Unconventional Computing Using Memristive Nanodevices: From Digital Computing to Brain-like Neuromorphic Accelerator

Mahyar Shahsavari

Mahyar SHAHSAVARI
Lille University of Science and Technology

Doctoral Thesis

Unconventional Computing Using Memristive Nanodevices: From Digital Computing to Brain-like Neuromorphic Accelerator

Supervisor: Pierre BOULET

Thèse en vue de l'obtention du titre de
Docteur en
INFORMATIQUE
DE L'UNIVERSITÉ DE LILLE
École Doctorale des Sciences Pour l'Ingénieur de Lille - Nord De France

14-Dec-2016

Jury:

Hélène Paugam-Moisy	Professor of University of the French West Indies (Université des Antilles), France (Rapporteur)
Michel Paindavoine	Professor of University of Burgundy (Université de Bourgogne), France (Rapporteur)
Virginie Hoel	Professor of Lille University of Science and Technology, France(Président)
Said Hamdioui	Professor of Delft University of Technology, The Netherlands
Pierre Boulet	Professor of Lille University of Science and Technology, France

Université de Lille

CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille

IRCICA

Acknowledgment

Firstly, I would like to express my sincere gratitude to my advisor Prof. Pierre Boulet for all kind of supports during my PhD. I never forget his supports specially during the last step of my PhD that I was very busy due to teaching duties. Pierre was really punctual and during three years of different meetings and discussions, he was always available on time and as a record he never canceled any meeting. I learned a lot form you Pierre, specially the way of writing, communicating and presenting. you were an ideal supervisor for me.

Secondly, I am grateful of my co-supervisor Dr Philippe Devienne. As I did not knew any French at the beginning of my study, it was not that easy to settle down and with Philippe supports the life was more comfortable and enjoyable for me and my family here in Lille. I have never forgotten our joyful time in concert and playing violin with ELV music group, your delicious foods and cakes, visiting London, Vienna, Kermanshah and Tehran all nice moments we spended with Philippe. With Philippe support we started a scientific collaboration with Razy University at my hometown city Kermanshah. We got the Gundishapour grants for two years collaboration between Lille 1 and Razy Universities In addition to signing a MoU for long term collaboration between two universities all thanks to Philippe efforts.

I would like to acknowledge and thanks to my thesis committee: Prof. Hélène Paugam-Moisy, Prof. Michel Paindavoine, Prof. Said Hamdioui and the president of the jury, Prof. Virginie Hoel for their insightful comments and encouragement, and reviewing my thesis.

My sincere thanks also goes to CRIStAL lab and Émeraude team colleagues, prof. Giuseppe Lipari, Dr Richard Olejnik, Dr Clément Ballabriga and specially to Dr Julien Forget. I will never forget Forget supports specially for the first time teaching at Polytech Lille. I appreciate my Émeraude team friends Antoine Bertout, Khalil Ibrahim Hamzaoui, Houssam Zahaf, Pierre Falez and Yassine sidlakhdar for nice discussions, coffee drinking and playing football together. I would like to thanks to our colleague Dr Fabien Alibart in IEMN lab for his technical consultancy during my research particularly during developing new synapse. I gratefully acknowledge my previous friends and colleagues in TUDelft in The Neterlands, my lovely and kind friends Faisal Nadeem, Arash Ostadzadeh, Mahmood Ahmadi that shared with me valuable knowledge and information. Thanks again to Prof. Said Hamdioui that we started working on memristor together in CE group at TUDelft.

Special thanks go to my Iranian friends in Lille Farzan, Hamid, Hamidreza, Ehsan, Babak, and Sina thanks for bing there for me. Eric and Leopoled, my lovely officemates and friends, I never forget your kindnesses in our calm office in M3 building. I am grateful of my Iranian friends and colleagues at Razi University, Prof. Mohsen Hayati Dr Mahmood Ahmadi, Dr Arash Ahmadi and one of my best friend Mazdak Fatahi that we started working on neural network research for his Master thesis. We have done many skype meetings that both of us learned a lot during those scientific discussions. Actually the last chapter of my thesis is related to his master thesis topic. I appreciate my father in law Prof. Mohammad Mehdi Khodaei for his guidance during my research as well as supporting us for starting collaboration between Razi and Lille 1 universities. I really Dr Mahmood Ahmadi, and Mazdak Fatahi for their supports and being kindly present specially during the time our French professors and colleagues

visited Kermanshah and Razi university. Without Mahmood helps and supports this collaboration was not feasible.

I am very grateful of my close family in Iran, my kind father, my lovely mother and two supportive brothers Mehdi and Mahziar and my little nephews Abolfazl and Ali who always prayed for me and encouraged me continuously not only during my PhD but also in whole my life.

Last, but not the least, I would like to express my appreciation to my better-half Hanieh, actually Dr Hanieh Khodaei. She was a kind wife, the best friend that we shared whole the beautiful, sad and even stressful moments of our PhD together. Thanks for all those understanding and supports. The best gift during my PhD was from my Almighty, Hana was my best gift, thanks God. Sorry Hana that I consisted part of the time that I should have played with you to my thesis and research.

Be yari Parvardegar Yekta
Mahyar Shahsavari,
December 2016

Abstract

By 2020, there will be 50 to 100 billion devices connected to the Internet. Two domains of hot research to address these high demands of data processing are the Internet of Things (IoT) and Big Data. The demands of these new applications are increasing faster than the development of new hardware particularly because of the slowdown of Moore's law. The main reason of the ineffectiveness of the processing speed is the memory wall or Von Neumann bottleneck which is comming from speed differences between the processor and the memory. Therefore, a new fast and power-efficient hardware architecture is needed to respond to those huge demands of data processing.

In this thesis, we introduce novel high performance architectures for next generation computing using emerging nanotechnologies such as memristors. We have studied unconventional computing methods both in the digital and the analog domains. However, the main focus and contribution is in Spiking Neural Network (SNN) or neuromorphic analog computing. In the first part of this dissertation, we review the memristive devices proposed in the literature and study their applicability in a hardware crossbar digital architecture. At the end of part I, we review the Neuromorphic and SNN architecture. The second part of the thesis contains the main contribution which is the development of a Neural Network Scalable Spiking Simulator (N2S3) suitable for the hardware implementation of neuromorphic computation, the introduction of a novel synapse box which aims at better learning in SNN platforms, a parameter exploration to improve performance of memristor-based SNN, and finally a study of the application of deep learning in SNN.

Résumé

On estime que le nombre d'objets connectés à l'Internet atteindra 50 à 100 milliards en 2020. La recherche s'organise en deux champs principaux pour répondre à ce défi : l'internet des objets et les grandes masses de données. La demande en puissance de calcul augmente plus vite que le développement de nouvelles architectures matérielles en particulier à cause du ralentissement de la loi de Moore. La raison principale en est est le mur de la mémoire, autrement appelé le goulet d'étranglement de Von Neumann, qui vient des différences de vitesse croissantes entre le processeur et la mémoire. En conséquence, il y a besoin d'une nouvelle architecture matérielle rapide et économe en énergie pour répondre aux besoins énormes de puissance de calcul.

Dans cette thèse, nous proposons de nouvelles architectures pour les processeurs de prochaine génération utilisant des nanotechnologies émergentes telles que les memristors. Nous étudions des méthodes de calcul non conventionnelles aussi bien numériques qu'analogiques. Notre contribution principale concerne les réseaux de neurones à impulsion (RNI) ou architectures neuromorphiques. Dans la première partie de la thèse, nous passons en revue les memristors existants, étudions leur utilisation dans une architecture numérique à base de crossbars, puis introduisons les architectures neuromorphiques. La deuxième partie contient la contribution principale : le développement d'une simulateur d'architectures neuromorphiques (N2S3), l'introduction d'un nouveau type de synapse pour améliorer l'apprentissage, une exploration des paramètres en vue d'améliorer les RNI, et enfin une étude de la faisabilité des réseaux profonds dans les RNI.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The two most important demands of humans using ICT devices and technologies are becoming two hottest topic of research in computer science and engineering namely Big Data and Internet of Things (IoT). In both domains, the way of processing data is quite important. In Big Data the clustering, classification, and prediction are not avoidable to use and process the data efficiently. In IoT, the smart devices are connected to other smart devices using different sensors. Machine learning recently proposed promising solution for processing data in these two domains. By 2020, there will be 50 to 100 billion devices connected to the Internet, ranging from smartphones, PCs, and ATMs (Automated Teller Machine) to manufacturing equipment in factories and products in shipping containers [7]. For the Big Data or sensory data not only an efficient processing algorithm is necessary but also finding a new fast, parallel and power-efficient hardware architecture is unavoidable.

Machine learning algorithms are widely used for data classification in software domain and using conventional hardware platform. These algorithms on nowadays computers consume a remarkable time and energy. The reason is that in conventional computing, the way of communicating between memory and central processing unit (CPU) is not efficient. The memory wall or Von Neumann memory bottelneck is the growing disharmony of communication speed between the CPU and memory outside the CPU chip. An important reason for this disharmony is the restricted communication bandwidth beyond chip boundaries, which is referred to as bandwidth wall as well. The CPUs access both data and program in memory using the same shared resources. Finally, CPUs spend most of their time idle.

Using emerging technologies such as memristors [1], there is possibility of performing both information processing and storing the computational results on the same physical platform [8]. Memristors have the potential to be a promising device for novel paradigms of computation as well as new generation of memory. The characteristics of memristor are promising to design a processing unit with local access memory rather than non-local and shared memory. The new high performance architecture for next generation of computation regarding to emerging technologies could be in logic or analog domain. Memristors have potential for both digital and analog paradigms of computations.

Another novel alternative architecture suitable for neural network and machine learning algorithms is proposed as Spiking Neural Network (SNN) system which is known as Neuromorphic architecture too. SNN is the known way to realize the neural network software abilities on a hardware architecture. The mammalian nervous system is a network of extreme complexity which is able to perform cognitive computation in a parallel and power-efficient manner. Understanding the principles of brain processing for computational modeling is one of the biggest challenges of the 21st century that led to the new branch of research e.g., neuromorphic computing. Neuromorphic engineering represents one of the promising fields for developing new computing paradigms complementing or even replacing current Von Neumann architecture [9]. The requirements for implementing a SNN architecture (neuromorphic) are providing electronic devices that can mimic the biological neural network components such as neurons and synapses. The Spiking neural model is an electrical model of physiological neuron

that has been implemented on the circuit using state-of-the-art technologies e.g., CMOS transistors or on low-power CMOS design using subthreshold regime transistor [10]. The synapse in biological neural network reacts as a plastic connection controller between two neurons. Recently, emerging devices in nano-scale have demonstrated novel properties for making new memories and Artificial synapse. One of those is the memristor that was hypothetically presented by Leon Chua in 1971 [11] and after a few decades, HP was the first to announce the successful memristor fabrication [1]. The unique properties in memristor nano-devices such as, extreme scalability, flexibility because of analog behavior, and ability to remember the last state make the memristor a very promising candidate to apply it as a synapse in Spiking Neural Network (SNN) [12].

1.2 Part I:Motivation, state-of-the-art and application of using emerging nanodevices for unconventional computing

The first part of this dissertation contains the mathematic and physical model of memristor, memristive nanodevice technologies, as well as different applications of this emerging technology. Memristor can remember its last state after the last power plugging and has a simple physical structure, high-density integration, and low-power consumption. These features make the memristor an attractive candidate for building the next generation of memories [13–15], an artificial synapse in Spiking Neural Network architectures [12, 16], and as a switch in crossbar array configurations [17]. Different device structures are still being developed to determine which memristor device can be presented as the best choice for commercial use in memory/flash manufacturing or in neuromorphic platforms. This is based on different factors such as size, switching speed, power consumption, switching longevity, and CMOS compatibility. A comprehensive survey particularly on recent research results and recent development of memristive devices seems be quite useful for future research work and developments. To better understanding how memristor can restore the data and how it could be flexible to modify the conductances to be replaced as a synapse, we have modeled the behavior of this device. In addition, we perform a classification of the devices based on the manufacturing technology. In this classification, we discuss different characteristics of various devices. We study the potential applications of a memristor built using specific technology. The advantages and disadvantages of each class of memristor with various type of device materials are discussed. Furthermore, we discuss potential applications of memristor nanodevice in nonvolatile memories such as RRAM, PCM, CBRAM, FeRAM and MRAM, digital computation, analog and neuromorphic domains.

In the second chapter, we discuss two methods for unconventional digital computing by memristive two-terminal devices [18]. Two main digital computation approaches, namely material implication (IMP) or stateful logic [19, 20] and programmable crossbar architecture [21, 22] are studied in this chapter. By applying memristor as a digital switch, a high memristance (memristor resistance) is considered as logic '0' and a low memristance is considered as logic '1'. First and foremost, based on the previous research works on IMP, we establish a functionally complete Boolean operation to be able to build all standard logic gates. Subsequently, building the digital gates has been performed using programmable crossbar architectures. Programmable crossbar architectures have been proposed as a promising approach for future computing architectures because of their simplicity of fabrication and high density, which support defect tolerance. At the end of Chapter 3, the comparison of two methods in digital computation is presented.

In the last chapter of part I, the basic definition of neuromorphic or Spiking Neural Network (SNN) architecture have been introduced. Neuromorphic computing has the potential to bring very low power computation to future computer architectures and embedded systems [23]. The main remarkable difference between conventional Von Neumann architecture and neuromorphic systems is in their use of memory structures. The way of communication between memory and central processing unit (CPU) in conventional computing is not very efficient which is known as Von Neumann memory bottelneck. CPUs spend most of their time idle because the speed difference between the CPU and memory. The solution that has been applied in Von Neumann architecture is memory

hierarchy. In other words, a limited amount of fast but costly memory sit closer to the processing unit, while most of the data is stored in a cheaper but larger memory. By proposing computing unit next to the local memory, neuromorphic brain-inspired computing paradigms offer an attractive solution for implementing alternative non von Neumann architectures, using advanced and emerging technologies [9,24]. Artificial neural network (ANN) is a mathematical model of the network of neurons in mammalian brain though SNN is an electronic hardware neuromorphic model of biological brain. SNNs provide powerful tools to emulate data processing in the brain, including neural information processing, plasticity and learning. Consequently, spiking networks offer solutions to a broad range of specific problems in applied engineering image detection, event detection, classification, speech recognition and many cognitive computation domain applications.

Neurons communicate together using spikes. In Chapter 4, we review different ways of coding data to spikes known as spike information coding methods. Furthermore, we study the network topologies and configurations. The interconnection among units can be structured in numerous ways resulting in numerous possible topologies. Two basic classes are defined: Feed-Forward Neural Networks (FFNN) and Recurrent (or feedback) Neural Networks (RNN) depicted in Figure 4.3. We add a discussion in more modern neural networks such as Convolutional Neural Networks (CNN) [25], and Deep Belief Networks (DBNs) [26]. Moreover, various spiking model of neurons as dynamic elements and processing units are reviewed. We discuss which model we have used in our platform and why we choose this model. Thanks to the plasticity property of synapse, in neural network system, we can basically say the synapse is where the learning happens. Therefore, in this chapter both synapse and learning are studied. Additionally, we discussed various classes of learning algorithms as well as a comprehensive study of Spike-Timing Dependent Plasticity (STDP) [27, 28] is presented. This comprehensive study includes presenting different models for STDP learning based on single or multiple pre- or postsynaptic spikes occurring across a synapse in an interval of time. Finally in Chapter 4, we applied lateral inhibition which is winner-take-all (WTA) [29, 30] in our platform as well as homeostasis as a method of neuron adaptation of learning.

1.3 Part II:Our contribution in spiking neural network architecture: Simulator, New synapse box, Parameter exploration and Spiking deep learning

The second part of the thesis consists of our contributions to neuromorphic computing and SNN architecture. In Chapter 5, we present and develop N2S3 (for Neural Network Scalable Spiking Simulator), an open-source event-driven simulator that is built to help design spiking neuromorphic circuits based on nanoelectronics. It is dedicated to intermediate modeling level, between low-level hardware description languages and high-level neural network simulators used primarily in neurosciences as well as the integration of synaptic memristive device modeling, hardware constraints and any custom features required for the target application. N2S3 has been developed from the ground up for extensibility, allowing to model various kinds of neurons and synapses, network topologies, learning procedures, reporting facilities, and to be user-friendly, with a domain specific language to easily express and run new experiments. For experimental set up, N2S3 is distributed with the implementation of two "classical" experiments: handwritten digit recognition on the MNIST dataset [25, 31] and the highway vehicle counting experiment [32].

In Chapter 6, with the combination of a volatile and a nonvolatile memristor we introduce and design a new artificial synapse box that can improve learning in spiking neural networks architectures [33]. This novel synapse box is able to forget and remember by inspiration from biological synapses. The nonvolatility is a unique property in memristor nanodevice to introduce it as a promising candidate in building next generation of non-volatile memory. However, by inspiring of physiological synapse, a synapse that can forget unimportant data (non-frequent spikes) and remember significant data (frequent spike trains) can support network to improve learning process.

Thanks to close collaboration with the nano-electronics research center in the University of Lille (IEMN), we have had the opportunity of studying the suitability of different types of memristors (TiO_2 , NOMFET, magnetoresistive, magnetoelectric) to build a spiking neural network hardware platform. To add the forgetting property to the synapse box, we have used a volatile memristor named NOMFET (Nanoparticle Organic Memory Field-Effect Transistor). We have merged NOMFET with a nonvolatile solid-state meristor nanodevice. At the end of this chapter, we evaluate the synapse box proposal by comparing it with a single non-volatile memory synapse by simulation on the MNIST handwritten digit recognition benchmark.

Specific application domains such as Big Data classification, visual processing, pattern recognition and in general sensory input data, require information processing systems which are able to classify the data and to learn from the patterns in the data. Such systems should be power-efficient. Thus researchers have developed brain- inspired architectures such as spiking neural networks. In Chapter 7, we surveyed brain- inspired architectures approaches with studying well-known project and architecture in this neuromorphic domain. For large scale brain-like computing on neuromorphic hardware we have recognized four approaches:

- Microprocessor based approaches where the system can read the codes to execute and model the behavior of neural systems and cognitive computation such as the SpiNNaker machine [34].

- Fully digital custom circuits where the neural system components are modeled in circuit using sate-of-the-art CMOS technology e.g., IBM TrueNorth machine [23].

- Analog/digital mixed-signal systems that model the behavior of biological neural systems, e.g. the NeuroGrid [35] and BrainScales [36] projects.

- Memristor crossbar array based systems where the analog behavior of the memristors emulate the synapses of a spiking neural network [37].

We have studied large scale brain-like architectures such as SpiNNaker, IBM TrueNorth, NeuroGrid, and BrainScales to recognize the cons and pros of these projects to have our optimized evaluation and exploration of required items and parameters for neuromorphic and brain-like computation platforms.

Spiking neural networks are widely used in the community of computational neuroscience and neuromorphic computation, there is still a need for research on the methods to choose the optimum parameters for better recognition efficiency. Our main contribution in Chapter 7 is to evaluate and explore the impact of several parameters on the learning performance of SNNs for the MNIST benchmark: the number of neurons in the output layer, the duration of the STDP window, various thresholds for adaptive threshold neurons, different distributions of spikes to code the input images and the memristive synapse fitting parameter. This evaluation has shown that a careful choice of a few parameters can significantly improve the recognition rate on this benchmark.

Deep learning is currently a very active research area in machine learning and pattern recognition society due to its potential to classify and predict as a result of processing Big data and Internet of Things (IoT) related data. Deep Learning is a set of powerful machine learning methods for training deep architectures. Considering the inherent inefficiency of learning methods from traditional Artificial Neural Networks in deep architectures, Contrastive Divergence (CD) has been proposed to train Restricted Boltzmann Machines (RBM) as the main building blocks of deep networks [38].

In Chapter 8 deep learning architectures are introduced in SNN systems. In SNN, neurons communicate using spikes [39]. Therefore, we have to design an architecture of neurons that are able to implement spiking data. Consequently, we present a framework for using Contrastive Divergence to train an SNN using RBM and spiking neurons. The obtained recognition rate or network accuracy of the architecture using CD algorithms for learning was 92.4%. This framework can open a new window toward the neruromorphic architecture designer to apply the state-of-the-art of machine learning learning algorithm in SNN architecture.

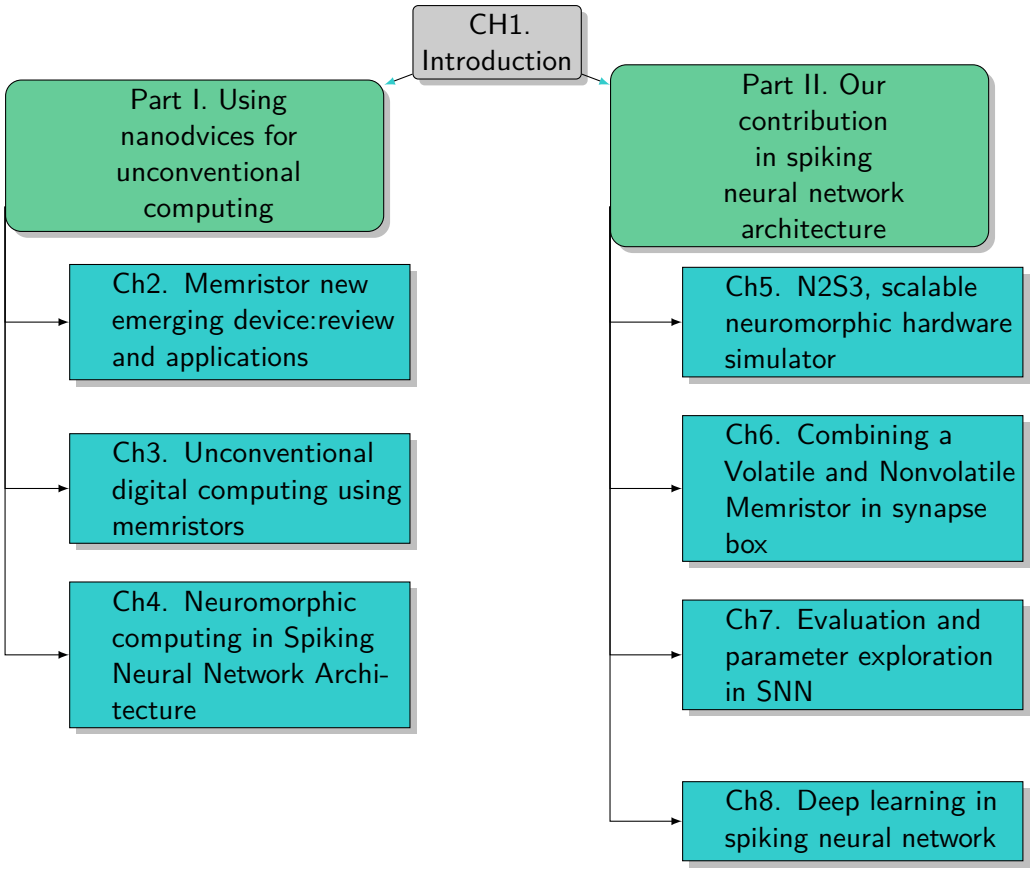

Figure 1.1: General overview of the manuscript.

1.4 Manuscript outline

This manuscript presents a spiking neural network platform suitable to hardware implementation with a focus on learning in SNN. Therefore in the first two chapters, we review cons and pros of memristor and memristive-based computation. The different models and technologies of memristor beside the applications is presented in chapter 2. The third chapter consists of using memristors in the unconventional digital manner. In the chapter 4, the principals of neuromorphic and spiking neural network architecture is described. Our contribution in designing and presenting spiking neural network architecture is presented in the second part of this manuscript. First of all, in the contribution part we present our neuromorphic hardware simulator. Second, we propose an artificial synapse box to improve learning by combining volatile and nonvolatile types of memristor devices. Furthermore, in Chapter 7, we discuss different parameters in SNN architecture and evaluate the effect of each parameters in learning. In Chapter 8, deep learning in spiking neural network is presented. Finally the conclusion and potential future work are explained. A general overview of this dissertation is shown in Figure 1.1.

Part I

Motivation, state-of-the-art and application of using emerging nanodevices for unconventional computing

Chapter 2

Memristor nanodevice for unconventional computing: review and applications

Abstract

A memristor is a two-terminal nanodevice. Its properties attract a wide community of researchers from various domains such as physics, chemistry, electronics, computer and neuroscience. The simple structure for manufacturing, small scalability, nonvolatility and potential of using in low power platforms are outstanding characteristics of this emerging technology. In this chapter, we review a brief literature of memristor from mathematic model to the physical realization and different applications. We discuss different classes of memristors based on the material used for its manufacturing. The potential applications of memristor are presented and a wide domain of applications are classified.

2.1 Introduction

Memristor has recently drawn the wide attention of scientists and researchers due to non-volatility, better alignment, and excellent scalability properties [40]. Memristor has initiated a novel research direction for the advancement of neuromorphic and neuro-inspired computing. Memristor remembers its last state after the last power plugging and has a simple physical structure, high-density integration, and low-power consumption. These features make the memristor an attractive candidate for building the next generation of memories [13]. In addition, from high-performance computing point of view, the memristor has the potential capability to conquer the memory bottleneck issue, by utilizing computational unit next to the memory [8]. Due to these unique properties and potentials of the memristor, neuroscientists and neuromorphic researchers apply it as an artificial synapse in Spiking Neural Network (SNN) architectures [12].

Memristor was predicted in 1971 by Leon Chua, a professor of electrical engineering at the University of California, Berkeley, as the fourth fundamental device [11]. Publishing a paper in the Nature journal by Hewlett Packard (HP) [1] in May 2008, announced the first ever experimental realization of the memristor, caused an extraordinary increased interest in this passive element. Based on the symmetry of the equations that govern the resistor, capacitor, and inductor, Chua hypothesized that fourth device should exist that holds a relationship between magnetic flux and charge. After the physical discovery of the memristor, several institutions have published the memristor device fabrications using a variety of different materials and device structures [1, 2, 41–43].

In 2009, Biolek *et al.* modeled nonlinear dopant drift memristor by SPICE [44]. One year later, Wei Lu, professor at the University of Michigan proposed a nanoscale memristor device which can mimic

the synapse behavior in neuromorphic systems [45]. Later on, in 2011 a team of multidisciplinary researchers from Harvard University published an interesting paper on programmable nanowire circuits for using in nanoprocessors [46]. Until June 2016, based on the Scopus bibliographic database, 2466 papers have been published in peer-reviewed journals and ISI articles which are related to memristor fabrication or applications of the memristor in different science and technology domains.

Memristors are promising devices for a wide range of potential applications from digital memory, logic/analog circuits, and bio-inspired applications [16]. Especially because the nonvolatility property in many types of memristors,they could be a suitable candidate for making non-volatile memories with ultra large capacity [14]. In addition to non-volatility, the memristor has other attractive features such as simple physical structure, high-density, low-power, and unlimited endurance which make this device a proper choice for many applications. Different device structures are still being developed to determine which memristor device can be presented as the best choice for commercial use in memory/flash manufacturing or in neuromorphic platforms. This is based on different factors such as size, switching speed, power consumption, switching longevity, and CMOS compatibility. The rest of the chapter is organized as follows: In Section 2, a general overview of the memristor is done and the electrical properties have been investigated. Section 3 presents memristor implementation and fabrication. We investigate various types of memristors based on the different materials that have been used in the fabrication. In Section 4, potential applications of Memristor has been studied. Section 5 deals with streams of research, we have investigated a research classification from the physics level to the system design. Finally, we describe a brief summary and the future work.

2.2 Memristor device overview and properties

In this section, we discuss the memristor nanodevice which is believed to be the fourth missing fundamental circuit element, that comes in the form of a passive two-terminal device. We discuss how it can remember its state, and what is its electrical model and particular properties.

2.2.1 Memristor a missing electrical passive element

Memristor is a contraction of "*memory* & *resistor,*" because the basic functionality of the memristor is to remember its state history. This characteristic proposes a promising component for next generation memory. Memristor is a thin-film electrical circuit element that changes its resistance depending on the total amount of charge that flows through the device. Chua proved that memristor behavior could not be duplicated by any circuit built using only the other three basic electronic elements (Resistor,Capacitor, Inductor), that is why the memristor is truly fundamental. As it is depicted in Figure 2.1, the resistor is constant factor between the voltage and current ($dv = R.di$), the capacitor is a constant factor between the charge and voltage ($dq = C.dv$), and the inductor is a constant factor between the flux and current ($d\varphi = L.di$). The relation between flux and charge is Obviously missing ($d\varphi = M.dq$) that can be interpreted by a fourth fundamental element such as memristor [11].

Obviously, in memristive devices, the nonlinear resistance can be changed and memorized by controlling the flow of the electrical charge or the magnetic flux. This control any two-terminal black box is called a memristor if, and only if, it exhibits a pinched hysteresis loop for all bipolar periodic input current signaling is interesting for the computation capability of a device similar to the controlling of the states of a transistor. For instance in an analog domain, one can control the state of a transistor to stay in an active area for amplification. Nevertheless, in the digital domain to stay in Off (cut-off) state for logic '0' and in On (saturated) state for logic '1' one can perform with controlling the gate voltage. The output current in MOSFET (Metal-Oxide semiconductor Field Effect Transistor) is managed by changing the gate voltage as well as in BJT (Bipolar Junction Transistor) the input current (base current) can control the output current (collector-emitter current). The main difference between the memristor and transistor for managing the states is that in transistor there is a third terminal to control the states however, in contrast a memristor is a two-terminal device and there is no extra terminal to control the device state. The challenge of using memristor as a computational component

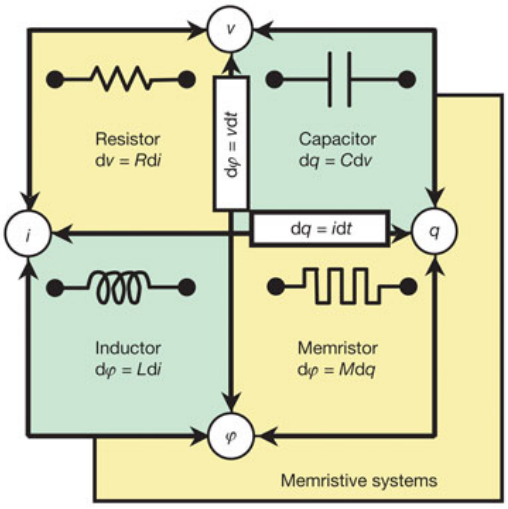

Figure 2.1: Relations between the passive devices and the anticipating the place of the fourth fundamental element based on the relations between charge (q) and flux (φ) (from [1]).

instead of transistor lies in the ability to control the working states as accurate as possible. Indeed, in a memristor both potentials for analog and digital computing have been presented. Consequently, using memristor in digital computing to make gate library or crossbar architecture as well as using memristor in analog domain (neuro-inspired or traditional) for computation are introduced in several work [8, 19–21, 47]. In next sections, we discuss different possibilities and our contributions to apply memristor in both digital and analog platforms.

2.2.2 Memristive device functionality

When you turn off the voltage, the memristor remembers its most recent resistance until the next time you turn it on, whether that happens a day later or a year later. It is worth mentioning that the duration to store the data in resistive form is dependent of the nano-device material. In other words, the volatility is different depending on the device materials in fabrication.

To understand the functionality of a memristor, let us imagine a resistor as a pipe which water flows through it. The water simulates the electric charge. The resistor obstruction of the flow of charge is comparable to the diameter of the pipe: the narrower the pipe, the greater the resistance. For the history of circuit design, resistors have had a fixed pipe diameter. But a memristor is a pipe that its diameter changes with the amount and direction of the water flows through it. If water flows through this pipe in one direction, it expands the pipe diameter (more conductive). But if the water flows in the opposite direction and the pipe shrinks (less conductive). Furthermore, let us imagine while we turn off the flow, the diameter of the pipe freezes until the water is turned back on. It mimics the memristor characteristic to remember last state. This freezing property suits memristors brilliantly for the new generation of memory. The ability to indefinitely store resistance values means that a memristor can be used as a nonvolatile memory.

Chua demonstrated mathematically that his hypothetical device would provide a relationship between flux and charge similar to what a resistor provides between voltage and current. There was no obvious physical interaction between charge and the integral over the voltage before HP discovery. Stanley Williams in [48] explained how they found the missing memristor and what is the relation between what they found and Chua mathematic model. In Figure 2.2, the oxygen deficiencies in the TiO_{2-x} manifest as bubbles of oxygen vacancies scattered throughout the upper layer. A positive voltage on the switch repels the (positive) oxygen deficiencies in the metallic upper TiO_{2-x} layer, sending them into the insulating TiO_2 layer below. That causes the boundary between the two materials to move down, increasing the percentage of conducting TiO_{2-x} and thus the conductivity of the entire switch.

Therefore, the more positive voltage causes the more conductivity in the cube. A negative voltage on

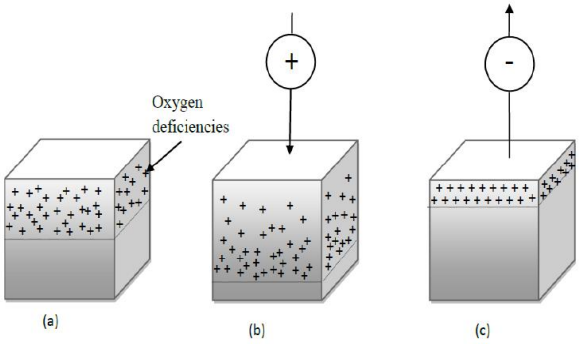

Figure 2.2: A material model of the memristor schematic to demonstrate TiO_2 memristor functionality, positive charge makes the device more conductive and negative charge makes it less conductive.

the switch attracts the positively charged oxygen bubbles, pulling them out of the TiO_2. The amount of insulating of resistive TiO_2 increases, thereby making the switch more resistive. The more negative voltage causes the less conductivity in the cube. What makes this switch a special device? When the voltage across the device is turned off–positive or negative–the oxygen bubbles do not migrate. They will freeze where they have been before, which means that the boundary between the two titanium dioxide layers is frozen. That is how the Memristor "remembers" the last state of conductivity as well as it proves the plasticity properties in memristor to be applied as a synapse in an artificial neural network architecture and neuromorphic platform.

2.2.3 Electrical model

When an electric field is applied to the terminals of the memristor, the shifting in the boundary between its doped and undoped regions leads to variable total resistance of the device. In Figure 2.3.a, the electrical behavior of memristor can be modeled as follows [1]:

$$v(t) = R_{mem} i(t) \tag{2.1}$$

$$R_{mem} = R_{ON} \frac{w(t)}{D} + R_{OFF}(1 - \frac{w(t)}{D}) \tag{2.2}$$

where $w(t)$ is the width of the doped region, D is the overall thickness of the TiO_2 bi-layer, R_{ON} is the resistance when the active region is completely doped ($w = D$) and R_{OFF} is the resistance, when the TiO_2 bi-layer is mostly undopped ($w \rightarrow 0$).

$$\frac{dw(t)}{dt} = \mu_v \frac{R_{ON}}{D} i(t) \tag{2.3}$$

which yields the following formula for $w(t)$:

$$w(t) = \mu_v \frac{R_{ON}}{D} q(t) \tag{2.4}$$

Where μ_v is the average dopant mobility. By inserting Equation (2.4) into Equation (2.2) and then into Equation (2.1) we obtain the memristance of the device, which for $R_{ON} \ll R_{OFF}$ simplifies to:

$$R_{mem} = M(q) = R_{OFF}(1 - \frac{\mu_v R_{ON}}{D^2} q(t)) \tag{2.5}$$

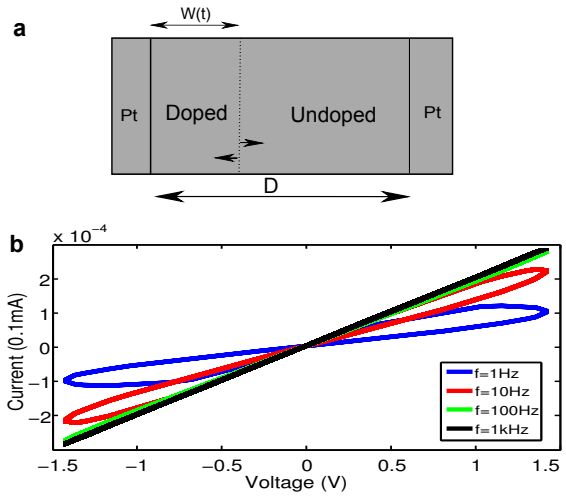

Figure 2.3: Memristor schematic and behavior: **a)** the memristor structure, the difference in applied voltage changes doped and undoped regions, **b)** current versus voltage diagram, which demonstrates hysteresis characteristic of a memristor, in the simulation we apply the sinusoidal input wave with an amplitude of 1.5v, different frequencies, $R_{ON} = 100\Omega, R_{OFF} = 15k\Omega, D = 10nm, \mu_v = 10^{-10} cm^2 s^{-1} V^{-1}$.

Equation (2.5) shows the dopant drift mobility μ_v and semiconductor film thicknesses D are two factors with crucial contributions to the memristance magnitude. Subsequently, we can write Kirchoff's voltage law for memristor given by:

$$v(t) = M(q)i(t) \tag{2.6}$$

By using Verilog-A HDL, we simulate the behavior of memristor, based on its behavioral equations. To investigate the characteristics of memristor in electrical circuits, the Verilog-A model of memristor behavior must be applied as a circuit element in the HSPICE netlist. In the HSPICE circuit, we apply a sinusoidal source to observe the memristor reaction in a simple circuit consisting of the memristor and the sinusoidal source. Figure 2.3.b depicts $i - v$ plot of memristor terminals that we measured in our simulation. This $i - v$ plot, which is the most significant feature of memristor [49], is namely called "pinched hysteresis loop". The $i - v$ characteristic demonstrates that memristor can "remember" the last electric charge flowing through it by changing its memristance. Therefore, we can use the memristor as a latch to save the data and also as a switch for computing. Moreover, in Figure 2.3.b, it is depicted that the pinched hysteresis loop is shrunk by increasing frequency. In fact, when the frequency increases toward infinity, memristor behavior is similar to a linear resistor.

2.3 Memristor classification based on different materials and applications

A memristor is generally made from a metal-insulator-metal (MIM) sandwich with the insulator usually consisting of a thin film like TiO_2 and a metal electrode like Pt. A memristor can be made from any Metal Insulator Metal (MIM) sandwich which exhibits a bipolar switching characteristic. It means that TiO_2 and Pt are not the only materials to fit the criteria for a memristor. For instance, Wan Gee Kim *et al.* [50] conducted a systematic approach using the HfO_2 and ZrO_2 as substitutes for TiO_2, also using TiN or Ti/TiN electrode instead of Pt. Basically, any two-terminal black box is called a memristor only if it can present a pinched hysteresis loop for all bipolar periodic input signals. Following we discuss four most significant materials for memristor fabrication namely:

- Resistive memristor
- Spintronic memristor

- Organic (Polymeric) memristor
- Ferroelectric memristor

2.3.1 Resistive Memristor

Before the memristor getting well-known, resistive materials have already been widely used in the resistive random access memories (ReRAM/RRAM) [51]. The storage function of ReRAM is realized by an intrinsic physical behavior in ReRAM, that is called resistive switching. The resistive material can be switched between a high resistance state (HRS) and a low resistance state (LRS) under an external electrical input signal. The TiO_2 memristor is a ReRAM fabricated in nanometre scale (2-3 nm) thin film that is depicted in Figure 2.3.a , containing a doped region and an undoped region. Strukov *et al.* [1] exploit a very thin-film TiO_2 sandwiched between two platinum (Pt) contacts and one side of the TiO_2 is doped with oxygen vacancies, which are positively charged ions. Therefore, there is a TiO_2 junction where one side is doped and the other is undoped. Such a doping process results in two different resistances: one is a high resistance (undoped) and the other is a low resistance (doped). The application of an external bias v(t) across the device will move the boundary between the two regions by causing the charged dopants to drift. How TiO_2 could change and store the state has been introduced in 2.2.2.

The obvious disadvantage of the first published TiO_2 memristor was its switching speed (operate at only 1Hz). The switching speed was not comparable with SRAM, DRAM and even flash memory. Flash exhibit writing times of the order of a microsecond and volatile memories have writing speeds of the order of hundreds of picoseconds. Many research groups in different labs published their fabrication results to demonstrate a faster switching speed device. In October 2011, HP lab developed a memristor switch using a SET pulse with a duration of 105 ps and a RESET pulse with a duration of 120 ps. The associated energies for ON and OFF switching were computed to be 1.9 and 5.8 pJ, respectively which are quite efficient for power-aware computations. The full-length D (Figure 2.3.a) of the TiO_2 memristor is 10 nm [52] that proposes high-density devices in a small area in VLSI.

