tau_{ij}$ such that $\gamma_j(r) = 1, \gamma_j(h) = 0$ and $\gamma_j(t - \tau_{ij}) = 1$
α is a scaling factor set to 10.
$P(u_i(t_{pre}), I_i(t_{pre}))$ is an activation/deactivation variable; that depicts the simplest form of a STDP learning rule as it follows:

If $u_i(t_{pre}) < \theta$ and $u_i(t_{pre}) + I_i(t_{pre}) > \theta$ and $u_i(t-1) = \eta_0$ and $u_i(t_{pre}) + I_i(t_{pre}) < \theta + 0.2$ then P=1	L.1
If $u_i(t_{pre}) < \theta$ and $u_i(t_{pre}) + I_i(t_{pre}) < \theta$ and $u_i(t_{pre}) + I_i(t_{pre}) > \eta_0$ and $Abs(I_i(t_{pre}) - u_i(t_{pre})) < 0.8$ then P=1	L.2
If $u_i(t_{pre}) > \theta$ and $u_i(t_{pre}) + I_i(t_{pre}) > \theta$ and $u_i(t-1) <> \eta_0$ and $u_i(t_{pre}) + I_i(t_{pre}) < \theta + 0.2$ then P=-1	L.3

If $u_i(t_{pre}) > \theta$ and $u_i(t_{pre}) + I_i(t_{pre}) > \theta$ and $u_i(t-1) = \eta_0$ then P=1	L.4

Note that the threshold is equal to zero ($\theta = 0$), then the learning rules (L.1, L.2, L.3 and L.4) described above would imply:
L.1 implies: Spike is not expected; the neuron may fire but negatively updates its synapse.
L.2 implies: Spike is received in phase time, (expected); the neuron won't fire but positively updates its synapse.
L.3 implies: Spike is received at exact time (as expected); the neuron will fire and positively updates its synapse.
L.4 implies: Spike is received late (the neuron has just fired i.e. $u_i(t-1) = \eta_0$); the neuron will fire but negatively updates its synapse.
Note that $I_i(t_{pre})$ is being catalyzed and should *remain in equilibrium with internal potential changes of the neuron*, thus and after synaptic changes the following reaction takes place:

$$I_i(t) = \sum_{j=1}^{n} ((w_{ij}(t-1) + \Delta w_{ij}(t_{pre}))\rho_j(t - \tau_{ij})) \tag{9}$$

Since,

$$w_{ij}(t) = w_{ij}(t-1) + \Delta w_{ij}(t_{pre}) \tag{10}$$

Then and finally, the result of all synaptic changes; that affects the internal voltage of the neuron, after being catalyzed at pre-synaptic activity, would be:

$$I_i(t) = \sum_{j=1}^{n} ((w_{ij}(t))\rho_j(t - \tau_{ij})) \tag{11}$$

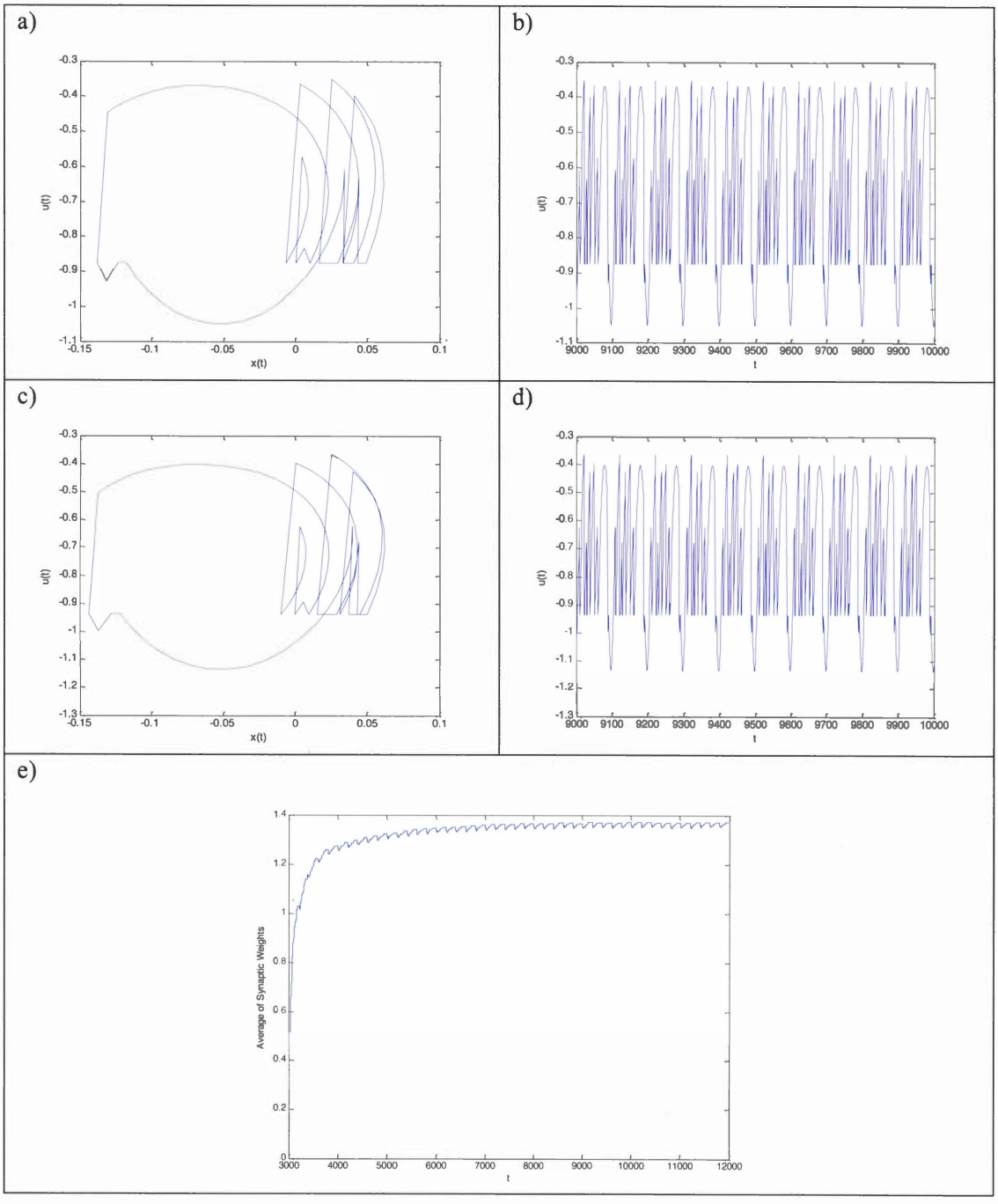

a)
u(t)
x(t)
b)
u(t)
t
c)
u(t)
x(t)
d)
u(t)
t
e)
Average of Synaptic Weights
t

Fig. 3. Displaying the activity of NDS Neurons when STDP is used: (a) and (c) are UPOs of two NDS Neurons, and their time series (b) and (d) respectively, showing synchronization and periodic output, (e) shows weights stabilization of the synaptic connections.