A research team at the University of Michigan led by Wei Lu [45] demonstrated another type of resistive memristor that can be used to build brain- like computers and known as amorphous silicon memristor. The Amorphous silicon memristor consists of a layered device structure including a co-sputtered Ag and Si active layer with a properly designed Ag/Si mixture ratio gradient that leads to the formation of a Ag-rich (high conductivity) region and a Ag-poor (low conductivity) region. This demonstration provides the direct experimental support for the recently proposed memristor-based neuromorphic systems.

Amorphous silicon memristor can be fabricated with a CMOS compatible simple fabrication process using only common materials which is a great advantage of using amorphous silicon devices. The endurance test results of two extreme cases with programming current levels $10nA$ and $10mA$ are 10^6 and 10^5 cycles respectively. We note the larger than 10^6 cycles of endurance with low programming currents are already comparable to conventional flash memory devices. Wei Lu team have routinely observed switching speed faster than 5ns from the devices with a few mA on-current. The switching in this device is faster than 5 ns with a few mA on-current that make it a promising candidate for high-speed switching applications. However, before the devices can be used as a switch, they need to go through a high voltage forming process (typically $\geq$ 10 V) which significantly reduces the performance of power efficiency of devices [53]. Moreover, the retention time (data storage period) is still short (a few months).

2.3.2 Spintronic Memristor

Spintronic memristor changes its resistance by varying the direction of the spin of the electrons. Magnetic Tunneling Junction (MTJ) has been used in commercial recording heads to sense magnetic flux. It is the core device cell for spin torque magnetic random access memory and has also been proposed for logic devices. In a spintronic device, the electron spin changes the magnetization state

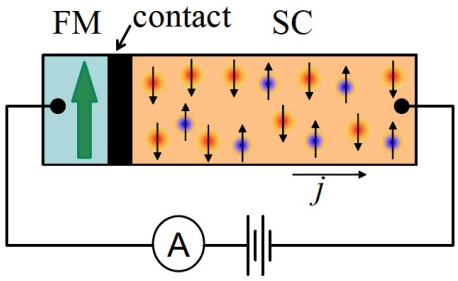

Figure 2.4: Spintronic memristor:Physical schematic of the circuit made of an interface between a semiconductor and a half-metal (ferromagnets with 100% spin-polarization at the Fermi level) (From [2]).

of the device. The magnetization state of the device is thus dependent upon the cumulative effects of electron spin excitations [54]. MTJ can be switched between a LRS and an HRS using the spin-polarized current induced between two ferromagnetic layers. If the resistance of this spintronic device is determined by its magnetization state, we could have a spintronic memristor with its resistance depending upon the integral effects of its current profile.

The use of a fundamentally different degree of freedom which allows for the realization of memristive behavior is thus desirabled by Pershin and Di Ventra [2]. They demonstrated that the degree of freedom is provided by the electron spin and memristive behavior is obtained from the broad class of semiconductor spintronic devices. This class involves systems whose transport properties depend on the level of electron spin polarization in a semiconductor which is influenced by an external control parameter (such as an applied voltage). Pershin and Di Ventra considered a junction with half-metals shown in Figure 2.4 (ferromagnets with 100% spin-polarization at the Fermi level), because these junctions react as perfect spin-filters, therefore they are more sensitive to the level of electron spin polarization. They observed memristor behavior in the $i - v$ curve of these systems. This means the proposed device is controllable and tunable. Furthermore, the device can be easily integrated on top of the CMOS. The integration of the spin torque memristor on CMOS is the same as the integration of magnetic random access memory cell on CMOS [55]. This has been achieved and commercialized in magnetic random access memory.

The potential applications of spintronic memristor are in multibit data storage and logic, novel sensing scheme, low power consumption computing, and information security. However, the small resistance ON/OFF ratio remains a notable concern for spintronic memristor devices.

2.3.3 Organic (Polymeric) Memristor

In 2005 Erokhin *et al.* [56] at the university of Parma reported a polymeric electrochemical element for the adaptive networks. Even though it was not called a memristor, however mainly its characteristics corresponds to the hypothetical memristor. At the heart of this device, there is a conducting channel, a thin polyaniline (PANI) layer, deposited onto an insulating support with two electrodes. A narrow stripe of solid electrolyte doped Poly Ethylene Oxide (PEO) which is formed in the central part of the channel and used for the redox reactions. The area of PANI under PEO is the active zone (see Figure 2.5). A thin silver wire is inserted into the solid electrolyte to provide the reference potential; such a wire is connected to one of the electrodes on the solid support, kept at the ground potential level.

Conductivity variations and memory properties of the organic memristor are due to the redox reactions occurring in the active zone, where PANI is reversibly transferred from the reduced insulating state into the oxidized conducting one [42]. In analogy with the nomenclature used in Field Effect Transistors (FETs), the two electrodes that are connected with the PANI film are called the source and drain electrodes, while the wire immersed in the PEO is called the gate electrode. In the normal

operation of the device, the source and the gate electrodes are kept at ground potential, and a voltage is applied to the drain electrode. Therefore, we can consider the organic memristor as a two-terminal device. The polymeric memristor, compared to the resistive memristor, can better meet the criteria of the theoretical memristor, as its resistance is generally governed by the charge transfer.

The Polymeric memristor was investigated in pulse mode, mimicking the synaptic behavior of signal transmission in neural systems. The phenomenon that the PANI conductivity, when connected to a positive signal excitation which is gradually increased similar to the synapse behavior in real biological neural systems, described in the Hebbian rule. Simple circuit based on the polymeric memristor has already been designed to realize both supervised and unsupervised learning in neural networks [42]. Organic materials present several advantages in terms of functionality, the deposition technique, costs and above all for the relative ease with which the material properties may be tailored by a chemical approach [57]. Organic memristor operates with very low power energy. The transformation to the conducting state occurs at potentials +0.4 - +0.6 V and transformation into the insulating state take place at potentials lower than +0.1 V [58].

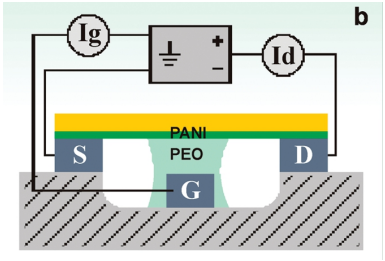

Figure 2.5: Organic (polymeric) Memristor: the active channel is formed by PANI on top of a support and two electrodes. The region between PANI and PEO is called the 'active zone', and conductivity transformation is performed here.

There is another type of organic memristor namely Nanoparticle Organic Memory Field Effect Transistor (NOMFET). NOMFET is made of conjugated molecules and metal nanoparticles (NPs). This device is initiated and fabricated by the institute of microelectronics and nanotechnology at Lille university [43]. NOMFET consists of a bottom-gate and source-drain contact organic transistor configuration. The gold NPs (5 nm in diameter) were immobilized into the source-drain channel by applying self-assembled monolayer covered by a thin film of Pentacene as it is shown in Figure 2.6. The NOMFET has the capability of mimicking synaptic properties as a volatile memory. We have used NOMFET in our research to make a new type of synapse which will be presented in next chapters of this thesis. Consequently, we will discuss it more in details.

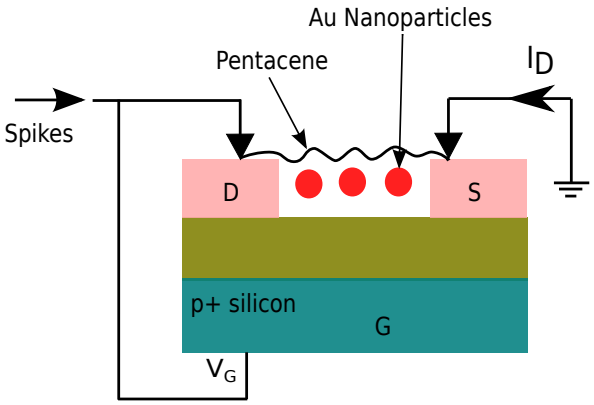

Figure 2.6: Physical structure of the NOMFET. It is composed of a p^+ doped bottom-gate covered with silicon oxide. Source and drain electrodes are made of gold and Au NPs are deposed on the inter-electrode gap before the pentacene deposition.

2.3.4 Ferroelectric Memristor

Ferroelectricity is a property of certain materials which have a spontaneous electric polarization that can be reversed by the application of an external electric field. The ferroelectric memristor is based on a thin ferroelectric barrier sandwiched between two metallic electrodes. Therefore, these two opposite polarization states can be used to represent binary bits '0' and '1', thus resulting in the advent of the Ferroelectric Random Access Memory (FeRAM). Due to FeRAM nonvolatility, ferroelectric materials have been widely used in automobile equipment, ID/smart card, Radio Frequency Identification (RFID) and other embedded memory applications [59].

Chanthbouala, A. *et al.* [41], showed voltage-controlled domain configurations in ferroelectric tunnel barriers yield memristive behavior with resistance variations exceeding two orders of magnitude and a 10 ns operation speed. They reported Piezoresponse Force Microscopy (PFM) images and electrical transport measurements as a function of the amplitude, duration and the repetition number of voltage pulses in the 10-200 ns range. In tunnel junctions with a ferroelectric barrier, switching the ferroelectric polarization induces variations of the tunnel resistance, with resistance contrasts between the ON and OFF states of several orders of magnitude. The low-resistance state (R_{ON}) corresponds to the ferroelectric polarization pointing up ($P\uparrow$), and the high-resistance state (R_{off}) corresponds to the ferroelectric polarization pointing down ($P\downarrow$). Positive and negative trains of pulses applied consecutively with +2.9 V and -2.7 V amplitude. In the analogy with the operation of FeRAM, the large ON/OFF ratio in ferroelectric tunnel junctions (FTJs) has so far been considered only for binary data storage, with the key advantage of non-destructive readout and simpler device architecture, however still non-CMOS compatible. In another study, Y.Kaneko *et al.* [3] presented a new transistor and implemented it by all oxide-based ferroelectric thin films, which include $SrRuO_3$ (SRO: bottom gate electrode), $Pb(Zr,Ti)O_3$ (PZT: ferroelectric), ZnO (semiconductor), and SiON (gate insulator) Figure 2.7. They have demonstrated the conductivity modulation of the interface between two oxides, ZnO and PZT, in a FeFET, is applicable for a nonvolatile memory which has the same memristive operation. Ferroelectric-based memristor cell can be expected to be very suitable for the nonvolatile memory array configuration and the future neuromorphic systems embedded in the intelligent transparent electronic applications [60].

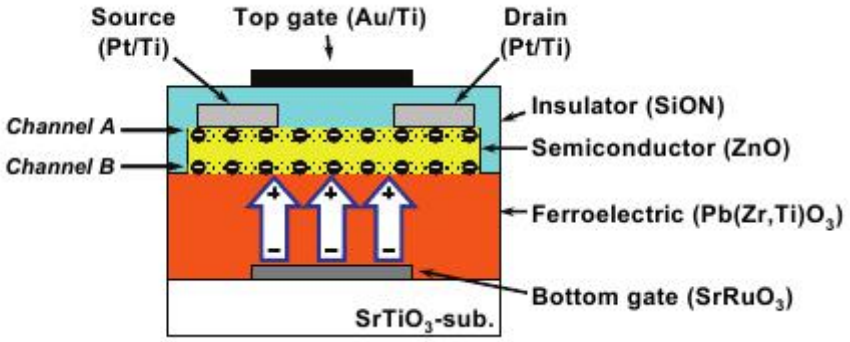

Figure 2.7: Ferroelectric Memristor, the OxiM transistor has dual channels at the upper and lower sides of the ZnO film, which are controlled independently by the top gate and the bottom gate, respectively. The bottom FET has the gate (SRO layer) and insulator (PZT ferroelectric layer) constituting a FeFET that has memory characteristics (from [3]).

2.3.5 Evaluation of Memristor with different materials

After 2008, there have been many articles related to memristors and memristive devices. Here we evaluate the four types of memristors using different materials. TiO_2 memristor is the first fabrication of memristor device which is considered as one of the most promising ones. There are hundreds of publications related to TiO_2 memristor . Recently, Stanley Williams research group enhanced the

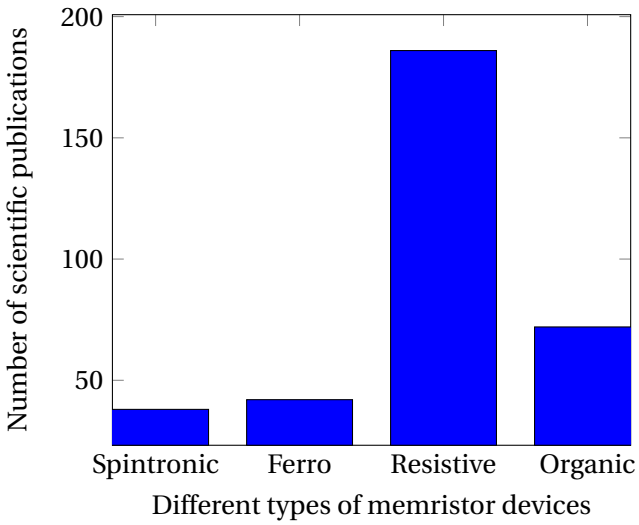

Figure 2.8: Number of publications for each type of memristors.

characteristics of TiO_2 memristor such as switching speed, programming endurance, and retention time which made TiO_2 memristor the best candidate to apply in commercial usages. Chen Yirin [54,55] assistant professor at the University of Pittsburgh, published 49 articles in memristor technology and application are working on spintronic memristor, Pershin and Di Ventra [2] are the other researchers who trying to improve spintronic memristive devices characteristics to use it as a nonvolatile memory. Organic materials present several advantages, therefore, could be the next proper candidate to fabricate memristive devices. The last material that we have evaluated here is ferroelectric memristor. After publishing a ferroelectric memristor in Nature materials journal [41] this type of fabrication is introduced as another option for building Memristor. We sum up specifications and characteristics of these five classes of memristor material in Table 7.1. Lei Wang *et al.* [61] discussed another type of memristors such as manganite memristor and Resonant-Tunneling Diode (RTD) memristor. We note that the magnetic memristor has the similar behavior to the ferroelectric type of memristive devices. RTD has more potential to react as a complementary device beside memristor for neural network applications e.g., Cellular Neural Networks (CNN) [62]. A quantitative comparison respecting the number of published paper specifically in fabrication and material of different classes of the memristors is presented in Figure 2.8.

Memristor	Advantage	Disadvantage	Applications	University-Lab
Resistive	small scale, fast switching, simple structure	still non-reliable for commercial	memory, logic gates, neuromorphic, analog devices	HP Lab
Spintronic	magnetic memory match technology	the small resistance ON/OFF ratio	Neuro-inspired systems, memory	University of Pittsburgh, US
Organic	relative ease with chemical materials, work with ultra-low power	slow switching	artificial synapse	Parma/Lille University
Ferroelectric	suitable for the nonvolatile memory array	slow switching, Non-CMOS Compatible	synapse, RRAM	Panasonic, Japan & Thales, France

Table 2.1: Table of different class of memristors based on different materials and its applications, the first university/lab announcement of the device is listed too.

2.4 Potential applications of memristors

In the previous section, we have discussed the different type of memristive materials. In this section, we study the potential application of memristors. we divided the applications into three main classes: nonvolatile memories, digital computing, and analog domain applications. The classification in addition to the practical application examples is depicted in Figure 2.9. This classification may cover most of the recent applications, however, the memristor is a novel device which new capabilities may be introduced soon in various areas of research. Therefore, the novel applications are dramatically anticipated.

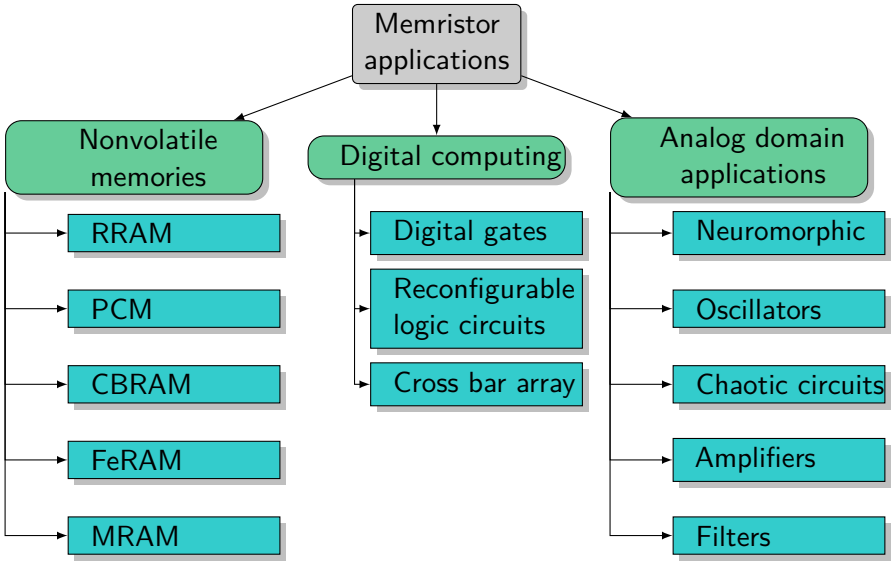

Figure 2.9: Different memristor applications in different domains.

2.4.1 Memristor-based nonvolatile memory

Memristor-based nonvolatile memory is the most obvious application of memristors [63]. The non-volatile, memristor-based memory cell compared to SRAM and DRAM, can exhibit non-volatility, good scalability, compatible with conventional CMOS electronics, and the last but not the least, it has no leakage power. Several types of memories are introduced using memristor nanodevice. Resistive RAM (RRAM) [64], Phase change memory (PCM) [65], Conductive Bridge memory (CBRAM) [66], ferroelectric memory (FeRAM) [41] and the spintronic memristor that can be a promising replacement for Magnetic memory (MRAM) [67]. A memory array of Memristors what is called a resistive RAM or RRAM is another form of memristor memory. RRAM operates faster than phase-change memory (PCRAM), and it has simpler and smaller cell structure than magnetic memory (MRAM).

2.4.2 Digital computing

Another possible application of memristor is digital design and computation. Memristors can be applied in hybrid CMOS-memristor circuits, or as a basic component to built the logic gates [68]. One remarkable logic application is using memristor as a reconfigurable switch (FPGA) [69]. The implication logic synthesis [20] and crossbar array architecture [21] are two alternatives presented a the new approaches to make efficient digital gate library. In next chapter of this study, we will discuss and develop these two methods in details.

2.4.3 Analog domain applications

Another further research area of memristive devices is analog domain application. Simple circuits of memristors with a single capacitor or inductor are discussed in [70]. There are several research work in analog domain using memristors such as neuromorphic computing, amplifiers, memristor oscillators, filters and chaotic circuits (see e.g. [71], [72], [73], [74]).

Most interesting and recent study field in analog domain that we will focus on it more in our research is neuromorphic or neuro-inspired computation. It is believed this approach is promising for yielding innovations in future cognitive computing, hence will get probably more attention in the near future of research. The key to the high efficiency of biological neural systems is the large connectivity between neurons that offers highly parallel processing power. Another key factor of high efficiency of biological neural network is the connection plasticity. The synaptic weight between two neurons can be precisely adjusted by the ionic flow through them and it is widely believed that the adaptation of synaptic weights enables the biological neural systems to learn. Memristors function similarly to the biological synapses. This characteristic makes Memristors proper building blocks in neuromorphic systems, where neurons and synapses are modeled as electronic devices [45]. Memristors can be made extremely small, therefore, by applying them, mimicking the functions of a brain would be possible in near future [48]. The brain-like computing capability of the memristor-based complex spike timing-dependent plasticity (STDP) learning networks is demonstrated by Afifi *et al* [75].

2.5 Streams of research

The ability to indefinitely store resistance values means that a memristor can be used as a nonvolatile memory. There are yet more potential applications that we did not mention previously , we point to the capacity of memristive networks in realizing demanding image processing and more specifically edge detection and face/object recognition [76]. Pershin and Di Ventra address the capability of memristors to perform quantum computation in addition to conventional neuromorphic and digital data processing [77]. Performing arithmetic operations in memristor-based structures is possible in both analog and digital approaches [78]. Another system level application is memristive Neuro-Fuzzy System [79].

However, the memristor potential goes far beyond instant-on computers to embrace one of the biggest technology challenges: mimicking the functionality of the brain. Within a decade, memristors could let us emulate, instead of merely simulate, networks of neurons and synapses. By replacing several specific transistors with a crossbar of memristors, circuit could be shrunk by nearly a factor of 10

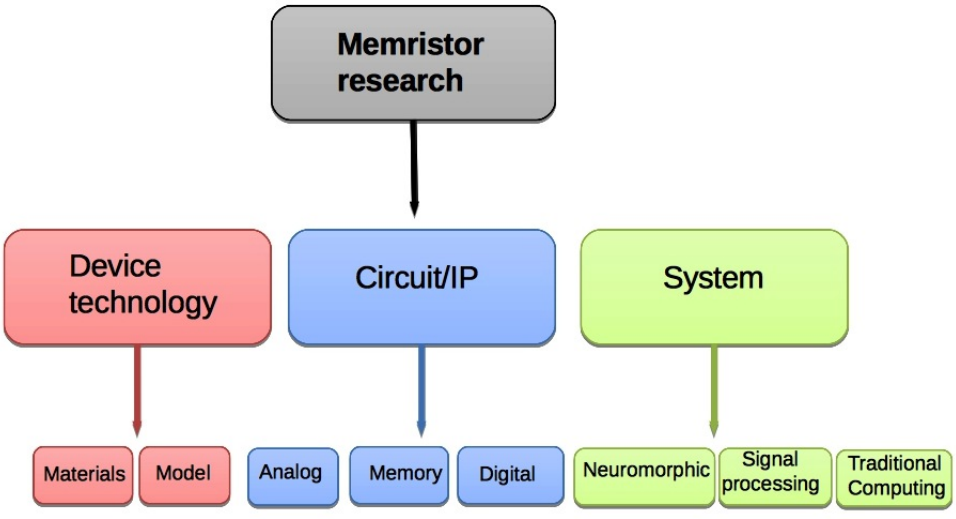

Figure 2.10: Classification of domain studies using memristive devices

in area and improved in terms of its speed relative to power consumption performance [48]. Computing with memristor is another interesting approach which memristor plays a crossbar switch role in the circuit. Using memristor-based circuit for performing arithmetic operations, signal processing application, dynamic load, oscillators, amplifiers, sensing application, artificial biological system, image encryption, and many other approaches are several applications of memristor that recently appeared in different research publications. In this context, we made a simple classification of a stream of research as shown in Figure 2.10 that could be useful for those who want to use this flexible device in their research studies. The hybrid approaches are not in the chart, here we mention that any sort of utilization can be implemented in hybrid circuit instead of using pure memristive circuit.

2.6 Conclusions and summary

In this chapter, we have done a feasibility study on memristive nanodevice from theoretical model to practical applications. we studied the cons and pros of using different memristor devices based on the applied material to manufacture the device. As each material or type of memristor has its own characteristics, consequently one can analyze and discover the different potential of application of each type of these memristive devices more conveniently. The better understanding of physics of different memristive materials leads to discover more sophisticated applications of these devices. Four general memristive devices are surveyed namely: Resistive, Spintronic, Organic (polymeric) and Ferroelectric memristive devices. The potential application as well as advantages versus disadvantages of using each one are presented too. The resistive memristor has been applied more than others in different research works from memory to artificial synapse. The Spintronic and Ferroelectric devices show promising properties to make new nonvolatile memories. The organic memristor is more appropriate to make artificial synapse in Spiking Neural Networks. The mix combination of those materials not only propose new research studies but also take the advantages of using more useful properties of each device.

The practical applications of memristive devices are presented subsequently. Three main domains of potential applications have been classified: a)nonvoaltile memory such as RRAM, PCM, CBRAM, FeRAM and MRAM; b)digital computing domain such as logic implication and crossbar array; c)analog domain of application such as neuro-inspired computing in spiking neural networks, oscillators, filters and amplifiers. Among these applications, we focus on computation approaches by using two digital and neuromorphic domains.

Chapter 3

Unconventional digital computing approach: memristive nanodevice platform

Abstract

Memristor is a two-terminal nanodevice that has recently attracted the attention of many researchers. Its simple structure, non-volatility behavior, high-density integration, and low-power consumption make the memristor a promising candidate to act as a switch in digital gates for future high-performance and low-power nanocomputing applications. In this chapter, we model the behavior of memristor by using Verilog-A tools. To investigate its characteristics in a circuit, we use the HSPICE simulator. Furthermore, a library of digital gates is provided by using two approaches to make digital gates: the first one is based on material implication (IMP) and the second one is based on crossbar arrays. Finally, we compare and evaluate both computation methods.

3.1 Introduction

It is important to be able to control the states of the conductance of memristor in order to use memristors for computing in general, analog or digital. As the memristor is applied in the different platform as a two terminal device for state modification new approaches are required. In traditional CMOS computing, using transistor state we can switch between '0' and '1' to be able to produce Boolean functions such as NOT, AND, OR and etc. In CMOS-based computing, the switching of the transistor is controlled by a third terminal, (Base in BJT and Gate in MOSFET). The story of the state controlling for memristor nanodevice is different as there are only input and output terminals (no terminal for controlling the states of the switch). Therefore, the new methods and approaches are studied to be able to use two-terminal devices for computing.

In this chapter, we study and discuss two methods for digital computing by memristive tow-terminal devices mainly based on our publication [18]. There are two main approaches, namely *material implication* (IMP) or stateful logic and *programmable crossbar architecture* that are discussed in this chapter. Material implication was studied by Whitehead and Russell in early 1900s in *Principia Mathematica* [80]. IMP can be implemented on memristor with couple of sequences of control voltages. In this study based on previous research works on IMP, we establish a functionally complete Boolean operation to be able to build other standard logic gates. The second method, combination of memristor with crossbar interconnect technology, has been demonstrated as offering the potential of creating configurable electronic devices with applications in digital computation, signal processing and artificial intelligence [22]. In this chapter, we analyze the application of crossbar array architecture in constructing logic building blocks. Our contribution is to present a comprehensive library to build

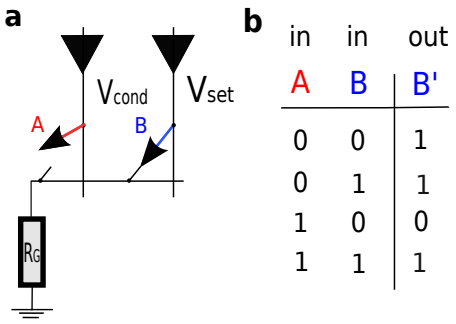

Figure 3.1: Memristor-based IMP: **a)** circuit schematic, **b)** IMP truth table.

a complete set of Boolean gates using both approaches. At the end of this chapter we evaluate and compare both methods.

The logic computing applications of memristor have been investigated by several researchers [17, 19–22, 81–84]. For instance, Borghetti et al. [17] used *material implication* (IMP) logic operation to carry out logic computation by using memristors. In IMP approach, A IMP B ($A \rightarrow B$) operation means 'if A, then B' and can be read as A implies B. IMP together with a FALSE gate are able to form a functionally complete set (any Boolean function can be expressed). By applying memristor as a digital switch, a high memristance (memristor resistance) is considered as logic '0' and a low memristance is considered as logic '1'. Another approach to make gates by a two-terminal device as a switch is the *programmable crossbar architecture* [21, 22]. The crossbar nanowire array architecture can be used to compute logic functions by using memristor as a switch between two nanowires [81].

3.2 Stateful implication logic

One of the basic potential applications of memristors is to utilize them in building blocks of logic gates. Therefore, by applying a digital pulse voltage to the memristor terminals, we have a switch with ON or OFF state. Unlike conventional CMOS, in memristor-based gates, data will be stored as a resistance rather than a voltage. In this case, the latches are non-volatile. Thus, R_{ON} displays logic '1' which means closed switch and R_{OFF} displays '0' for presenting open switch. In contrast to the three-terminal CMOS-based transistor as a switch, in a two-terminal switch, there is no terminal to control ON or OFF states of the switch. Consequently, instead of conventional Boolean logic, we should find other substitutes to create a logic gate.

IMP (Figure 3.1.a) is a way to use one memristor to control the other one. IMP is recognized as a promising method for making gates by memristors [17, 20, 83, 85]. In IMP structure, memristors have different roles in different stages of the computing process: input, output, computational logic element, and a latch depending upon which write, read, computing and storing processes are taking place, respectively.

To figure out how IMP operates, imagine A as a question and B as an answer to that question. If the question is wrong, any answer (wrong or correct) makes a true output (logic 1), as depicted in Figure 3.1.b. The only case for the false (logic 0) output would be a wrong answer to a correct question. So the implication logic is equivalent to function $(\neg A) \vee B$. Figure 3.1.a shows the basic circuit of memristors A and B to perform implication logic, which are formed by the vertical nanowire crossing over the horizontal nanowire connected to a load resistor R_G. After the operation of material implications, the result is stored as the state of switch B (B') while A is left unchanged.

To switch between logic '1' to logic '0' (and vice versa), we need a tri-state voltage driver with a high impedance output state when it is undriven. V_{set} is a negative voltage which should be applied to its corresponding tri-state driver. V_{set} can switch memristor to conductive state with low resistance R_{ON}. Similarly, the positive voltage V_{clear} is required to change the memristive switch state to low-conductance (high-resistance) state R_{OFF}. It is important to mention that the magnitude of V_{set}

and V_{clear} must be larger than device threshold voltage for switching 'ON' and 'OFF'. In order to remain in a specific state (line 3 in truth table 3.1.b), the V_{cond} is applied as a negative voltage with a magnitude smaller than V_{set}. Consequently, tri-state drive —since is not in high-impedance state— is pulsed by one of the V_{set}, V_{clear} or V_{cond}. By applying V_{cond} and V_{set} to A and B simultaneously, the memristive IMP operates properly. Although the conditional voltage (V_{cond}) is not necessary, except for the case AB='10' (third line in the truth table Figure 3.1.b), it is possible to either apply V_{cond} or use high-impedance (HZ) for all other cases. If V_{set} is applied to B alone, it would be unconditionally logic '1', nevertheless applying V_{cond} by itself to A does not change its state. On the other hand, if both voltages are applied together, the present state of switch A determines the next state of switch B. If A='0', it means memristor A is in high resistance state (R_{OFF}). Therefore, there is a small voltage drop across R_G. In this case, B will be set and A is left unchanged. Alternatively, if present state of A='1', switch A is in low resistance state and V_{cond} drops across R_G, so both A and B remain unchanged. It should be noted that the R_G value must be chosen such that R_{ON}<R_G<R_{OFF}, where R_{ON} and R_{OFF} are resistance states of 'ON' and 'OFF' switches, respectively.

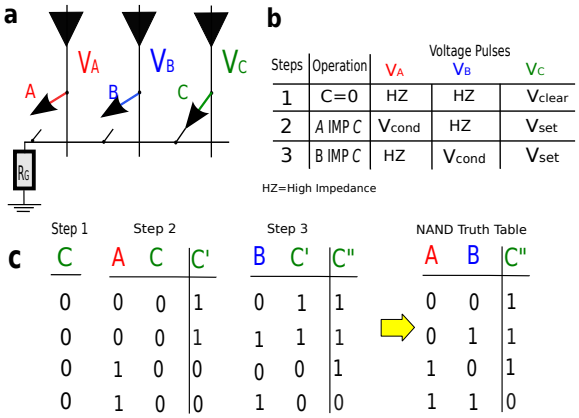

Figure 3.2: NAND configuration with IMP: **a)** circuit schematic, **b)** required voltages for controlling the process, **c)** sequential truth table to obtain NAND.

3.2.1 Functionally complete Boolean operations

In Boolean logic algebra, a set of logic operations is called functionally complete if all other logic functions can be constructed from combining the members of this set. The single-element sets *NAND* (AND, NOT) and *NOR* (OR, NOT) are functionally complete operations. In addition, it has been demonstrated that all Boolean expressions can then be written in one of the standard normal forms using only a (IMP) operation and a *false* (Inverse) operation [17, 83]. In fact, to show that the memristive implication logic is a functionally complete operation; the easiest way is to synthesize the NAND function with it.

A circuit with three memristors A, B, and C is illustrated schematically in Figure 3.2.a. Assume two implication operations being performed subsequently, first A IMP C and then B IMP C. Hence A and B are inputs and C is output. The final output result is represented with variable C$''$ in Figure 3.2.c. Firstly, V_{clear} should be applied to switch C to create the *false* operation ($C' = (C \text{ IMP } 0)$). By applying V_{cond} and V_{set} pulses to A and C respectively, the second step would be performed. Finally, $C'' = (B \text{ IMP } C)$ is yielded by applying V_{cond} to V_B and V_{set} to V_C (see Figure 3.2.b and Figure 3.2.c). In other words, the resulting state of switch C can be written as:

$$C = B \text{ IMP } (A \text{ IMP } C) = \neg B \vee ((\neg A) \vee C)$$

if $C = 0$ initially then

$$\begin{aligned} C &= \neg B \vee ((\neg A) \vee 0) = (\neg B) \vee (\neg A) \\ &= \neg (A \wedge B) \end{aligned}$$

which is a NAND operation. Similarly, we can produce equal IMP structure of other logic operations. In Table 6.1 different Boolean logic gates are listed. Obviously, we can demonstrate all logic relations in Table 6.1, not only by applying Boolean logic rules but also by checking the truth tables of both sides of relations.

Table 3.1: Different logic operations made by IMP operations and the number of required memristors

Logic Operation	Equal IMP functions	#Device
NOT A	A IMP 0	3
A AND B	{A IMP (B IMP 0)} IMP 0	4
A NAND B	A IMP (B IMP 0)	3
A OR B	(A IMP 0) IMP B	3
A NOR B	{(A IMP 0) IMP B} IMP 0	4
A XOR B	(A IMP B) IMP {(B IMP A) IMP 0}	3

3.3 Crossbar architecture

Programmable crossbar architectures have been proposed as a promising approach for future computing architectures because of their simplicity of fabrication and high density, which support defect tolerance [22, 86, 87]. In such architectures, assume that each junction within the crossbar can be utterly configured to activate an electronic device, such as a resistor, diode, transistor, or recently memristor. In fact, attractive features of memristor, such as simple physical structure, non-volatility, high-density, and unlimited endurance make this nano-device one of the best choices to play a switch role in the crossbar junctions. The memristive-based crossbar opens new windows to explore advanced computer architecture, different from the classical Von Neumann architecture [86]. On the other hand, in crossbar architecture, memory and logic operators are not separated. The memory can perform logic implementations on the same devices which store data. This is because during the operation process, control signals determine which elements act as logic gates and which ones act as memory cells.

3.3.1 Memristive switches in crossbar architectures

Each junction in a crossbar could be connected or disconnected by replacing a memristor as a switch in the junction point between two vertical and horizontal nanowires as depicted in Figure 3.3.a. Such a switch is in high resistance (R_{OFF}) (Figure 3.3.c) for open state or low resistance (R_{ON}) (Figure 3.3.d) for close state, similar to the states of memristor in Section 3.2. The memristive switch retains its state as long as the voltage drop across it, which is not more than the required threshold voltage to change the memristor state. In Figure 3.3.a the input voltage (v_{in}) either could be connected to the output voltage of an external CMOS circuitry or be connected to the output of another latch from the previous stage. As we have already mentioned, data in our architecture is saved as the resistance of the memristive latch. However, for data transferring, an input voltage is necessary. Indeed, this input voltage is driven by the state of the input impedance (R_{in} in Figure 3.4), which is a memristive switch. This voltage can be disconnected which means the floating state (for instance, a very high impedance R_{OFF}). The floating state input happens when the input memristor (R_{in} in Figure 3.4.a) is OFF. Subsequently, the input voltage can also be a fixed negative voltage ($-V_f$) with less magnitude than the threshold, while the input memristor (R_{in} in Figure 3.4.b) is ON.

The memristive switches can be configured as an inverting or non-inverting mode as it is depicted in Figure 3.4.c and Figure 3.4.d, respectively. If the stored data in the receiving switch is the logical

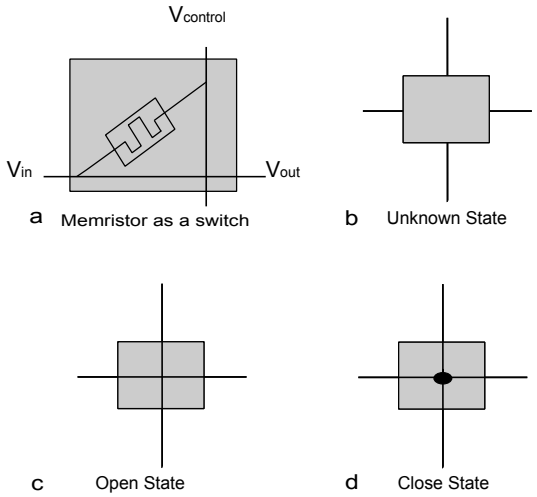

Figure 3.3: Different states of a switch in a crossbar array.

complement of the input data, the configuration is in the inverting mode. If the stored data is the same as the input data the configuration is in the non-inverting mode. Different control signals make different configurations. Inverting configuration requires three steps to perform the appropriate latch operation:
1) We preset the switch unconditionally open (Figure 3.4.a) by applying a positive voltage (more than the positive threshold) to the control of the vertical nanowire and also by forcing input to the high impedance mode. A diode is also required to provide a low-impedance path to the ground to protect the junction.
2) If the input voltage is logic '1' then R_{in}=R_{ON} and $-V_f$ drops across R_G, so the voltage across memristor is not enough to close it and the switch remains OFF. On the other hand, when input is logic '0', then R_{in}=R_{OFF} and the memristor switch turns ON. Thus, the junction has held the inverted state of the input (see Figure 3.4.b).
3) The input signal must be in the high impedance (disconnected). The state of the switch is read out onto the horizontal nanowire. This is accomplished by driving the vertical nanowire with a small voltage whose magnitude is less than the threshold control voltage (can not change the switch state). We call this voltage v_R (Read voltage).
For non-inverting configuration, as depicted in Figure 3.4.d, the pull-down resistor R_G is not required when writing the state of the driving switch to the receiving one. The steps for appropriate latch operation are the same as the 'inverting configuration' except grounding the vertical nanowire of the driving switch rather applying a negative voltage in conditionally close step. In this case, by applying the negative voltage to the vertical control line, the receiving switch will be close if the driving switch is close (low-impedance path to the ground) and the receiving switch remains open if the driving switch is open. Thus, the state of the first switch is replicated in the second switch.

3.3.2 Configurable crossbar array logic gates

In this section, we demonstrate how to make a gate by using the crossbar approach. We start using crossbar to make a 3-input AND gates. Figure 3.5.a shows a crossbar array which consisting of 8 memristors. We need three inputs A, B, and C, each of which is assumed to be impedance encoded. Basically, various voltages are applied to vertical nanowires to control the memristive switches. The inputs are connected to the horizontal wires as represented in Figure 3.5.a. The crossbar array functions as an AND gate by applying the following sequences.

1. All junctions are unconditionally opened by applying a positive voltage higher than the threshold voltage to the v_{in}, v_{and}, v_{out} control vertical lines.

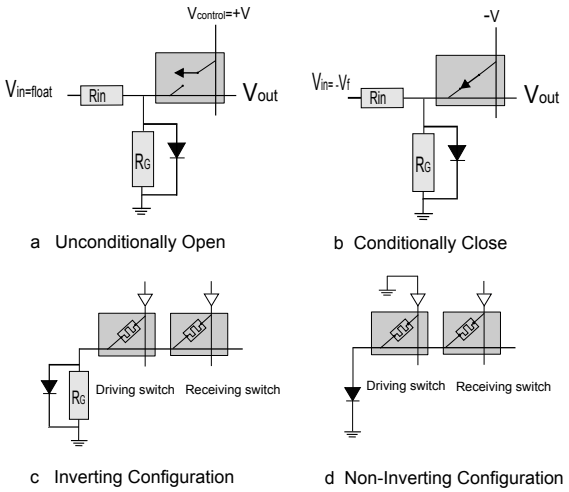

Figure 3.4: Different states and configurations of the memristive switches in a crossbar array.

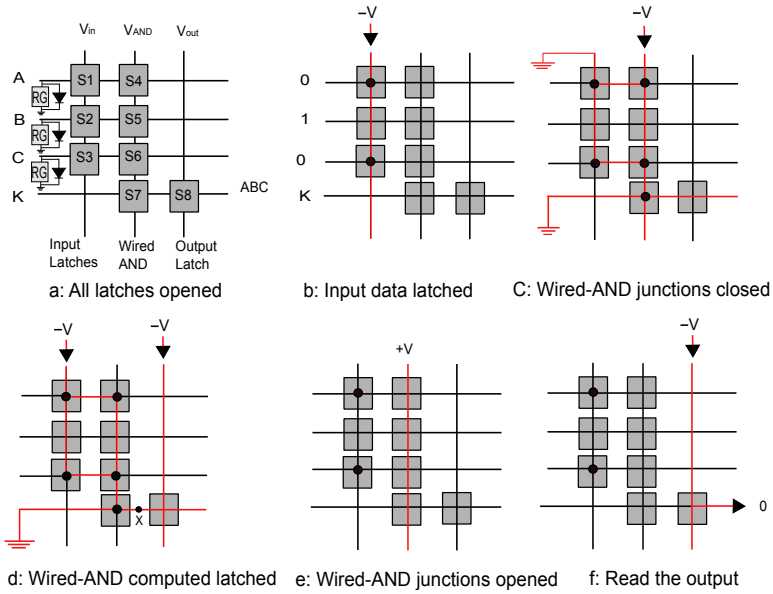

Figure 3.5: The crossbar array architecture for AND function with memristor switches: **a)** by applying the positive voltages all switches become open (clear), **b)** capture data in the input latches, **c)** transfer data to the wired-AND switches, **d)** if all inputs are '1' (open) then the X spot is not negative so the output switch has enough voltage across it to be ON, **e)** open all wired-AND switches to be prepared to read output, **f)** read output.

2. By driving v_{in} with a negative voltage, the input data (A, B, C) are latched in the input switches. The inputs are in the inverting configuration thus '0' and '1' inputs, close and open the switches, respectively (Figure 3.5.b).

3. In this step, by driving the v_{in} and K input to the ground, as well as v_{and} to the negative voltage (Figure 3.5.c) the input data are latched to the wired-AND junctions.

4. In this step, the vertical control lines of the input and output (v_{in}, v_{out}) are activated by a negative voltage to capture the result to the output memristor (S8). The voltage in point X in Figure 3.5.d can determine output switch state. If there is at least one closed input (also wired-AND) in the route from v_{in} to the X point, the negative voltage efficacy causes the output switch to stay open. The reason is that the potential difference across the output switch is not enough to close it. It is noteworthy to mention that the voltage at X point is yielded from the voltages dividing between the resistance of switches in the route and S7 that connected to the

ground in the last horizontal nanowire line.

5. By driving a positive voltage to v_{and}, all junctions in this column will be opened (Figure 3.5.e).

6. The AND of the three inputs is stored in the output switch and is ready to be read out.

If the output switch is at inverse mode (inverting configuration), the crossbar array becomes a 3-input NAND gate. Moreover, the crossbar array can also become a 3-input NOR gate, if R_G's are removed while input data are applied to the circuit (Non-inverting configuration). By inverting the last output with the recent situation of the inputs, the crossbar architecture becomes a 3-input OR gate. Consequently, all logic gates are created except XOR and XNOR, which require a little different structures. First, we create an exclusive-NOR gate, subsequently, by inverting the output, XOR is obtained. To implement XNOR, two midterms should be OR'ed together. The crossbar array operation for creating XNOR is similar to AND crossbar array in Figure 3.5. However, three additional steps are required. To simplify the crossbar array, we apply two variables A and B (Figure 3.6). In the following, we discuss these sequences step by step.

1. All junctions are unconditionally opened by applying a positive voltage to the v_{in}, two v_{AND}'s, and v_{out} control lines.

2. By driving v_{in} with a negative voltage, the input data (A, B, $\bar{A}$, $\bar{B}$) are latched in the input switches. The inputs are in the inverting configuration, therefore, '0' and '1' inputs close and open the switches, respectively.

3. In this step, by driving the v_{in} and K input to the ground, and v_{AND1} to the negative voltage, the input data are latched to the wired-AND for the first minterm.

4. In this step, the vertical control lines of input and output (v_{in}, v_{out}) are activated by a negative voltage for first minterm to capture the result ($\bar{A}\bar{B}$) to the output.

5. By applying the positive voltage to v_{AND1} (the wired-AND), junctions become open for the first minterm.

6. For the second minterm implementation, step 3 should be repeated by applying a negative voltage to v_{AND2}.

7. In this step, the control lines of input and output (v_{in}, v_{out}) are activated by a negative voltage for the second minterm to capture the result (AB) to the output. It is worth to note that performing OR of the two minterms is sequential rather than concurrent. It means that if the output switch is set into the close state by the first minterm, the value of the second minterm has no effect. Otherwise, changing the state of the output switch is dependent on the second minterm.

8. By applying the positive voltage to the v_{AND2}, the wired-AND junctions become open for the second minterm.

9. The XNOR of two inputs A and B is stored in the output switch and it is ready to be read out.

It is also possible to make other Boolean logic functions by applying NAND combinations. However, it requires a larger number of memristive switches. Furthermore, it nullifies the most important capability of memristors, i.e. configurability.

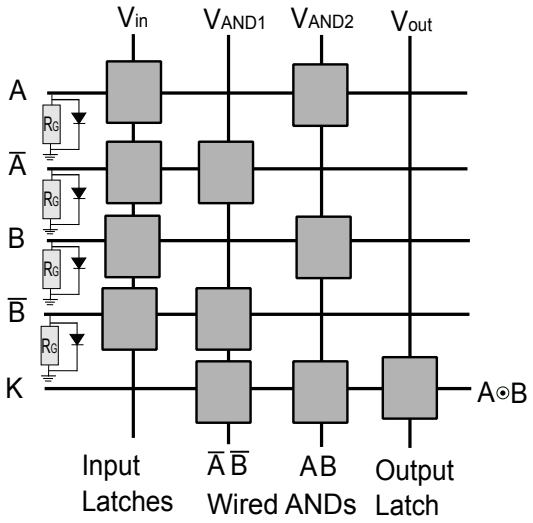

Figure 3.6: Crossbar array architecture for exclusive-NOR function.

3.4 Evaluation

In this section, we present an evaluation of both the IMP and crossbar array approaches. Both approaches have significant advantages over the CMOS-based logic gates, such as having computation and storing using the same physical unit, non-volatility, and small scalability because of the nature of memristive switches. By applying the IMP method, the number of memristors to make a gate can be saved. The less number of memristive switches causes less power consumption and cost. Table 3.2 provides the number of memristors required to make various logic gates in both cases. We note

Table 3.2: The number of memristive switches to make logic gates for the imp and crossbar array approaches.

Logic Operation	#IMP	#Crossbar Array
$S \leftarrow$ NOT A	2	3
$S \leftarrow A$ AND B	4	6
$S \leftarrow A$ NAND B	3	6
$S \leftarrow A$ OR B	6	6
$S \leftarrow A$ NOR B	6	6
$S \leftarrow A$ XOR B	7	11

that the number of memristors to make gates in the IMP approach in Table 3.1 is different from Table 3.2. When logic operations listed in Table 3.1 operate on two variables A and B, the original logic would be lost during the implication process. For instance, in case of NAND, only B is changed and in case of XOR both A and B will change. Therefore, to store data we require the additional number of memristive switches. For a more detailed explanation see [78]. Despite the overhead introduced in the crossbar approach in terms of number of memristors, it can dynamically adapt its logic function by controlling voltages. Therefore, based on the run-time requirements, the crossbar array approach is adaptable and reconfigurable.

3.5 Conclusions

In this chapter, based on the electrical model of the memristor is represented in Chapter 2, we show how physical properties of this device can be utilized for digital circuit applications. Memristor is a two-terminal device, therefore, we need new approaches to apply it to operate as a switch. IMP

logic with memristor is studied as the first method to create digital gates. We demonstrated that by using IMP and the inverse function (NOT), it is possible to produce all digital Boolean functions. In the IMP approach, the number of memristors to make gates are fewer than the crossbar array approach. We proposed the crossbar array architecture as the second novel promising technique for nanocomputing binary paradigm. The crossbar array method with memristive switches has been investigated comprehensively. Reconfigurability is the most significant advantage of using the crossbar array architecture. It is interesting to note that in both computing techniques, the storing and computing units are physically the same. This property has the potential to overcome the memory bottleneck. For future work, one can organize a Programmable Logic Device (PLA) platform in such a way that the crossbar array operates as a programmable AND array beside IMP as a fixed connection for the OR array.

Chapter 4

Neuromorphic computing in Spiking Neural Network architecture

4.1 Introduction

The mammalian nervous system is a network of extreme complexity which is able to perform cognitive computation in a parallel and power-efficient manner. Understanding the principles of the brain processing for computational modeling is one of the biggest challenges of the 21st century that led to the new branch of research e.g., neuromorphic computing. Neuromorphic engineering represents one of the promising fields for developing new computing paradigms complementing or even replacing current Von Neumann architecture [9].

The main remarkable difference between conventional Von Neumann architecture and neuromorphic systems is in their use of memory structures. The way of communication between memory and central processing unit (CPU) in conventional computing is not efficient. The memory and CPU communication suffer from what is called Von Neumann memory bottelneck. The CPUs access both data and program in memory using the same shared resources. CPUs spend most of their time idle because the speed of CPU is much more than memory due to the quality of materials applied to manufacturing the transistors in CPU and different memories.

If we want to apply better quality of memory such as SRAM, regarding to the high demands of memory usages the machine would be costly. To improve the efficiency of nowadays computation platforms, the applicable solution is what commonly known as the cache hierarchy; in other words, a limited amount of fast but costly memory sit closer to the processing unit, while most of the data is stored in the cheaper but larger memory as it is shown in Figure 4.1.a. To execute computational tasks, instruction codes and data stored in the memory are fetched to the processor, and after execution, pushed back to the memory unit, via a memory bus. Subsequently, it would be operating system (OS) duty to manage the data around these different levels of memory to optimize the system speed by consisting frequently-used data to the closer memory with better quality and speed rate. On the other hand, the multi-core platforms are commonly used in the new hardwares and the memory hierarchy management would be more significant and difficult too. By proposing computing unit next to the local memory, neuromorphic brain-inspired computing paradigms offer an attractive solution for implementing alternative non von Neumann architectures, using advanced and emerging technologies [24].

Neuromorphic systems are electronic implementations inspired from neural systems that is known as neuro-inspired computation system. The idea of creating circuit model for a neuron system refers back at least to 1907, where a neuron is modeled by a resistor and a capacitor [88]. However, the first neuromorphic term was coined by Carver Mead [89] using Very Large Scale Integration (VLSI) technology to propose an implementation of neural system hardware. Mahowald and Mead implemented the first silicon retina model with considering of adaptivity and energy efficiency by

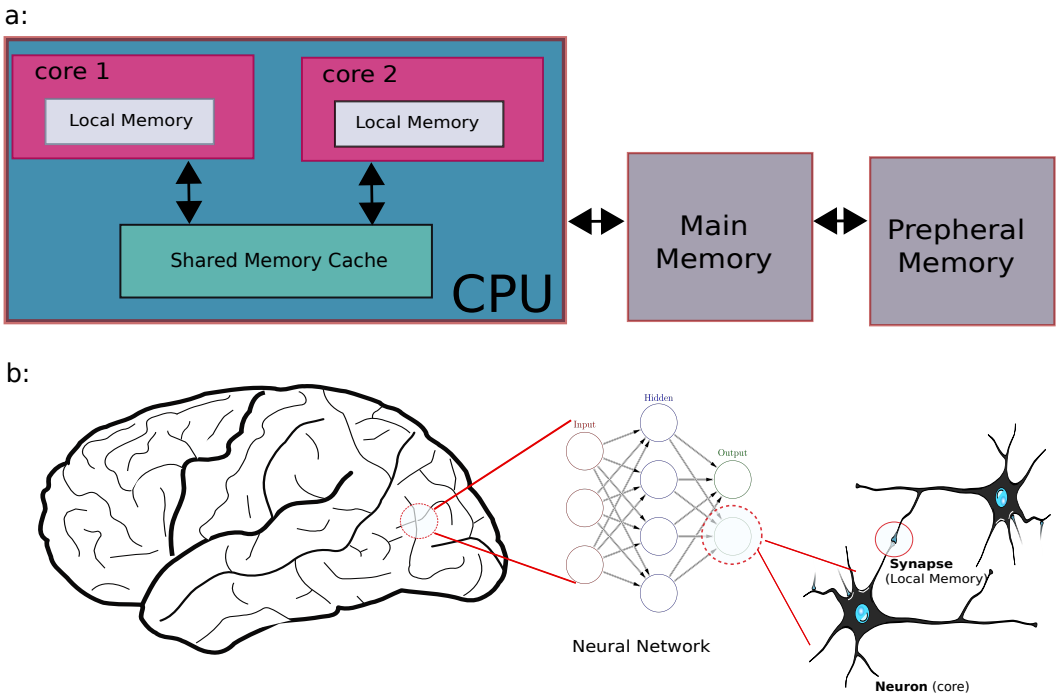

Figure 4.1: Computational architecture **a)** Von Neumman architecture, fast and costly memory are closer to cores in multiprocessor platforms as cashes and local memory as well as inexpensive and slower memory are in other layers close to magnetic memory to save the cost of CPU (memory hierarchy). **b)** Neuromorphic architecture inspired from neural networks in the biological brain, capable to conquer Von neumann bottelneck issue, performing parallel and cognitive computing, as well as considering that the synapses are local memories connected to each neurons as computational cores.

simulating retina functionalities [90]. Tobi Delbruck built on the idea of adaptive photoreceptor circuits developed in [91] and presented approaches for enhancing retinomorphic sensors consist of 128×128 pixel Dynamic Vision Sensor (DVS). DVS established a benchmark in neuromorphic vision domain with introducing Address Event Representation (AER) sensory data in which each individual pixel processed the normalized time derivative of the sensed light and provided an output in the form of spikes of the pixel addresses. In addition to vision sensory neuromorphic research, there are several neuromorphic studies using auditory and olfactory sensors [92–94] for review study in neuromorphic research using different sensory inputs, we refer the readers to [95].

More close to our research, in 2014 two distinguished articles were published that increased the scientists attentions to the general neuromorphic platforms as novel computing architectures. Merolla et al. [23] in an IBM research was sponsored by DARPA, have demonstrated a computing hardware consist of the compact modular core for large-scale neuromorphic system architecture. The cores combine digital neurons with the large synaptic array. This general purpose neuromorphic processor was built using thousands of neurosynaptic cores are involved one million neurons and 256 million of reconfigurable synapses. The second notable work published in 2014 was Spiking Neural Network Architecture (SpiNNaker) project [34]. The SpiNNaker project is a decade old, comprehensive description of the project is announced in [34]. SpiNNaker project aims to deliver a massively parallel million core architectures whose interconnections are inspired by the connectivity properties of the mammalian brain. The hardware platform is suitable to model the large-scale spiking neural networks in biological real time. Neuromorphic and neuro-inspired computing is now being adapted by an increasing number of academic and industrial different research teams. In recent few years, there have been many valuable publications explaining the use of novel materials are able to emulate some of the properties observed in biological synapses [24, 35, 45, 96–98].

Our work focuses on an alternative approach aimed at high performance computation to realize the compact, parallel, cognitive and energy-efficient architecture structure that emulate the style of

computation of the biological brain, using the Spiking Neural Network (SNN) structure, modeling the neurons as computational cores next to memristive artificial synapses as local memories to skip memory delay bottelneck similar to what is shown in Figure 4.1.b. Therefore, it is necessary to define, analyze and verify the efficient models of network topology, neuron and synapse models based on state-of-the-art technologies besides choosing the optimized learning model adapted to our platform and devices. The structure of Chapter 4 is followed by reviewing SNN and more significantly the functionality of various spike information codings. In the same section, we discuss different neural network topologies. Furthermore in the Section 4.3, different models of neuron is presented. Synapse and learning are explained in the Section 4.4 which various methods of spike-timing-dependent plasticity (STDP) [27,28] are studied comprehensively. The state-of-the-art of the most important neuromorphic platforms and projects in the world is presented in 4.5. Lateral inhibition and Homeostasis have been discussed at the discussion part of this chapter.

4.2 Spiking Neural Networks

Artificial neural networks (ANN) can generally be categorized into three generations. The first generation of neural network consisted of McCulloch and Pitts neurons [99] that the output signals are limited to discrete '0' or '1' binary values. Perceptrons, Hopfield network, Boltzmann machine and multilayer networks with threshold units are ANN examples that are classified in first generation. The second generation of neural network, by using a continuous activation function such as sigmoid, polynomial or exponential functions, the output can take analog values between '0' and '1'. Due to using analog output the network requires less neurons than the first generation class. Radial Basis Function (RBF) networks and Multi-Layer Perceptrons (MLP) are categorized under second generation class. The third generation of neural network model are networks which employ spiking neurons as computational units. In this model, the precise firing times of neurons are used for information coding. Spiking neural networks belong to the third generation of neural networks.

Indeed, artificial neural network in the first and second generation is a mathematical model of mammalian brain though, SNN is an electronic hardware neuromorphic model of the biological brain. Networks composed of spiking neurons are able to process significant amount of data using a relatively small number of spikes [100]. Due to the similarity between the biological neurons and spiking models functionality, SNNs provide powerful tools to emulate data processing in the brain, including neural information processing, plasticity and learning. Consequently, spiking networks offer solutions to a broad range of specific problems in applied engineering image detection, event detection, classification, speech recognition and many cognitive computation domain applications.

4.2.1 Spike information coding

Spike is the language of neuron communication in SNN architectures. One of the key unresolved questions in neuroscience is how information processed in the brain. The nature of the neural code is an unresolved topic of research in neuroscience. However, based on what is known from biology, a number of neural information encoding have been proposed:

1. **Rate coding**
   The rate of spikes in a time-window is counted for the information transmission. It is also called as frequency coding (Figure 4.2.a). As the intensity of a stimulus increases more, the firing rate of spikes increases more too. Rate encoding is motivated by the observation that biological neurons eager to fire more often for stronger stimuli. There are two types of rate coding namely spike-count rate and time-dependent firing rate. In spike-count rating by counting the number of spikes that are generated during a trial and dividing by the duration of the trial, we calculate the temporal average of rating. In independent firing rate, the average number of spikes over trial happens during a short interval between times t and t+Δt, divided by the duration of the

interval. Brette [101] has compared these two approaches in rate information coding in more details.

2. **Latency coding**
   In this model, information is supposed to be contained in the exact timing of a set of spikes relative to each other as it is shown in Figure 4.2.b. It is already proved that precisely timed patterns of spikes have been postulated to play a significant role in the networks of neuron in different functions [102]. Precise spike timing is one of the important parameters that control variety forms of synaptic plasticity. Latency coding by using sequences of spikes are mainly observed in feed-forward networks since noise and dynamics of recurrent networks can disrupt spike timing precision, some attempts to harvest precise spiking timing in recurrent networks have been done for example by exploring the idea of reservoir computation [103].

3. **Phase coding**
   This model generates the times of emitted spikes based on the time point in a periodic signal. In this (Figure 4.2.c) method the spike trains can encode information in the phase of a pulse respecting to the background oscillations. Phase coding method has been used both in models and experimentally. Phase coding has been suggested for the hippocampus as well [104]. Spiking networks exploring the phase coding strategy have recently been applied in tasks as olfactory systems or robot navigation [105].

4. **Rank-coding (spike-order coding)**
   In this method of spike coding, information is encoded by the order of spikes in the activity of a group of neurons as it is depicted in Figure 4.2.d. Rank-coding approach has been suggested to describe ultra-fast categorization observed in the visual system. This model assumes that each neuron emits only a single spike during a presentation of the image. This method can be implemented in a feed-forward network with inhibitory feedback connections. Thorpe and others [106] developed a spiking neural model that was able to classify static images with a processing speed comparable to that observed in humans one.

5. **Population coding**
   This coding model is a method to introduce stimuli by applying the joint activities of the group of neurons. In population coding, each neuron has a distribution of responses to the certain set of inputs, and the responses of group of neurons will be combined to present a value for the inputs (Figure 4.2.e). During the last two decades, the theory has focused on analyzing the methods in which different parameters that characterize neuronal responses to external stimuli affect the information content of these responses. Recent challenge in population coding is to develop a theory that can generate predictions for specific readout mechanisms for example for visual target information [107].

6. **Sparse coding**
   This model of coding generally refers to a representation where a few number of neurons are active, with the majority of the neurons inactive or showing low activity see Figure 4.2.f. Sparse coding has been suggested as a guiding principle in neural representations of sensory input, specially in the visual sensory system. It is also discussed that sparse coding offers a useful solution to the problem of representing natural data because such a scheme allows the system to take advantage of the sparse structure of the sensory environment. It is believed that the natural environment is inherently sparse and codes that using this structure can be both metabolically efficient and useful for learning. Sparseness can be defined over a population of neurons at a specific point in time (population sparseness) or it can be measured for a single neuron over a certain time-window [108].

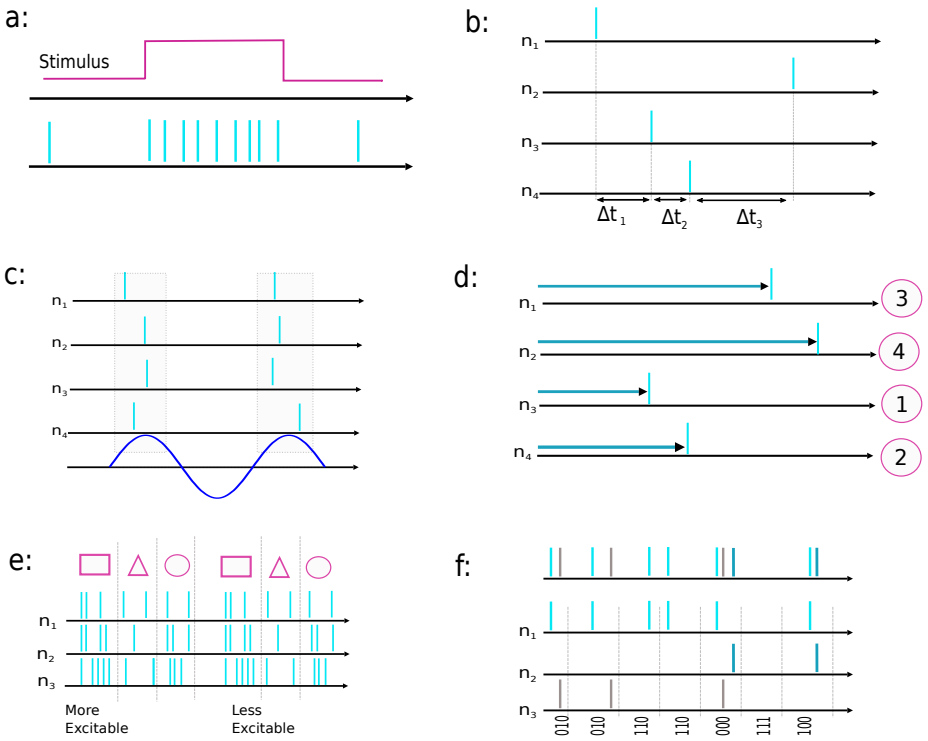

Figure 4.2: Spike information coding strategies **a)**Rate coding, **b)**Latency coding, **c)**Phase coding, **d)**Rank-coding (spike-order coding), **e)**Population coding, **f)** Sparse coding.

4.2.2 Network topology

The interconnection structure of neurons in a network of neurons is called topology, architecture or graph of an artificial neural network. The manner in which the interconnection is structured intimately is linked to the learning algorithms applied to train the neural networks. Indeed, the interconnection can be structured in numerous ways results in numerous possible topologies that are divided into two basic classes namely: Feed-Forward Neural Networks (FFNN) and Recurrent (or feedback) Neural Networks (RNN) depicted in Figure 4.3.

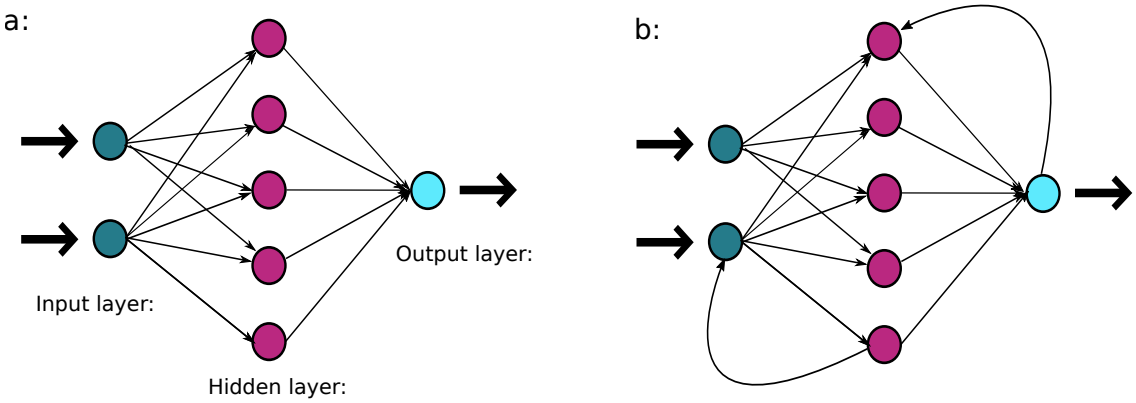

Figure 4.3: two main topologies of artificial neural network architectures **a)**Feed-Forward Neural Networks (FFNN), **b)** Recurrent Neural Networks (RNN).

Feed-Forward Neural Networks (FFNN)

The FFNN is divided into two different structure called single-layer FFNN and multilayer FFNN. The single-layer is structured as an input and output layer which is strictly a feed-forward or acyclic graph. We do not count the input layer because no calculation is performed in input nodes (neurons). The multilayer FFNN has one or more hidden layers between input and output layers similar to Figure 4.3.a which has one hidden layer. By adding one or more hidden layers, the neural network can extract the higher-order statistics which is particularly valuable when the size of the input layer is large [109]. Among the known types of neural networks (NN), the feed-forward neural networks are the mostly used because of their simplicity, flexible structure, good qualities of representation, and their capability of universal approximation. Respecting to the way of interconnectivity of the nodes (neurons) there are two kinds of feed-forward architecture:

- **fully connected**
  In this configuration, every node in each layer of the network is connected to every other node in the next layer. In fact, we can call them globally connected networks. The Restricted Boltzmann Machine (RBM) could be an example of fully connected FFNN.

- **partially connected**
  In this configuration, some communication links are missing. The convolutional neural networks is a good example for the partially connected FFNN. Partially connected topologies present a suitable alternative with a reduced degree of redundancy and thus a potential for increased efficiency of neural networks.

Recurrent Neural Networks (RNN)

The RNN is distinguished from FFNN in that it has at least one *feedback* loop connection. Recurrent neural networks can be single-layer or multilayer as well. Unlike feed-forward neural networks, recurrent networks retain a state that can represent information from an arbitrarily long context window. Although recurrent neural networks have traditionally been difficult to train, and often contain thousands of parameters, recent studies in network architectures, optimization techniques, and parallel computation have enabled successful large-scale learning to use RNN [110]. *Hopfield* [111] network is an example of the recurrent artificial neural network that is used to store one or more stable vectors. The stable vectors can be considered as memories that the network recalls them when provided with similar vectors that operate as a queue to the network memory. Other example of RNN is *Elman* network [112] that refers as a simple Recurrent Network is the special case of recurrent artificial neural networks. This type of artificial neural network has the memory that allows it to both detect and generate time-varying patterns.

Modern Neural networks

Here, we discuss recent feed-forward promising neural network which has been applied in different sensory computation applications.

- **Convolutional Neural Networks (CNN)** Convolutional network is a multi-layer feed-forward network architecture in which neurons in one layer receive inputs from multiple neurons in the previous layer and produce an output which is a threshold or sigmoidal function of the weighted sum of its inputs. The connectivity pattern between the nodes of one layer and the node of the subsequent layer, responsible for the weighted sum operation forms the convolution kernel. Each layer mainly has one or few number of convolution kernels that link the activity of a set of neurons from one layer to the target neuron of the next layer [25]. Convolutional neural networks which have been explored intensively within the neuromorphic community for visual processing tasks [113]. They are normally implemented on CPUs and GPUs which consume a significant amount of power. In recent years, System-On-Chip (SOC) solutions and FPGA platforms have

been used to implement these networks for increasing their performance while decreasing their power consumption.

- **Deep Belief Networks (DBN)** Deep learning is currently an extremely active research area in machine learning and cognitive computing society. It has obtained many successes in a wide area of applications such as speech recognition, computer vision, and natural language processing. Deep Belief Networks (DBNs), introduced by Hinton and his colleagues as a special type of deep neural networks with generative model properties [26]. This network is strctured as interconnected pairs of Restricted Boltzmann Machines. An adaptation of the neural model to allow transfer of parameters to a 784-500-500-10 layer spiking DBN was described in [97] with good performance on the MNIST digit database. DBN architecture has been implemented on a Xilinx Spartan-6 LX150 FPGA [114] with very promising classification performance results (92%) on the same MNIST database. This FPGA implementation of the DBN (also called Minitaur) contains 32 parallel cores and 128 MB of DDR2 as main memory

4.3 Spiking neuron model

The neuron is a dynamic element and processing unit that emits output pulses whenever the excitation exceeds some threshold. The resulting sequence of pulses or "spikes" contains all the information that is transmitted from one neuron to the other one. In this section, we compare the biological, artificial and spiking neuron and furthermore, we explain various model of spiking neuron models.

4.3.1 Biological, artificial and spiking neuron

A biological neuron is an electrically excitable cell that processes and transmits information by electrochemical signals. Chemical signaling occurs via synapses, specialized connections with other cells. A typical physiological neuron can be divided into three anatomical and functional parts, called *dendrites, soma* and *axon* as it is shown in Figure 4.4.a. The soma is the central part of the neuron. It contains the nucleus of the cell, where most protein synthesis occurs. The soma is considered as a central processing unit that performs an important nonlinear processing. The dendrites of a neuron are cellular extensions with many branches. Dendrites typically are considered as inputs of the neuron. The axon carries nerve signals away from the soma and typically is considered as neuron output. Neurons have only one axon, but this axon may and will usually undergo extensive branching, enabling communication with many target cells. Another term which is necessary to know in the physiological neuron is *action potential* which is a short-lasting event in which the electrical membrane potential of a cell rapidly rises and falls. It plays a central role in cell-to-cell communication. Action potentials are also called "nerve impulses" or *spikes*, and the temporal sequence of them generated by a neuron is called *spike train*. A neuron that emits an action potential is said to fire.

The artificial model of the neuron is a mathematical model of the physiological neuron. The basic computational element (neuron) is often called a node, unit or *perceptron*. Each input has an associated weight w, which can be modified and react like a biological synapse. The unit computes the f function of the weighted sum of its inputs x_i:

$$u_j = \sum_{1}^{i} w_{ji} x_i \tag{4.1}$$

$$y_j = f(u_j + b_j) \tag{4.2}$$

It is obvious in Figure 4.4.b that x_1, x_2, x_3,...x_i are neuron inputs, w_{ji} is the synaptic weights between neuron j and neuron i, b_j is bias, f is known as *activation function* or *transfer function* and y_j is output of the neuron. Based on the model and application of neural networks, there are several types of activation functions such as threshold or step function, linear function, and Non-linear (Sigmoid) function. Here to be able to Understand how neural network works we explain the functionality of

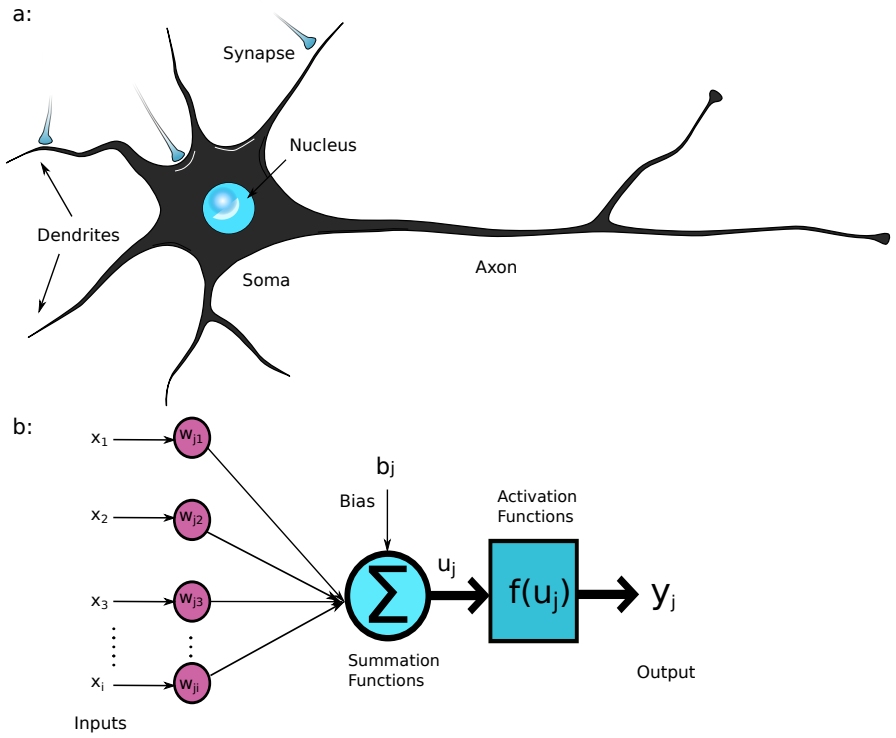

Figure 4.4: The structure of a neuron **a)**Physiological neuron, **b)** Artificial neuron model.

neuron using threshold function. Respecting to the input connections in Figure 4.4.b, we can define a threshold for transfer function f by defining threshold θ. Here, we choose $\theta = 0$ in the way we could perform a binary classification.

$$y_j = \begin{cases} 1 & \text{if} \quad u_j \geq 0 \\ 0 & \text{if} \quad u_j < 0 \end{cases} \tag{4.3}$$

where u_j is the induced local field of the neuron; which is,

$$u_j = \sum_{1}^{i} w_{ji} x_i + b_j \tag{4.4}$$

Such a model of neuron is referred to McCulloch and Pitts [99].

The Spiking neural model is an electrical model of physiological neuron that can be implemented on the circuit using traditional devices or state-of-the-art technologies e.g., CMOS transistors or on hardware platforms e.g., FPGAs. In Spiking model the neurons communicate using spikes and the input spikes make an action potential firing if inside a neuron reaches to the desired threshold (can be compared to threshold activation function in the artificial model of the neuron). Different models of the spiking neuron are proposed that here we study the main models.

Hodgkin-Huxley model

The first electrical model and in other words the first spiking model of neuron is Hodgkin-Huxley neuron model [115] which got the Nobel Prize in Physiology or Medicine. Hodgkin and Huxley performed experiments on the giant axon of the squid and found three different types of current: sodium, potassium and leak current. It was demonstrated that the ionic permeability of the membrane can be highly dependent on the membrane potential. The schematic diagram of the Hodgkin-Huxley model is shown in Figure 4.5 where E_{rest} is the membrane potential, C is the membrane capacitance, the leakage channel is described by an independent R and the conductance of this leakage is calculated $g_L = \frac{1}{R}$ the conductance the other ion channels ($g_{Na} = \frac{1}{R_{Na}}$ and $g_K = \frac{1}{R_K}$) is voltage and time dependent. The ionic

current is divided into components carried by sodium and potassium ions. Each element of the ionic current is determined by a driving force which may easily be measured as an electrical potential, E_{rest} as resting membrane potential, E_{Na} and E_K sodium and potassium potentials respectively. Current can be carried through the membrane either by charging the membrane capacitance or by moving ions through the resistances in parallel with the capacitance.

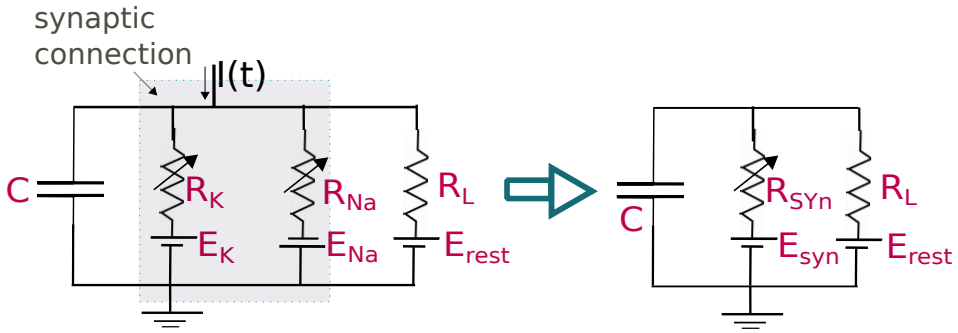

Figure 4.5: Electrical circuit represents Hodgkin-Haxley model of the neuron. **a)**Details circuit model of the neuron with sodium and potassium channels effects and leakage current, **b)** Equivalent circuit for more simplicity in solving equations.

The equivalent circuit of Hodgkin-Hulxey model is shown in the left side of Figure 4.5 that by representing the Krichhoffs law and using this circuit we can write following equations:

$$I_L(t) = \frac{V_C(t) - E_{rest}}{R_L} \tag{4.5}$$

$$I_{syn}(t) = C\frac{dV_C(t)}{dt} + \frac{V_C(t) - E_{rest}}{R_L} \tag{4.6}$$

Solving the equation 4.6 leads to an exponential answer (Equation 4.7) that can model the behavior of membrane potential.

$$V_C(t) = v_\infty(1 - \exp(-\frac{t}{\tau}) + E_{rest} \tag{4.7}$$

Respecting to the synaptic current charging if there is enough input current to membrane the neuron will fire. We note that $\tau = RC$ in Equation 4.7 is the time constant for charging and discharging the membrane.

Integrate-and-Fire (I&F) neurons

Integrate-and-Fire (I&F) neuron model are derived from the Hodgkin-Huxley neuron model. There is an important type of I&F neuron model which is named *Leaky-Integrate-and-Fire (LIF)*. There are other types of I&F models such as Quadratic-Integrate-and-Fire (QIF). The Leaky-Integrate-and-Fire (LIF) neuron model is a well-studied model of the neuron. There are three reasons for using LIF in our platform.

- The fabricated model with recent CMOS technology is available [116, 117].
- LIF works effectively in spiking and event-based networks [118].
- LIF models are quite fast to simulate, and particularly attractive for large-scale network simulations [119].

Neurons integrate the spike inputs from other neurons they are connected to. These input spikes change the internal potential of the neuron, it is known as neuron's membrane potential or state variable. When this membrane potential passes a threshold voltage due to integrated inputs, the action

potential occurs, in other words, the neuron fires. The model is described by the neuron membrane potential:

$$\tau_n \frac{\mathrm{d}v}{\mathrm{d}t} = -v(t) + RI_{syn}(t) \tag{4.8}$$

$$I_{syn}(t) = \sum_j g_{ij} \sum_n \alpha(t - t_j^{(n)}) \tag{4.9}$$

where, $v(t)$ represents the membrane potential at time t, $\tau_n = RC$ is the membrane time constant and R is the membrane resistance. Equation 4.8 describes a simple parallel resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of $I_{syn}(t)$ is due to the capacitor. The total input current, $I_{syn}(t)$, is generated by the activity of pre-synaptic neurons. In fact, each pre-synaptic spike generates a post-synaptic current pulse. The total input current injected to a neuron is the sum over all current pulses which is calculated in Equation 4.9. Time $t_j^{(n)}$ represents the time of the n_{th} spike of post-synaptic neuron j, and g_{ij} is the conductance of synaptic efficacy between neuron i and neuron j. Function $\alpha(t) = q\delta(t)$, where q is the injected charge to the artificial synapse and $\delta(t)$ is the Dirac pulse function. If $I_{syn}(t)$ is big enough where action potential can pass the threshold voltage, neuron fires. It means there are enough input spikes in a short time window. When there is no or only a few spikes in a time window, the neuron is in the leaky phase and the state variable decreases exponentially. The duration of this time window depends on $\tau_n = RC$. The equation is analytically solvable and thus we use the answer of Equation 4.8 in the network simulation when there is an input spike to improve the simulation performance. In Figure 6.2, you can see the Matlab model of a single neuron. When the input voltage passes the threshold, the neuron fires and resets to resting state. The membrane potential stays for a definite period, which is called the *refractory* period, below the reset value.