By following this mode of STDP learning, NDS Neurons are able to stabilize the weights of their connections and synchronize their outputs to identical periodic spiking patterns, as we can see in figure 3. We note that very fast stabilization takes place (Fig. 3e). In comparison to Aoun (2007), enhancements where made in rule number 1 and rule number 4. Now, these rules allow an inhibitory effect on the synapse. Also, to avoid overweight, the neuron input is normalized (7) and boundaries were set through the learning rules (L.1 to L.4). Furthermore, by introducing $\rho_j(t-\tau_{ij})$ instead of $\gamma_j(t-\tau_{ij})$ while calculating the input to the neuron, then the spike has more information to provide to the postsynaptic neuron (i.e. it embeds, at its onset, the duration of potentiation of the presynaptic neuron).

IV. Signature Coding and System Architecture

Online Signature techniques rely on many features that characterize the uniqueness of a signature like the speed, the acceleration, the torque, the X and Y coordinates, the pen pressure, the series of pen-ups and pen-downs… So, we plan using such features; provided from user samples and template signatures that are available from the signature database SVC2004 (Yeung, Chang, Xiong, George, Kashi, Matsumoto & Rigoll, 2004).

It was shown in Lei & Govindaraju (2005) that the X, Y coordinates, with the speed of writing and the inclination over the x axis are the utmost significant features that make a signature unique. Also, since the velocity and the angle above the x axis already incorporate x and y (For instance, the angle α above the x axis can be derived from $\cos(\alpha) = SpeedVector / x$ (Lei & Govindaraju, 2005)), then using x and y as the signature features is enough and adequate to analyze online signatures.

The input data given by Yeung et al. (2004) is originated from a pen tablet, so it should be normalized to a consistent range of values. We choose a range from -1 to 1 (Fig. 5a and 5b). Furthermore, it was shown in Lupu et al. (2009), that TESPAR (Time Encoding Signal Processing and Recognition) (King, 2004) is a reliable coding scheme for online signature recognition. A TESPAR stream of symbols is built using the minimum or maximum of a function with the zero crossing points. To encode the signal (e.g. the signature), we build a similar coding scheme like the TESPAR coding (King, 2004) by using the stationary points (relative minima and relative maxima) of the function x times y (Fig. 5d). We decided to use the product x times y, because multiplication has order preservation (i.e. if one of the values is negative and the other is positive then the order is negative. However, addition for instance doesn't preserve the order of operation…). In our coding scheme, a symbol stream of 0's and 1's is generated according to the following:

If the value of x times y is not a stationary point, then the symbol is equal to 0
If the value of x times y is a stationary point, then the symbol is equal to 1

Finally, a signature is transformed to a symbol stream of 0's and 1's (e.g. 00101000…), as in Fig. 5e, which is considered a spike train and fed as input to the network. This process that starts from normalizing the raw data to generating spike trains is called the digitization

process as indicated in the system architecture in Fig. 4.

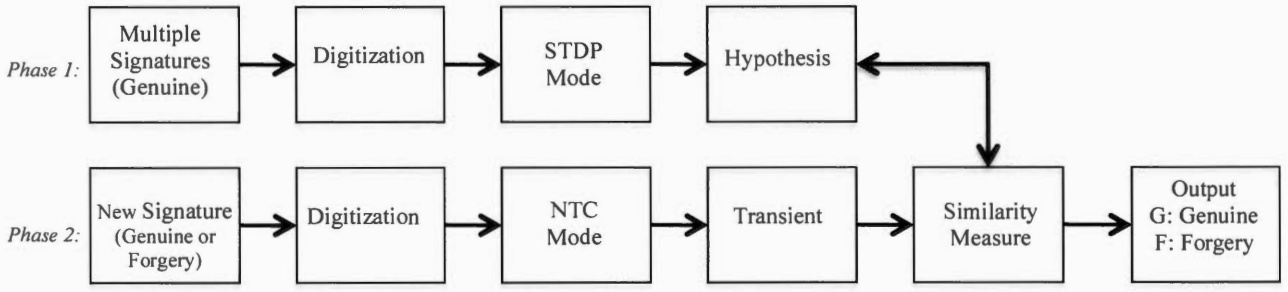
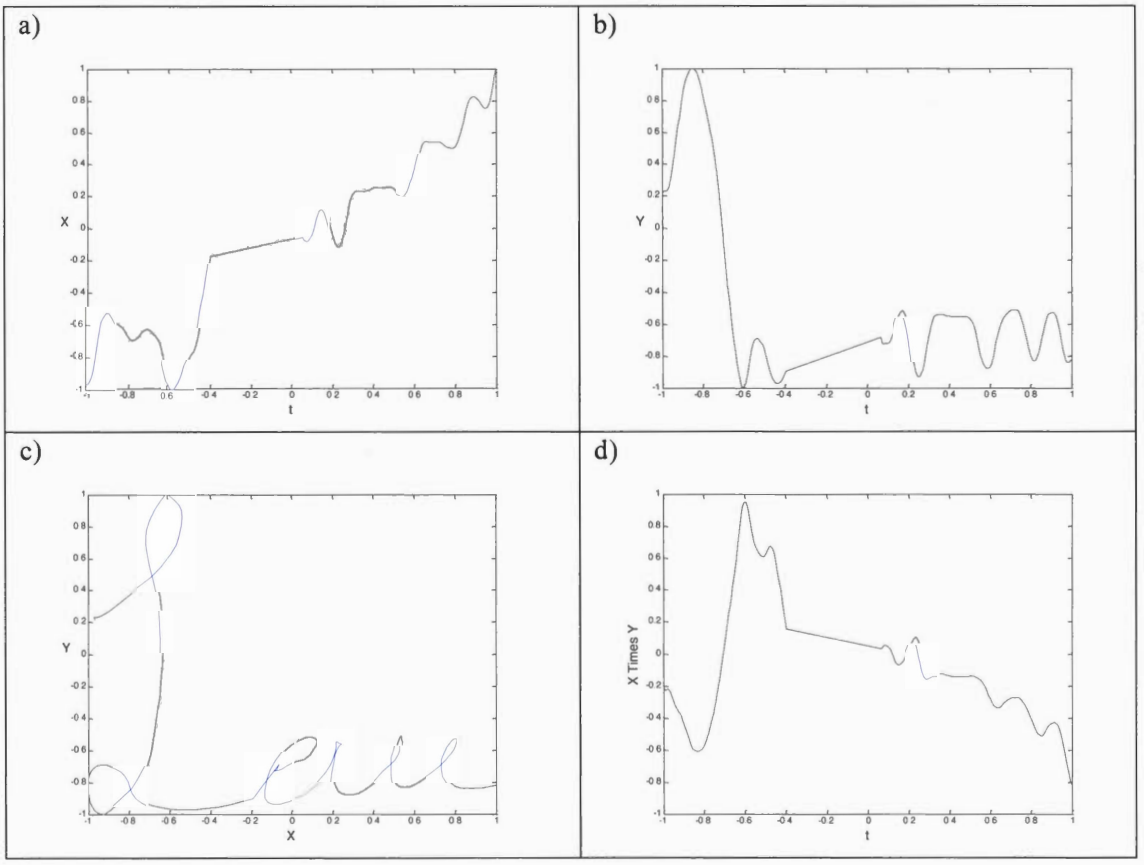