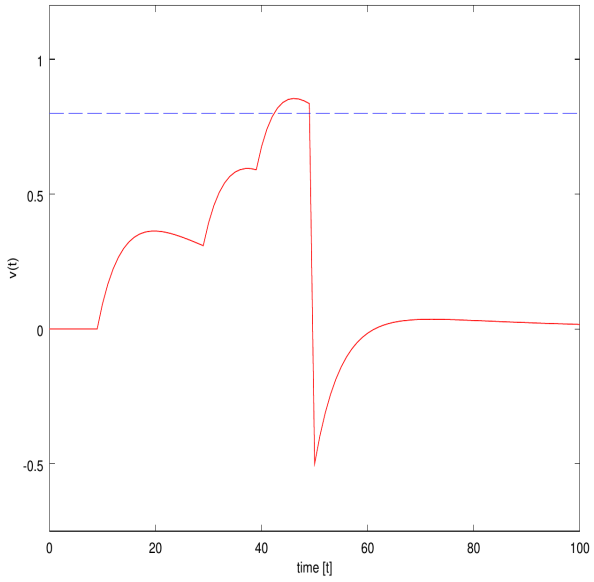

Figure 4.6: Simulation of a single LIF neuron in Matlab, the input spikes are applied in t=[10, 30, 40, 50] ms. Between 10 and 30 there is more decrease than between 30 and 40.

Izhikevich neuron model

Izhikevich neuron model [4] combines the biological plausibility of Hodgkin-Huxley model and the computational efficiency of integrate-and-fire neurons. Using this model, we can simulate tens of thousands of spiking cortical neurons in real time. The model has two main characteristics it is

computationally simple as well as capable of producing rich firing patterns that physiological neuron could produce.

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = 0.04V(t)^2 + 5V(t) + 140 - u(t) + I(t) \tag{4.10}$$

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = a.(b.V(t) - u(t)) \tag{4.11}$$

$$\text{if} V(t) \geq 30mV, \text{then} \begin{cases} V(t) \leftarrow c \\ u(t) \leftarrow u(t) + d \end{cases} \tag{4.12}$$

Where $V(t)$ and $u(t)$ are variables without any dimension, and a, b, c, and d are parameters without dimension. $V(t)$ represents the membrane potential of the neuron and $u(t)$ represents a membrane recovery variable, which accounts for the activation of K^+ ionic currents and inactivation of Na^+ ionic currents, and it provides negative feedback to $V(t)$. Synaptic currents or injected dc-currents are delivered via the variable $I(t)$. The parameter a describes the time scale of the recovery variable $u(t)$. Smaller value results in slower recovery. The parameter b presents the sensitivity of the recovery variable $u(t)$ to the subthreshold fluctuations of the membrane potential $V(t)$. Greater values couple $V(t)$ and $u(t)$ more strongly resulting in possible subthreshold oscillations and low-threshold spiking dynamics. The parameter c represents the after-spike reset value of the membrane potential $V(t)$ caused by the fast high-threshold K^+ conductances. Finally, the parameter d describes after-spike reset of the recovery variable $u(t)$ caused by slow high-threshold Na^+ and K^+ conductance. Different firing behaviors can occur in biological spiking neurons and Izhikevich model can produce them is shown in Figure 4.7.

4.4 Synapse and learning

Synapse is a specialized structure with highly plastic characteristics enabling two neurons to exchange spike signals between themselves in other words, adjusting the connection strength between neurons. Thanks to the plasticity property of synapse, we can basically say the synapse is where the learning happens in neural network system. A physiological synapse connects the axon of a presynaptic neuron (the neuron before the synapse) to the dendrite of a postsynaptic neuron (the neuron after the synapse). Two behavioral types of biological synapses are defined:chemical and electrical.

The chemical synapse is the primary definition of neurotransmitters between presynaptic and postsynaptic neurons. A neurotransmitter through a chemical synapse consists of three parts. The axon potential causes the presynaptic neuron to release a chemical substance into the synaptic *cleft* which is an intracellular space between the two neurons. The neurotransmitter then diffuses through the synaptic cleft. Moreover, the neurotransmitter causes a change in the voltage of the membrane of the postsynaptic neuron. In biological neural system, a synapse is *excitatory* if the neurotransmitter causes an increase in the voltage of the postsynaptic neuron and *inhibitory* if it causes a reducing voltage in postsynaptic neuron. An electrical synapse consists of a group of *gap junctions* occurring close together. Gap junctions are tiny channels in the cell membrane that directly connect the cytoplasms of two cells [120]. The basic mechanism of synaptic transmission is well established. A presynaptic spike depolarizes the synaptic terminal, leading to a calcium flow through presynaptic calcium channels, causing vesicles of neurotransmitter to be released into the synaptic cleft. The neurotransmitter binds temporarily to postsynaptic channels, opening them and allowing ionic current to flow across the membrane. Modeling this complete electrochemical behavior is rather challenging. The purpose of our study is not to model the exact behavior of synapse suitable for neuroscience study. The purpose of our study is to design a neuromorphic system appropriate for hardware implementation. Therefore, the behavior of synapse, neuron and model of neuron are studied to compare with recent techniques in addition to recent alternative technologies for hardware implementations.

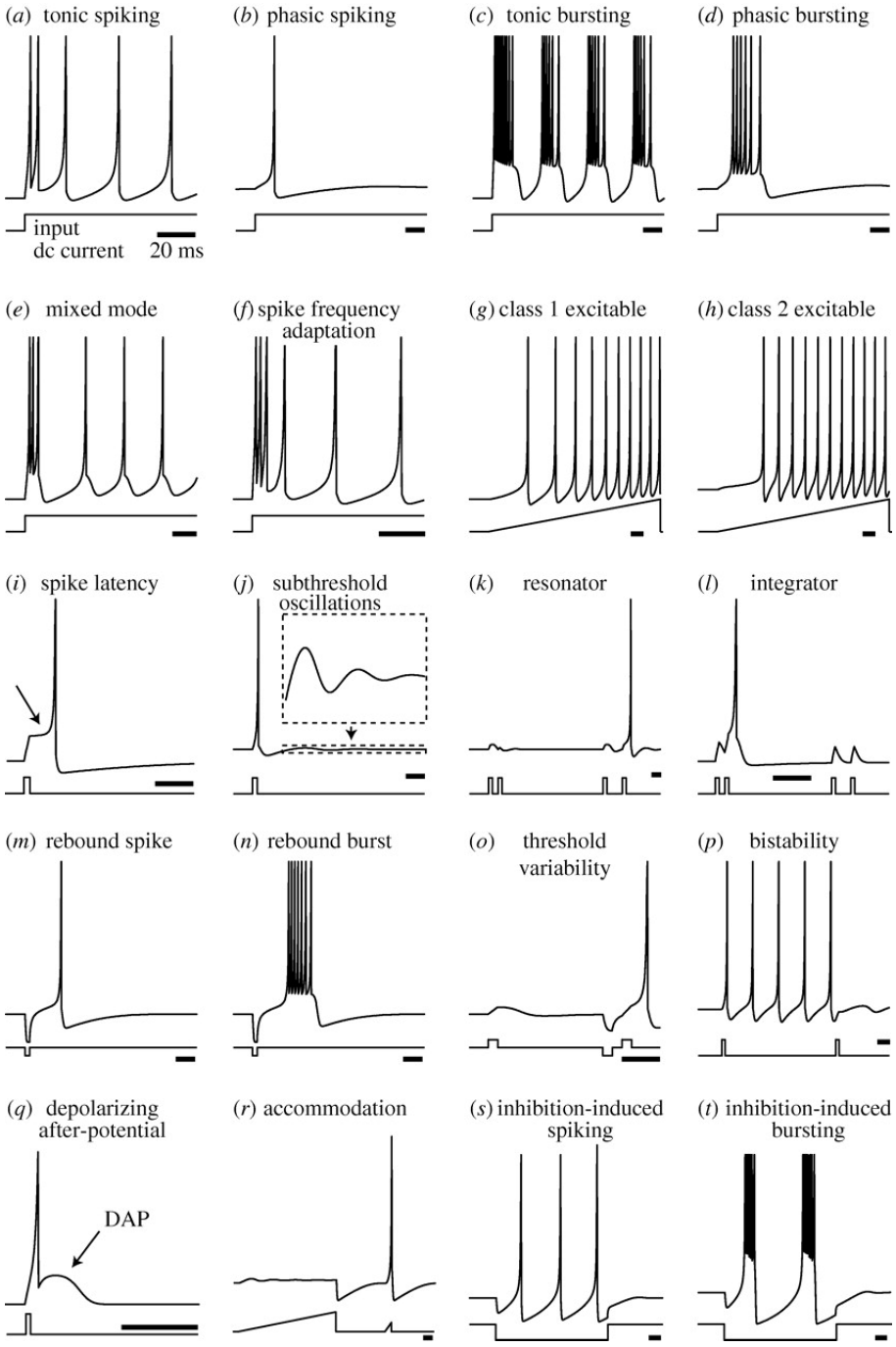

Figure 4.7: Different Known types of neurons correspond to different values of the parameters a, b, c, and d could be reproduced by Izhikevich model From [4].

4.4.1 Synaptic learning and plasticity

To be able to model a proper synapse to contribute in learning process in an efficient way in neural system, we need to analyze how learning happens in synapse. Neurons and synapses are the two basic computational units in the brain. The human brain consists of 10^{11} neurons and an extremely large number of synapses, 10^{15}, which act as a highly complex interconnection network among the neurons.

Subsequently, each neuron is connected to 1000-10000 synapses [121]. Neuron computation is performed by integrating the inputs coming from other neurons and producing spikes as based on variety of the connections. The synapses contribute to the computation by modifying their connection strength as a result of neuronal activity, which is known as the synaptic plasticity. This synaptic

plasticity is believed as the basis of adaptation and learning, even in traditional neural network models where several synaptic weight updating rules are based on Hebb's law [122, 123].

Classes of learning algorithms

The primary significance of any type of neural networks is the property of learning from the environment to improve the performance of neural network. There are several types of *learning algorithms*.Although interconnection configuration of neural network is important in learning however, learning algorithms generally differ from each other in the way in which they adjust synapse weights. Simon Haykin, mentioned five different basic algorithms for learning in his book [109] namely memory-based, Hebbian, error-correction, competitive, and Boltzmann learning. Memory-based learning functionality is based on memorizing the training data explicitly. Hebbian and competitive learning are inspired by neurobiology. Error-correction is working using optimum filtering rule and Boltzmann learning is based on ideas borrowed from statistical mechanics. In general, learning algorithms can be divided into *supervised or with teacher* learning, *semi-supervised* learning, and *unsupervised or without teacher* learning algorithms (see Figure 4.8).

- Supervised algorithms
  Teacher has the knowledge of environment and this knowledge will be shared with the network as some examples of inputs and their corresponding outputs. The supervision is continued letting a modification rule adjust the synapses until the desired computation emerges as a consequence of the training process. Then the supervision process is stopped and network must have the similar outputs with the specific inputs while the supervision was working. Error-correction algorithms which include the back-propagation using *gradient descent* is an example of supervised algorithms, other well-known supervised algorithms are support vector machines (SVM) and Bayesian type of learning algorithms. In fact, we put label on the data in training and check those labels in testing. This type of algorithms are used for regression and classifications.

- Semi-supervised algorithms
  Semi-supervised learning falls between supervised learning and unsupervised learning. Labeled data are often difficult, expensive, and time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect. Semi-supervised uses large amount of unlabeled data, together with the labeled data, to build better classifiers. Intuitively, in semi-supervised learning we can consider the learning problem as an exam and labeled data as the few example problems that the teacher solved in the course. The teacher also provides a set of unsolved problems. Semi-supervised learning requires less human effort and gives higher accuracy, therefore it is of great interest both in theory and in practical application.

- Unsupervised algorithms
  There is no teacher and environment is unknown for the network too. There is no labeled data output in unsupervised learning. Unsupervised learning can be thought of as finding patterns in the data above and beyond what is considered as pure unstructured noise. One very simple classic example of unsupervised learning is clustering. Hebbian plasticity is a form of unsupervised learning, which is useful for clustering input data but less appropriate when a desired outcome for the network is known in advance.

Short-term and Long-term plasticity

Physiological synapses have an inherent dynamics, that controls how the pattern of amplitudes of postsynaptic responses depends on the temporal pattern of the incoming spike train. Indeed, each effective spike evokes a spike response in the postsynaptic neuron that is fewer (depression) or bigger (facilitation or potentiation) than the previous one. The strength of synaptic connections or weights are

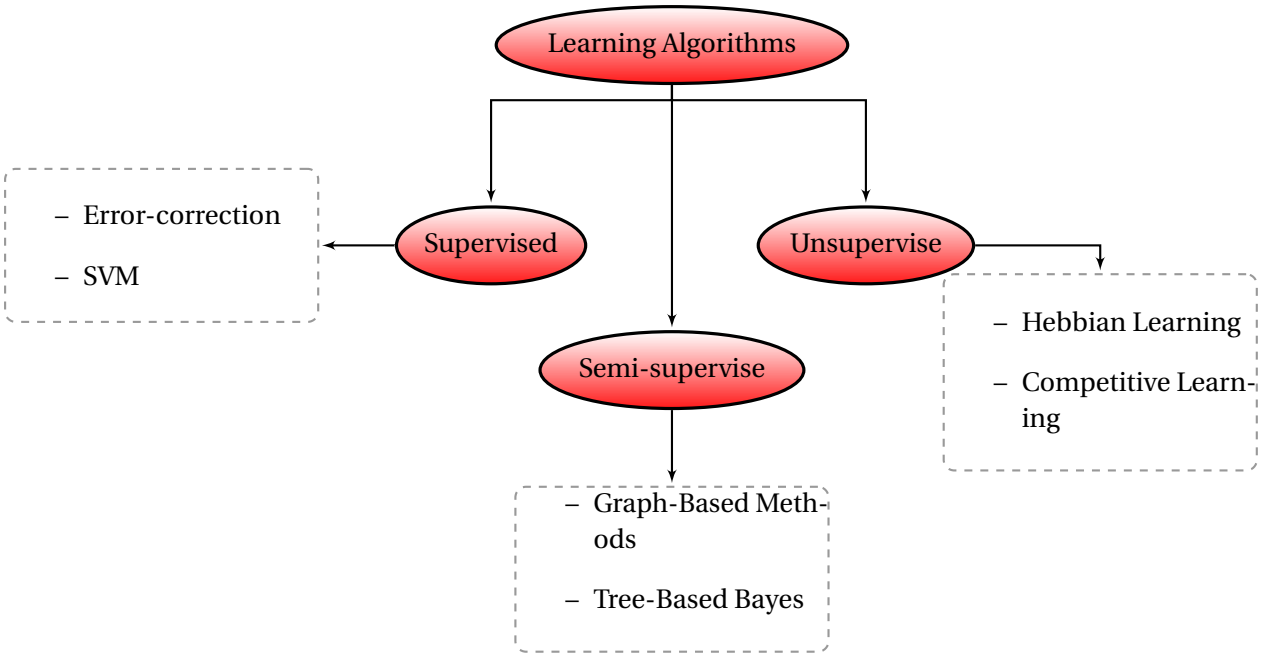

Figure 4.8: Different learning classifications.

caused by memorizing events, underling the ability of the brain to memorize. In the biological brain, short-term plasticity refers to a number of phenomena that affect the probability that a presynaptic action potential opens postsynaptic channels and that takes from milliseconds to tens of seconds. Short-term plasticity is achieved through the temporal enhancement of a synaptic connection, which then quickly decays to its initial state. Short-term plasticity depends on the sequence of presynaptic spikes Figure 4.9.

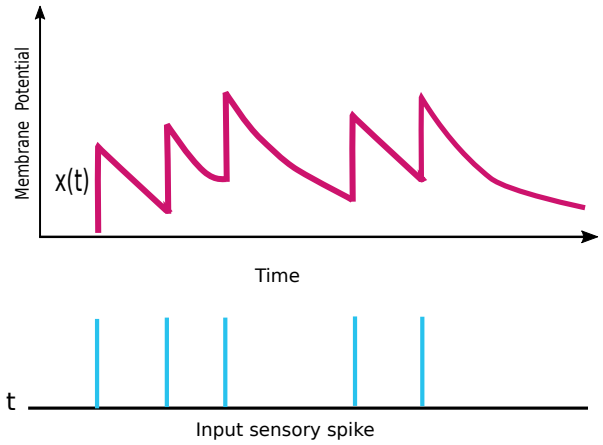

Figure 4.9: Implementation of plasticity by local variables which each spike contributes to a trace x(t). The update of the trace depends on the sequence of presynaptic spikes

.

In local learning process, iteration of stimulation leads to a more stable change in the connection to achieve long-term plasticity. Long-term plasticity is sensitive to the presynaptic firing rate over a time scale of tens or hundreds of seconds [124]. In general, synapses can exhibit potentiation and depression over a variety of time scales, and multiple components of short- or long-term plasticity. Thus, four combination are possible from short and long term plasticity: Short-term potentiation (STP), short-term depression (STD), Long-term potantiation (LTP) and long-term depression (LTD) [125].

Spike-Timing Dependent Plasticity (STDP)

Most of the plasticity models employed in the neuroscience and neuromorphic approach were inspired by Hebb's (1949) postulate that explains the way that synapse connection weight should be modified: *When an axon of cell A is near enough to excite cell B or 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.*

Local learning rules aim to deal with information encoded by precise spike timing in local synaptic memory. One of the most commonly studied and used rules is spike-timing-dependent plasticity (STDP) [27, 28] that can be considered as a spike-based producing of Hebbian learning. Based on the STDP modification rule, the synaptic changing is reinforced while both the pre- and post-synaptic neurons are active, nothing prevents the synapses from strengthening themselves boundlessly, which causes the post-synaptic activity to explode [126] . Indeed, the plasticity depends on the time intervals between pre- and postsynaptic spikes or in the other words, the concept of timing-LTP/LTD. The basic mechanisms of plasticity in STDP is derived from the long term potentiation (LTP) and the long term depression (LTD). Pre-synaptic spikes that precede post-synaptic action potentials produce long-term potentiation (LTP), and pre-synaptic spikes that proceed post-synaptic action potentials generate long-term depression (LTD).

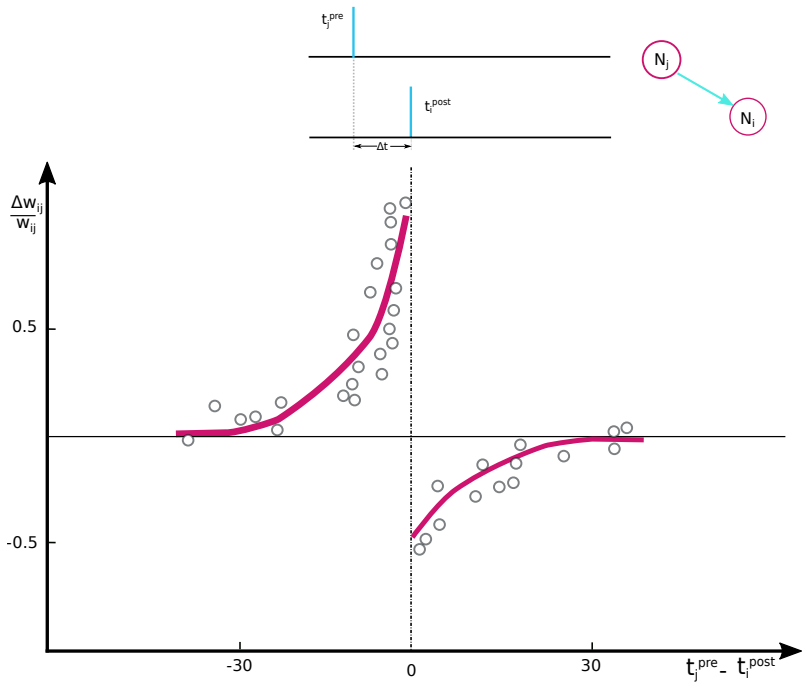

Figure 4.10: Basic of spike-timing-dependent plasticity. The STDP function expresses the change of synaptic weight as a function of the relative timing of pre- and post-synaptic spikes.

The basic configuration of STDP learning is depicted in Figure 4.10. The rate of weight changing Δw_{ji} of a synapse from a presynaptic neuron j to postsynaptic neuron i depends on the relative timing between presynaptic spike and postsynaptic spikes. Let us name the presynaptic spike arrival times at synapse j by t_j^{pre} where $pre = 1, 2, 3, ...$ counts the presynaptic spikes. Similarly, t_i^{post} with $post = 1, 2, 3, ...$ labels the firing times of the postsynaptic neuron. The total weight change w_{ji} induced by Equation 4.13 is then ([27])

$$\Delta w = \sum_{pre=1}^{n} \sum_{post=1}^{m} W(x)(t_i^{post} - t_j^{pre}) \tag{4.13}$$

where W(x) is called a STDP learning function. Based on Zhang et al. [127] in their experimental work

presented W(x) as:

$$W(x) = \begin{cases} A_+ e^{(\frac{-x}{\tau_+})} & \text{if} \quad x \geq 0 \\ -A_- e^{(\frac{x}{\tau_-})} & \text{if} \quad x < 0 \end{cases} \tag{4.14}$$

where the parameters A_+ and A_- depend on the current value of the synaptic weight w_{ij}. The time constants τ_+ and τ_- are on the order of 10 ms.

Different models for STDP learning

Multiple pre- or postsynaptic spikes occurring across a synapse in an interval of time, the plasticity modification depends on their timing in a more complex manner. For instance, pair-based STDP models present "pre-post-pre" and "post-pre-pos" triplets of spikes with the same pairwise intervals should induce the same plasticity, however experimental studies demonstrated that these two triplet patterns have different effects [5, 128].

- **Pair-based STDP**
  In this model of spike counting in the STDP interpret the biological evidence in terms of a pair-based update rule, i.e. the modification of a synaptic weight depends on the temporal difference between pairs of pre- and postsynaptic spikes:

$$\begin{cases} W_{inc}(x) & = F_{inc}(w).e^{(-\frac{|\Delta t|}{\tau_+})} \quad \text{if} \quad \Delta t > 0 \\ W_{dec}(x) & = -F_{dec}(w).e^{(-\frac{|\Delta t|}{\tau_-})} \quad \text{if} \quad \Delta t < 0 \end{cases} \tag{4.15}$$

  In Equation 4.15, $\Delta t = t_i^{post} - t_j^{pre}$ is the temporal difference between the post- and the presynaptic spikes, and $F_{inc}(w)/F_{dec}(w)$ presents the dependence of the update on the current synaptic weight. A pair-based model is fully specified by defining the form of $F_{inc}(w)/F_{dec}(w)$ as well as determining which pairs are taken into account to perform a new modification. A pair-based weight modification rule can be implemented using two local variables: one for a low-pass filtered version of the presynaptic spike train and another one for the postsynaptic spike train as it is shown in Figure 4.11. Let us suppose that each spike from presynaptic neuron j contributes to a trace $x_j(t)$ at the synapse weight then we can write:

$$\frac{dx_j(t)}{dt} = -\frac{x_j(t)}{\tau_{pre}} + \sum_{t_j^{pre}} \delta(t - t_j^{pre}) \tag{4.16}$$

  where t_j^{pre} represents the history of the firing times of the presynaptic neuron. In particular, the variable is increased by an amount of one at the arrival time of a presynaptic spike and reduces exponentially with time constant τ_{pre} afterwards. Similarly, each spike from postsynaptic neuron i contributes to a trace $x_i(t)$:

$$\frac{dx_i(t)}{dt} = -\frac{x_i(t)}{\tau_{post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{4.17}$$

  where t_i^{post} presents the firing times of the postsynaptic neuron. Similar to presynaptic spike, a decrease of the weight is induced proportionally to the momentary value of the postsynaptic trace $x_i(t)$. The steady-state average for synaptic strength in pair-based STDP has a stable nontrivial mean if the depression window is larger than the potentiation window [5]. This fixed point is unique, so the mean of the steady-state distribution of synaptic weights converges to this value regardless of its initial value. The stability of the mean is not a sufficient condition for the steady-state distribution of synaptic strengths to be fully stable, each synapse must also have a stable deviation from the mean. The connection strength of a particular synapse can be presented as $w = w + \delta w$, where δw is the deviation of the synapse from the mean. If the deviation is going to grow over time, the synapses will drift away from the mean and the distribution will be partially stable. If the deviation tends to decrease, the synapses will cluster around the mean and the distribution will be stable.

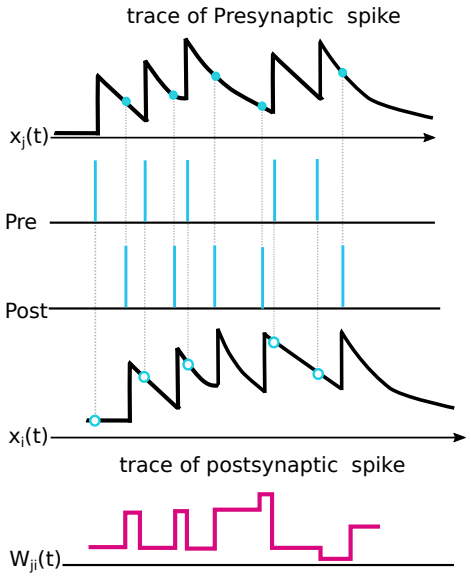

Figure 4.11: Pair-based STDP using local variables. The spikes of presynaptic neuron j leave a trace $x_j(t)$ and the spikes of the postsynaptic neuron i leave a trace $x_i(t)$. The update of the weight W_{ji} at the moment of a postsynaptic spike is proportional to the momentary value of the trace $x_j(t)$ (filled circles). This gives the amount of potentiation due to pre-before-post pairings. Analogously, the update of W_{ji} on the occurrence of a presynaptic spike is proportional to the momentary value of the trace $x_i(t)$ (unfilled circles), which gives the amount of depression due to post-before-pre pairings

■ **The triplet model**

The standard pair-based STDP models predict that if the repetition frequency is increased, the strength of the depressing interaction becomes greater, leading to less network potentiation. The frequency-dependence of STDP experiments can be accounted for if one assumes that the basic building block of potentiation during STDP experiments is not only a pair-wise interaction but also could be a triplet interaction between two postsynaptic spikes and one presynaptic spike. Pfister & Gerstner [129] to propose the triplet model, which takes into account interactions of spikes beyond pre-post pairings. This model is based on sets of three spikes, one presynaptic and two postsynaptic. For a pre-post-pre triplet, the first presynaptic spike enforces extra depression on the synapse, additionally for a post-pre-post triplet the first postsynaptic spike enforces extra potentiation. The triplet model sums the contributions of all previous pre- and postsynaptic spikes as well as all pre-post pairings. Pfister & Gerstner [129] also provided a version of the triplet model based only on nearest neighboring spikes, but the qualitative behavior of both all to all and nearest neighboring versions is similar.

Similarly to pair-based rules, each spike from presynaptic neuron j contributes to a trace $x_j(t)$ at the synapse:

$$\frac{dx_j(t)}{dt} = -\frac{x_j(t)}{\tau_{pre}} + \sum_{t_j^{pre}} \delta(t - t_j^{pre}) \tag{4.18}$$

where t_j^{pre} presents the firing times of the presynaptic neuron. In contrast with pair-based STDP, each spike from postsynaptic neuron i contributes to a fast trace $x_i(t)$ and a slow trace $x_i'(t)$ at the synapse:

$$\frac{dx_i(t)}{dt} = -\frac{x_i(t)}{\tau_{1post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{4.19}$$

$$\frac{dx_i'(t)}{dt} = -\frac{x_i'(t)}{\tau_{2post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{4.20}$$

where $\tau_{1post} < \tau_{2post}$, how the triplet model works is depicted in Figure 4.12. In this model, LTD is induced as in the standard STDP pair model in Equation 4.15, i.e. the weight change is proportional to the value of the fast postsynaptic trace $x_i(t)$ evaluated at the arrival of a presynaptic spike. The new feature of the rule is that LTP is pursued by a triplet effect: the weight change is proportional to the value of the presynaptic trace $x_j(t)$ evaluated at the arrival time of a postsynaptic spike as well as to the slow postsynaptic trace $x'_i(t)$ from previous postsynaptic spike. The main functional advantage of a triplet STDP rule is that it can be mapped to a Bienenstock-

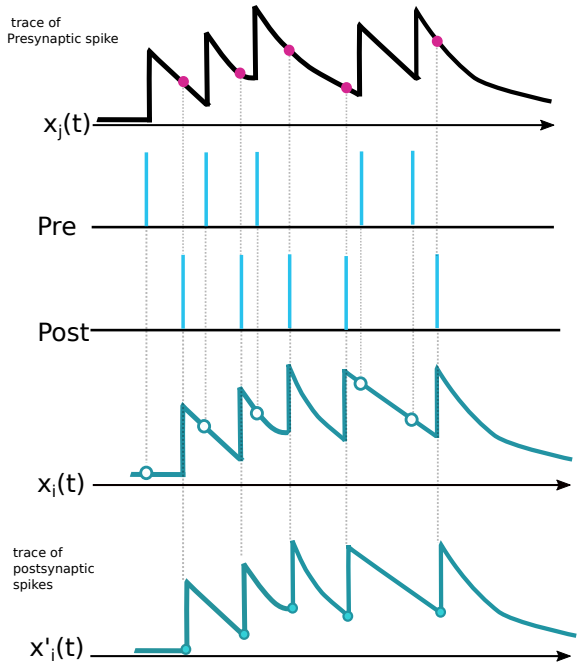

Figure 4.12: Triplet STDP model using local variables. The spikes of presynaptic neuron j contribute to a trace $x_j(t)$, the spikes of postsynaptic neuron i contribute to a fast trace $x_i(t)$ and a slow trace $x'_i(t)$. The update of the weight W_{ji} at the arrival of a presynaptic spike is proportional value of the fast trace $x_i(t)$ (green unfilled circles), as in the pair-based model. The update of the weight W_{ji} at the arrival of a postsynaptic spike is proportional to the value of the trace $x_j(t)$ (red filled circles) and the value of the slow trace $x'_i(t)$ just before the spike (green filled circles).

Cooper-Munro learning rule [130]. It means if we assume that the pre- and postsynaptic spike trains are managed by Poisson statistics, the triplet rule presents depression for low postsynaptic firing rates and potentiation for high postsynaptic firing rates.

- **Suppression model**

  Plasticity experiments using triplets of spikes demonstrated different effects than the hippocampal results. In the synapses of the visual cortex of rats, pre-post-pre triplets induce potentiation while post-pre-post triplets induce depression. These results led Froemke et al. [131] to develop the suppression model, in which STDP is induced by nearest neighbor pre- and postsynaptic spikes. In this model of STDP, the plasticity is computed from the standard pair-based STDP curve, however the impact of the presynaptic spike in each pair is suppressed by previous presynaptic spikes and, similarly, the plasticity induced by the postsynaptic spike in each pair is suppressed by previous postsynaptic spikes as it is shown in Figure 4.13.

  The suppression is maximal after each pre- or postsynaptic spike, and it decreases exponentially as the interval between consecutive pre- or postsynaptic spike increases. In a post-pre-post sequence of spikes, the timing of the first post-pre pairing was the best predictor for the synaptic weight modification. Moreover, in a pre-post-pre sequence of spikes, the first pre-post pair

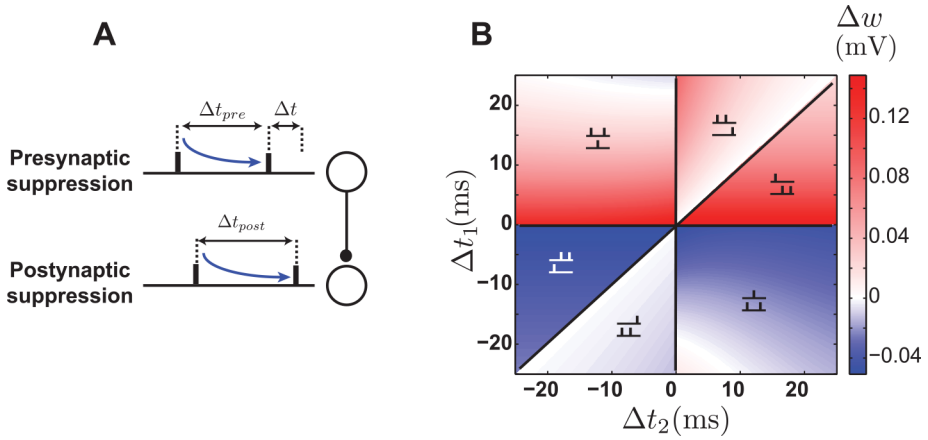

Figure 4.13: The suppression STDP model. A) Spike interactions in the suppression model, in which the impact of the presynaptic spike in a pair is suppressed by a previous presynaptic spike (top), and the impact of the postsynaptic spike is suppressed by a previous postsynaptic spike (bottom). B) Plasticity in the suppression model induced by triplets of spikes: pre-post-pre triplets induce potentiation (top left), and post-pre-post triplets induce depression (bottom right), From [5].

induces potentiation, nevertheless the amount of depression induced by the second post-pre pair is suppressed by the first presynaptic spike. In the suppression STDP model, synaptic weight modification is presented by

$$\Delta w = (1 - e^{-\frac{\Delta t_{pre}}{\tau_{pre}}})(1 - e^{-\frac{\Delta t_{post}}{\tau_{post}}}) \times \begin{cases} A_{inc}.e^{(-\frac{\Delta t}{\tau_{inc}})} & \text{if} \quad \Delta t \geq 0 \\ -A_{dec}.e^{(\frac{\Delta t}{\tau_{dec}})} & \text{if} \quad \Delta t < 0 \end{cases} \tag{4.21}$$

where Δt_{pre} is the interval between the presynaptic spike in the pair and its preceding presynaptic spike, and Δt_{post} is the interval between the postsynaptic spike and its preceding spike. This model introduces a proper fit to triplet and quadruplet protocols particularly in the visual cortex, and also represents a much better prediction for synaptic changing due to natural spike trains [131]. Nonetheless, it does not predict the increase of LTP with the repetition frequency.

■ **Voltage dependence model**

Experimental model of Spike-Timing Dependent Plasticity recommends that synaptic weight modifications are caused by the tight temporal correlations between pre- and post- synaptic spikes. However, other experimental protocols where presynaptic spikes are paired with a fixed depolarization of the postsynaptic neuron (e.g. under voltage clamp) show that postsynaptic spikes are not necessary to induce long-term potentiation and depression of the synapse [132].

It has been discussed whether the voltage dependence is more fundamental than the dependence on postsynaptic spike. In fact, voltage dependence alone can produce a behavior similar to STDP learning, as the membrane potential reacts in a particular manner in the vicinity of a spike it means high shortly before a spike, and low shortly after. Alternatively, a dependence on the slope of the postsynaptic membrane potential has been shown to regenerate the properties of STDP weight change curve. The voltage effects caused by back-propagating spikes is implicitly contained in the mechanistic formulation of STDP models outlined above. In particular, the fast postsynaptic trace $x_i(t)$ in the triplet model can be considered as an approximation of a backpropagating action potential. In contrast, a standalone STDP rule does not automatically generate a voltage dependence. Furthermore, synaptic effects caused by subthreshold depolarization in the absence of postsynaptic firing cannot be modeled by standard STDP or triplet models.

■ **The NMDAR-based model**

The NMDAR-based model was proposed for the first time in [133] and “NMDAR-based model”,

is phenomenologically based on the kinetics of the N-Methyl-D-Aspartate receptoras. It is a description for the main STDP experiments and resemble both the triplet and suppression models and and it is sensitive to spike interactions beyond pre-post pairings. The NMDAR-based model is proposed to have three states, rest, up and down. Every presynaptic spike moves a portion of the NMDARs in the rest state into the up state, and every postsynaptic spike transitions a portion of the rest-state into the down state.The NMDAR goes exponentially back to the rest state while there is no spike.

This model introduce two second messengers called “up” and “down” messengers, which cause to potentiation and depression, respectively which can be in active or inactive states. The arrival of presynaptic spike causes to a fraction of the inactive down messengers a transition to the active state. Similarly, when a postsynaptic spike arrives in the synapse, it shifts a portion of the inactive up messengers into their active state. The messengers go back to their inative states when there is no spike. Subsequently, when a presynaptic spike arrives, the synapse is depressed proportionally to the value of active down messenger, provided that this is greater than a threshold θ^{dn} . Similarly, each postsynaptic spike leads synapse to potentiate proportionally to the amount of active up messenger provided that it is greater than a threshold θ^{up}. Therefore, the presynaptic spike has three roles in this model: it transmits resting NMDARs into the up state, it activates the down messenger, and it induces depression. The postsynaptic spike also plays three roles: it movement resting NMDARs into the down state, it activates the up messenger, as well as it induces potentiation see Figure 4.14. Shortly, the specific property of the NMDAR-

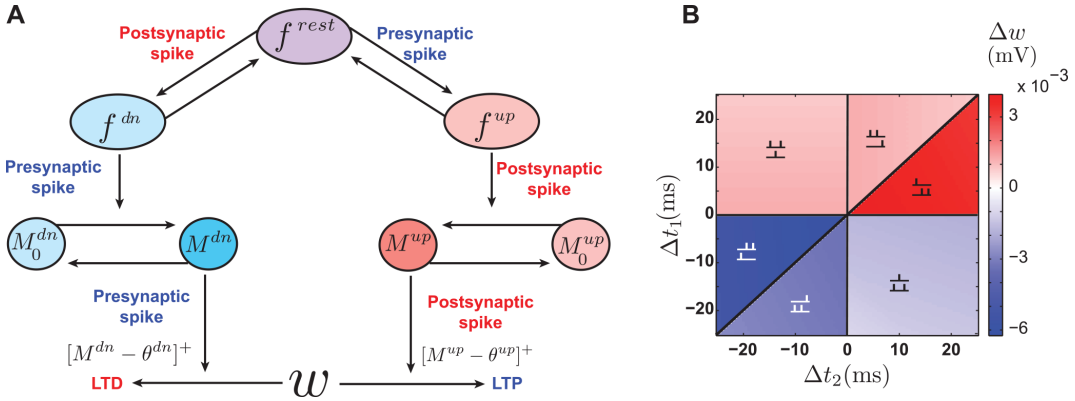

Figure 4.14: The NMDAR-based model. **A)** Schematic illustration of spike interactions in the NMDAR-based model. The presynaptic spike up-regulates f rest, activates M dn and depresses the synapse. The postsynaptic spike down-regulates f_{rest}, activates M_{up} and potentiates the synapse. **B)** The effect is asymmetric, with pre-post-pre triplets inducing potentiation (top left) and post-pre-post depression (bottom right), From [5].

based learning model compared to the pair-based model is the possibility of a stable synaptic distribution and anti-Hebbian competition when the maximum depression is significantly larger than the maximum potentiation.

■ **Other methods**

In addition to the reviewed methods above, there are other types of STDP models for learning such as supervised [134] and reinforcement learning [135]. However, due to the unsupervised nature of STDP learning that is interesting for neuro-inspired computation, we do not focus on them in this study. Pair-based STDP models can be categorized into three classes: weight dependence, spike-pairing scheme and delay partition. Choosing each category should be made consciously and take into account the relevant available experimental findings. The recent available evidences shows that both potentiation and depression are dependent on the weight. Accordingly it is recommended to begin with very simplified models. Moreover, we know that STDP models which assume some weight dependence generate different behavior

from the additive model. The pair-based and triplet models are partially stable and use Hebbian competition. The Suppression and NMDAR-based have more stability but they use anti-Hebbian competition. The main challenge in this domain is to perform analytical and simulation studies that are able to identify and characterize their composite effects, and investigate their functional consequences.

4.5 Hardware spiking neural network systems

Specific application domains such as Big Data classification, visual processing, pattern recognition and in general sensory input data, require information processing systems which are able to classify the data and to learn from the patterns in the data. Such systems should be power-efficient. Thus researchers have developed brain-inspired architectures such as spiking neural networks. For large scale brain-like computing on neuromorphic hardware, there are four approaches:

1. Microprocessor based approaches where the system can read the codes to execute and model the behavior of neural systems and cognitive computation such as the SpiNNaker machine [34].

2. Fully digital custom circuits where the neural system components are modeled in circuit using state-of-the-art CMOS technology e.g., IBM TrueNorth machine [23].

3. Analog/digital mixed-signal systems that model the behavior of biological neural systems, e.g. the NeuroGrid [35] and BrainScales [36] projects.

4. Memristor crossbar array based systems where the analog behavior of the memristors emulate the synapses of a spiking neural network.

In the following, we give some details about these approaches and compare their performance.