Fig. 4. System Architecture

The system architecture is divided into two phases following the diagram provided in Fig. 4. In the first phase, STDP is On and a set of different spike input patterns (genuine signatures of a person) is provided to the network for learning. All neurons converge to an identical spikes output pattern, we call it: The Hypothesis. This Hypothesis is a generalization of the genuine signatures of the person. It is used in the testing phase (Phase 2) for comparison.



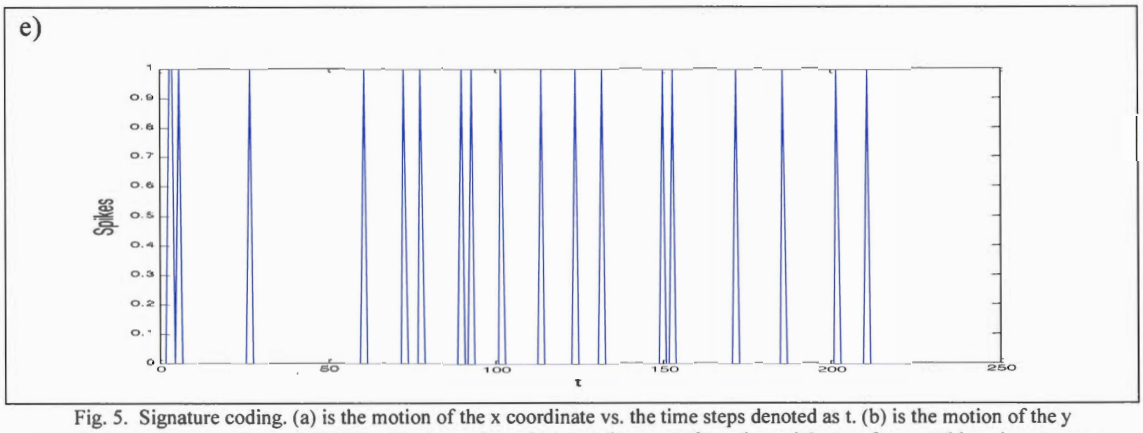


Fig. 5. Signature coding. (a) is the motion of the x coordinate vs. the time steps denoted as t. (b) is the motion of the y coordinate vs. the time steps denoted as t. The input data of (a) and (b) comes from the serial port of a pen tablet where a person signs and is normalized to the range -1 and 1. We generate (c), (d) and (e) from this input data. (c) is the plot of y vs. x which only shows the signature. (d) is the plot of the product of x and y vs. t, which will be used to generate a spike train. (e) is a spike train that indicates the stationary points of the product of x and y.

Similar scenario occurs in the testing phase, however, in this phase STDP is Off; the weights of the connections are now stabilized, so they are kept fixed and won't undergo anymore changes. We have a single input spikes pattern to test (a single signature that we don't know if it is Genuine or Forgery). This single pattern is propagated to all neurons. In the testing phase, we apply NTC.

The Separation Property (SP), which is the first necessary and sufficient condition for NTC, says that similar inputs to the network should cause similar transients and different inputs should cause different transients (Crook, 2007). So, the transient spikes pattern that is generated from the network in the testing phase is compared to the Hypothesis that was generated in the training phase. The comparison is carried using a similarity measure called: Hamming Distance. Hamming distance indicates the number of errors (bit differences) between two patterns. Thus, each transient will have a hamming distance. A high hamming distance means: no similarity between the transient and the hypothesis, which indicates forgery. A low hamming distance means: little errors between the transient and the hypothesis, which indicates genuineness.

V. Experimental Results

We have 6 NDS Neurons recurrently connected to each other with no self-feedback connections, as in Fig. 1.-

Each input is connected to all neurons. All the weights of the connections are set to random values between 0.05 and 0.3 (Crook, Goh & Hawarat, 2007). All neurons start with random initial conditions (i.e. the internal voltage u of every neuron is set to a random value between -1 and 0).

First, the network runs for 3000 time steps so the neurons can evolve their dynamics. They run in isolation (i.e. The connections delay feedback control τ is set to 0 for all neurons).

At time step 3001, the delay feedback control τ is set to 200 and every neuron in the network is fed with a digitized signature of the same user. Note that τ is set to 200 so the neuron state time window contains the signature time steps, this is because we selected signatures from the database in (Yeung, Chang, Xiong, George, Kashi, Matsumoto & Rigoll, 2004) that have time length less than 200.

The network runs for 9000 steps. In these 9000 steps, the network is in learning mode because STDP is governing the weight updates. It was observed that 9000 steps are more than enough to ensure stabilization. All the neurons in the network stabilize to the same spikes output pattern, that we call The Hypothesis, as in Fig. 6a.

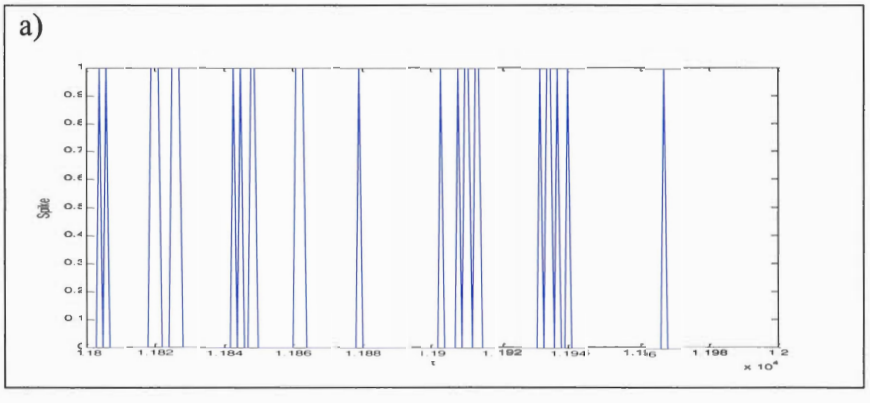
At time step 12000, STDP is turned Off and the network is fed with spike trails of 0's. The network runs for 6000 steps with input trails of 0's and this is to ensure that in the testing phase, no neuron will have input residue from the past.

At time step 18000, the Network (i.e. the pool of NDS Neurons) is fed with a Genuine Signature for testing. The Network runs for another 6000 steps and the neurons converge to a transient pattern as in Fig. 6b.

Again, the network runs for another 6000 steps with input trails of 0's to clean the path for future inputs and ensure that no trace of previous input exists in the dynamics of the network.

At time step 24000, the pool of NDS Neurons is fed with a Forgery signature and runs for another 6000 steps. The Network converges to a new transient pattern as in Fig. 6c.

As we can see in Figure 6, the genuine transient spikes pattern (Fig. 6b) - caused by a genuine signature - has many correlations with the Hypothesis (Fig. 6a) and less errors than the forgery transient spikes pattern (Fig. 6c) - caused by a forgery signature - which has less correlations but more errors when compared to the Hypothesis.



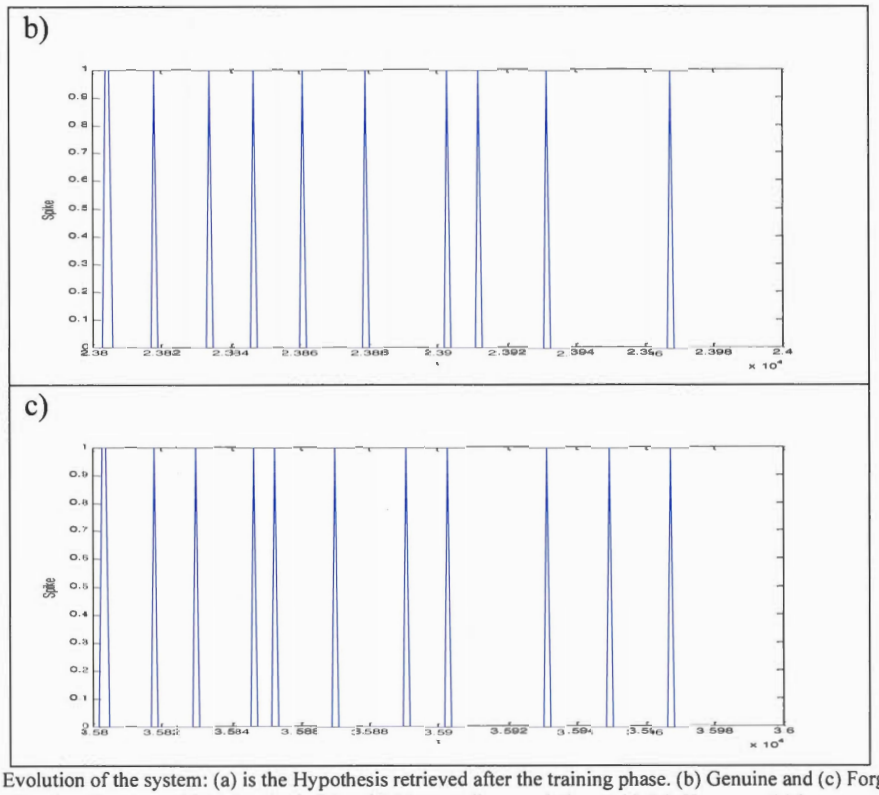


Fig. 6. Evolution of the system: (a) is the Hypothesis retrieved after the training phase. (b) Genuine and (c) Forgery are Transients of testing signatures. Note that, both (b) and (c) have spike correlations with (a). However, (c) has more errors than (b) when compared to (a), because (c) is a Transient of a Forgery signature.

VI. Analysis Using F-MEASURE

In a test experiment (e.g. Table 1, First row), after the network has generated its Hypothesis, the system is fed with 3 genuine signatures one after the other (separated by trails of 0's, like in the previous section).

TABLE I. Accuracy of the system based on the F-MEASURE

USER	GENUINE			Threshold	GENUINE			FORGERY			F-measure
1	33	34	31	32.6	32	29	31	31	34	34	0.86
2	31	34	32	32.6	29	29	29	36	37	36	1
3	32	35	37	34.6	34	37	33	36	38	39	0.80
4	37	38	40	37.6	38	36	36	40	39	39	0.80
							Average Accuracy:				0.86

The Hamming distance of each of their transients is recorded. These hamming distances are: 33, 34 and 31. The average of these hamming distances is calculated and is equal to: 32.6 (We call it: Threshold). The Approximation Property (AP) of NTC, which is the second necessary and sufficient condition for computing with perturbations, says that a readout mechanism should exist and is able to classify the transients (Crook, 2007). This Threshold acts as the AP.

Now, the network is fed with 6 signatures (3 Genuine and 3 Forgery), one after the other. The goal of the system is to detect which of the signatures is Genuine and which is Forgery by comparing the hamming distance of each of their transients to the Threshold. To be considered Genuine, a transient should have a hamming distance ≤ Threshold. To be considered Forgery, a transient should have a hamming distance > Threshold.

To evaluate the accuracy of the system (i.e. the percentage of decisions that are correct) we use the F-measure. F-measure is the harmonic mean of Precision and Recall.

$$Fmeasure = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

$$Precision = \frac{TP}{(TP + FP)} \qquad Recall = \frac{TP}{(TP + FN)}$$

TP (True Positive) means the signature is Genuine and it is truly considered Genuine because its transient has Hamming Distance ≤ Threshold.
FP (False Positive) means the signature is Genuine. But, it is considered as Forgery because its transient has hamming distance > Threshold. However, the hamming distance should have been ≤ Threshold because it is Genuine.
FN (False Negative) means the signature is Forgery. But, it is considered Genuine because its transient has Hamming Distance ≤ Threshold. However, the Hamming Distance should have been > Threshold because it is Forgery.

For the first row in Table 1, we have:
Precision = 3 / (3 + 0) = 1; Recall = 3 / (3 + 1) = 0.75
F-Measure = 2 * 0.75 * 1 / (0.75 + 1) = 0.857142 ~ 0.86

As another example, we take the third experiment (Table 1, User 3), we have:
Precision = 2 / (2 + 1) ~ 0.666667; Recall = 2 / (2 + 0) =1
F-Measure = 2 * 0.666667 * 1 / (0.666667 + 1) ~ 0.8

As we can see in Table 1, we run four experiments and calculate their F-measures, in order to estimate the accuracy of the system. The results show that the system is reliable in classifying time varying signals; which are Genuine and Forgery Online Signatures, by showing 86% of Accuracy on average.