SpiNNaker is a massively parallel and processor-based (ARM processor) system with the purpose of building large scale spiking neural networks simulations. It is highly scalable and capable to simulate a network from thousands to millions of neurons with varying degree of connectivity. It proposes to integrate 57,600 custom VLSI chips based on the AER (Address Event Representation) communication protocol [136]. Each chip contains 18 fixed-point advanced RISC ARM968 processing cores next to the custom routing infrastructure circuits which is dedicated 96 kB of local memory besides 128 MB of shared Dynamic Random Access Memory (DRAM) as it is depicted in Figure 4.15.a. The router memory consists of a three-state 1024×32 bits Content Addressable Memory (CAM) and a 1024×24 bits Random Access Memory (RAM). Going more to the details, each ARM core has a local 32 kB instruction memory and 64 kB data memory. Regarding to the architecture and design properties, SpiNNaker offers very fast simulation of large scale neural networks. It has a remarkable flexibility for arbitrary connectivity for network architecture and various neurons, synapses and learning algorithms. However, the system still uses von Neumann architecture with a large extent of memory hierarchies found in conventional computers with memory wall bottleneck issues. Although using low-power ARM processors dedicated to power-efficient platforms used in training and robotic applications with four to 48 nodes, SpiNNaker consumes a relatively small amount of power. However, the largest machine with the ability to simulate of one percent of a human brain and incorporating over a million ARM processor cores, still requires up to 75 kW of electrical power.

IBM designed a scalable, flexible and non-von Neumann full custom spiking neural network named “TrueNorth”. Although TrueNorth uses transistors as digital gates, they use event-driven method to communicate in fully asynchronous manner. The structure of TrueNorth consists of 5.4 billion transistors to build 4096 neurosynaptic cores. Each core includes 256 digital LIF neurons, 256×256 binary programmable synapses, and asynchronous encoding/decoding and routing circuits. Each synapse has a binary behavior that can be individually turned on or off and can be assigned to model one type of inhibitory and two types of excitatory synapse with different weights. Neuron dynamics has a global 1 kHz clock and so is discretized into 1 ms time steps. Regarding to the synaptic

matrix, each neuron can be connected to one up to 256 neurons of a destination core. The routing in TrueNorth is less flexible than in SpiNNaker, however TrueNorth can distribute the system memory includes core synaptic matrix and routing table entries (Figure 4.15.b) The architecture thus supports dynamics of connectivity that includes feed-forward, recurrent, and lateral connections. The power consumption is 20 mW/cm^2, though the traditional central processing unit (CPU) is 50 to 100 W/cm^2. In this platform the synapses do not implement any plasticity mechanism, therefore they are not able to perform on-line learning.

The BrainScales project (Brain-inspired multiscale computation in neuromorphic hybrid systems) is the successor of FACETS [137] project. This project proposes the design and implementation of a custom analog/digital mixed-signal simulation engine that is able to implement the differential equations with an acceptable accuracy. This computational neuroscience model is provided by neuroscientists, and reproduces the results obtained from numerical simulations executed on conventional computers. The Heidelberg University BrainScales project (HICANN chip) aims to produce a wafer-scale neural simulation platform, in which each 8 inch silicon wafer integrates 50×106 plastic synapses and 200,000 biologically realistic neuron circuits (see Figure 4.15.c). In order to have a scalable size with maximum number of processors on the wafer, relatively small capacitors have been applied for modeling the synapses and neurons. Accordingly, using the large currents generated by the above-threshold circuit and the small capacitors, the BrainScales circuits are not able to achieve the long time-constants required for interacting with real-time environments. However, the speed of network components operations compared to biological elements reactions is accelerated by a factor of 10^3 or 10^4 which can reduce the simulation time dramatically. Furthermore, it needs large bandwidth and fast switching and still high-power circuit for propagating spikes across the network [9].

NeuroGrid is another big project developed at Stanford University that emulates neuromorphic engineering vision, sub-threshold network components circuits and uses analog/digital mixed-signal to model continuous time for network elements. This meuromorphic platform simulates a million neurons with billions of synaptic connections in real-time. Such as TrueNorth and BrainScales the architecture of Neurogrid is non-von Neumann. Neurogrid emulates four network elements: axon, dendrite, soma and synapse. Only the axon circuit is digital and the other elements are modeled in the analog circuits due to the better energy efficiency. NeuroGrid consists of 16 standard CMOS "NeuroCores" (see Figure 4.15.d) integrated on a board that works using 3 W of power energy connected in a tree network, with each NeuroCore consisting of a 256×256 array of two-compartmental neurons. The synaptic circuits are shared among the neurons while different spikes can be assigned to the same synapse. The main goal of neuromorphic systems is to interact with real physical environments and process the natural signals with physiological time-scales, Neurogrid has long time constants in the range of tens of milliseconds. Consequently, this long time constants limitation causes difficulty in using typical VLSI for design and implementation. Neurogrid and BrainScales similarly use the temporal dynamic of memory elements to store the state of the network. Accordingly, these two projects have the capability of local learning using the STDP learning rule.

An alternative to these architectures, that has been proposed by several authors [12, 14, 43, 47, 138], is to use memristive devices as synapses in neuromorphic circuits. This has the potential to lower the energy consumption by a large proportion. It has also been showed that the memristors can emulate the STDP learning rule, and thus lead to unsupervised learning circuits. We have thus chosen to study this kind of architecture and, in particular, to check how some parameters of the architecture or of the devices influence the learning capabilities of the circuit.

4.6 Discussion

Still for a network simulation and implementation of neuromorphic spiking system, we need more techniques such as homeostasis and lateral inhibition to support learning process for an optimized system. Homeostasis is used in the SNN to adapt the threshold level of neurons to learning in SNN. Another consideration is lateral inhibition while we are using unsupervised learning methods such as

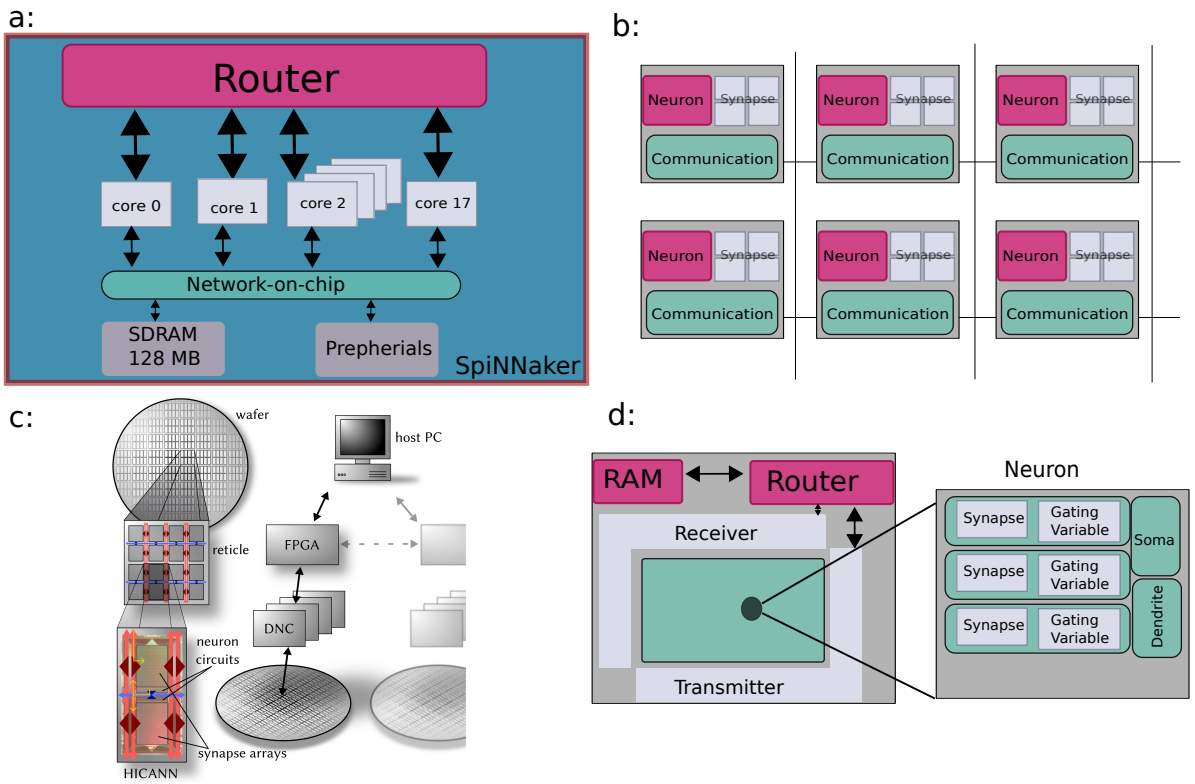

Figure 4.15: Large scale spiking neural network systems, **a)** Principal architectural parts of a SpiNNaker processing node, **b)** In TrueNorth, conceptual blueprint of an architecture like the brain, tightly integrates memory, computation, and communication in distributed modules that operate in parallel and communicate via an event-driven platform. **c)** Schematic of HICANN board in BrainScales project, **d)** The chip comprises a 256 × 256 array of neuron elements, an asynchronous digital transmitter for sending the events generated by the neurons, a receiver block for accepting events from other sources, a router block for communicating packets among chips, and a memory blocks for supporting different network configurations.

STDP. Here we discuss Winner Take-All (WTA) method.

Homeostasis

Homeostasis addresses a general principle that safeguards the stability of natural and artificial neural systems, where stability is understood in its more classical sense of robustness against external perturbations. Homeostasis is a fundamental concept in neuropsychology, psychophysiology and neuroscience. Homeostasis will be defined as negative feedback control. In physiological neural systems, the synaptic input of a neuron is changing over time due to the external neural drive and learning results of synaptic plasticity. From a perspective of metabolic cost, a restricted activity range of a neuron is really meaningful.

In STDP learning, the synaptic input of a neuron may strongly increase or decrease for a long time and as a result the neural activity will be drifting to an extremely high or low level. Homeostasis is a neuron property that regulates the firing threshold to prevent a neuron to be hyperactive [139]. The idea is to use an adaptive threshold for the membrane potential. If the neuron is too much active in a short time window the threshold grows gradually; likewise, when a neuron is not active in a certain time window the threshold is reduced slightly.

$$\frac{\mathrm{d}V_{th}}{\mathrm{d}x} = \gamma(fr_{mean} - fr_{target}) \tag{4.22}$$

where fr_{mean} is the mean activity (or firing rate) of a neuron, fr_{target} is the target activity, and γ is a multiplicative positive constant. Consequently, the activity of the neuron is bounded in a homeostatic range to encode the synaptic input more effectively to improve the STDP learning [140].

Winner-take-all

In a winner-take-all (WTA) network, in output layer or partially output layers, neurons compete with each other based on their output activities, which leads to an adaptation only of the weights of the neuron with the highest output activity [141]. In unsupervised learning using spike coding and plasticity learning. Without competition, all the neurons would behave alike and no specialization takes place in the neurons. The theoretical analysis shows that winner-take-all is a surprisingly powerful computational method compared with threshold gate (McCulloch-Pitts neuron) and sigmoidal gate [142]. There have been many implementations of winner take all (WTA) computations in recurrent networks in the literature [29, 30]. Also there have been many analog VLSI implementations of these circuit [30, 143]. In WTA, after the competition, only one neuron will be the most active for some inputs and the rest of the neurons will eventually become inactive for those inputs. Physiologically plausible learning methods can be mainly classified as dense, local, or sparse. Competitive learning such as WTA is a local learning rule as it activates only the unit that fits the input pattern best and suppresses the others through fixed inhibitory connections.

The simplest competitive computational model is a hard WTA that computes a function f_{WTA}:$\mathbb{R}^n \rightarrow \{0,1\}^n$ whose output $\langle b_1, ..., b_n \rangle = f_{WTA}(x_1, ..., x_n)$ satisfies

$$b_i = \begin{cases} 1 & \text{if} \quad x_i > x_j \quad \text{for all} \quad j \neq i \\ 0 & \text{if} \quad x_i < x_j \quad \text{for some} \quad j \neq i \end{cases} \tag{4.23}$$

Therefore in the case of inputs $x_1, ..., x_n$ a single output b_i has values 1 that marks the position of the biggest input x_i. Wolfgang Maass [142] introduced two types of WTA namely *k-WTA* and *soft-WTA*. In k-WTA, b_i has value 1 if and only if x_i is among the k largest inputs. In soft-WTA the i_{th} output is an analog variable r_i whose value reflects the rank of x_i among the input variables. We use WTA in our research that will be presented in the next sections.

4.7 Conclusion

Neuromorphic computation using Spiking Neural Networks (SNN) is proposed as an alternative solution for future of computation to conquer the memory bottelneck issue in recent computer architecture. Different spike codings have been discussed to improve data transferring and data processing in neuro-inspired computation paradigms. Choosing the appropriate neural network topology could result in better performance of computation, recognition and classification. The model of the neuron is another important factor to design and implement SNN systems. The speed of simulation and implementation, ability of integration to the other elements of the network, and suitability for scalable networks are the factors to select a neuron model. The learning algorithms are significant consideration to train the neural network for weight modification. As the most frequent used unsupervised method for network training in SNN is STDP, we analyzed and reviewed the various methods of STDP. Furthermore, in the next chapters, we use STDP as the main learning algorithm to train the network. The sequential order of pre- or postsynaptic spikes occurring across a synapse in an interval of time leads to defining different STDP methods. Based on the importance of stability as well as Hebbian competition or anti-Hebbian competition the method will be used in weight modification. To step further more confidently in next part of our work, we surveyed the most significant projects that cause making neuromorphic platform. The advantages and disadvantages of each neuromorphic platform have been introduced in this chapter.

Part II

Our contribution in spiking neural network architecture: Simulator, New synapse box, Parameter exploration and Spiking deep learning

Chapter 5

N2S3, an Open-Source Scalable Spiking Neuromorphic Hardware Simulator

Abstract

One of the most promising approaches to overcome the end of Moore's law is neuromorphic computing. Indeed, neural networks already have a great impact on machine learning applications and offer very nice properties to cope with the problems of nanoelectronics manufacturing, such as a good tolerance to device variability and circuit defects, and a low activity, leading to low energy consumption. We present here N2S3 (for Neural Network Scalable Spiking Simulator), an open-source simulator that is built to help design spiking neuromorphic circuits based on nanoelectronics. N2S3 is an event-based simulator and its main properties are flexibility, extensibility, and scalability. One of our goals with the release of N2S3 as open-source software is to promote the reproducibility of research on neuromorphic hardware. We designed N2S3 to be used as a library, to be easily extended with new models and to provide a user-friendly special purpose language to describe the simulations.

5.1 Introduction

Neuromorphic computing is a suitable alternative for conventional computation to take the advantages of low power computing for future computer architectures [23]. In fact, parallel neuro-inspired computing, by performing computation and storage on the same devices, can overcome the von Neumann bottleneck. Several large projects are based on neuromorphic systems, such as the EU Human Brain Project [144], the DARPA/IBM SYNAPSE project [145], and deep learning research led by Google and Facebook, among others.

Recently, emerging nano-scale devices have demonstrated novel properties for producing new memories and unconventional processing units. One of those is the memristor, that was hypothetically presented by Leon Chua in 1971 [11]; after a few decades, HP was the first to announce a successful memristor fabrication [1]. The unique properties of memristors, such as extreme scalability, flexibility thanks to their analog behavior, and their ability to remember their last state, make memristors very promising candidates to be used as synapses in Spiking Neural Networks (SNN) [12].

Given their potential of very low power execution and their capability to handle natural signals, we focus on SNN. A comprehensive introduction and literature review about SNN was published by Paugam-Moisy and Bohte in 2012 [146]. The authors explore the computational capabilities of SNN, their learning capabilities, and their simulation.

Brette *et al.* [119] surveyed and discussed the existing work on SNN simulation in 2007. All the simulators discussed in this chapter as well as the more recent Brian [147] target the simulation of biological SNN. More recently, Bichler *et al.* [37] proposed Xnet, a C++ event-driven simulator dedicated to the simulation of hardware SNN. In our work, we share the goals of Xnet: "intermediate modeling level,

between low-level hardware description languages and high-level neural networks simulators used primarily in neurosciences", and "the integration of synaptic memristive device modeling, hardware constraints and any custom features required for the targeted application". In addition to these goals, we put an emphasis on *flexibility and usability* to allow the study of various kinds of hardware designs (possibly by other researchers than us), *scalability* to simulate large hardware SNNs, and *software engineering best practices* (robust and extensible software architecture for maintainability, extensive test suite, continuous integration, open-source distribution).

In this chapter we present N2S3 (Neural Network Scalable Spiking Simulator, pronounced "Nessie", hence its logo), an event-driven simulator dedicated to the exploration of hardware SNN architectures [148]. The internals of N2S3 are based on the exchanges of messages between concurrent actors [149], mimicking the exchange of spikes between neurons. N2S3 has been developed from the ground up for extensibility, allowing to model various kinds of neuron and synapse models, various network topologies (especially, it is not restricted to feed-forward networks), various learning procedures, various reporting facilities, and to be user-friendly, with a domain specific language to easily express and run new experiments. It is available as open-source software at `https://sourcesup.renater.fr/wiki/n2s3` so that its users can share their models and experimental settings to enable others to reproduce their results. In this spirit, N2S3 is distributed with the implementation of two "classical" experiments: handwritten digit recognition on the MNIST dataset [25, 31] and the highway vehicle counting experiment [32].

Figure 5.1: N2S3 Logo

In the remainder of this chapter, we will first detail the fundamental architectural choices of N2S3, then the models that are already implemented in N2S3, and finally the features that N2S3 provides to implement and run experiments.

5.2 Event-Driven Simulation Architecture

5.2.1 Event-Driven vs Clock-Driven Simulation

SNN are essentially defined by standard differential equations, but because of the discontinuities caused by the spikes, designing an efficient simulation of spiking neural networks is a non-trivial problem. There are two families of simulation algorithms: event-based simulators and clock-based ones. Synchronous or "clock-driven" simulation simultaneously updates all the neurons at every tick of a clock, and is easier to code, especially on GPUs, for getting an efficient execution of data-parallel learning algorithms. Event-driven simulation behaves more like hardware, in which conceptually concurrent components are activated by incoming signals (or events).

Event-driven execution is particularly suitable for untethered devices such as neurons and synapses, since the nodes can be put into a sleep mode to preserve energy when no interesting event is happening. Energy-aware simulation needs information about active hardware units and event counters to establish the energy usage of each spike and each component of the neural network. Furthermore, as the learning mechanisms of spiking neural networks are based on the timings of spikes, the choice of the clock period for a clock-based simulation may lead either to imprecision or to a higher computational cost.

There is also a fundamental gap between this event-driven execution model and the clock-based one: the first one is independent on the hardware architecture of computers on which it is running. So, event-driven simulators can naturally run on a grid of computers, with the caveat of synchronization issues in the management of event timings.

5.2.2 Technological Choices: Scala and Akka

To address our concurrency and distributability requirements (i.e., ability to scale out a simulation on several computers to handle large networks) we have chosen to use the Scala programming language [150] along with the Akka actor library [149].

5.2.3 Software Architecture

The architecture of N2S3 is organized in Akka actors. Our choice for an actor model aims mainly for the distribution and scalability of large neural networks. Each entity of the simulation is deployed in a concrete actor that is assigned to a concrete machine at runtime. The core of a typical neural network simulation in N2S3 is composed mainly by the following network entities:

Containers Containers are network entities in charge of organizing the network structure. Their main responsibility is to contain network entities and dispatch messages to their children.

Neurons The basic building blocks of a neural network. In N2S3, a neuron is composed of both its nucleus (the soma) and its incoming connections (dentrites and associated synapses).

Each N2S3 actor has a network container capable of containing one or multiple entities. This particularity allows us to control how distribution and parallelism is managed in each simulation. Figure 5.2 illustrates the architecture with an example network. In this figure, the simulation is made of one input and a neural network organized in two layers. The input of the simulation resides in one actor, the hidden layer is split in one actor per neuron, and all neurons of the output layer reside in a single actor. The reason why one would put several neurons in the same or a different actor is based on the scalability and the synchronization choices for the experiment. The topology of the network (discussed in more details in Section 5.3.3) is one of the major influence on these questions.

The communication between simulation entities requires to identify each object by a URI made of a pair `(actor, local_identifier)`. The component `actor` is the Akka actor that contains the entity and `local_identifier` is a path that identifies each object uniquely within its actor.

Since actors are inherently concurrent, one concern is how the temporal order of messages is guaranteed during the simulation. To do so, N2S3 allows to configure several levels of synchronization to be used by the simulation designer. On one end of the spectrum, N2S3 may make use of a unique synchronizer for the simulation, which will ensure that no causality issues happen but can create a bottleneck that will affect the performance of the simulation. On the other end of the spectrum, we can also configure N2S3 to use a synchronization mechanism which is local to each neuron. The latter policy enables better parallelism, but may cause some temporal consistency problems.

5.3 Neuron, Synapse, Network Modeling

5.3.1 Neuron Modeling

The neuron is a dynamic element and processing unit that emits output pulses whenever the excitation exceeds some threshold. The resulting sequence of pulses or spikes contains all the information that is transmitted from one neuron to another one. As we have chosen event-driven simulation, the state of a neuron is updated only when an event representing a spike is received.

We provide by default a *Leaky-Integrate-and-Fire (LIF)* neuron model [118]. There are several reasons for using a LIF model:

- CMOS technology fabrication of this model is available [116, 117];
- it is fast enough to simulate, and in particular it is effective for large-scale network simulations [119].

In LIF, neurons integrate the spikes coming from other neurons. These spikes can change the internal potential of the neuron, which is known as the *membrane potential* or *state variable* of the neuron. When the membrane potential reaches a given threshold voltage after integrating enough input spikes, an action potential occurs; in other words, the neuron fires, generating an output spike. This is done by updating the membrane potential using the value computed when receiving the last spike and the analytical solution to the differential equations defining the behavior of the LIF neuron.

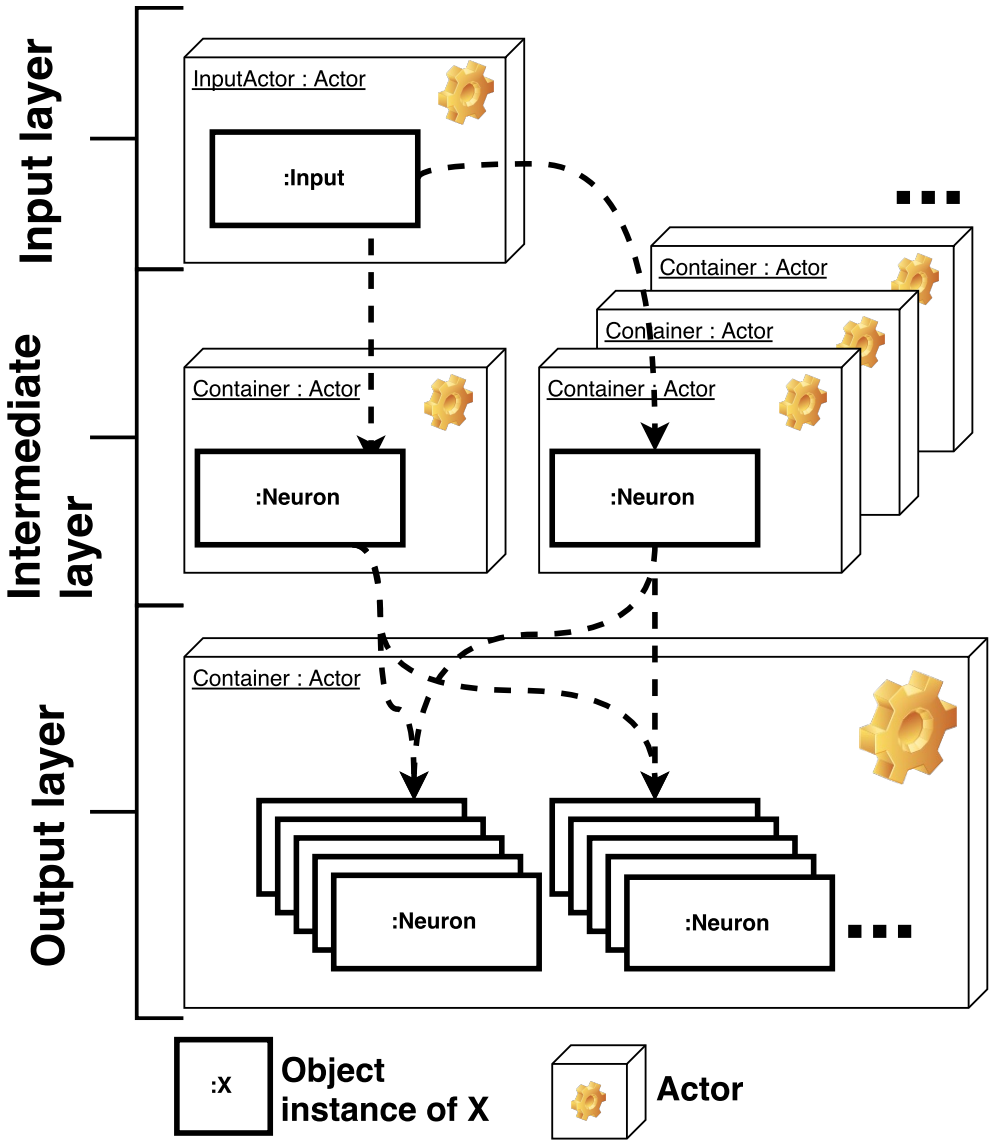

Figure 5.2: **N2S3 Architecture.** A network is organized in actors that may contain one or more network entities. Such entities could be for example, neurons, inputs or any other.

In order to improve the learning capabilities of neural networks, we provide two refinements of the neuron model: homeostasis and a refractory period. Homeostasis [139] dynamically adapts the threshold of each neuron to the activity of this neuron to prevent a neuron from being over-active or inactive. To allow more specialization of the neurons, we include a refractory period after the firing of a neuron during which it ignores incoming spikes.

5.3.2 Synapse modeling and learning

A synapse operates as a plastic controller between two neurons. This plasticity is believed to be the origin of the learning and memorization capabilities of the brain [151]. Hence, hardware spiking neural networks usually use some kind of synaptic plasticity to enable learning. In N2S3, we have modeled and simulated hardware synapses and one standard learning behavior, namely Spike Timing-Dependent Plasticity (STDP).

STDP is a local weight modification based on the timing difference between presynaptic spikes and postsynaptic spikes. It increases the connectivity between the neurons when there is a temporal

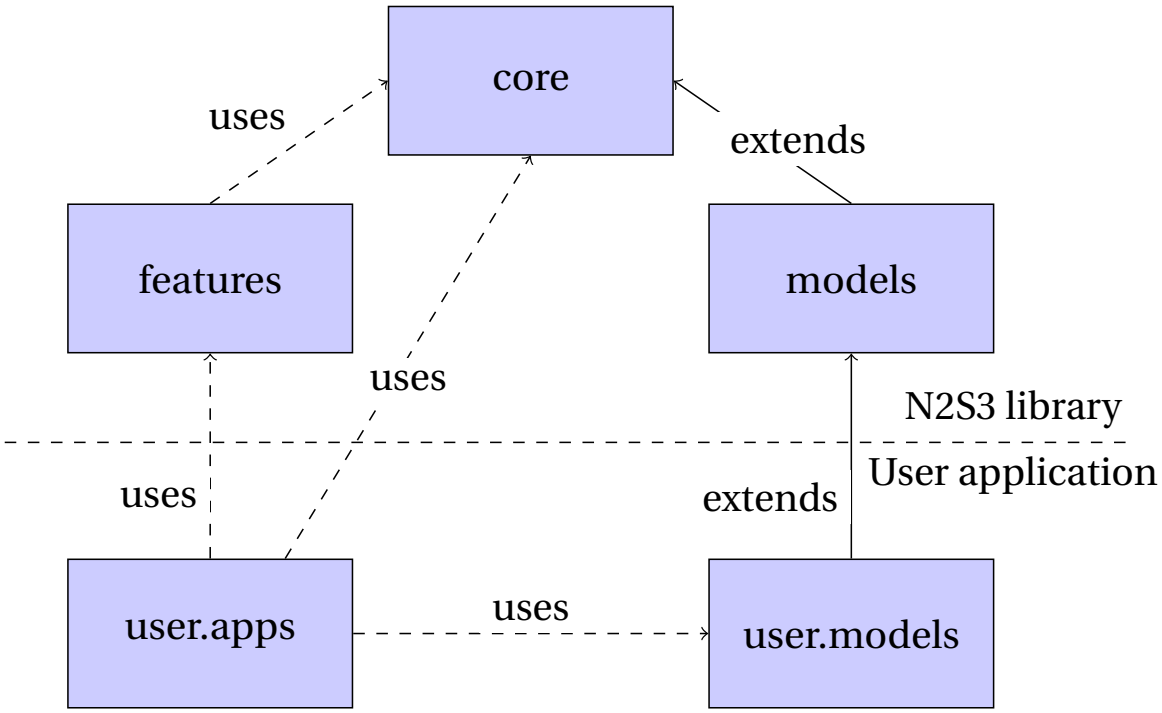

Figure 5.3: N2S3 Packages

causality between the spikes they emitted. This is implemented by locally remembering the past spikes and updating the synaptic weight according to the chosen STDP rule.

Using a memristor as a nonvolatile synaptic memory has been proposed in previous work [12, 43, 47, 152]. The basic artificial synapse model that we provide is taken from [153, 154], which describes a nonvolatile memristor suitable for event-driven and STDP computation. In addition, we designed a new model of synapse which is able to forget unimportant events and remember significant events by combining a nonvolatile memristor and a volatile one [33].

5.3.3 Network Topologies

Another purpose of N2S3 is to allow an easy exploration of different neural network configurations. In order to facilitate the network construction, neurons are gathered into groups. Those groups, which usually represent neuron layers, are organized into a specified shape. This information allows the automation of the builder and of the visualization creation.

Inside a group one can specify the use of lateral inhibition to implement the winner-take-all rule [142] that states that a spiking neuron inhibits its neighbors during a short time period to allow the specialization of the neurons and hence to enhance unsupervised learning.

Neuron groups can be connected to each other. Connection patterns are managed by different connection policies. By default, N2S3 uses unidirectional full connections (each neuron of the input group is connected to every neuron of the output group). Other policies are also bundled with N2S3, such as the *one-to-one* policy, which connects each neuron of the input group to only the corresponding neuron of the output group, and a random policy, which creates connections according to a given probability distribution, e.g. to enable Reservoir Computing [155] modeling.

Several builders have the responsibility to create different topologies. They can use the information of an existing layer (e.g., the shape of the neuron group), if there is one, to construct a new one. Builders bundled with N2S3 include, for instance, builders for the typical layers of Convolutional Neural Networks [31].

5.4 N2S3 Features

5.4.1 Input Processing

The simulator uses a piped stream system to provide stimuli to the network entities. The input process typically starts by an input reader, which reads data from files or any external source, followed by a number of streams that filter the input data before feeding it to the network.

Input readers provided with N2S3 allow to read data in a variety of format, including standard formats such as Address Event Representation (AER), a data format used by spike-based cameras, or MNIST, used in a standard dataset for handwritten digit recognition (see Section 5.4.5 for details about the data).

Subsequent filter streams available in N2S3 include coding streams, which convert raw numerical data into sequences of spike timings, spike presentation streams (e.g., repeating input spike over a given period, or shuffling spikes), and modifier streams, that alter the input spikes (e.g., by adding noise). Users are free to use one or multiple input readers and to combine any number of filter streams in any order; they may also easily create their own readers and filters.

5.4.2 Visualization tools

Users may observe simulation outputs (spikes, weight values...) through network observers. Network observers follow the observer pattern by subscribing to events in the simulation (e.g., when a spike happens), perform some calculations on such events, and make them visible to the user. Examples of such observers range from textual loggers to dynamic visualizations of the spikes of each neuron. Concretely, N2S3 provides a spike activity map of the network, a synaptic weight evolution visualizer, and the calculation of evaluation metrics (e.g. recognition rates, confusion matrices...).

5.4.3 Experiment Specification Language

N2S3 includes a dedicated internal Domain Specific Language (DSL) that aims to simplify the creation of simulations. At a higher level of abstraction, users can design experiments (network topology, neuron and synapse properties, observation units...) without having to deal with core features such as synchronization or actor policies. The snippet of code below illustrates (1) the creation of a layer with 18 standard LIF neurons with a 20 mV threshold, (2) its full connection to an existing layer (named `outputLayer`), and the addition of two observers: (3) a standard weight observer and (4) a custom (i.e., defined by the user) spike observer. The DSL also allows the definition of different stages for the simulation (e.g., splitting the simulation into a training phase and a test phase).

```
val hiddenLayer =
  n2s3.createNeuronGroup() // (1)
    .setNumberOfNeurons(18)
    .setDefaultNeuronConstructor( () => {
      new QBGNeuron()
        .setProperty(NeuronThreshold, 20 millivolts)
    })
hiddenLayer.connectTo(outputLayer) // (2)
n2s3.createSynapseWeightGraphOn(hiddenLayer, outputLayer) // (3)
n2s3.addNetworkObserver(new SpikeLogger) // (4)
```

At a lower level, the DSL can specify how neurons are organized within actors (e.g., deploying all neurons within the same actor or each neuron in a specific actor). Users may use pre-existing policies or define their own.

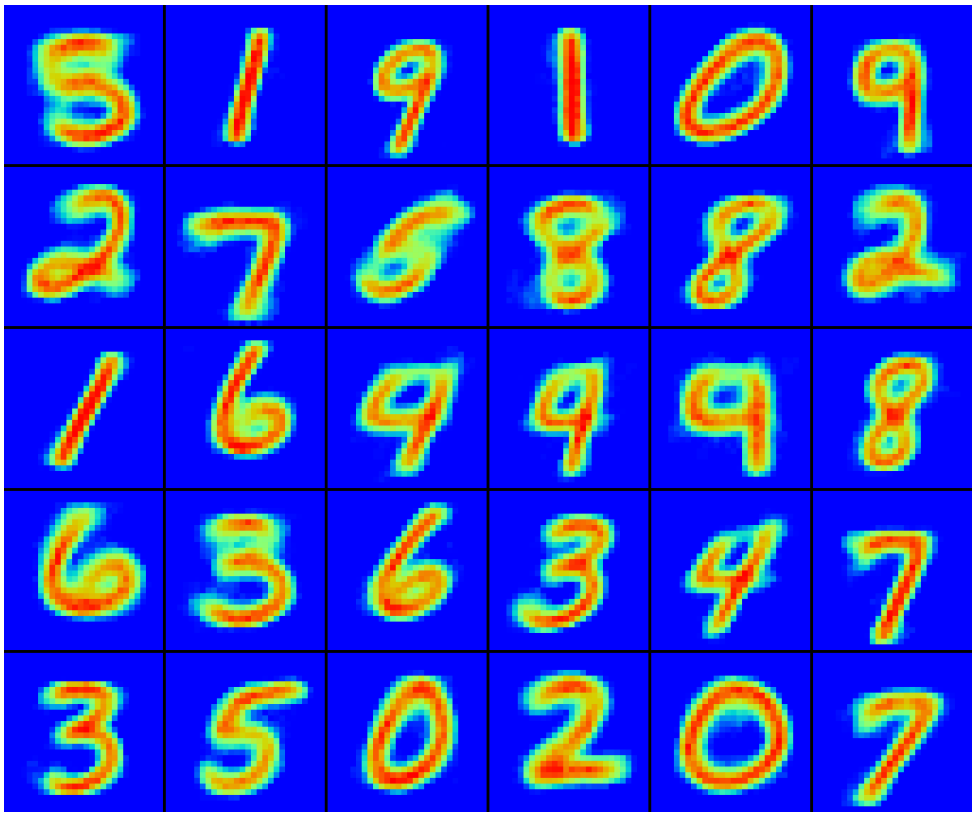

Figure 5.4: Heat map of the synaptic weights after learning the MNIST data base with 30 neurons on the output layer.

5.4.4 Software Engineering Practices

To ensure the quality of the simulator's core and its surrounding features, we follow several good practices of sofware engineering. We use automated tests (unit tests and integration tests) to validate the basic behavior of the simulator. These unit tests are run both locally and within a continuous integration environment to detect regressions in the project as soon as possible. N2S3's codebase is open-source, and its developers perform periodic code reviews on it.

5.4.5 Standard experiments

N2S3 comes with a set of pre-implemented experiments from the literature. These implementations both demonstrate the features and simulation accuracy of N2S3 and provide code snippets to allow users to understand the ESL and help them designing their own simulations. Specifically, experiments for two standard tasks are provided with N2S3: handwritten digit recognition on the MNIST dataset and car counting on the Freeway dataset. In both cases N2S3 provide simulation results comparable to those reported in the literature.

MNIST [25] is a standard dataset for automatic handwritten digit recognition. Since its creation, it has been extensively used to evaluate algorithms for machine learning, computer vision and document recognition. The task is to learn to identify which digit is represented on each image. The dataset includes 60,000 greyscale images of size 28×28 pixels, divided into a training set (50,000 images) and a test set (10,000). The implementation of SNN learning on MNIST provided with N2S3 follows the experimental settings from [156]: 28×28 inputs (1/pixel) and one output layer (10 to 300 neurons). Figure 5.4 shows the result of learning the MNIST database with 30 neurons on the output layer and winner-take-all activated to enhance learning. The initial synaptic weights are random.

Freeway [136] is an AER (Address Event Representation) video of a Freeway in Pasadena. It is a standard video demonstrating the capabilities of spike-based cameras. The task here is to count the

Figure 5.5: Input data for the freeway experiment comming from a spike-based camera. The spikes represent a variation of intensity for a given pixel and are generated asynchronously.

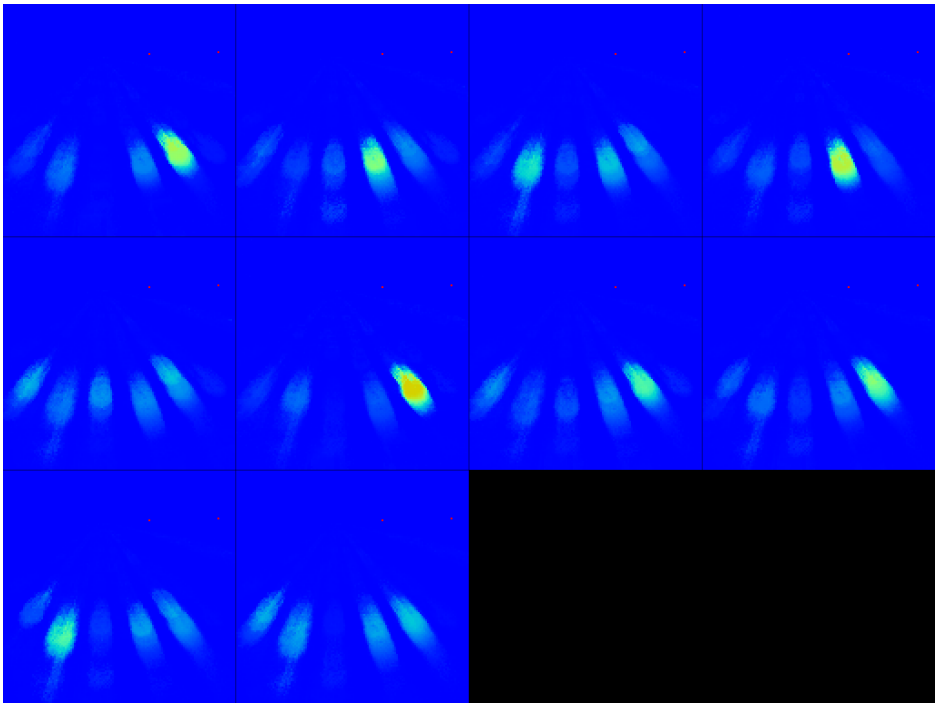

Figure 5.6: Heatmap showing the reconstruction of the contribution of each input pixel to the activity of the 10 output neurons for the freeway experiment. One can see that some neurons have clearly specialized to detect vehicles on a particular lane.

number of vehicles passing over each of the six lanes of the freeway. The input data is the spikes recorded by a spiking camera of resolution 28 × 28 pixels over a sequence of 78.5 seconds (5.2 millions spikes). The implementation of the Freeway experiment provided with N2S3 reproduces the experimental architecture and settings described in [32]: 2 × 28 × 28 inputs, one hidden layer (60 neurones), one output layer (10 neurons), with lateral inhibition on every layer. Figure 5.5 represents the input of the neural network and figure 5.6 shows the result of the unsupervised learning process.

5.5 Conclusion

We have presented in this chapter a spiking neural network simulator, N2S3. This simulator is dedicated to nanoelectronics-based hardware neural networks. Its simulation strategy is event-based, for precision and flexibility sakes. As one of the main goals of N2S3 is to promote open research data, i.e. the open distribution of models and experiments for research result reproducibility, it is distributed as open-source software (at `https://sourcesup.renater.fr/wiki/n2s3`), and a lot of effort has been put in the software architecture, maintainability and enabling features (I/O, graphical outputs, logging, domain specific language). N2S3 is distributed with a few neuron and synapse models. It can simulate not only feed-forward networks but also convolutional or even recurrent networks. Finally, it is extensively extensible.