VII. Discussion

Our approach and the basis of Cognitive Informatics:
Wang et al. (2013) say that "the internal information processing mechanisms and processes of the brain and the mind form the basis of Cognitive Informatics". They are the first to coin the term Cognitive Informatics in the International Conference on Cognitive Informatics (ICCI) in 2002. According to Wang et al., *"Cognitive informatics* studies the natural intelligence and the brain from a theoretical and a computational approach…" (Wang et al., 2013). Furthermore, they consider Cognitive Informatics as the third level in the hierarchical study of brain science where neuroinformatics and brain informatics are the preceding two levels. In our study, presented herewith, we approached the exploit of chaotic nonlinear dynamics in the goal of artificially imitating cognitive processes and their out coming brain

states, like learning a stimulus and performing its memory recall. This is achieved by representing the generalization of a set of stimulus as an orbit in the state space of a chaotic attractor. We call this orbit a Hypothesis and we consider it as a memory block.

> *"Specifically, we investigated the exploit of the theory of chaotic nonlinear dynamics via NTC and STDP, in order to imitate cognitive processes; like memory formation and recall, which are explained by the theory of chaotic neurodynamics"* (Aoun & Boukadoum, 2014)

In this regard, we refer to the discussion of Perlovsky & Kozma (2007) on NeuroCognition; where he describes brain processes based on chaotic neurodynamics, and we relate it to our theoretical and computational approach. Perlovsky says "A complex brain state is characterized by a trajectory over a chaotic attractor landscape. The system dynamics may reside for a brief time period in a localized attractor basin, before it transits to another basin. The presence of a given sensory stimulus may constrain the trajectory to a lower dimensional attractor basin. Once the sensory stimulus is removed, the trajectory switches to a higher-dimensional dynamics (less concrete, less conscious state) until the next sensory stimulus constrains the dynamics again (to a concrete perception)". This is well observed in the activity of our pool of NDS Neurons. First, when they are faced with a stimulus, they converge to a single UPO in the NDS Neurons attractor phase space. Then, they are able to switch to another UPO when they are faced with another stimulus. The UPO represents the concrete perception of the stimulus. The fact that NDS Neurons have an infinite number of UPOs in their dynamic reservoir (Crook, Goh & Hawarat, 2005) makes them favorable in modeling such form of cognition. Accordingly, our approach settles in the context of Cognitive Informatics (Wang et al., 2013) and its objectives in brain science studies.

Spike Coding, Periodicity, Synchronization and Modeling Attention as Consciousness to Stimuli:

Pulse Frequency Modulation (PFM) is a "signal transformation function that converts an analog sensory signal into a sequence of impulses where the frequency of pulses per second (pps) is proportional to the intensity of the stimuli" (Wang, 2013). In reference to Wang (2013), *'The Pulsed Frequency Modulation Principle of Neurology'* (PFMPN) states that the "general form of neural signals in the central nervous system (CNS) and Peripheral Nervous System (PNS) is uniformed PFM" (Wang, 2013). PFMPN falls under the umbrella of the traditional rate-coding paradigm of neural coding. This is because PFMPN considers the classical and trivial interpretation of the response of a neuron to a stimulus, based on the rate of the neuron's spikes output firing activity that is being generated due to the intensity of the presented stimulus (i.e. PFMPN solely considers the frequency of the neuron's spikes that are being fired in a specific time window, relatively to the amplitude of the electrical current caused by the corresponding stimulus). For instance, the interpretation of the mean firing rate of spikes provides the strength of the input stimulus being presented to the neuron. Notwithstanding the fact that rate-coding paradigm is a convenient and trivial neural coding methodology which is classically considered in the understanding, depiction and analysis of neural information processing. Above and beyond, we acknowledge that synchrony and correlation of spike timing in a population of neurons could "convey information beyond the firing rate" (Mass & Bishop, 1999). For instance, periodic signaling and oscillations serve as a reference signal and are found in the hippocampus and the olfactory cortex where the

oscillations repeat periodically while there is no change in the input stimulus. These periodic patterns of spikes are driven by a background oscillation (similar to NDS Neurons' behavior in the pool). This assumes that "the reference signal is not a single event, but a periodic signal" (Mass & Bishop, 1999). For instance, it was shown that the spatial location of a rat is correlated with phase spikes during oscillations that occur in the hippocampus (Mass & Bishop, 1999; O'Keefe & Recce, 1993).
We refer to Izhikevich (2006): "Synchronization ... should be so rare and difficult to occur by chance that when it happens, even transiently in a small subset of the network, it would signify something important, something meaningful, e.g., a stimulus is recognized, two or more features are bound, attention is paid" (Izhikevich, 2006). Thus, synchronization with exact spikes timing correlations is very adequate to neural information processing and could delineate higher level of cognitive processing capabilities like 'attention as consciousness to stimuli'. This makes the synchronization process presented herein this work very relevant.

On Chaos:
It was conceived that eye movements are indicators of cognitive mechanisms that interlay thinking, perception (Wang, 2014). Furthermore, their occurrence is essential for the manifestation of Long Term Memory (LTM) (Wang, 2014). Hence, it was shown that decreased nonlinear complexity of Electroencephalography (EEG) time series and their chaotic activity in Rapid Eye Movement (REM) sleep, indicates diminished ability of humans (as is the case of patients with schizophrenia) to process information (Keshavan, Cashmere, Miewald & Yeragani, 2004). Thus, chaotic activity; analyzed through the dynamics of population of neurons, is an important feature that characterizes the performance of cognitive functions. This strengthens the assumption that Chaos "may be the chief property that makes the brain different from an artificial-intelligence machine" and "Chaos constitutes the basic form of collective neural activity for all perceptual processes and functions as a controlled source of noise, as a means to ensure continual access to previously learned sensory patterns, and as the means for learning new sensory patterns..." (Skarda & Freeman, 1987).

Emulating chaotic neurodynamics:
Emulating chaotic neurodynamics is a fundamental strategy in the design of new models of artificial neural networks, and it will have enormous benefits in enhancing neural information processing, particularly memory storage and retrieval. The current paradigm to model the human memory relies on learning with prototypes. Then, anything that deviates from the learned prototypes during the recall stage is attributed to noise and it is hoped that the generalization capability of neural networks will be able to filter it out. However, prototypes are not natural in real life, and learning is accomplished with exemplars. Whether the mind extracts prototypes from the exemplar and how, if so, is still an open question. Nonlinear dynamics might provide the answer. For instance, strange attractors in a chaotic neural network could model the memory process not as one where specific prototypes are learned, but rather as one where all the exemplars associated with a memory recall are stored in the orbit of the strange attractor, the latter serving as "global prototype". In this way of viewing the memory processed, the notion of single point prototypes disappears and exemplars are no longer prototypes corrupted with noise; instead, they are the actual building blocks of memory.

We support this memory-processing hypothesis by referring to Freeman (1994) theory of chaotic neurodynamics. Freeman suggests that memory states are based on strange attractors. Specifically, he gives as an example the wings of the Lorenz Attractor (Lorenz, 1963) to be considered as memory states.

Freeman postulates:

"It is the Hebbian nerve cell assembly that provides the basis for generalization from examples to significant classes of input, and that guides the whole bulb during a state transition, when the bulb has been destabilized by input... The altered state is manifested by a brief aperiodic oscillation called a "burst" that constitutes the cortical response to the stimulus... One way to view the operation of responding to input is to postulate that the olfactory system has a global attractor that resembles a Lorenz attractor in having wings, but instead of two it has many wings, one for each class of input that it can discriminate" (Freeman, 1994).

In our case, the global attractor resides in the dynamics of a network of NDS Neurons, where each of its wings is a UPO. This UPO is a neural code; it is considered as a memory block and represents a class of inputs. Inputs are generalized using Aoun (2010) model of STDP, which is a form of Hebbian plasticity. They are processed using NTC (Crook, 2007), which is a form of computing on perturbations.

VIII. Conclusion

In this work, we presented enhancements to our model of STDP within NDS Neurons (Aoun, 2010). Also, we showed that this model allow these neurons to stabilize their connections' weights, so they can perform Nonlinear Transient Computation (NTC) (Crook, 2007) and learn time varying signals (e.g. Signature). A network of chaotic spiking neurons called Nonlinear Dynamic State (NDS) Neurons was able to learn the genuine signature of a person and detect its forgeries.

The signature example is a one-class classification problem. For multiple classes, we will have multiple Hypotheses. In consequence, a different readout mechanism should be considered, like a Naïve Bayes classifier.

Finally, a NDS Neuron has a theoretically infinite storage capacity (Crook, Goh & Hawarat, 2005, 2007), which is a wide range of Unstable Periodic Orbits (UPOs), considered as neural states that it can settle onto. In our case, a neural state is the manifestation of a dynamic signature. We used a network of 6 neurons only to learn and classify genuine versus forgery signatures, coded as input spikes patterns (i.e. bitstream), each of length 200. However, a Multi Layer Perceptron would have required a minimum of 401 neurons (200 for the input layer, 200 for the hidden layer and 1 neuron for the output layer)... This emphasizes the hypothesis of using NDS Neurons to optimize the memory capacity of artificial neural networks (Crook, Goh & Hawarat, 2005, 2007).

IX. Acknowledgment

Thanks to Prof. Nigel Crook from Oxford Brookes University for hints and communication on Nonlinear Transient Computation. Thanks to Chantal Khalaf for her

support and technical assistance. Thanks to M.D.

REFERENCES

Alhawarat, M. (2007). *Learning and memory in chaotic spiking neural models* (Ph.D. thesis). Oxford Brookes University.

Alhawarat, M. (2015). Analysis of the Dynamics of a Nonlinear Neuron Model. *Applied Mathematical Sciences*, Hikari Publishing, 9(126), 6295-6315.

Alhawarat, M., Nazih, W., Eldesouki, M. (2013a). Analysis of a chaotic spiking neural model: The NDS neuron. In *Proceedings of the Third International Conference on Advances in Computing and Information Technology*. (pp. 109 – 121). http://dx.doi.org/10.5121/csit.2013.3412

Alhawarat, M., Nazih, W., Eldesouki, M. (2013b). Studying a chaotic spiking neural model. *International Journal of Artificial Intelligence and Applications*, 4(5), 107 - 119. http://dx.doi.org/10.5121/ijaia.2013.4508

Alhawarat, M., Scheper, T., Crook, N.T. (2014). Investigation of a chaotic spiking neuron model. *International Journal of Computer Applications*, 99(17), 1 - 8. http://dx.doi.org/10.5120/17462-8258

Aoun, M.A. (2010). STDP within NDS neurons. In *Proceedings of the 7th International Symposium on Neural Networks* (ISNN), *Shanghai, China*, Printed in Lecture Notes in Computer Science (LNCS 6063) - Advances in Neural Networks Part I., Liqing Zhang, Bao-Liang Lu, and James Kwok (Eds.), Germany: Springer-Verlag Berlin Heidelberg, 2010, pp. 33-43.

Aoun, M.A., Boukadoum, M. (2014). Learning algorithm and neurocomputing architecture for NDS Neurons. In *IEEE 13th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC 2014)*, London, UK (pp. 126-132). doi: 10.1109/ICCI-CC.2014.692145

Crook, N., Goh, W.J, Alhawarat, M. (2007). Pattern recall in networks of chaotic neurons. *Biosystems*, 87(2-3), 267–274.

Crook, N., Goh, W.J., Alhawarat, M. (2005). The nonlinear dynamic state neuron. In *ESANN 2005: Proceedings of the 13th European Symposium on Artificial Neural Networks*, Bruges, Belgium, Verleysen, M. (ed.), Belgium: D-side Publishing, (pp. 37–42).

Crook, N.T. (2007). Nonlinear transient computation. *Neurocomputing*, 70(7-9), 1167–1176.

Freeman, W.J. (1994). Neural Networks and Chaos. *Journal of Theoretical Biology*, 171(1), 13–18.

Izhikevich, E.M. (2006). Polychronization: Computation with Spikes. *Neural Computation*, 18(2), 245–282.

Jaeger, H., Maass, W., Principe, J. (2007). Editorial: Special issue on echo state networks and liquid state machines. *Neural Networks*, 20(3), 287-289.

Jain, A., Griess, F., Connell, S. (2002). On-line signature verification. *Pattern Recognition*, 35(12), 2963–2972.

Keshavan, M.S., Cashmere, J.D., Miewald, J., Yeragani, V.K. (2004). Decreased nonlinear complexity and chaos during sleep in first episode schizophrenia: a preliminary report. *Schizophrenia Research*, 71(2-3), 263-272.

King, R.A (2004). Waveform Coding Method. *United States Patent No: US6,748,354 B1*.

Langton, C.G. (1990). Computation at the edge of chaos. *Physica D*, 42.