We are planning to periodically release new versions in a timeboxed fashion and to perform more quality measurements in the near future, for example: continuous benchmarking and performance testing to detect non-functional regressions, adding automated code quality and architectural rules enforcement to detect potential bugs, ensuring the respect for architectural and programming idioms, and agilizing the code reviews.

On the functional side, future work include new models of neurons and synapses, new learning algorithms and procedures, noise modeling, energy modeling, automatic design space exploration, and new applications.

Chapter 6

Combining a Volatile and Nonvolatile Memristor in Artificial Synapse to Improve Learning in Spiking Neural Networks

Abstract

With the end of Moore's law in sight, we need new computing architectures to satisfy the increasing demands of big data processing. Neuromorphic architectures are good candidates to low energy computing for recognition and classification tasks. We propose an event-based spiking neural network architecture based on artificial synapses. We introduce a novel synapse box that is able to forget and remember by inspiration from biological synapses. Two different volatile and nonvolatile memristor devices are combined in the synapse box. To evaluate the effectiveness of our proposal, we use system-level simulation in our Neural Network Scalable Spiking Simulator (N2S3) using the MNIST handwritten digit recognition dataset. The first results show better performance of our novel synapse than the traditional nonvolatile artificial synapses.

6.1 Introduction

Neuromorphic computing has the potential to bring very low power computation to future computer architectures and embedded systems [23]. Indeed parallel neuromorphic computing, by doing computation and storage in the same devices can overcome the Von-Neumann bottelneck. Neuromorphic computing is introduced as an appropriate platform for Big Data analysis and Cloud Computing. Furthermore, many huge projects are running based-on neuromorphic system such as the EU Human Brain Project [144], the DARPA/IBM SYNAPSE project [145] and deep learning research by Google and Facebook among others.

Recently, emerging devices in nano-scale have demonstrated novel properties for making new memories and unconventional processing units. One of those is the memristor that was hypothetically presented by Leon Chua in 1971 [11] and after a few decades, HP was the first to announce the successful memristor fabrication [1]. The unique properties in memristor nano-devices such as, extreme scalability, flexibility because of analog behavior, and ability to remember the last state make the memristor a very promising candidate to apply it as a synapse in Spiking Neural Network (SNN) [12].

In the recent years, there have been several research works using non-volatile resistive nanodevice as a synapse to build a SNN hardware [12,23,157]. Forgetting in the biological brain is an important key of adaptive computation, as without forgetting the biological memory soon becomes overwhelmed by the details of every piece of information ever experienced. Consequently, some studies have been done

using volatile memory as a synapse in brain-like computing [43, 158, 159]. In this work, we combine both volatile and non-volatile types of artificial synapses. It leads to make a synapse which can forget if the information is not important as well as remember if it is significant data.

Thanks to close collaboration with the nano-electronics research center in the University of Lille (IEMN), we have the opportunity of studying the suitability of different kinds of memristors (TiO_2, NOMFET, magnetoresistive, magnetoelectric) to build a spiking neural network hardware platform. Due to the demonstrated potential of NOMFET (Nanoparticle Organic Memory Field-Effect Transistor) [43, 158] to play the role of a synapse, we use it as a volatile synapse in neuromorphic accelerator. The non-volatile device could be any solid-state memristor. We have chose here the resistive memory presented in [153] as non-volatile memory.

We evaluate the synapse box proposal by comparing it with a single non-volatile memory synapse by simulation on the MNIST handwritten digit recognition benchmark. We use Leaky Integrate and Fire (LIF) neurons in Restricted Boltzmann Machine (RBM) network topology. To run the simulations, we introduce the Neural Network Scalable Spiking Simulator (N2S3), a simulation framework for architecture exploration of neuromorphic circuits.

In the next section we describe the architecture including the neuron and synapse models, as well as the network topology and the unsupervised training algorithm. Then, in Section 6.3 we present the experimental evaluation of the two different synapses using the MNIST handwritten digit dataset and the new spiking neural network simulator we propose, N2S3.

6.2 Circuit Design of Neuron and Synapse in RBM Network

In the biological brain, neurons are computing units. Here we define our computing unit by inspiration from a biological neuron model. The synapse operates as a plastic controller between two neurons. The manufacturability is out of scope of this research, however in the synapse box, we have applied the real parameters of volatile synapse beside the model of nonvolatile Resistive RAM. As there are varieties of nonvolatile memristor fabrications, finding appropriate one that is compatible with NOMFET seems not too complicated. It worth mentioning that most of the nanodevices have been reported as Resistive RAMs (nonvolatile) as well as NOMFET (volatile) are compatible with CMOS circuits [158].

6.2.1 Leaky Integrate-and-Fire neurons

The Leaky-Integrate-and-Fire (LIF) neuron model is a well-studied model of neuron. We discussed it in this dissertation in 4.3.1. We have applied this model of LIF neurons in our work. If there is enough spikes in the short period of time which depend on the resistor and capacitor value, the neuron reaches to the threshold. Vice versa, if there is no or small number of spikes connected to input of neuron the action potential decrease and neuron is in leaky phase.

6.2.2 New artificial synapse using memristors

Here, we define our computing unit by inspiring from biological neuron model. The synapse operates as a plastic controller between two neurons. This plasticity is believed as origin of learning and memory in brain. Figure 6.1 shows a simple circuit of two neurons connected with an adaptive synapse (memristor).

Before the discovery of a memristor nanodevice, by using state-of-the-art technology, 12 transistors were combined to mimic the behavior of memristor to perform the STDP learning method [160]. Therefore, using a two-terminal and scalable device such as the memristor could save remarkable amount of power and cost specially in modeling large scale Spiking Neural Networks. To model biological synapses, not only do we need a device to be able to store the last activity, but it must also have enough flexibility to achieve Spike Timing-Dependent Plasticity (STDP) for learning. Using memristor as a nonvolatile synaptic memory has been proposed in several works [12, 43, 47, 152]. By using nonvolatile memory, we can guarantee to store the last synaptic weight which is necessary for

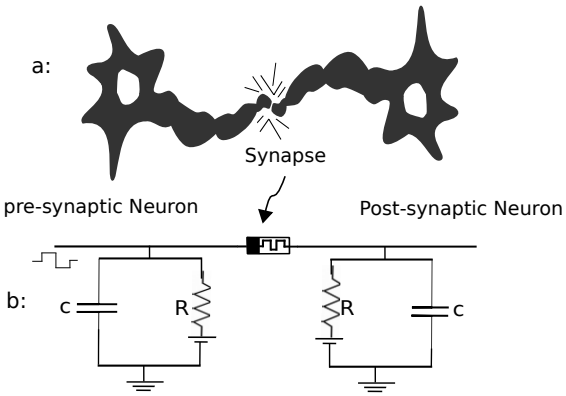

Figure 6.1: **a)** Schematic of two simple biological neurons connected with synapse, **b)** Leaky Integrated & Fire model of neuron connected with artificial synapse (memristor)

network training but the synapse can not forget. To be able to have a synapse which is able to forget, scientists used a volatile memory cell [158, 159].

6.2.3 New plasticity learning method

Forgetting is a memory mechanism that helps brain having better functionality. In fact, it is believed that forgetting helps the brain to remember. However, remembering details of many daily activities and information such as shopping list, novel book details or newspapers not only are unnecessary to remember but also might interfere with brain functionality for innovative thinking and data analysis. Basically human brain skips details of insignificant information and remembers the most important, unique and surprising events and information. In neuroscience, memorization is believed to achieve as a result of two types of synaptic plasticity: Short-Term Potentiation (STP) and Long-Term Potentiation [151]. STP is achieved through the temporal changing of a synaptic connection and the decrease to its initial state soon after. In LTP, stimulation iteration causes permanent synaptic weight achievement as it is depicted in Figure 6.2. Shorter iteration interval leads to more efficient LTP. By inspiring of this biological theory, we propose a new artificial synapse with the ability to forget insignificant data while storing significant information. The novel synaptic box includes one organic transistor and one resistive RAM.

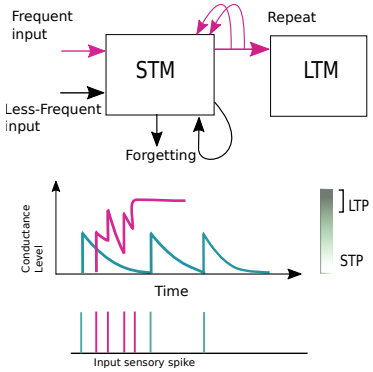

Figure 6.2: Memorization inspired from biology, the data is stored in Long-Term Memory (LTM) if the spikes are repeated in a certain time-window, otherwise Short-Term Memory (STM) will store temporary data.

6.2.4 Combining a volatile and nonvolatile memristor to make a new artificial synapse

Resistive RAM is modeled in our previous work [18] and is used here as a nonvolatile memristor in the synapse box. As it is shown in Figure 6.3.b by changing the doped-undoped regions of device, the conductance will be changed. Bigger doped region leads to more conductivity. Therefore by controlling this boundary between two regions, the conductivity is controlled. The behavior of memristor can be modeled as follows [1]:

$$v(t) = R_m i(t) \tag{6.1}$$

$$R_m = R_{ON}\frac{w(t)}{D} + R_{OFF}\left(1 - \frac{w(t)}{D}\right) \tag{6.2}$$

where R_m is the variable resistance of memristor, $w(t)$ is the width of the doped region, D is the overall thickness of device, R_{ON} and R_{OFF} are device resistances while the active region is completely doped ($w = D$) and mostly undopped ($w \to 0$) respectively (Figure 6.3.b). To model the changing of the conductance, we use the model extracted from Equation 6.2 and introduced in [154] by considering $g_{max} = \frac{1}{R_{ON}}$ and $g_{min} = \frac{1}{R_{OFF}}$ as the maximum and minimum device conductance respectively.

Organic synaptic memory is a novel memristive device with capability of mimicking synaptic properties especially forgetting ability that we discussed it in Section 2. Here, we use a Nano-particle Organic Memory Field Effect Transistor (NOMFET) as an organic memristive device made of conjugated molecules and metal nanoparticles (NPs) which is fabricated by the Institute of Electronics, Microelectronics and Nanotechnology (IEMN) at Lille university [43]. We use NOMFET as our volatile device in the synapse box. In the most recent fabrication process [159], NOMFET works at 1 V with a typical response time in the range 100–200 ms. NOMFET is designed particularly for neuro-inspired computing architectures [158]. NOMFET uses charge trapping/detrapping in an array of gold nanoparticles (NPs) with the SiO_2/pentacene interface designed to mimic dynamic plasticity of a biological synapse as depicted in Figure 6.3 [158]. The NOMFET is used as a two-terminal device by connecting drain (D) and gate (G) together and using this terminal as an input. The source (S) is used as output of the device. Equation 5 shows the behavior of NOMFET as a memristor:

$$i_{ds}(t) = g(q_{np}(t), v_{ds}(t), t)v_{ds} \tag{6.3}$$

where g is the conductance of the device, $v_{ds}(t)$ is the applied voltage and q_{np} is the charges trapped in the NP. For more details of physical structure and behavior of NOMFET refer to [158, 159].

Figure 6.3.c is the synapse box schematic that we apply in our simulation platform to take the advantages of both nonvolatile and volatile artificial synapses. The equivalent circuit of transistor is depicted in Figure 6.3.d. Actually, weight modification follows the STP rule until reaching the LTP threshold in NOMFET. The modification of nonvolatile device is based on STDP learning. Indeed the NOMFET reacts similar to a high-pass filter (HPF). The stimuli spikes with low frequency are not qualified to pass in forgetting Phase. In LTP, stimuli spikes which have more frequency pass to interfere in learning phase (Figure 6.2).

6.2.5 Network topology and learning

By using unsupervised learning inspired by biological neural networks, we propose a fully connected network architecture similar to Restricted Boltzmann Machine [161]. To figure out the correlation between the data, STDP helps to adjust the weight if the sensory input spikes are frequent enough to pass the STP and remain in LTP phase. In STDP, if there is output spike in pre-synaptic neuron and shortly after in post-synaptic neuron, the conductance of the synapse between two neurons increases. On the other hand, if the post-synaptic neuron spikes shortly before the pre-synaptic neuron, the conductance of synapse between two neurons decreases. For more comprehensive explanation for STDP, we refer you to Chapter 4. To see how plasticity in memristor helps targeting STDP achievement we refer you to [162].

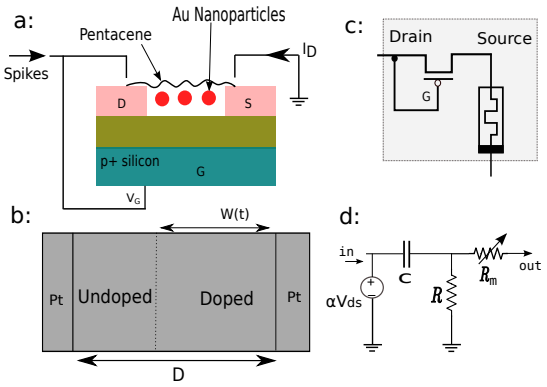

Figure 6.3: Artificial synapse: **a)** Schematic view of the NOMFET as a volatile memory, **b)** TiO_2 based nonvolatile memory, **c)** Synapse box schematic, **d)** Equivalent circuit with simple elements

The simulator architecture in our work is event-driven, there is no clock to synchronize the inputs and outputs. Furthermore, by inspiring of biological behavior of brain computing, we apply lateral inhibition to reduce the activity of the neighbors of winner neurons. This method is known as winner-take-all (WTA) strategy [163]. The neuron which reaches the threshold first sends an inhibitory signal to all other neurons in the same layer to reset their states during inhibition time.

6.3 Experimental Validation

In order to check the effectiveness of the synapse, we propose a spiking neural network simulator. Our requirements for the simulator are: speed (thus event-driven simulation and concurrency), scalability (thus high-level abstraction and distributability), and adaptability (possibility to model different synapses, soma, and network topology). After the presentation of the simulator, we describe our experimental setting and discuss the simulation results that show an improvement in recognition rate for the synapse box with respect to the simple nonvolatile synapse.

6.3.1 MNIST recognition improvement

We have used the MNIST training dataset of handwritten digits [164] to train and test the performance of neural networks based on the synapse box. The training set consists of 60000 digits between 0 and 9 and each handwritten number is a 28 × 28 pixel image. In this simulation, we present the full dataset (60000 images) and full images. Each pixel is connected to one input buffer neuron. Pixel intensity is between 0 to 255 and is transfered to 0 to 22 Hertz spiking frequency using a Poisson distribution during a 350 ms presentation window. Based on previous similar work [165], we have chosen a delay of 150 ms between two images. Therefore, there is sufficient time for membrane potentials of all neurons to reset back to initial values. The network connection weights are between 0 and 1 initialized using a Gaussian distribution.

The hardware platform is a 4 core i7-3687U CPU (2.10GHz × 4). We have simulated different network topologies consisting of 2 fully interconnected layers, with a fixed input neuron number ($28 \times 28 = 784$) and different output neuron number. The neuron model is LIF and we evaluate two types of synapses: non-volatile (NV) and proposed synapse box (volatile/nonvolatile or VNV).

To measure and evaluate the network classification accuracy after a fully unsupervised learning period consisting of the presentation of the full MNIST data set, we label the output neurons using 10000 samples of MNIST: After training, we stop all synaptic modification processes such as STDP, STP and LTP. We assign a number class to the output neuron which has most frequent firing rate during the presentation of the 10000 labelling samples. Using these labels, we then evaluate the recognition rate of the network on 10000 different test samples by comparing the known class of the input with the class of the most firing output neuron. As it was observed in similar works [165] and [153], the

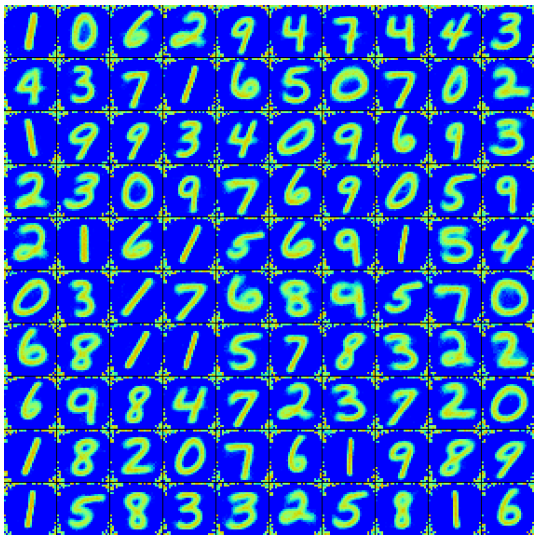

Figure 6.4: Synaptic weights (conductance of non volatile memristor) learned in simulation using the synapse box with 100 output neurons. The weights in the corners are random because they were always filtered out by the volatile memristor and thus are never modified or even read.

recognition rate depends on the number of neurons and synapses, and the number of repetitions of the presentation of the dataset to the network. In the experiments, we present 60000 digit the dataset the smallest number of time that is necessary for learning. That is 1 time for less or equal to 50 output neurons, 2 times for less or equal to 100 output neurons and 3 times for more output neurons. With these numbers of presentations, we obtain recognition rates that are comparable to the state of the art. The running time of the simulations are also comparable to those of similar experiments though it is difficult to make accurate comparisons.

An example of the conductance weights learned in N2S3 on the MNIST dataset is shown in Figure 6.4 for 100 output neurons. As it is obvious in the figure, the border of each digit did not pass the NOMFET high-pass filter because of low frequency changes. This is the impact of forgetting properties in synapse box to skip unimportant data or noise. To demonstrate the functionality of the synapse box, we compare the recognition rate of networks of the same topology but using different synapse models: a simple nonvolatile synapse (NV) model and the volatile-nonvolatile (VNV) synapse box model. We have run 10 simulations for each number of output neuron and each synapse. The results are summerized in Figure 6.5 showing the distribution of the recognition rates for each configuration. We can conclude that using the synapse box improves the recognition rate in average by a small but consistent margin.

Although it is not shown in Figure 6.5, we have also made the comparison using 300 output neuron and the best recognition rate we have obtained is 89.4 %.

6.4 Conclusion

In this study, we have introduced a novel synapse box with the possibility to forget and remember inspired from biological synapse properties. This synapse box is composed of a volatile memristor (NOMFET) followed by a nonvolatile resistive RAM. The volatile memristor acts like a high-pass filter to enhance short term plasticity and the nonvolatile resistive RAM enables long term potentiation. Both work together in the spike timing dependant plasticity unsupervized learning process.

In addition, in this work we have also announced and used our event-based simulator, N2S3 (Neural Network Scalable Spiking Simulator). It is specifically designed to simulate hardware spiking neural networks. N2S3 is quite flexible to explore different network architectures, synapse and neuron models to help design hardware architectures and VLSI circuits. To evaluate and verify the new synapse

Table 6.1: Comparing network architecture efficiency for two synapses: nonvolatile (NV) v.s volatile-nonvolatile (VNV) synapse

#Output neurons	#synapses	Training repetition	NV	VNV
20	15680	2	67,17%	68,73%
		3	68,35%	70,68%
30	23520	2	75,05%	77,29%
		3	76,50%	78,91%
50	39200	2	77,79%	79,60%
		3	79,88%	82,12%
100	78400	2	82,32%	84,83%
		3	85,11%	86.76%
300	235200	2	86,23%	87,59%
		3	88,23%	89,46%

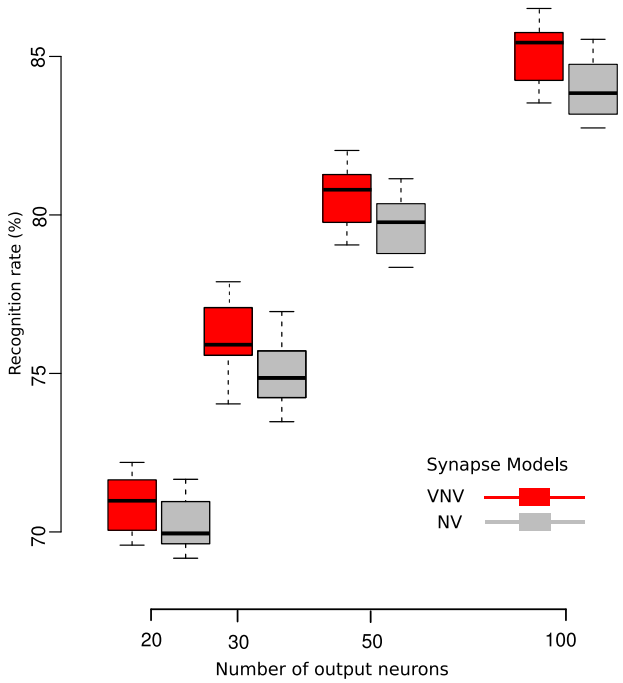

Figure 6.5: Recognition rate as a function of number of output neurons. In the box plot for each number of neuron, we compare the recognition rate of the two synapse models. The whiskers of the box plot represent the minimum and maximum recognition rates of the 10 simulations.

box as well as the functionality of the simulator, we have used the MNIST handwritten digit dataset. The first results demonstrate an improvement in recognition rate by using the synapse box over a single nonvolatile memristor synapse. We will continue to explore the various parameters and device combinations to help design the most efficient hardware neural networks as possible.

For future works, we also propose to study different neural network topologies such as deep belief, recurrent and convolutional neural networks to evaluate the synapse box benefits and costs (area, energy, manufacturability, variability) in other contexts.

Chapter 7

Evaluation methodology and parameter exploration to improve performance of memristor-based spiking neural network architecture

Abstract

The brain-inspired spiking neural network neuromorphic architecture offers a promising solution for a wide set of cognitive computation tasks at a very low power consumption. Due to the practical feasibility of hardware implementation, we present a memristor-based model of hardware spiking neural networks which we simulate with N2S3 (Neural Network Scalable Spiking Simulator), our open source neuromorphic architecture simulator. Although Spiking Neural Networks (SNN) are widely used in the community of computational neuroscience and neuromorphic computation, there is still a need for research on the methods to choose the optimum parameters for better recognition efficiency. With the help of our simulator, we analyze and evaluate the impact of different parameters such as number of neurons, STDP window, neuron threshold, distribution of input spikes and memristor model parameters on the MNIST hand- written digit recognition problem. We show that a careful choice of a few parameters can significantly improve the recognition rate on this benchmark.

7.1 Introduction

Neuroinspired computing has the potential to carry very low power computation to future computer architectures and embedded systems [23]. Indeed parallel neuromorphic computation, by performing computation and storage in the same devices can overcome the Von-Neumann bottleneck. There are two broad types of brain-inspired cognitive computations: abstract artificial neural networks (ANNs) and closer to biology spiking neural networks (SNNs) [166]. Machine learning algorithms such as classical ANNs and more recently deep belief networks [26, 167] are widely used for data classification and clustering. However, ANNs are a highly abstract mathematical model of neurons that are designed to be executed on digital traditional von Neumann processing platforms (more and more using accelerating units such as GPUs). ANNs only use rate coding to represent neuronal activity and are not capable of taking into account the precise relative timing of spikes that are significant when dealing with natural signals that are more and more significant in the internet of things. The data should be coded to spikes to be processed in SNN, which is known as spike coding (versus rate coding in ANN). SNNs offer online real time unsupervised learning through continuous weight updating which is performed on local synaptic weights. This temporal and spatial locality is important in hardware implementations of neural networks because it frees this architecture of the memory bottleneck of Van

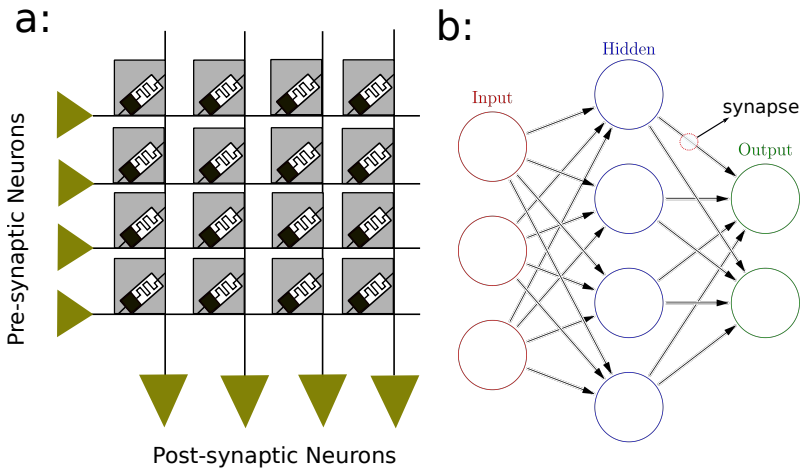

Figure 7.1: Neuromorphic vs SNN, **a)** The memristive synapse connects the spiking neurons in configurable crossbar array suitable for stdp unsupervised learning, the presynaptic neurons are considered as inputs and postsynaptic neurons play output rolls. **b)** The spiking neural network two layers of this three layers could similarly operates as crossbar array.

Neumann architectures. Recent advances in nanotechnology have provided neuromorphic computing architecture with novel memristive devices which have the capability of mimicking synaptic plasticity, such as resistive switching memory (RRAM) [64, 98, 138], phase change memory (PCM) [65, 168, 169], Conductive Bridge memory (CBRAM) [37, 66, 170], and ferroelectric memory (FeRAM) [41, 171]. The advantages of using these memristive nanodevices to model the behavior of synapses are their unique properties, such as scalability, flexibility because of their analog behavior, manufacturability on top of CMOS technology to make a crossbar array (shown in Figure 7.1) and ability to remember the last state in a SNN [12]. Fortunately due to close collaboration with the nano-electronics research center in the University of Lille (IEMN), we have the opportunity to study the appropriateness of various classes of memristors (e.g.,TiO_2, NOMFET, Magnetoelectric) to build a SNN hardware platform by using real parameters. The non-volatile resistive switch could be any solid-state RRAM. We have chosen here the resistive memory presented in [153] as non-volatile memory because it is representative of such devices.

It is widely believed that plasticity is the key of learning in the biological brain [172]. Consequently, with the latest proposals to use the memristive nano-devices as synapses, we implement an efficient and well-studied unsupervised learning rule known as spike timing dependent plasticity (STDP) [5,173]. In this study, we show that the memristive synapse is adapted to unsupervised STDP learning in SNNs. We explain the required technology and architecture with system-level simulation. For implementation and results, we use the MNIST dataset of handwritten digits [164] to test the performance of neural networks. Although, there are various research works using STDP learning in SNN architecture and neuromorphic VLSI implementations, none of them evaluates and analyzes the key parameters that improve learning and SNN recognition performance in those architectures. Our main contribution is to evaluate and explore the impact of several parameters on the learning performance of SNNs for the MNIST benchmark: the number of neurons in the output layer, the duration of the STDP window, various thresholds for adaptive threshold LIF neurons, different distributions of spikes to code the input images and the memristive synapse fitting parameter.

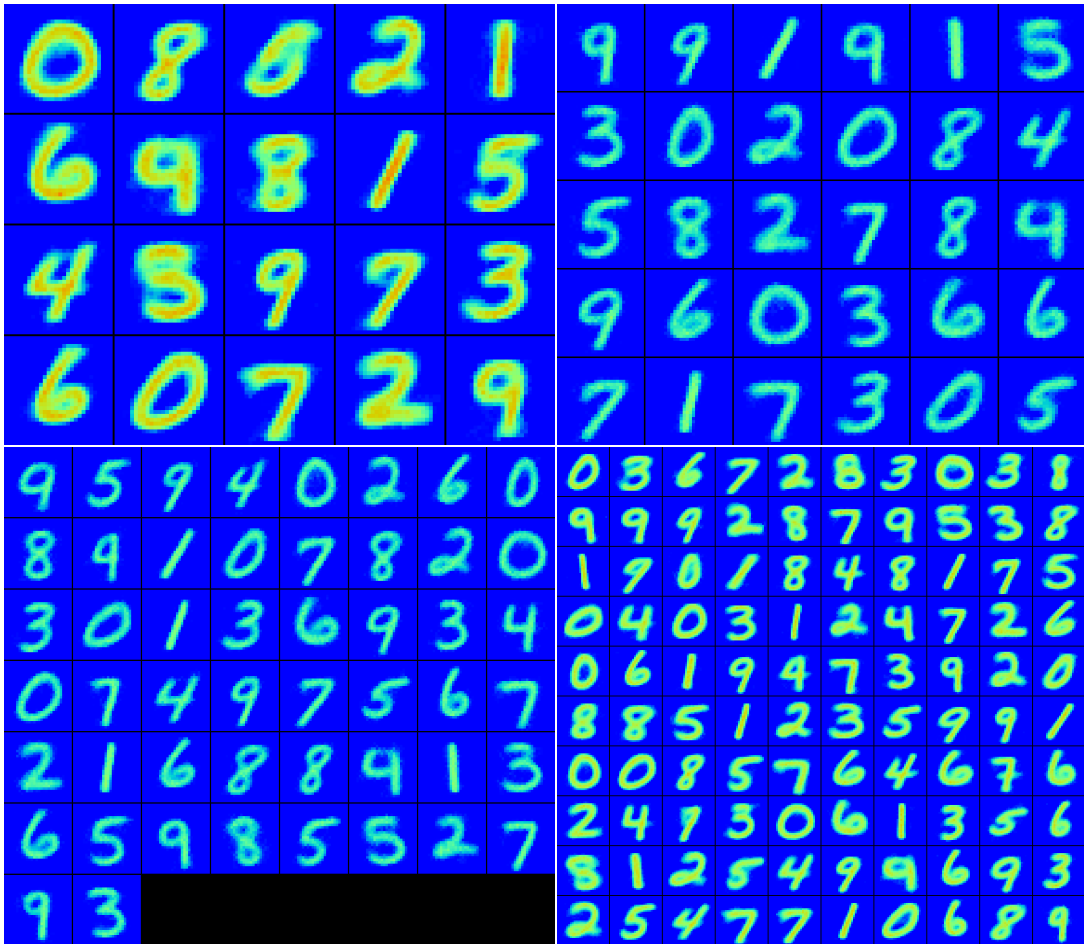

Figure 7.2: Sample heat maps of the synaptic weights after network training with four different numbers of output neurons (20, 30, 50 and 100).

7.2 Experimental evaluation of the influence of four parameters on the classification of handwritten digits

In this section, we evaluate and explore the effects of four parameters on the performance of the network architecture namely the distribution of input spike trains, the STDP window duration, the neuron thershold, and the synapse β parameter. These parameters will be fully described in the following. The full MNIST dataset is presented twice to the networks with four different numbers of output neurons. A sample of output results is shown in Figure 7.2 for 20, 30, 50 and 100 output neurons.

Simulation has been done using our event-based Neural Network Scalable Spiking Simulator (N2S3). N2S3 has been developed to implement, evaluate and simulate spiking neuromorphic hardwares. We vary each parameter independently to asses its influence on the recognition rate. The default value for these parameters are taken from the values listed in the literature. We call this set of parameters the baseline.

- Input spike train distribution: Poisson.
- STDP window duration: 25 ms.
- Neuron threshold: 15 mV.
- Synapse β parameter: 1.5.

At the end of this section, in section 7.2.5, we compare the baseline with the best parameters obtained separately and discuss the results.

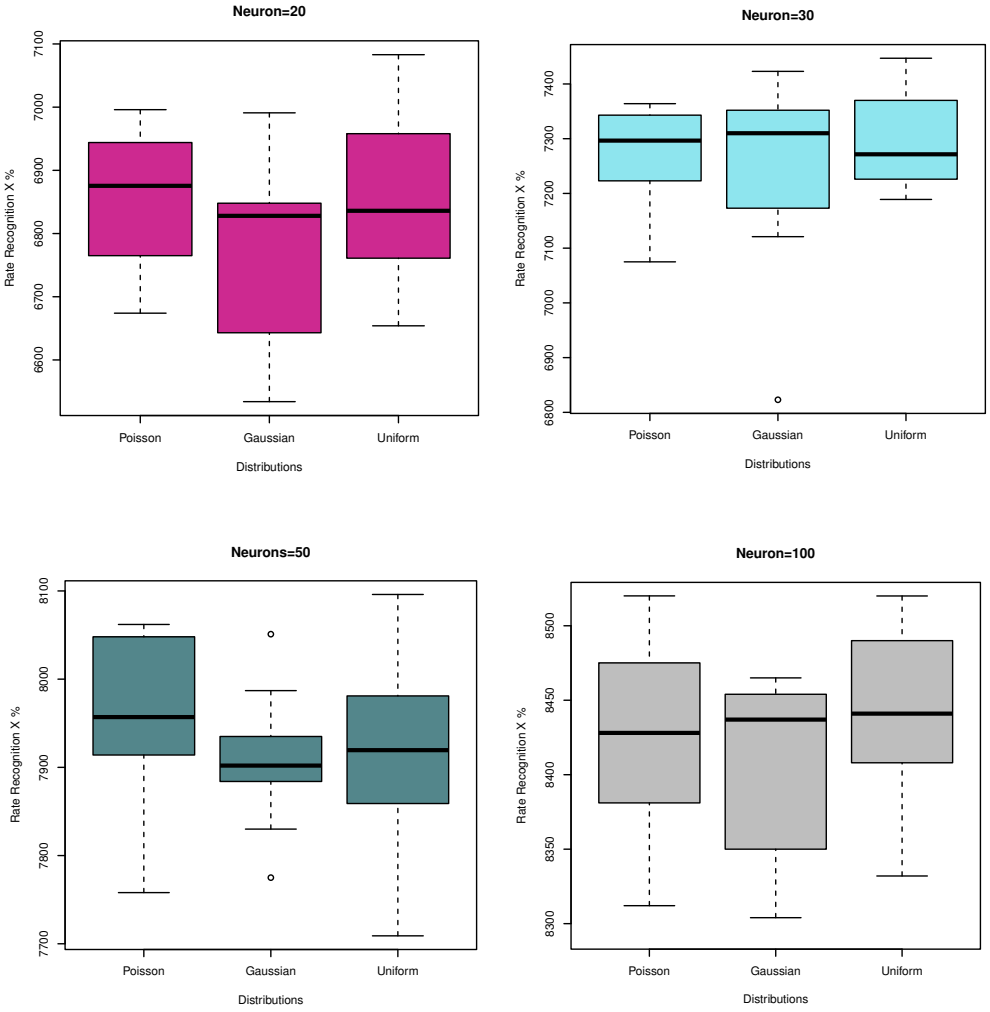

Figure 7.3: The comparison of three different distributions for generating spike train by transferring MNIST database pixels to spike train. These distributions are tested with different number of neurons=20, 30, 50, 100.

In all the simulations, as the weight initialization is random, each simulation has been repeated 10 times and the results are shown in box plots with the default parameters of the box plot function of the R statistical tool.

7.2.1 Effect of spike distribution

As the platform is working using spikes for processing the data, we need to generate spikes train of the images of MNIST dataset to extract a spike-based dataset from a frame-based dataset respecting to intensity of each pixel of images. The pixel intensity is between 0 to 255 and should be transfer to the spike train to be readable by SNN. For instance, Diehl and Cook [165] use a method which the maximum pixel intensity of 255 is divided by 4, resulting in input firing rates between 0 and 63.75 Hz. Yet there is a question that with which interval time between two spikes we generate the spikes? Therefore, statistical distribution rules can help to generate appropriate spike train when encode the pixels.

Poisson input spike train distribution

Here we explain a particular class of random process called a Poisson spike train process. Define $\rho(t)$ response function to the input stimuli:

$$\rho(t) = \sum_{i=1}^{k} \delta(t - t_i) \tag{7.1}$$

where k is the total number of spikes in the spike train, and t_i defines the time each spikes occurs. The unit impulse signal is defined as:

$$\delta(t) = \begin{cases} 1 & \text{if} \quad t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

The instantaneous firing rate (e.g., of a sensory neuron) can now be formally defined to be the expectation of the sensory input response function which is $r(t) = \langle \rho(t) \rangle$. The average spike count between times t_1 and t_2 can then be defined from the instantaneous firing rate:

$$\langle n \rangle = \int_{t_1}^{t_2} r(t)\, dt, \tag{7.3}$$

the probability of a spike occurring during a given brief time interval is equal to the value of the instantaneous firing rate during that time interval times the length of the interval:

$$P_{(\text{one spike in } (t-\Delta t, t+\Delta t))} = r(t)\Delta t. \tag{7.4}$$

We assume the instantaneous firing rate r is constant over time. This is called a homogeneous Poisson process. From Equation 7.3 it is derived $\langle n \rangle = r\Delta t$ for any interval $\Delta t = t_2 - t_1$. Equation 7.4 can be used to generate a Poisson spike train by first subdividing time into a bunch of short intervals, each of duration Δt. Then generate a sequence of random numbers $x[i]$ uniformly distributed between 0 and 1. For each interval, if $x[i] \leq r\Delta t$, we generate a spike otherwise no spike. This procedure is appropriate only when Δt is small enough for e.g., millisecond range. Using Poisson distribution, we made an event-based version of MNIST [174] and it is available open-source online (https://github.com/MazdakFatahi/evt-MNIST). We refer to intensity of pixels as the probability that a spike occurs within an interval.

Other distributions of the input spike trains

In our simulation, the pixel intensity between 0 and 255 is transferred to 0 to 22 Hz spiking frequency using different probability distributions during a 350 ms presentation window. Based on previous similar work [165], we have chosen a delay of 150 ms between two images. Therefore, there is enough time for membrane potentials of all neurons to reset back to their initial value. The network connection weights are between 0 and 1 initialized using a Gaussian distribution. We compare uniform, Gaussian and Poisson distribution using different numbers of output neurons in Figure 7.3. This figure shows that the choice of the input distribution does not have a significant impact on the recognition rate of the network. That means that the classification of the inputs is done mainly on the frequency of the inputs regardless of the precise timing of the spikes.

Figure 7.4 depicts recognition rate for different number of neurons using three various spike train distribution. This figure demonstrate that increasing the number of neuron increase the recognition rate. We expected such a behavior because increasing the number of neurons will increase the number of synapse in the network therefore the chance of learning is increased due to more precise coding. It is worth to note that in general the relation between the number of neuron and learning performance is not always increasing ratio while we increase the number of neurons. The reason is that the random selection of number of neurons might cause either overfitting or underfitting problems.

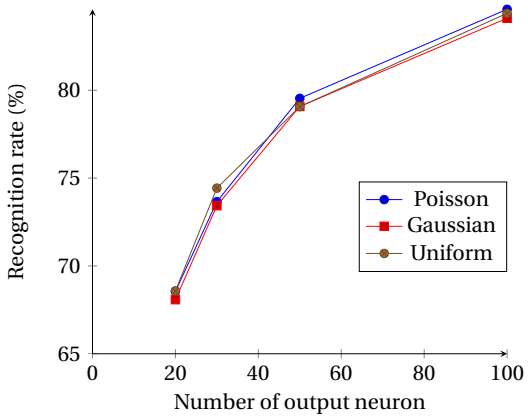

Figure 7.4: The recognition rate of the network using different number of neurons and three spike train distributions.

7.2.2 Effect of STDP window duration

Activity-dependent modification of synaptic weights because of STDP depends on the correlations between pre- and postsynaptic firing over timescales of tens of milliseconds [175]. Contrarily to a biological simulation, as we simulate hardware SNNs, we can choose some parameters. The way STDP is implemented in hardware is a very important parameter. We use a simplified form of STDP (from [176]) where the weights of the synapses are always slightly decreased except when there is a temporal correlation between a presynaptic firing and a postsynaptic firing during a time window of duration STDP window. We look here at the influence of this STDP window duration on the learnig capabilities of our networks.

The duration of the STDP window is related to the frequency of the input spike trains. As the maximum frequency is 63.75 Hz, the STDP window duration should be of approximately the same duration as the corresponding period, 15.7 ms, or higher. To be able to evaluate the optimum STDP window duration, we have started using a 15 ms duration and increasing to 65 ms by increments of 10 ms as it is depicted in Figure 7.5.

The results show a low performance using 15 ms regarding to other STDP window durations. We have a remarkable improvement from 15 ms to 25 ms and the best results are obtained using the range between 35 ms and 55 ms. At 65 ms, the recognition rate starts to decrease. Our interpretation is that a too short duration does not allow to capture all the high intensity pixels in the input, and a too long duration is not specific enough as it captures too many mid range pixels and thus is not specific enough. One would need to do additional experiments to check how the STDP window influences the learning speed of the network. Indeed, here we just check the final network state after two presentations of the 60000 images of the MNIST dataset, but the convergence speed of the SNN may well depend also on the STDP window duration. Figure 7.6 illustrates the average recognition rate of neural networks using various number of neurons.

7.2.3 Effect of neuron threshold

In our hardware simulation, we use the Leaky-Integrate-and-Fire (LIF) neuron model. This type of neuron model is fit for SNN architectures [118] due to several properties such as avalablility of low-power CMOS design using subthreshold regime transistor [10], fast to simulate, and particularly efficient for large-scale network simulations [119].

The LIF model is described in Chapter 4, Equations 4.8 and 4.9. If injected currents from synapses are large enough, they cause the action potential to pass the threshold voltage, and the neuron fires. It means there are enough input spikes in a short time-window. When there is no or only a small number of spikes in a time-window, the neuron is in the leaky phase and the state variable decreases exponentially. The duration of this time window depends on $\tau_n = RC$.

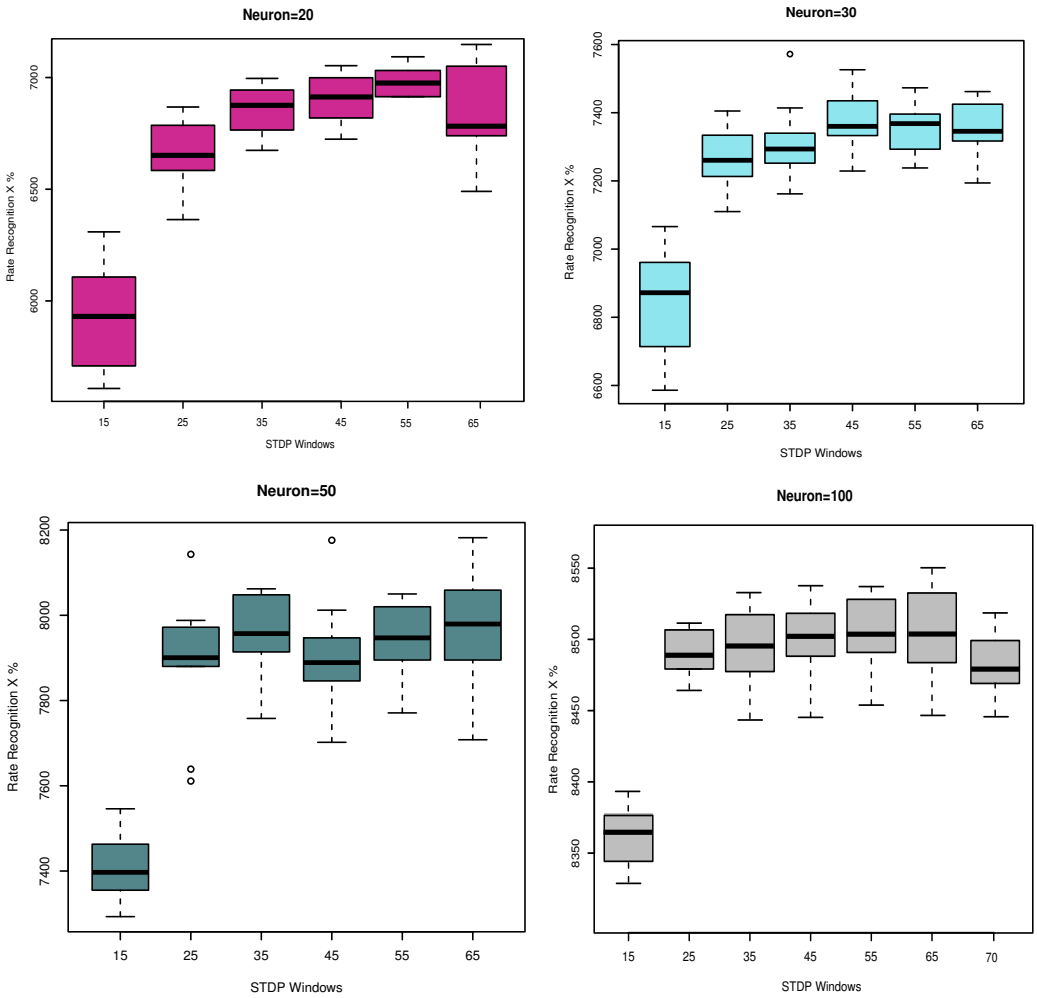

Figure 7.5: The comparison of different STDP-window duration for using different number of neurons=20, 30, 50, 100. The MNIST digits dataset after converting to the corresponding spikes to the pixels densities, are presenting to the network for 350 ms for (each frame). The 150 ms pause between each digit presenting are considered. This figure illustrates the performance of neural network using four various number of neurons and different STDP-windows.

To simulate the LIF neuron, we have to define a threshold for the neuron. Furthermore, we used the homeostasis technique to improve the network stability and performance. It means that if one or a group of neurons are too active when reading the input data, we slightly increase the threshold and vice versa if one or some neurons are inactive during the training process the threshold is decreased slightly. In this section, we verify and analyze the spiking neural network performance while using different thresholds for the same neurons to reach the best threshold value. To be able to trace only the impact of different thresholds on the system performance, we use the same homeostasis for all neurons with the same rate of increasing and reduction. The range of threshold values have been chosen to cover the minimum and maximum action potential in the network. Figures 7.7 and 7.8 shows the effect of the different thresholds on the recognition rate. The results show that the neuron thresholds between 25 and 35 mV lead to the best recognition rates. However, for the larger numbers of neurons choosing a threshold of 45 mV also leads to an acceptable performance. In contrast, for the networks with the smaller numbers of neurons the lower thresholds lead to a good recognition rate in the network. One could think that it is easily explained because in the networks with more neurons, each neuron is connected to a larger number of other neurons and thus receives more incoming spikes and thus reach its threshold sooner. But here, the number of inputs of the neurons is constant and

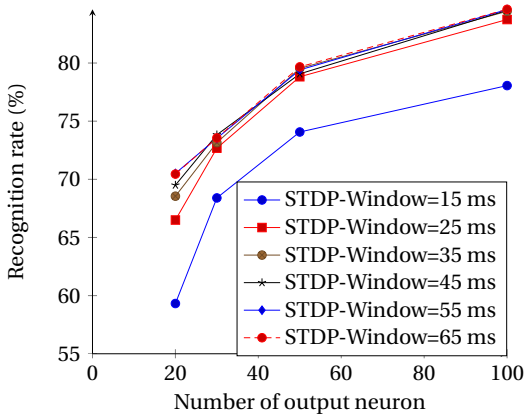

Figure 7.6: The recognition rate of the network using different number of neuron and six different STDP-wnidows.

equal to $28 \times 28 = 784$, the number of pixels in the image. Actually, the winner-takes-all rule increases the connectivity of the neurons inside the layer, but for inhibitory purposes. In general, the optimal value of the neuron threshold depends on the network architecture, and more generally on the activity of the network. In our case, the differences are not large enough to conclude and we would need to run a much larger number of simulations to check if this shift on the optimal value is real or just a random effect. In Figure 7.8, we compared all four threshold values for having a better comparison. The mean of the ten times running values is considered for this figure. As it is obvious from Figure 7.7 small threshold value has better performance with small number of neurons and larger threshold value has better performance with only the larger number of neurons. However if we want to choose one threshold for neurons in network, definitely the average values (35 and 25 mV) demonstrate better performance as it is more obvious in Figure 7.8.

7.2.4 Effect of synapse β parameter

The quality of the synapse and the rate of conductance change of synapses causes influence on the learning capabilitiesof SNNs. For instance using binary synapses which switch from zero to one will cause much lower performance in learning than using an analog synapse. Therefore the amount of change after updating the weights is important. The conductance change does not only depend on the STDP rule for modification but also on the characteristics of the synapse itself. Here, we evaluate the network learning performance when changing parameters in the synapse to modulate the synaptic weight change rate.

For the model of memristor as an artificial synapse, we use the model introduced in [153, 154] which is inspired from experimental memristive devices measurements [45]. The increasing and decreasing of conductance is presented by Equation 7.5 and Equation 7.6 respectively.

$$\Delta g_{inc} = \alpha_{inc} e^{-\beta \frac{g - g_{min}}{g_{max} - g_{min}}} \tag{7.5}$$

$$\Delta g_{dec} = \alpha_{dec} e^{-\beta \frac{g_{max} - g}{g_{max} - g_{min}}} \tag{7.6}$$

g_{max} and g_{min} are the maximum and minimum of conductances of the memristor. α_{inc} and α_{dec} characterize the conductance step and β is a fitting parameter. To evaluate the impact of synaptic conductance on learning in the neural network, we vary the β fitting parameter because it directly affects the amplitude of the weight modifications. The other parameters in the simulation are fixed to $g_{max} = 1$, $g_{min} = 0.0001$, $\alpha_{inc} = 0.01$ and $\alpha_{dec} = 0.005$. The initial conductance of the nanodevices before starting to train the network are chosen randomly around the mid-range. We observed no remarkable effective impact on the network performance, which is a proof of neural network robustness

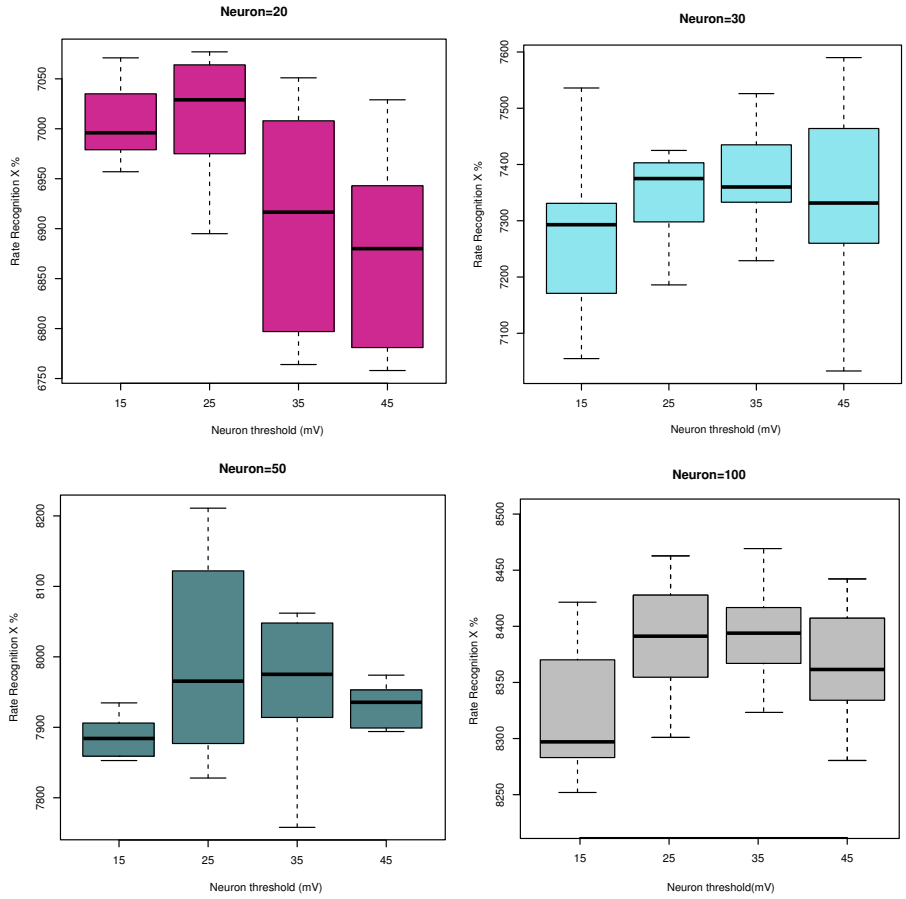

Figure 7.7: The comparison of various threshold (15, 25, 35, 45 mV) for using different number of neurons=20, 30, 50, 100. The threshold between 25 and 35 mV demonstrate better performance in the network, however, in the network with smaller number of neurons the neuron with less threshold have still acceptable performance and on the contrary in larger neural networks the neuron with more firing threshold voltage (such as 45 mV using for the 100 output neurons) have demonstrated acceptable performance too.

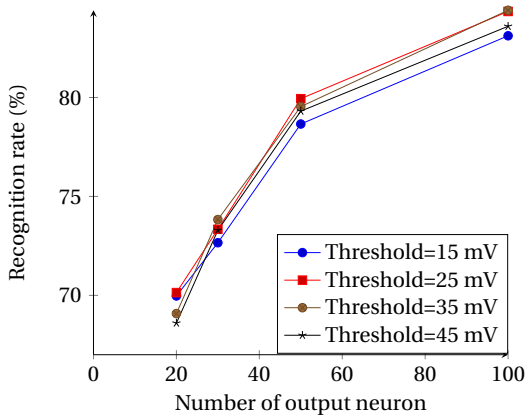

Figure 7.8: comparing network performance using various number of neuron with different threshold

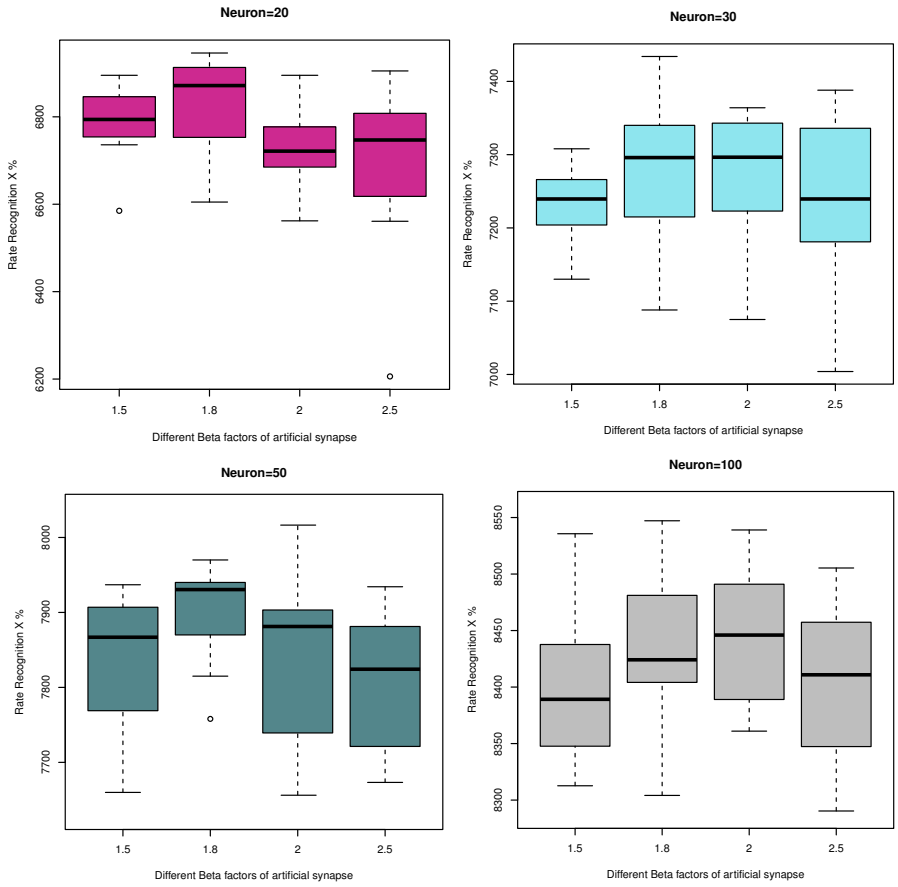

Figure 7.9: The comparison of various fitting parameters (β) for using different number of neurons=20, 30, 50, 100. The results demonstrate better performance using β between 1.8 and 2. However, the differences are not distinguishable which is a prove of memristor devices robustness to the variations.

	Synapse	Distrib.	STDP w.	Thres.	β
Best	VNV	Poisson	55 ms	35 mV	2
Baseline	NV	Poisson	25 ms	15 mV	1.5

Table 7.1: Best parameters vs. baseline parameters

to variations as is reported in [176]. The results for four different number of neurons in the output are presented in Figure 7.9 and 7.10. Although the network performance variation using different β parameters is not remarkable, the results demonstrate slightly better performance using β between 1.8 and 2.

7.2.5 Discussion

As a final experiment we compare the baseline (values inspired by the litterature) with the best values for all the evaluated parameters, including the volatile/non volatile synapse of Chapter 6. The parameters are listed in table 7.1. Other network elements and parameters that are not listed in table 7.1 are the same for both models.

We have evaluated both models using the same number of output neurons that were used in the parameter evaluation sections with the addition of 300 neurons for the output layer. The results in Figure 7.11 demonstrate enhanced performance using the best obtained parameters rather than baseline parameters. These results, as most simulations shown in this study, show that the variation of

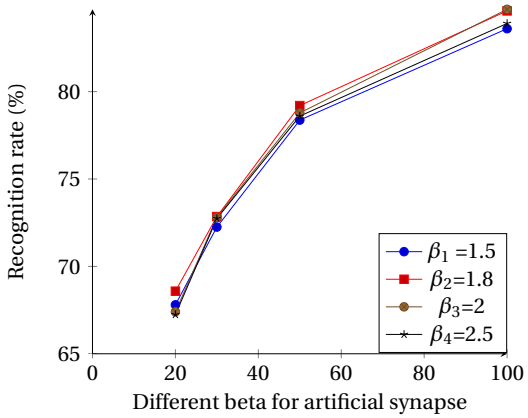

Figure 7.10: comparing network performance using various number of neurons with different fitting parameter (β) for synapse model.

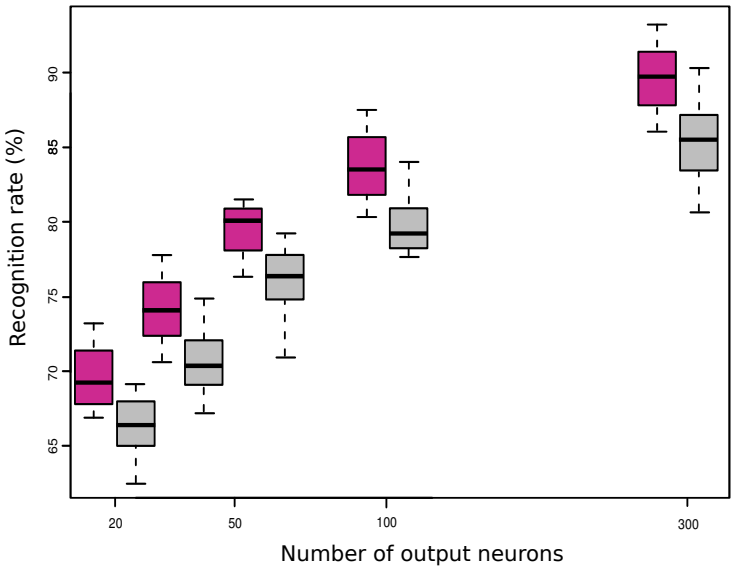

Figure 7.11: Using the best parameters significantly improves the recognition rate.

recognition rate due to the random initialization of the synaptic weights is around 6 to 8 points with half of the results in a 3 to 4 point range. The ranges of variation of the best version and of the baseline overlap, so one does not have the guarantee that using theoretically better parameters will lead to a better performing network, it is just statistically better.

As a summary, we can conclude from this study that:

- The parameters that have a significant influence on the learning rate are the number of neurons in the SNN, the kind of synapse (volatile or synapse box), the duration of the STDP window and the neuron threshold value.
- The distribution of the spikes in the input spike trains and the β fitting parameter of the volatile synapse do not have a significant impact on the recognition rate.
- One can significantly improve the recognition rate of a SNN by choosing better parameters.
- There is a relatively large spread of the recognition rate due to the random initialization of the synaptic weights in the simulations.

Thus a design space exploration should concentrate on the parameters that influence significantly the recognition rate, first of all the architecture of the network (number of neurons, topology), and

then tune the other parameters. This design space exploration should be based on a statistical analysis of the performance of the network by running several simulations for each combination of parameter values. Finally, researchers should explain clearly what they mean when they give a single number as the recognition rate of a given SNN, is it a mean of several rates, the best rate or just a single rate.

7.3 Conclusions

We have presented an empirical study of the influence of several parameters on the recognition rate of memristor based hardware SNNs on the MNIST benchmark. This study is based on simulations run with N2S3, an open source simulator we have developped to help design neuromorphic circuits.

This study has shown that not all parameters have a significant influence on the learning rate of SNNs and thus that a design space exploration should concentrate first on the architecture of the network; then, the kind of device used as a synapse, the STDP mechanism used and its parameters, and the threshold of the neuron. This study is only valid on the MNIST benchmark and should be complemented by similar studies on other test cases to confirm these findings, especially test cases using natural data in spiking form where the precise relative timings of the input spikes would necessitate more precise STDP mechanisms than the simplified one used in this study that is only sensitive to the average frequency of the input spike trains.

In the future, we will explore different models of synapses and neurons, more complex network topologies and STDP mechanisms, and enhance the N2S3 simulator with automatic design space exploration facilities that will concentrate on optimizing the most significant parameters discovered in this study. In addition to the recognition rate (or classification capabilities), we will also evaluate other performance measures such as the power consumption of the circuit or its convergence speed.

Chapter 8

Deep learning in spiking neural network

Abstract

Deep learning recently is used in several state-of-the-art studies and developments due to its ability in pattern recognition data classification. Most of the research work and applications are in machine learning domain. In this chapter, we introduce a framework to use the proposed rate based version of Contrastive Divergence (CD) updating weight rule. Respecting to the rate aspect of neural coding, we develop a Spike-Based Deep Belief Network (S-DBN) with Leaky Integrate-and-Fire (LIF) neurons and the model is evaluated using ORL face detection dataset. The proposed method provides a suitable framework that can be used in implementing deep architectures in the neuromorphic hardware systems.

8.1 Introduction

Deep learning is currently very active research area in machine learning and pattern recognition society due to its potential to classify and predict as a result of processing Big data and Internet of Things (IoT) related data. The implementation on neuromorphic hardware platforms emulating large-scale networks of spiking neurons may present important advantages from the perspectives of scalability, power dissipation and real-time interfacing, and better recognition performance in neural network learning [177]. Considering brain mechanisms, prerequisite for an intelligent system is utilizing multi- level architecture such that each level of the model provides extracted features for the next one. This approach of feature extraction leads the system to implement complex functions using higher level of abstraction. Deep Learning is a set of powerful machine learning methods for training deep architectures. Considering the inherent inefficiency of learning methods from traditional Artificial Neural Networks in deep architectures, Contrastive Divergence (CD) has been proposed to train Restricted Boltzmann Machines (RBM) as the main building blocks of deep networks [38]. Deep architectures can be developed by stacking RBMs and training the layers using Contrastive Divergence (CD) as the most efficient algorithm in deep learning in a greedy layer-wise approach [26].

Despite the proficiency of deep architectures in machine learning tasks, implementing these models in common platforms can be a very time and resource consuming process. The conventional digital computers because of using Von Neumann architecture, waste too many energy specifically through storing results and retrieving data in/from memory elements [178]. The neuro-inspired computational platform which has memory next to the computation unit, could be an alternative solution for efficient computation using CD algorithm. Spiking Neural Networks (SNN) are more close to biological neurons rather than abstract mathematical models. The important aspect of spiking model of neurons is its potential for hardware implementation as a very specific and power efficient hardware accelerator. In SNN the neurons communicate using spikes [39]. Therefore, we have to design an NN architecture that implement spiking data. In this chapter, we present a framework for using Contrastive Divergence to train an SNN using RBM and spiking LIF neurons. This framework

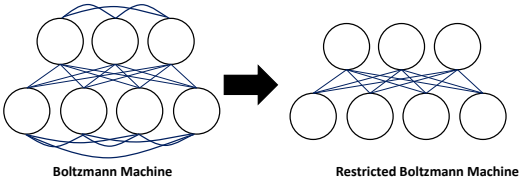

Figure 8.1: Restricted Boltzmann Machine is a network of neurons which neurons in one layer are connected to all neurons in the next layer.

can open a new window toward the neruromorphic architecture designer to apply the state-of-the-art of machine learning learning algorithm in SNN architecture.

To overcome the challenge of using spiking neurons in machine learning domain, one uses the abstract model of neuron. OConnor *et al.* [97] applied an abstract model of Leaky Integrate-and-Fire (LIF) neuron called Siegert neuron [179] to approximate the average of output firing rates of LIF neurons. Using this abstract model as a unit of a Deep Belief Network (DBN) and exploiting the standard Conterastive Divergence (CD) algorithm, they have trained an DBN and the adjusted weights are transferred to a functionally equivalent Spiking platform of LIF neurons. Using such an offline learning approach, all the weights are fixed and no weight adjustments can be performed in application platform.

Spike-Timing Dependent Plasticity (STDP) as the main Hebbian-based learning method, is used for updating the synaptic weights in SNN [39]. This spike-based learning rule in a time-averaged can be interpreted as anti-Hebbian and Hebbian correlations between input and output spikes [180]. In [177] utilizing an abstract model based on neural sampling a STDP-based version of CD has been proposed and a single RBM is trained online successfully.

In this work, regrading the rate aspect of neural coding, we proposed a rate coded contrastive divergence as an online learning rule for spike-based RBM. Consequently, using the proposed learning approach, an acceptable accuracy of a Spike Based DBN is demonstrated.

Finally, the remainder of this study is organized as follows. In Section 8.2, we discuss the mathematical backgrounds of the CD and compare the spike coding with the rate coding in the neural architecture. In Section 8.3, after studying the preliminary principles of an online model, a contrastive divergence adapted to the spike-based RBMs is proposed. The proposed approach is developed and verified in Section 8.4 for a single spiking RBM and also for the desired Spike-Based Deep Belief Network. The model is evaluated in Section 8.5. Finally, the conclusion as well as future possible works have been discussed in Section 8.6.

8.2 Restricted Boltzmann Machine and Contrastive Divergence

Restricted Boltzmann machines, as a model of artificial neural networks, have been driven from Boltzmann machines. RBM consists of binary stochastic units connected to each other using bidirectional edges. It can represent a probabilistic distribution to learn the basic features of an unknown distribution using observed data, which is considered as the training data. Generally, training in Boltzmann machine, as fully Recurrent Neural Network (RNN), is involved with a large number of complex computations. Therefore, applying some restrictions to the Boltzmann machine topology leads to a less complex structure called Restricted Boltzmann Machine (Figure 8.1). From a structural point of view, RBM has one visible and one hidden layer, where all the units in the visible and hidden layers are symmetrically connected, but there is no visible-visible or hidden-hidden connection. The structure of the Boltzmann machine is related to the proposed structure in 1982 by John Hopfield [111]. The Hopfield model is driven from thermodynamic systems and can be quantified through equilibrium energy. Each state of the Boltzmann machine can be expressed as a value called energy of the state.

Equation 8.1 presents the energy function of a given RBM as a restricted type of Boltzmann machine.

$$E(V,H) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_i a_i v_i - \sum_j b_j h_j, \tag{8.1}$$

where E is the total energy of the network, v_i is the state of visible i^{th} unit, h_j is the state of j^{th} hidden unit, w_{ij} is the weight between v_i and h_j, a_i and b_j are the biases. The assigned probability to each configuration of the network states are:

$$p(V,H) = \frac{1}{Z} e^{-E(V,H)}, \tag{8.2}$$

where $Z = \sum_{VH} e^{E(V,H)}$ is a partition function. RBM, as a generative model, tries to generate an internal representation of its environment. Increasing the log-probability of the generating input data vector using equation 8.2 leads to contrastive divergence [181] updating weight rules for RBM:

$$\Delta w_{ij} = \eta(< v_i h_j >_{data} - < v_i h_j >_{model}). \tag{8.3}$$

In equation 8.3, $< v_i h_j >_{data}$ represent the expectation under the distribution specified by input data vector and $< v_i h_j >_{model}$ is the expectation under the distribution specified by internal representation of the RBM model. Also the probability of h_j using a given V as the input data vector, for each j in hidden layer is 1 with probability $p(h_j = 1|V)$:

$$p(h_j = 1|V) = \sigma(b_j + \Sigma_i v_i w_{ij}), \tag{8.4}$$

when $\sigma(x)$ is the logistic sigmoid function and can be defined as:

$$\sigma(x) = \frac{1}{1+e^x} \tag{8.5}$$

In contrastive divergence using Gibbs updating chain (initiated with training data) and iteration sampling for a limited time, the approximated value for $< v_i h_j >_{model}$ can be computed perfectly [181, 182]. A single step of Gibbs sampling using H as a given hidden vector can be expressed as

$$p(v_i = 1|H) = \sigma(a_i + \Sigma_j h_j w_{ij}). \tag{8.6}$$

According to [182] only one step of contrastive divergence (CD_1), can provide an acceptable approximation of gradient of the log probability of the training data and iterating the process for k times provides a more precise value:

$$(\Delta w_{ij})^k = \eta(< v_i h_j >^0 - < v_i h_j >^k), \tag{8.7}$$

when $< v_i h_j >^0$ is equal to $< v_i h_j >_{data}$ and $< v_i h_j >^k$ means iterating Gibbs sampling for k times. Also the biased will be updated using these equations:

$$(\Delta a_i)^k = \eta({v_i}^0 - {v_i}^k), \tag{8.8}$$

and

$$(\Delta b_j)^k = \eta({h_j}^0 - {h_j}^k). \tag{8.9}$$

8.3 Deep learning in artificial neural networks versus spiking neural networks

Artificial Neural Networks as a tool of artificial intelligence can demonstrate high level of accuracy in machine learning problems and specifically in face recognition applications [183, 184]. On the other side the brain inspired models and specifically Spiking Neural Networks are very suitable to

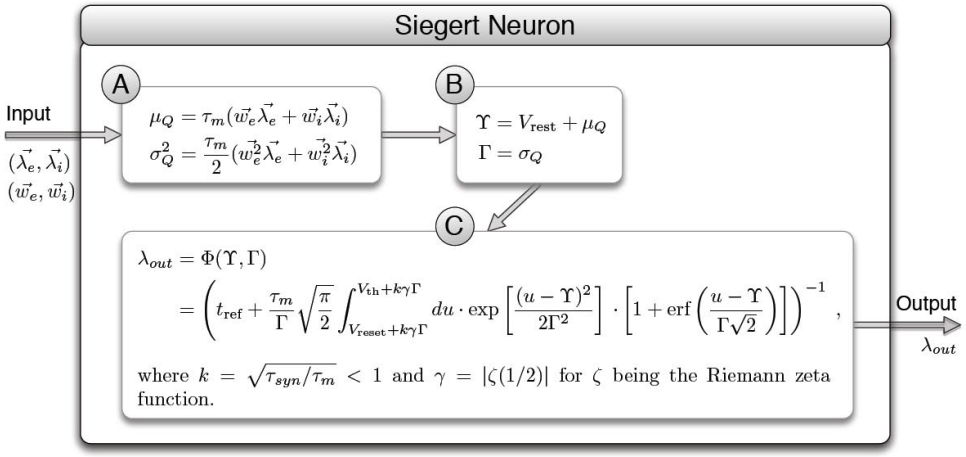

Figure 8.2: Siegert abstract neuron model [6]

be implemented in VLSI. Therefore, to have a biological model of neural networks in hardware level SNN is applied. The SNN can be trained using an unsupervised Hebbian-based learning rules such as Spike-Timing Dependent Plasticity (STDP). However respecting to the deep architecture of the brain to have a deep SNN, we have to use efficient algorithms for deep architectures.

Spiking model of neuron uses short pulses (spikes) to coding data [39]. Since the shape of spikes are generally the same, then the rate and the time of spikes are the determinant parameters in data transferring in biological models. Comparing the biological inspired model to machine learning algorithms, it is obvious that the time parameter has no role in data coding in machine learning domain. Consequently, we have to overcome this gap between ANN and SNN if we want to use machine learning algorithms (e.g., ANN) in SNN platforms.

In this study, we use the Siegert neuron model that can approximate the mean firing rate of Leaky Integrate-and-Fire neurons with Poisson-process inputs. Siegert abstract model using mathematical equations can estimate the input-output rate transfer function of Leaky Integrate-and-Fire neurons. Figure 8.2 shows the equation of the Siegert neuron [6, 179]. SNNs use spike trains with randomly distribution of spike times. To be able to use Siegert model as a unit in our network, we assume the spike trains are Poisson-process with specific firing rates (λ_{in}). As it is depicted in Figure 8.2, the Siegert model receives excitatory and inhibitory inputs where λ_e is the excitatory and λ_i is the inhibitory rates of incoming spike trains from pre-synaptic neurons. w_e and w_i are the corresponding synaptic weights. Due to more simplicity, we did not categorize the neurons to excitatory and inhibitory neurons. We assume λ_{in} as total input spike rates from pre-synaptic neurons as well as w which indicate the both of w_e and w_i. By normalizing the values of pixels of an image, we can convert the density of each pixel to the spike rate of corresponding pixel such that brighter pixel has higher spike rate and darker one has less firing rate. LIF neuron parameters can be set by adjusting the corresponding variables in Siegert equation. Table 8.1 shows the given values for the membrane time constant (τ_m), the resting potential (V_{rest}), the reset potential (V_{reset}), the threshold potential (V_{th}) and the absolute refractory time (t_{ref}).

In the following sections stacking the proposed RBM with Siegert units, we developed a Deep Belief Network. In this network, after training phase, we transferred the weight matrix to a DBN with LIF units using Brian simulator [147] to evaluate the proposed model. According to the equality between Siegert transfer function and LIF transfer function, we can use the LIF neurons with the same parameters (Table 8.1).

Table 8.1: LIF parameters

Parameter	Description	value
τ_m	Membrane time constant	5sec
V_{rest}	Resting potential	0
V_{reset}	Reset potential	0
V_{th}	Threshold potential	5mv
t_{ref}	Absolute refractory time	2ms

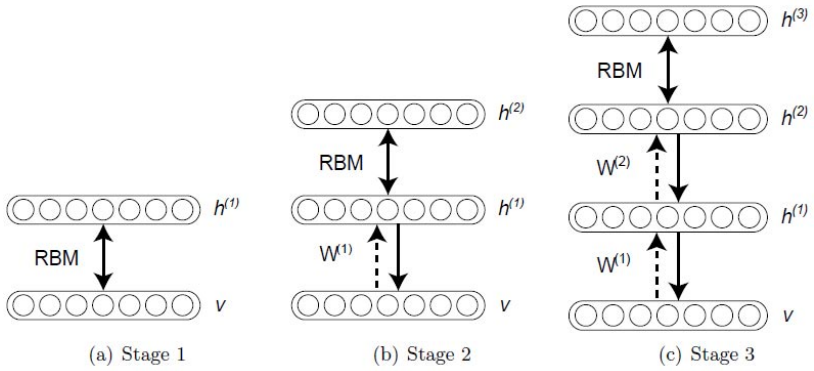

Figure 8.3: Stacking RBMs as the main building blocks of DBN

Figure 8.4: Some sample images from ORL dataset

8.4 Developing and Training Deep Belief Network with Siegert Units

The RBMs with Siegert units can be trained in machine learning domain. In fact in this model the mean firing rate of input spike trains will be used instead of spike trains.

Hinton *et al.* introduced an efficient Deep Belief Networks (DBN) for the first time [26]. They have proposed a greedy layer wised algorithm to train the DBN by training each RBM sequentially. Figure[1] 8.3 illustrates the stages of stacking RBMs. In this work, we use the Olivetti Research Laboratory (ORL) face detection dataset from Cambridge, UK [2]. This dataset originally contains 400 grey scale face images of 40 distinct volunteers. Figure 8.4 shows some images of ORL dataset. In this chapter, we use a resized version of ORL that has been proposed in [185]. The input images with 32×32 pixels are used as input vectors with 1024 elements. According to [186], in each step of RBM training process,a

[1] http://deeplearning.net/tutorial/

[2] The source of the Database of Faces: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

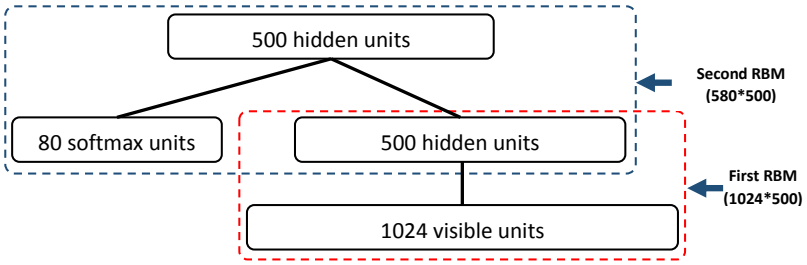

Figure 8.5: The proposed DBN with Siegert neurons for learning ORL

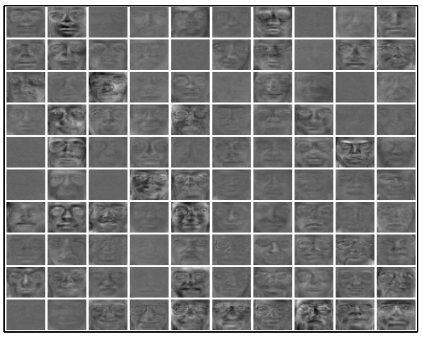

Figure 8.6: Visualizing the learned features by hidden units

mini-batch of training data consisted of a small subset of all training data is used. The proposed DBN architecture, respecting to the size of training vectors, is depicted in Figure 8.5. The first RBM has 1024 (32×32) units in visible layer and 500 units in hidden layer. This RBM is trained without any label (unsupervised learning) to provide abstract features for the second one. After training the first RBM, the second RBM is trained using the extracted features generated in previous step. In the recent RBM as a classifier, we use the joint of 500 extracted values and 80 softmax units. The labels of ORL images have been converted to softmax vectors. For the first class of images the two first bits are one and others are zero. For the next one, the two first bits are zero, the next two bits are one, and the others are zero and so on.

We divided the ORL images into two subsets. The first one is the training set and consisted of 8 images from 10 of each class. These images are used to train the model and are not used for testing. The second one is the test images containing 2 of 10 from each class. Because of the full connections between each visible unit and each hidden unit, the dimensions of the arriving connections at each hidden unit are the same as the dimension of input images. Figure 8.6 shows the corresponding weights of connections between all visible units and 100 randomly selected hidden units. The weight vectors has been reshaped as 32×32 images. During the learning process the hidden units have learned some specific features such that they can be triggered only with those specific features. Visualizing these weight vectors (Figure 8.6), displays the learned features by each hidden unit. Therefore, it is a appropriate monitoring method to study the network learning process [186]. As we can see in Figure 8.7, the training process needs too many iterations to reach a proper result. For this model which is implemented in Matlab, after about 800 iterations, the results are close to 90%. The maximum value, 93.2%, corresponds to iteration 1910^{th}. In this study, we use the free energy function (equation 8.10) to find the predicted label. Using this function, each possible label has been tested to find the configuration with the lowest energy. The corresponding label to the recent configuration is assumed as the predicted label [186].

$$F(V) = -\sum_i v_i a_i - \sum_j \log(1 + e^{x_j}) \tag{8.10}$$

where $x_j = b_j + \sum_i v_i w_{ij}$.

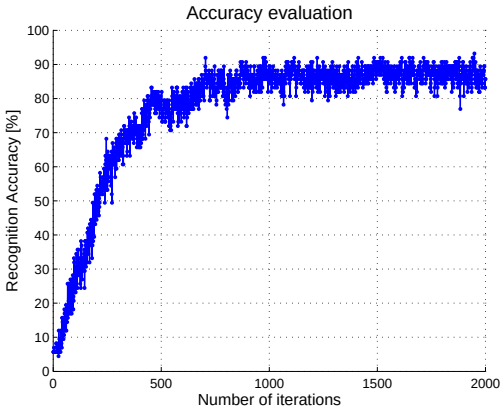

Figure 8.7: Accuracy of the proposed DBN with Siegert neurons in face recognition on ORL dataset

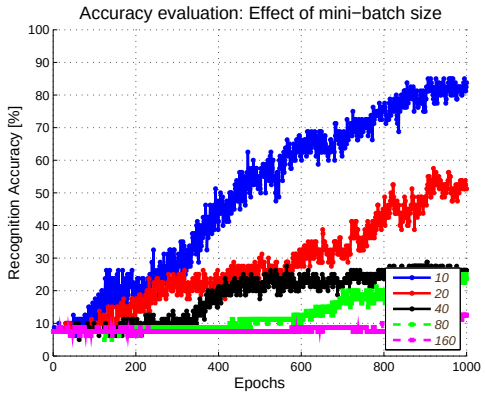

Figure 8.8: Decreasing the number of epochs and increasing the mini-batch size can reduce the model accuracy

The accuracy of the model depends on the learning parameters [186, 187]. For example Figure 8.7 shows the results when the mini-batch size is 4. The effect of changing the parameters of model using less iterations and using various mini-batch sizes is depicted in Figure 8.8 . Obviously, less iteration and larger mini-batch size lead to less accuracy. Eventually, if one is interested in more precise accuracy, it can be possible through adjusting the learning parameters and also the Siegert neuron parameters (Table 8.1).