Legenstein, R., Maass, W. (2006). *New directions in statistical signal processing: From systems to brain*. In S. Haykin, J. C. Principe, T. Sejnowski, & J. McWhirter (Eds.). Cambridge: MIT Press, (pp. 127–154).

Legenstein, R., Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3), 323-334, ISSN 0893-6080, http://dx.doi.org/10.1016/j.neunet.2007.04.017.

Lei, H., Govindaraju, V. (2005). A comparative study on the consistency of features in on-line signature verification. *Pattern Recognition Letters*, 26(15), 2483–2489.

Lorenz, E.N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2), 130–141.

Lupu, E., Emerich, S., Beaufort, F. (2009). On-line signature recognition using a global features fusion approach. *Acta Technica Napocensis Electronics and Telecommunications*, 50(3), 13–20.

Maass, W., Legenstein, R. A., Bertschinger, N. (2005). Methods for estimating the computational power and generalization capability of neural microcircuits. In *L. K. Saul, Y. Weiss, & L. Bottou (Eds.), Advances in neural information processing systems*, 17 (pp. 865–872). Cambridge: MIT Press.

Maass, W., Markram, H. (2004). On the computational power of recurrent circuits of spiking neurons. *Journal of Computer and System Sciences*, 69(4), 593–616.

Maass, W., Markram, H. (2004). On the computational power of recurrent circuits of spiking neurons. *Journal of Computer and System Sciences*, 69(4), 593–616.

Maass, W., Natschläger, T., Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11), 2531–2560.

Mass, W., Bishop, C.M. (1999). *Pulsed Neural Networks*, MIT Press.

Natschläger, T., Bertschinger, N., Legenstein, R. (2004). At the Edge of Chaos: Real-time Computation and Self-Organized Criticality in Recurrent Neural Network. *Proceedings of NIPS* 2004, pp. 145-152.

O'Keefe, J., Recce M.L. (1993). Phase Relationship Between Hippocampal Place Units and the EEG Theta Rhythm. *Hippocampus*, vol.3, no. 3, pp. 317-330.

Perlovsky, L. I., Kozma, R. (2007). *Neurodynamics of Cognition and Consciousness Understanding Complex Systems*, L. I. Perlovsky and R. Kozma (Eds.), Springer-Verlag Berlin Heidelberg, (pp. 1–8).

Pyragas, K. (2003). Time-delayed feedback control method and unstable controllers. In *Proceedings of the 2003 International Conference on Physics and Control - Volume 2* (August 20 - 22, 2003). PHYCON, IEEE Computer Society, Washington, DC, vol. 2, pp. 456–467.

Rao, R.P, Sejnowski, T.J. (2001). Spike-Timing-Dependent Hebbian Plasticity as Temporal Difference Learning. *Neural Computation*, 13(10), 2221–2237.

Rössler, O.E. (1976). An equation for continuous chaos. *Physics Letters A*, 57(5), 397–398.

Skarda, C.A., Freeman W.J. (1987). How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, 10, 161–195.

Wang, Y. (2013). Formal Models and Cognitive Mechanisms of the Human Sensory System. *International Journal of Software Science and Computational Intelligence*, 5(3), 55–75.

Wang, Y. (2014). Unveiling the Cognitive Mechanisms of Eyes: The Visual Sensory Vs. the Perceptive Browser of the Brain. *International Journal of Cognitive Informatics and Natural Intelligence*, 8(1), 36-50.

Wang, Y., Fariello, G., Gavrilova, M.L., Kinsner, W., Mizoguchi, F., Patel, S., Patel, D., Pelayo, F.L., Raskin, V., Shell, D.F., Tsumoto, S. (2013). Perspectives on Cognitive Computers and Knowledge Processors. *International Journal of Cognitive Informatics and Natural Intelligence*, 7(3), 1–24.

Yeung, D.Y., Chang, H. Xiong, Y., George, S., Kashi, R., Matsumoto, T., Rigoll, G. (2004). SVC2004: First International Signature Verification Competition. In *Proceedings of the International Conference on Biometric Authentication (ICBA)*, Hong Kong.

BIBLIOGRAPHY

Aihara K., Takabe T. and Toyoda M. (1990). Chaotic neural networks. *Physics Letters A, Volume 144, Issues 6–7,* 333-340. (http://www.sciencedirect.com/science/article/pii/037596019090136C)

Aleksander, I. and Morton, H. (1995). *Introduction to Neural Computing*. 2nd Edition, International Thomson Computing Press.

Alhawarat, M.O.I. (2007) Learning and memory in chaotic spiking neural models. Ph.D. thesis, Oxford Brookes University.

Aoun, M. A. (2010). STDP within NDS neurons. *Advances in Neural Networks-ISNN 2010.* Springer Berlin Heidelberg, pp. 33-43.

Aoun, M. A. and Boukadoum, M. (2014). Learning algorithm and neurocomputing architecture for NDS Neurons. *Proceedings of 2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2014.*

Aoun, M. A. and Boukadoum, M. (2015). Chaotic Liquid State Machine. *International Journal of Cognitive Informatics and Natural Intelligence*. 9, 4, 1-20. http://dx.doi.org/10.4018/IJCINI.2015100101

Aubin D. and Dalmedico A.D. (2002). Writing the history of dynamical systems and chaos: Longue Duree and revolution, disciplines and cultures. *Academic Press, Historia Mathematica, 29(3), August 2002*, 273-339.

Bi, G., and Poo, M. (1998). Synaptic modifications in cultured hippoc-ampal neurons: dependence on spike timing, synaptic strength, and post- synaptic cell type. *J. Neurosci.* 18, 10464–10472.

Brette, R., & Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology, 94 5*, 3637-42.

Cajal, R. S. (1933). *Histology*, 10th ed., Wood, Baltimore.

Clopath C., Gerstner W. (2010). Voltage and spike timing interact in STDP – a unified model. *Front. Synaptic Neurosci. 2*: 25.10.3389/fnsyn.2010.00025

Crook, N.T., Goh, W.J., Hawarat, M. (2005). The nonlinear dynamic state neuron. In: Verleysen, M. (Ed.), *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN'2005)*, April, d-side, Belgium, 37 - 42.

David, E. and Greental, I. (2017). Genetic Algorithms for Evolving Deep Neural Networks arXiv.org > cs > arXiv:1711.07655. *Originally in ACM Genetic and Evolutionary Computation Conference (GECCO)*, 1451-1452, Vancouver, Canada, July 2014.

Eliasmith, C. (2003). Moving beyond metaphors: Understanding the mind for what it is. *Journal of Philosophy C(10),* 493-520.