8.5 Evaluating the model

Thanks to the equality between the Siegert neuron's transfer function and the LIF transfer function, without any adjustments, the trained weight matrix in the previous section can be copied to a network with the same topology consisted of LIF neurons with the same parameters (Tabel 8.1). To develop such a network with LIF neurons, we use the Brian simulator. Brian is a simulator for Spiking Neural Networks. This simulator is written in the Python programming language. Because of using development tools such as SciPy module, Python provides very fast routines for mathematical operations and specifically matrix operations [147]. We have developed a Deep Belief Network in Brian simulator with same topology as the one that has been implemented in Matlab. In this model since we want to test the accuracy of the model in a more realistic situation, the equation of LIF neuron (Equation 8.11) is applied besides the described parameters in Table 8.1.

$$\tau_m \frac{dv}{dt} = -(v(t) - v_{rest}) + RI(t) \tag{8.11}$$

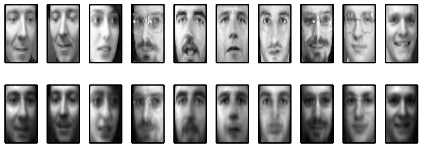

Figure 8.9: The upper row shows 10 of the training images and the lower one illustrate the corresponding reconstructed images

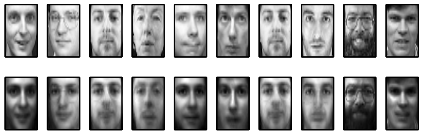

Figure 8.10: The upper row shows 10 of the test images and the lower one illustrate the predicted images

We know the spiking model uses spike trains instead of the real numbers. In Section 8.4, we discussed the assumption to use the normalized value of the pixels to produce the related firing rate. Consequently in this section to test the proposed spiking model, we have to convert the firing rate to the spike trains. In [174], we have presented more in details the converting approach for MNIST handwritten digits. In Brian simulator, using *PoissonGroup* function, we can generate spike trains with specified firing rates. Therefore, the network can be tested with spike trains corresponding to the density of pixels of the test images. Having the trained weights matrix, despite of the Matlab model, the labels are not used in the second RBM for training however, they are used as the outputs of the model for classifying the input images [188]. Respecting to the generative characteristic of DBNs, the model not only can reconstruct the input images as the internal representations of the given images (Figure 8.9), but also can generate the corresponding learned labels as the predictions of the model. To evaluate the model in a spiking framework after transferring the weights matrix, the generated spike trains of test images are passed through the entire of the network. Furthermore, comparing the predicted labels with the original labels, we compute the model accuracy. Figure 8.10 shows predicted images respecting to the input test images. In addition, as it was predictable, the results of the model with LIF neurons have not changed considerably. However because of the difference between floating number precision in Matlab and Brian simulator reloading the weights matrix in Python can cause a small decreasing in accuracy. The accuracy of the model in Brian simulator is reduced to 92.4%.

8.6 Conclusion and future works

The ability of brain-like computing has motivated us to evaluate a model with biological structure in face recognition application. Regarding the brain as a deep neural network, we have proposed a deep neural network with spiking neurons to understand if the brain-like models are suitable for face recognition applications. The proposed model is different from a traditional neural network. Indeed this model is a Spike-Based deep model with biological inspired neurons (Leaky Integrate-and-Fire neurons). Considering the results of Deep Belief Networks in various tasks in Machine learning domain, a DBN with two RBMs has been developed in Matlab with Siegert units. Consequently, the trained weights matrix is transferred to a Spiking DBN with LIF neurons. The recent model is simulated in the Brian simulator and the results show the capability of the Spiking Deep Belief Networks in a simple face recognition application.

For future works, we will take into account using other types of deep models such as Deep Autoencoders or Convolutional Neural Networks. In this work, we had to train the model in one platform and using the trained weights in another one. Indeed our model is an offline training model. Using the

same platform for training and utilizing the model leads us to an online model of training, which can be considered as the next work. Additionally, we are looking for a way to perform the same work in a single platform more suitable for hardware implementation by using the emerging nanodevices such as memristor to realize the synapse model.

Chapter 9

Conclusion

The growing use of ICT devices and technologies in our daily life increases the demand for high performance, fast (parallel) and power-efficient computations. In present computing architecture, the way of communicating between memory and central processing unit (CPU) is not efficient particularly while we are facing the problem of Big Data processing. The reason of this inefficiency is called the memory wall issue or Von Neumann memory bottelneck. This memory bottelneck issue is because of speed disparity between the CPU and memory outside the CPU chip.

As we are reaching to the end of Moore's law, the possibility of increasing the pulse clock of CPUs in order to increase the speed of system is facing serious problems. The multi-core processor was a solution for the power dissipation problem. However, memory hierarchy and memory management would be more significant and difficult in multi-core processors. Therefore, the memory wall problem is actually worse in multicores.

New unconventional computing techniques using the emerging nano-devices such as memristors have been proposed as solutions to Von Neumann architecture problem. Here in this dissertation, we study the different possibilities of using this nano-device both as a digital logic and analog memristive switch. Therefore, we began our research with a feasibility study of using memristive devices in different domains. We briefly review the literature about memristors from mathematic model to the physical realization and its applications. We discuss different classes of memristors based on the material used to build the devices. Four general memristive devices are studied in the first chapter: resistive, spintronic, organic (polymeric) and ferroelectric memristors. The potential application as well as advantages versus disadvantages of using each one are presented too. The resistive memristor has been used more than others in different research works from memory to artificial synapse. The spintronic and ferroelectric devices show promising properties to make new nonvolatile memories. The organic memristor is more appropriate to make artificial synapse in Spiking Neural Networks.

In the digital domain, using the capability of memristors to have the processor and the memory in the same unit, we propose two different architectures using non-volatile memristors. The first approach is based on material implication and the second one is based on crossbar arrays. A library of logic gates are provided using both methods. We compare both methods and the reconfigurability is the most significant advantage of using the crossbar array architecture.

Neuromorphic computation using Spiking Neural Networks (SNNs) has been proposed as another alternative solution for the future of high performance computation. Neurons communicate together using spikes. Therefore different spike codings have been discussed to improve data transferring and data processing in neuro-inspired computation paradigms. Choosing the appropriate neural network topology could result in better performance of computation, recognition and classification. The model of the neuron is another important factor to design and implement SNN systems. The speed of simulation and implementation, the ability of integration to the other elements of network, and the suitability for scalable networks are the factors to select an optimized neuron model. The sequential order of pre- or postsynaptic spikes occurring across a synapse in an interval of time leads

to define different learning methods. In chapter 4, we studied the differences among the methods to help to choose the most adapted STDP method to a given network.

We present N2S3 (for Neural Network Scalable Spiking Simulator), an open-source simulator that is built to help design spiking neuromorphic circuits based on nanoelectronics. N2S3 is an event-based simulator and its main properties are flexibility, extensibility, and scalability. N2S3 has been developed from the ground up to model various kinds of neuron and synapse models, various network topologies, various learning procedures, various reporting facilities, and to be user-friendly, with a domain specific language to easily express and run new experiments. Experimental set up for two standard tasks are provided in N2S3: handwritten digit recognition on the MNIST dataset and car counting on the Freeway dataset.

Furthermore, we have introduced a novel synapse box with the possibility to forget and remember inspired from biological synapse properties. This synapse box is composed of a volatile memristor (NOMFET) followed by a nonvolatile resistive RAM. The volatile memristor acts like a high-pass filter to enhance short term plasticity and the nonvolatile resistive RAM enables long term potentiation. To evaluate and verify the new synapse box as well as the functionality of the N2S3 simulator, we have used the MNIST handwritten digit dataset. The first results demonstrate an improvement in recognition rate by using the synapse box over a single nonvolatile memristor synapse.

We reviewed the most important existing platform in neuromorphic computing domain. Additionally We have presented an empirical study of the influence of several parameters on the recognition rate of memristor based hardware SNNs on the MNIST benchmark. This study has shown that not all parameters have a significant influence on the learning rate of SNNs and thus that a design space exploration should concentrate first on the architecture of the network; then, the kind of device used as a synapse, the STDP mechanism used and its parameters, and the threshold of the neuron. This study is only valid on the MNIST benchmark and should be complemented by similar studies on other test cases to confirm these findings, especially test cases using natural data in spiking form where the precise relative timings of the input spikes would necessitate more precise STDP mechanisms than the simplified one used in this study that is only sensitive to the average frequency of the input spike trains.

Deep learning recently is used in several state-of-the-art studies and developments due to its ability in pattern recognition data classification. Finally, we have proposed a deep neural network with spiking neurons to verify the functionality of the brain-like models in SNN for face recognition applications. We introduced a framework to use the rate based version of Contrastive Divergence (CD) updating weight rule. Indeed this model is a spike-based deep model with biological inspired leaky integrate-and-fire neurons. Considering the results of Deep Belief Networks (DBNs) in various tasks in machine learning domain, a DBN with two restricted boltzmann machines has been developed in Matlab with Siegert units. Then the trained weights matrix is transferred to a spiking DBN with LIF neurons. The recent model is simulated in the Brian simulator and the results show the capability of the Spiking Deep Belief Networks in a simple face recognition application is comparable with traditional machine learning performance.

To introduce the potential of the future works, there are several possibilities not only in unconventional computing domain but also in using emerging technologies such as memristor. The mix combination of memristive devices using different materials propose new research studies on the capability of both devices such as what we have proposed in Chapter 6.

As a proposition for the next study in digital domain, one can organize a Programmable Logic Device (PLA) platform in such a way that the crossbar array operates as a programmable AND array beside material implication as a fixed connection for the OR array. Therefore new generation of FPGA could be the next target of such a study.

For future work in neuromorphic and SNN field, as they are not well-studied yet, there are plenty of spaces for research in this domain. Studying different neural network topologies such as deep belief, recurrent and convolutional neural networks to evaluate the different memristive synapses considering the costs (area, energy, manufacturability, variability). We can explore different models of synapse and neurons, more complex network topologies and STDP mechanisms, and enhance the

N2S3 simulator with automatic design space exploration facilities that will concentrate on optimizing the most significant parameters discovered in this study. In addition to the recognition rate (or classification capabilities), we can also evaluate other performance measures such as the power consumption of the circuit or its convergence speed.

Bibliography

[1] Dmitri B. Strukov, Gregory S. Snider, Duncan R. Stewart, and R. Stanley Williams. The missing memristor found. *Nature*, 453(7191):80–83, May 2008.

[2] Yurij V. Pershin and Massimiliano D. Ventra. Spin Memristive systems. *Physical Review B.*, 78:113309, 2008.

[3] Y. Kaneko, H. Tanaka, M. Ueda, Y. Kato, and E. Fujii. A novel ferroelectric memristor enabling nand-type analog memory characteristics. In *Device Research Conference (DRC), 2010*, pages 257 –258, june 2010.

[4] E. M. Izhikevich. Simple model of spiking neurons. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(6):1569–1572, 2003.

[5] Baktash Babadi and L. F. Abbott. Stability and Competition in Multi-spike Models of Spike-Timing Dependent Plasticity. *PLOS Comput Biol*, 12(3):e1004750, March 2016.

[6] Florian Jug, Johannes Lengler, Christoph Krautz, and Angelika Steger. Spiking networks and their rate-based equivalents: does it make sense to use siegert neurons? *Swiss Society for Neuroscience*, 2012.

[7] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Sensing as a Service Model for Smart Cities Supported by Internet of Things. *arXiv:1307.8198 [cs]*, July 2013. arXiv: 1307.8198.

[8] Massimiliano Di Ventra and Yuriy V. Pershin. Memcomputing: a computing paradigm to store and process information on the same physical platform. *CoRR*, abs/1211.4487, 2012.

[9] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, August 2015. arXiv: 1506.03264.

[10] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, Johannes Schemmel, Gert Cauwenberghs, John Arthur, Kai Hynna, Fopefolu Folowosele, Sylvain Saighi, Teresa Serrano-Gotarredona, Jayawan Wijekoon, Yingxue Wang, and Kwabena Boahen. Neuromorphic Silicon Neuron Circuits. *Frontiers in Neuroscience*, 5, May 2011.

[11] L. Chua. Memristor-The missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, September 1971.

[12] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Letters*, 10(4):1297–1301, April 2010.

[13] Yenpo Ho, Garng M. Huang, and Peng Li. Nonvolatile memristor memory: device characteristics and design implications. pages 485–490, 2009.

[14] Julien Borghetti, Zhiyong Li, Joseph Straznicky, Xuema Li, Douglas A. A. Ohlberg, Wei Wu, Duncan R. Stewart, and R. Stanley Williams. A hybrid nanomemristor/transistor logic circuit capable of self-programming. *Proceedings of the National Academy of Sciences*, 106(6):1699–1703, 2009.

[15] G. Ch Sirakoulis and S. Hamdioui. Editorial note on memristor models, circuits and architectures. *International Journal of Unconventional Computing*, 12:247–250, December 2016.

[16] Garrett S. Rose. Overview: Memristive devices, circuits and systems. In *ISCAS*, pages 1955–1958. IEEE, 2010.

[17] Julien Borghetti, Gregory S. Snider, Philip J. Kuekes, Joshua J. Yang, Duncan R. Stewart, and R. Stanley Williams. 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature*, 464(7290):873–876, April 2010.

[18] Mahyar Shahsavari, M. Faisal Nadeem, S. Arash Ostadzadeh, Philippe Devienne, and Pierre Boulet. Unconventional digital computing approach: memristive nanodevice platform. *physica status solidi (c)*, 12(1-2):222–228, January 2015.

[19] Garrett S. Rose, Jeyavijayan Rajendran, Harika Manem, Ramesh Karri, and Robinson E. Pino. Leveraging memristive systems in the construction of digital logic circuits. *Proceedings of the IEEE*, 100(6):2033–2049, 2012.

[20] E. Lehtonen, J. Poikonen, and M. Laiho. Implication logic synthesis methods for memristors. pages 2441–2444, May 2012.

[21] I. Vourkas and G.C. Sirakoulis. A novel design and modeling paradigm for memristor-based crossbar circuits. *Nanotechnology, IEEE Transactions on*, 11(6):1151–1159, 2012.

[22] Philip J. Kuekes, Duncan R. Stewart, and Stanley R. Williams. The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits. *Journal of Applied Physics*, 97(3), 2005.

[23] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, August 2014.

[24] B. Rajendran and F. Alibart. Neuromorphic Computing Based on Emerging Memory Technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 6(2):198–211, June 2016.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[26] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[27] Richard Kempter, Wulfram Gerstner, and J. Leo van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498–4514, April 1999.

[28] S. Song, K. D. Miller, and L. F. Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, September 2000.

[29] Dezhe Z. Jin and H. Sebastian Seung. Fast computation with spikes in a recurrent neural network. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 65(5 Pt 1):051922, May 2002.

[30] M. Oster and Shih-Chii Liu. A winner-take-all spiking network with spiking inputs. In *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004*, pages 203–206, December 2004.

[31] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[32] Olivier Bichler, Damien Querlioz, Simon J. Thorpe, Jean-Philippe Bourgoin, and Christian Gamrat. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32:339–348, August 2012.

[33] Mahyar Shahsavari, Pierre Falez, and Pierre Boulet. Combining a Volatile and Nonvolatile Memristor in Artificial Synapse to Improve Learning in Spiking Neural Networks. In *12th ACM/IEEE International Symposium on Nanoscale Architectures (Nanoarch 2016)*, Beijing, China, July 2016.

[34] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The SpiNNaker Project. *Proceedings of the IEEE*, 102(5):652–665, May 2014.

[35] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen. Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations. *Proceedings of the IEEE*, 102(5):699–716, May 2014.

[36] Johannes Schemmel, Daniel Briiderle, Andreas Griibl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. pages 1947–1950. IEEE, May 2010.

[37] O. Bichler, D. Roclin, C. Gamrat, and D. Querlioz. Design exploration methodology for memristor-based spiking neuromorphic architectures with the Xnet event-driven simulator. In *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 7–12, July 2013.

[38] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

[39] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[40] Hai (Helen) Li and Miao Hu. Compact model of memristors and its application in computing systems. pages 673–678, March 2010.

[41] André Chanthbouala, Vincent Garcia, Ryan O. Cherifi, Karim Bouzehouane, Stéphane Fusil, Xavier Moya, Stéphane Xavier, Hiroyuki Yamada, Cyrile Deranlot, Neil D. Mathur, Manuel Bibes, Agnès Barthélémy, and Julie Grollier. A ferroelectric memristor, June 2012.

[42] T. Berzina, S. Erokhina, P. Camorani, O. Konovalov, V. Erokhin, and M.P. Fontana. Electrochemical control of the conductivity in an organic memristor: A time-resolved x-ray fluorescence study of ionic drift as a function of the applied voltage. *ACS Applied Materials and Interfaces*, 1(10):2115–2118, 2009.

[43] Fabien Alibart, Stéphane Pleutin, David Guérin, Christophe Novembre, Stéphane Lenfant, Kamal Lmimouni, Christian Gamrat, and Dominique Vuillaume. An Organic Nanoparticle Transistor Behaving as a Biological Spiking Synapse. *Advanced Functional Materials*, 20(2):330–337, January 2010.

[44] Z. Biolek, D. Biolek, and V. Biolková. Spice model of memristor with nonlinear dopant drift. *Radioengineering*, 18(2):210–214, 2009.

[45] Sung H. Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.*, 10(4):1297–1301, March 2010.

[46] H. Yan, H.S. Choe, S. Nam, Y. Hu, S. Das, J.F. Klemic, J.C. Ellenbogen, and C.M. Lieber. Programmable nanowire circuits for nanoprocessors. *Nature*, 470(7333):240–244, 2011.

[47] Giacomo Indiveri, Bernabe Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38):384010, September 2013. arXiv: 1302.7007.

[48] R. Williams. How we found the missing memristor. *IEEE Spectr.*, 45(12):28–35, December 2008.

[49] L.O. Chua and Sung Mo Kang. Memristive devices and systems. *Proceedings of the IEEE*, 64(2):209–223, Feb. 1976.

[50] Wan Gee Kim, Min Gyu Sung, Sook Joo Kim, Jong Hee Yoo, Te One Youn, Jang Won Oh, Jung Nam Kim, Byung Gu Gyun, Taeh Wan Kim, Chi Ho Kim, Jun Young Byun, Won Kim, Moon Sig Joo, Jae Sung Roh, and Sung Ki Park. Dependence of the switching characteristics of resistance random access memory on the type of transition metal oxide; tio2, zro2, and hfo2. *Journal of The Electrochemical Society*, 158(4):H417–H422, 2011.

[51] Rainer Waser, Regina Dittmann, Georgi Staikov, and Kristof Szot. Redox-Based Resistive Switching Memories-Nanoionic Mechanisms, Prospects, and Challenges. *Advanced Materials*, 21(25-26):2632–2663, July 2009.

[52] A.C. Torrezan, J.P. Strachan, G. Medeiros-Ribeiro, and R.S. Williams. Sub-nanosecond switching of a tantalum oxide memristor. *Nanotechnology*, 22(48):485203, 2011.

[53] S.H. Jo, K.-H. Kim, T. Chang, S. Gaba, and W. Lu. Si memristive devices applied to memory and neuromorphic circuits. pages 13–16, 2010.

[54] Xiaobin Wang, Yiran Chen, Haiwen Xi, Hai Li, and D. Dimitrov. Spintronic Memristor Through Spin-Torque-Induced Magnetization Motion. *Electron Device Letters, IEEE*, 30(3):294–297, March 2009.

[55] Xiaobin Wang and Yiran Chen. Spintronic memristor devices and application. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 667–672, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association.

[56] Victor Erokhin, Tatiana Berzina, and Marco P. Fontana. Hybrid electronic device based on polyaniline-polyethyleneoxide junction. *Journal of Applied Physics*, 97(6):064501, 2005.

[57] F. Pincella, P. Camorani, and V. Erokhin. Electrical properties of an organic memristive system. *Applied Physics A: Materials Science & Processing*, 104:1039–1046, September 2011.

[58] Victor Erokhin and Marco P. Fontana. Organic memristive device and its application for the information processing. In *ICECS*, pages 926–929. IEEE, 2010.

[59] L. Wang and S. Gai. The next generation mass storage devices - Physical principles and current status. *Contemporary Physics*, 55(2):75–93, April 2014.

[60] S.-M. Yoon, S. Yang, S.-W. Jung, C.-W. Byun, M.-K. Ryu, W.-S. Cheong, B.H. Kim, H.C. Oh, S.-H.K. Park, C.-S. Hwang, S.-Y. Kang, H.-J. Ryu, and B.-G. Yu. Polymeric ferroelectric and oxide semiconductor-based fully transparent memristor cell. *Applied Physics A: Materials Science and Processing*, 102(4):983–990, 2011.

[61] Lei Wang, CiHui Yang, Jing Wen, Shan Gai, and YuanXiu Peng. Overview of emerging memristor families from resistive memristor to spintronic memristor. *Journal of Materials Science: Materials in Electronics*, 26(7):4618–4628, March 2015.

[62] Xiaofang Hu, Gang Feng, Shukai Duan, and Lu Liu. Multilayer RTD-memristor-based cellular neural networks for color image processing. *Neurocomputing*, 162:150–162, August 2015.

[63] S. Hamdioui, L. Xie, H.A. Du Nguyen, M. Taouil, and K.L.M. Bertels. Memristor based computation-in-memory architecture for data-intensive applications. In *Proc. 18th Design, Automation & Test in Europe conference*, Grenoble, France, March 2015.

[64] Cong Xu, Xiangyu Dong, Norman P. Jouppi, and Yuan Xie. Design implications of memristor-based rram cross-point structures. In *DATE*, pages 734–739. IEEE, 2011.

[65] T. Driscoll, H.-T. Kim, B.-G. Chae, M. Di Ventra, and D. N. Basov. Phase-transition driven memristive system. *Applied Physics Letters*, 95(4):043503, July 2009.

[66] Qi Liu, Shibing Long, Hangbing Lv, Wei Wang, Jiebin Niu, Zongliang Huo, Junning Chen, and Ming Liu. Controllable Growth of Nanoscale Conductive Filaments in Solid-Electrolyte-Based ReRAM by Using a Metal Nanocrystal Covered Bottom Electrode. *ACS Nano*, 4(10):6162–6168, October 2010.

[67] Hai Li and Yiran Chen. An overview of non-volatile memory technology and the implication for tools and architectures. In *DATE*, pages 731–736, 2009.

[68] Qiangfei Xia, Warren Robinett, Micheal W. Cumbie, Neel Banarjee, Thomas J. Cardinali, Joshua J. Yang, Wei Wu, Xuema Li, William M. Tong, Dmitri B. Strukov, Gregosry S. Snider, Gilberto Medeiros=Ribeiro, and R. Stanley Williams. Memristor-CMOS Hybrid Integrated circuits for reconfigurable logic. *Nanoletters*, 2009.

[69] Jason Cong and Bingjun Xiao. mrfpga: A novel fpga architecture with memristor-based reconfiguration. In *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures*, NANOARCH '11, pages 1–8, Washington, DC, USA, 2011. IEEE Computer Society.

[70] Yogesh N. Joglekar and Stephen J. Wolf. The elusive memristor: properties of basic electrical circuits.

[71] T.A. Wey and W.D. Jemison. Variable gain amplifier circuit using titanium dioxide memristors. *IET Circuits, Devices and Systems*, 5(1):59–65, 2011.

[72] M. Itoh and L.O. Chua. Memristor oscillators. *International Journal of Bifurcation and Chaos*, 18(11):3183–3206, 2008.

[73] S. Shin, K. Kim, and S.-M. Kang. Memristor applications for programmable analog ics. *IEEE Transactions on Nanotechnology*, 10(2):266–274, 2011.

[74] B. Muthuswamy. Implementing memristor based chaotic circuits. *International Journal of Bifurcation and Chaos*, 20(5):1335–1350, 2010.

[75] A. Afifi, A. Ayatollahi, and F. Raissi. Implementation of biologically plausible spiking neural network models on the memristor crossbar-based cmos/nano circuits. pages 563–566, 2009.

[76] X. Hu, S. Duan, L. Wang, and X. Liao. Memristive crossbar array with applications in image processing. *Science China Information Sciences*, 55(2):461–472, 2012.

[77] Y.V. Pershin and M. Di Ventra. Neuromorphic, digital, and quantum computation with memory circuit elements. *Proceedings of the IEEE*, 100(6):2071 –2080, june 2012.

[78] Farnood Merrikh-Bayat and Saeed Bagheri Shouraki. Memristor-based circuits for performing basic arithmetic operations. *Procedia CS*, 3:128–132, 2011.

[79] F. Merrikh-Bayat and S. Bagheri Shouraki. Memristive neuro-fuzzy system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2012.

[80] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. 1927. 2nd Edition).

[81] Tezaswi Raja and Samiha Mourad. Digital logic implementation in memristor-based crossbars - a tutorial. *Electronic Design, Test and Applications, IEEE International Workshop on*, 0:303–309, 2010.

[82] Julien Borghetti, Zhiyong Li, Joseph Straznicky, Xuema Li, Douglas A. A. Ohlberg, Wei Wu, and Duncan R. Stewart. A hybrid nanomemristor/transistor logic circuit capable of self-programming. *PNAS*, 106, 2009.

[83] E. Lehtonen and M. Laiho. Stateful implication logic with memristors. In *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 33–36, July 2009.

[84] L. Xie, H. A. D. Nguyen, M. Taouil, S. Hamdioui, and K. Bertels. Boolean logic gate exploration for memristor crossbar. In *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6, April 2016.

[85] F. S. Marranghello, V. Callegaro, M. G. A. Martins, A. I. Reis, and R. P. Ribas. Factored Forms for Memristive Material Implication Stateful Logic. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):267–278, June 2015.

[86] G. Snider. Computing with hysteretic resistor crossbars. *Applied Physics A*, 80:1165–1172, 2005.

[87] I. Vourkas, D. Stathis, G. C. Sirakoulis, and S. Hamdioui. Alternative Architectures Toward Reliable Memristive Crossbar Memories. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(1):206–217, January 2016.

[88] L Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *J Physiol Pathol Gen*, 9:620–635, 1907.

[89] C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, October 1990.

[90] M. A. Mahowald and C. Mead. The silicon retina. *Scientific American*, 264(5):76–82, May 1991.

[91] T. Delbruck and C. A. Mead. Adaptive photoreceptor with wide dynamic range. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, volume 4, pages 339–342 vol.4, May 1994.

[92] Rahul Sarpeshkar, Richard F. Lyon, and Carver Mead. A Low-Power Wide-Dynamic-Range Analog VLSI Cochlea. In Tor Sverre Lande, editor, *Neuromorphic Systems Engineering : Neural Networks in Silicon*, pages 49–103. Kluwer Academic, Boston, MA, 1998.

[93] Shih-Wen Chiu and Kea-Tiong Tang. Towards a Chemiresistive Sensor-Integrated Electronic Nose: A Review. *Sensors*, 13(10):14214–14247, October 2013.

[94] S. C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck. Asynchronous Binaural Spatial Audition Sensor With 2 64 4 Channel Output. *IEEE Transactions on Biomedical Circuits and Systems*, 8(4):453–464, August 2014.

[95] Anup Vanarse, Adam Osseiran, and Alexander Rassau. A Review of Current Neuromorphic Approaches for Vision, Auditory, and Olfactory Sensors. *Frontiers in Neuroscience*, 10, March 2016.

[96] L. Shi, J. Pei, N. Deng, D. Wang, L. Deng, Y. Wang, Y. Zhang, F. Chen, M. Zhao, S. Song, F. Zeng, G. Li, H. Li, and C. Ma. Development of a neuromorphic computing system. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 4.3.1–4.3.4, December 2015.

[97] Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Neuromorphic Engineering*, 7:178, 2013.

[98] Mirko Prezioso, Farnood Merrikh-Bayat, Brian Hoskins, Gina Adam, Konstantin K. Likharev, and Dmitri B. Strukov. Training and Operation of an Integrated Neuromorphic Network Based on Metal-Oxide Memristors. *Nature*, 521(7550):61–64, May 2015. arXiv: 1412.0611.

[99] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943.

[100] Rufin VanRullen, Rudy Guyonneau, and Simon J. Thorpe. Spike times make sense. *Trends in Neurosciences*, 28(1):1–4, January 2005.

[101] Romain Brette. Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain. *Frontiers in Systems Neuroscience*, page 151, 2015.

[102] Sander M. Bohte. The evidence for neural information processing with precise spike-times: A survey. *Natural Computing*, 3(2):195–206, June 2004.

[103] Filip Ponulak and Andrzej Kasiński. Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Computation*, 22(2):467–510, February 2010.

[104] T. Masquelier, E. Hugues, G. Deco, and S.J. Thorpe. Oscillations, phase-of-firing coding, and spike timing-dependent plasticity: An efficient learning scheme. *Journal of Neuroscience*, 29(43):13484–13493, 2009.

[105] H. T. Chen, K. T. Ng, A. Bermak, M. K. Law, and D. Martinez. Spike Latency Coding in Biologically Inspired Microelectronic Nose. *IEEE Transactions on Biomedical Circuits and Systems*, 5(2):160–168, April 2011.

[106] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6-7):715–725, July 2001.

[107] Maoz Shamir. Emerging principles of population coding: in search for the neural code. *Current Opinion in Neurobiology*, 25:140–148, April 2014.

[108] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, June 1996.

[109] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[110] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv:1506.00019 [cs]*, May 2015. arXiv: 1506.00019.

[111] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, April 1982.

[112] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, March 1990.

[113] A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. volume 2, pages 1097–1105, 2012.

[114] D. Neil and S. C. Liu. Minitaur, an Event-Driven FPGA-Based Spiking Network Accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(12):2621–2628, December 2014.

[115] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, August 1952.

[116] E. Chicca, D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. Del Giudice. A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Transactions on Neural Networks*, 14(5):1297–1307, September 2003.

[117] Shih-Chii Liu and R. Douglas. Temporal coding in a silicon network of integrate-and-fire neurons. *IEEE Transactions on Neural Networks*, 15(5):1305–1314, September 2004.

[118] Wolfgang Maass and Christopher M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, Cambridge, MA, USA, 1999.

[119] Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James M. Bower, Markus Diesmann, Abigail Morrison, Philip H. Goodman, Frederick C. Harris, Milind Zirpe, Thomas Natschläger, Dejan Pecevski, Bard Ermentrout, Mikael Djurfeldt, Anders Lansner, Olivier Rochel, Thierry Vieville, Eilif Muller, Andrew P. Davison, Sami El Boustani, and Alain Destexhe. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, December 2007.

[120] Edward H. Hu and Stewart A. Bloomfield. Gap junctional coupling underlies the short-latency spike synchrony of retinal alpha ganglion cells. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 23(17):6768–6777, July 2003.

[121] David A. Drachman. Do we have brain to spare? *Neurology*, 64(12):2004–2005, June 2005.

[122] R. G. Morris. D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain Research Bulletin*, 50(5-6):437, December 1999.

[123] D.O. Hebb. The organization of behavior: A neuropsychological theory. D. O. Hebb. John. *Wiley, New York*, (3), 1949.

[124] Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478, June 2008.

[125] Peter Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005.

[126] Pierre Yger and Matthieu Gilson. Models of Metaplasticity: A Review of Concepts. *Frontiers in Computational Neuroscience*, page 138, 2015.

[127] Li I. Zhang, Huizhong W. Tao, Christine E. Holt, William A. Harris, and Mu-ming Poo. A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, 395(6697):37–44, September 1998.

[128] Huai-Xing Wang, Richard C. Gerkin, David W. Nauen, and Guo-Qiang Bi. Coactivation and timing-dependent integration of synaptic potentiation and depression. *Nature Neuroscience*, 8(2):187–193, February 2005.

[129] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity. *The Journal of Neuroscience*, 26(38):9673–9682, September 2006.

[130] E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 2(1):32–48, January 1982.

[131] Robert C. Froemke and Yang Dan. Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature*, 416(6879):433–438, March 2002.

[132] Per Jesper Sjöström, Gina G. Turrigiano, and Sacha B. Nelson. Endocannabinoid-dependent neocortical layer-5 LTD in the absence of postsynaptic spiking. *Journal of Neurophysiology*, 92(6):3338–3343, December 2004.

[133] W. Senn, H. Markram, and M. Tsodyks. An algorithm for modifying neurotransmitter release probability based on pre- and postsynaptic spike timing. *Neural Computation*, 13(1):35–67, January 2001.

[134] Y. Frégnac and D. E. Shulz. Activity-dependent regulation of receptive field properties of cat area 17 by supervised Hebbian learning. *Journal of Neurobiology*, 41(1):69–82, October 1999.

[135] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation*, 18(6):1318–1348, June 2006.

[136] K.A. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):416–434, May 2000.

[137] Karsten Wendt, Matthias Ehrlich, and René Schüffny. A Graph Theoretical Approach for a Multistep Mapping Software for the FACETS Project. In *Proceedings of the 2Nd WSEAS International Conference on Computer Engineering and Applications*, CEA'08, pages 189–194, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[138] J.J. Yang, M.D. Pickett, X. Li, D.A.A. Ohlberg, D.R. Stewart, and R.S. Williams. Memristive switching mechanism for metal/oxide/metal nanodevices. *Nature Nanotechnology*, 3(7):429–433, 2008.

[139] Eve Marder and Jean-Marc Goaillard. Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, 7(7):563–574, July 2006.

[140] C. Li and Y. Li. A Review on Synergistic Learning. *IEEE Access*, 4:119–134, 2016.

[141] S. Grossberg. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23(3), 1976.

[142] Wolfgang Maass. On the Computational Power of Winner-Take-All. *Neural Computation*, 12(11):2519–2535, November 2000.

[143] P. Hafliger. Adaptive WTA With an Analog VLSI Neuromorphic Learning Chip. *IEEE Transactions on Neural Networks*, 18(2):551–572, March 2007.

[144] Henry Markram. The human brain project. *Scientific American*, 306(6):50–55, June 2012.

[145] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha. The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, November 2009.

[146] Hélène Paugam-Moisy and Sander Bohte. Computing with Spiking Neuron Networks. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 335–376. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-540-92910-9_10.

[147] Dan F. M. Goodman, Romain Brette, Dan Goodman, and Romain Brette. Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*, 2:5, 2008.

[148] Mahyar Shahsavari, Philippe Devienne, and Pierre Boulet. N2s3, a Simulator for the Architecture Exploration of Neuromorphic Accelerators. In *2nd International Workshop on Neuromorphic and Brain-Based Computing Systems (NeuComp 2015) in DATE Conference, Grenoble, France*, 2015.

[149] Derek Wyatt. *Akka Concurrency*. Artima Incorporation, USA, 2013.

[150] Martin Odersky, Philippe Altherr, Vincent Cremet, Iulian Dragos, Gilles Dubochet, Burak Emir, Sean McDirmid, Stéphane Micheloud, Nikolay Mihaylov, Michel Schinz, Erik Stenman, Lex Spoon, and Matthias Zenger. An overview of the Scala programming language (2nd Edition). Technical Report LAMP-REPORT-2006-001, École Polytechnique Fédérale de Lausanne (EPFL), 2006.

[151] S. J. Martin, P. D. Grimwood, and R. G. Morris. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual Review of Neuroscience*, 23:649–711, 2000.

[152] H. Kim, M. P. Sah, C. Yang, T. Roska, and L. O. Chua. Memristor Bridge Synapses. *Proceedings of the IEEE*, 100(6):2061–2070, June 2012.

[153] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat. Immunity to Device Variations in a Spiking Neural Network With Memristive Nanodevices. *IEEE Transactions on Nanotechnology*, 12(3):288–295, May 2013.

[154] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat. Learning with memristive devices: How should we model their behavior? In *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 150–156, June 2011.

[155] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, April 2007.

[156] D. Querlioz, W. S. Zhao, P. Dollfus, J.-O. Klein, O. Bichler, and C. Gamrat. Bioinspired Networks with Nanoscale Memristive Devices that Combine the Unsupervised and Supervised Learning Approaches. pages 203–210, 2012.

[157] Giacomo Indiveri, Bernabe Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38):384010, September 2013. arXiv: 1302.7007.

[158] Fabien Alibart, Stéphane Pleutin, Olivier Bichler, Christian Gamrat, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Dominique Vuillaume. A Memristive Nanoparticle/Organic Hybrid Synapstor for Neuroinspired Computing. *Advanced Functional Materials*, 22(3):609–616, February 2012.

[159] Simon Desbief, Adrica Kyndiah, David Guérin, Denis Gentili, Mauro Murgia, Stéphane Lenfant, Fabien Alibart, Tobias Cramer, Fabio Biscarini, and Dominique Vuillaume. Low voltage and time constant organic synapse-transistor. *Organic Electronics*, 21:47–53, June 2015.

[160] G. Indiveri. Neuromorphic bistable VLSI synapses with spike-timing-dependent plasticity. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 1091–1098, Cambridge, MA, USA, December 2003. MIT Press.

[161] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[162] Teresa Serrano-Gotarredona, Timothée Masquelier, Themistoklis Prodromakis, Giacomo Indiveri, and Bernabe Linares-Barranco. STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Frontiers in Neuroscience*, 7:2, 2013.

[163] Bernhard Nessler, Michael Pfeiffer, and Wolfgang Maass. STDP enables spiking neurons to detect hidden causes of their inputs. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1357–1365. Curran Associates, Inc., 2009.

[164] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[165] Peter U. Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, page 99, 2015.

[166] Sukru Burc Eryilmaz, Siddharth Joshi, Emre Neftci, Weier Wan, Gert Cauwenberghs, and H.-S. Philip Wong. Neuromorphic architectures with electronic synapses. pages 118–123. IEEE, March 2016.

[167] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[168] Sukru B. Eryilmaz, Duygu Kuzum, Rakesh Jeyasingh, SangBum Kim, Matthew BrightSky, Chung Lam, and H.-S. Philip Wong. Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Neuromorphic Engineering*, 8:205, 2014.

[169] Stefano Ambrogio, Nicola Ciocchini, Mario Laudato, Valerio Milo, Agostino Pirovano, Paolo Fantini, and Daniele Ielmini. Unsupervised Learning by Spike Timing Dependent Plasticity in Phase Change Memory (PCM) Synapses. *Neuromorphic Engineering*, page 56, 2016.

[170] Manan Suri, Damien Querlioz, Olivier Bichler, Giorgio Palma, Elisa Vianello, Dominique Vuillaume, Christian Gamrat, and Barbara DeSalvo. Bio-Inspired Stochastic Computing Using Binary CBRAM Synapses. *IEEE Transactions on Electron Devices*, 60(7):2402–2409, July 2013.

[171] Y. Nishitani, Y. Kaneko, and M. Ueda. Artificial synapses using ferroelectric memristors embedded with CMOS Circuit for image recognition. In *72nd Device Research Conference*, pages 297–298, June 2014.

[172] Wulfram Gerstner, Raphael Ritz, and J. Leo Hemmen. Why spikes? hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, 69(5-6):503–515, 1993.

[173] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215, 1997.

[174] Mazdak Fatahi, Mahmood Ahmadi, Mahyar Shahsavari, Arash Ahmadi, and Philippe Devienne. evt_mnist: A spike based version of traditional MNIST. *arXiv:1604.06751 [cs]*, April 2016. arXiv: 1604.06751.

[175] Patrick J. Drew and L. F. Abbott. Extending the effects of spike-timing-dependent plasticity to behavioral timescales. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8876–8881, June 2006.

[176] D. Querlioz, O. Bichler, and C. Gamrat. Simulation of a memristor-based spiking neural network immune to device variations. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 1775–1781, July 2011.

[177] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Neuromorphic Engineering*, 7:272, 2014.

[178] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky. Memristor-Based Multilayer Neural Networks With Online Gradient Descent Training. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2408–2421, October 2015.

[179] Arnold J. F. Siegert. On the First Passage Time Probability Problem. *Physical Review*, 81(4):617–623, February 1951.

[180] R. Kempter, W. Gerstner, and J. L. van Hemmen. Intrinsic stabilization of output rates by spike-based Hebbian learning. *Neural Computation*, 13(12):2709–2741, December 2001.

[181] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[182] Max Welling and Geoffrey E Hinton. A new learning algorithm for mean field boltzmann machines. In *Artificial Neural Networks—ICANN 2002*, pages 351–357. Springer, 2002.

[183] Wojciech K Mleczko, Tomasz Kapuściński, and Robert K Nowicki. Rough deep belief network-application to incomplete handwritten digits pattern classification. In *Information and Software Technologies*, pages 400–411. Springer, 2015.

[184] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2006.

[185] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. Learning a spatially smooth subspace for face recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Machine Learning (CVPR'07)*, 2007.

[186] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, volume 7700, pages 599–619. Springer, 2012.

[187] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[188] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM, 2008.