Encyclopedia Britannica Online: http://www.britannica.com/eb/article-9022470

Freeman, W. J. (1991). The physiology of perception. *Scientific American*, vol. 264, no. 2, pp. 78-85.

Freeman, W. J. (1994). Neural Networks and Chaos. *Journal of Theoretical Biology*, 171 (1),13–18.

Gerstner W. and vanHemmen J.L. (1992). Associative memory in a network of 'spiking' neurons. Network: Computation in Neural Systems *3*(2), 139-164.

Gerstner W., VanHemmen J.L. and Cowan J.D. (1996). What matters in neuronal locking. *Neural Computation 8*(8), 1653-1676.

Gerstner W. (2000). Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking. Neural Computation *12*(1), 43-89.

Hebb, D.O. (1949). *The Organization of Behavior*. New York: Wiley & Sons.

Heeger, D. (2000). Poisson model of spike generation, *Handout, University of Standford,* 5.

Herbert, S. (1983). Why should machines learn?, *in R. S. Michalski, J. G. Carbonell & T. M. Mitchell, eds, "Machine learning: An Artificial Intelligence Approach", 1, Morgan Kaufmann Publishers, Inc.*

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA*, *79*(8) 2554–2558.

Ingraham, R. L. (1991). *A Survey of Nonlinear Dynamics: "chaos Theory".* World Scientific, ISBN 9810207778.

Izhikevich, E.M. (2001). Resonate-and-fire neurons. *Neural networks : the official journal of the International Neural Network Society, 14 (6-7)*, 883-94.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*, 1569–1572.

Izhikevich, E. M., Desai, N. S., Walcott, E. C., & Hoppensteadt, F. C. (2003). Bursts as a unit of neural information: selective communication via resonance. *Trends in Neurosciences, 26*(3), 161-167. doi:10.1016/s0166-2236(03)00034-1

Izhikevich, E. M. (2006). Polychronization: Computation with Spikes. *Neural Comput.* 18 (2), 245-282.

Izhikevich, E. M. (2010). *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. The MIT Press Cambridge, Massachusetts London, England.

Korn H. and Faure P. (2003). Is there chaos in the brain? II. Experimental evidence and related models. *Comptes Rendus Biologies*, 326(9), 787-840. https://doi.org/10.1016/j.crvi.2003.09.011. (http://www.sciencedirect.com/science/article/pii/S1631069103002002)

Larsen-Freeman D. (1997). Chaos/Complexity Science and Second Language Acquisition. *Applied Linguistics*, 18(2), 141-165. https://doi.org/10.1093/applin/18.2.141.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database. *AT&T Labs. Available online: http://yann.lecun.com/exdb/mnist.*

Lorenz, E. N.(1963). Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences, 20*, 130-141.

Maass, W., Schnitger, G. and Sontag, E. (1991). *On the computational power of sigmoid versus boolean threshold circuits.* Proc. of the 32nd Annual IEEE Symposium on Foundations of Computer Science.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, *10*(9), 1541–1741.

Maass, W. and Schmitt, M. (1997). On the Complexity of Learning for a Spiking Neuron. *Communications of the ACM, 54-61.*

Maass W. & Bishop C.M. (1999). *Pulsed Neural Networks*, MIT Press.

Maass, W., Natschlager, T. and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation. 14* (11), 2531–2560.

Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science 275*, 213–215.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *7,* 115–133.

Natschlaeger T., Maass W., and Markram H. (2002). The "liquid computer": A novel strategy for real-time computing on time series. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8 (1), 39-43.

Naud, R., Marcille, N., Clopath, C., & Gerstner, W. (2008). Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics, 99*, 335 - 347.

Price, K., Storn, R.M., Lampinen, J.A. (2005). *Diffentential evolution: a practical approach to global optimization*. Springer, Heidelberg.

Rieke, F., Warland, D., van Steveninck, R. R. and Bialek, W. (1997). *Spike: Exploring the Neural Code.* Computational Neurosciences. MIT Press, Cambridge.

Rössler, O. E. (1976). An equation for continuous chaos, *Physics Letters A*, *57*(5), 397-398,

Rumelhart, D. E., Hinton G. E. and Williams R. J. (1986). Learning Internal Representations by Error Propagation. *David E. Rumelhart, James L. McClelland, and the PDP research group*. (editors), *Parallel distributed processing: Explorations in the microstructure of cognition, 1*, MIT Press.

Shadlen, M. N., and Newsome, W. T. (1998). The variable discharge of cortical neurons: Implications for connectivity, computation and information coding. *Journal Neuroscience, 18*, 3870–3896.

Smale, S. (1998). Chaos: Finding a horseshoe on the beaches of Rio. *Mathematical Intelligencer, Springer-Verlag,* 20 (1), 39–44.

Smale, S. (1999). Mathematical problems for the next century. In V. I. Arnold, M. Atiyah, P. Lax, B. Mazur. *Mathematics: frontiers and perspectives. American Mathematical Society.* 271–294.

Song, S., Miller K.D. and Abbott L.F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat Neurosci.* (9). 919-926.

Tsuda, I. and Kuroda, S. (2001). Cantor coding in the hippocampus. *Japan Journal of Industrial and Applied Mathematics*, *8*(2), 249–258. https://doi.org/10.1007/BF03168573

Valadez-Godinez, S., Gonzalez, J. and Sossa, H. (2017). Efficient Pattern Recognition Using the Frequency Response of a Spiking Neuron. *Springer-Verlag, LNCS 10267*, 53–62.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27.*

Valiant, L. G. (2003).Three Problems in Computer Science. *Journal of the ACM*, Vol. *50*(1), 96-99.

Vázquez, R.A. (2010). Pattern Recognition Using Spiking Neurons and Firing Rates. *Springer-Verlag LNAI 6433*, 423–432.

Wang, H., Gerkin, R. C., Nauen, D. W., & Bi, G. (2005). Coactivation and timing-dependent

integration of synaptic potentiation and depression. *Nature Neuroscience Nat Neurosci, 8*(2), 187-193. doi:10.1038/nn1387

Wang, Y., Fariello, G., Gavrilova, M.L., Kinsner, W., Mizoguchi, F., Patel, S., Patel, D. Pelayo, F.L., Raskin, V., Shelland D.F. and Tsumoto, S. (2013). Perspectives on Cognitive Computers and Knowledge Processors. *International Journal of Cognitive Informatics and Natural Intelligence*, *7*(3), 1–24.

Weisstein, E. W. Dynamical System. From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/DynamicalSystem.html.

Yang, D. and Poo, M.M. (2006) Spike Timing-Dependent Plasticity: From Synapse to Perception. *Physiol Rev, 86*, 1033–1048. doi:10.1152/physrev.00030.2005