Master Thesis

In-memory Reservoir Computing

Exploiting memristive devices in Spiking Neural Networks

Filippo Moro

With the supervision of Prof. G.Indiveri and Dr. M.Payvand
Prof. C.Ricciardi as Relator

Master Degree in Nanotechnologies for ICTs'

Academic year 2018/2019

Abstract

Neuromorphic computing has often lent on analysis of biological systems to improve its performances. One of the key properties of the Nervous System is Plasticity, i.e. the capacity of its components (Neuron and Synapses) to modify themselves, functionally and structurally, in response to experience and injury. In the brain, plasticity is mainly attributed to Synapses, which control the flow of ions between Neurons through neurotransmitters with a certain, variable in time, weight. Neurons also play a role in plasticity mechanisms, since their excitability can be varied as well as the leakages of internalized ions to keep a healthy firing regime, minimizing energy consumption and maximizing information transfer. These plasticity mechanisms applied to Neural Networks not only increase the plausibility to biology but also contribute to improving the performances of the system in several tasks. This is particularly true for Liquid State Machines, a computing paradigm based on Spiking-Neural-Networks, in the framework of Reservoir Computing. Different forms of plasticity are present in the Brain, and in turn also in Brain-inspired Neural Networks: the most popular one is Spiking-Time-Dependent-Plasticity (STDP), a form of synaptic Hebbian plasticity; Synaptic-Normalization (SN) is also a common homeostatic feature; Intrinsic-Plasticity (IP) is instead a less investigated property for Neuromorphic systems, probably due to the difficulty of implementing it in hardware devices. The co-action of such mechanisms has been shown to boost the performance in the framework of Reservoir Computing for Artificial-Neural-Networks (SORN, Lazar 2009), while it remains to be investigated for more biologically plausible Spiking-Neural-Networks (Liquid State Machines).
From the hardware standpoint, conventional CMOS based Neuromorphic hardware struggles to implement such plasticity mechanisms, particularly Intrinsic Plasticity: to update the parameters of the network, external memory is required to be operated, transferring the information of such modified biases back and forth from the registers to the computational units. This, of course, leads to the well known Von Neumann bottleneck problem, for which the transfer of the information across the device limits the device speed itself. The rise of Memristor opens the doors for new architectures that are able to store memory within the Neuron's scheme, creating advanced circuits embedding both the Neuron's circuit and its biases. This scheme enables the parameters of the Neuron to be modified locally, learning is enabled in situ. Exploiting the programmability of memristors, the resistances of the Neuron can be set to a target value and automatically updated every cycle without the need of storing their state in external memories and allowing for a real Non-Von Neumann architecture to be conceived. The work aims to show that technologically plausible In-Memory Mixed Signals architectures allow for the development of algorithms, implementing plasticity mechanisms, able to improve the performance of Liquid-State-Machines in temporal tasks.

Acknowledgment

This work is dedicated to the people who supported me in my way toward my graduation. My colleagues in the Master program, companions of the difficulties of the studies, thanks to which coming every day at lesson was not a burden. My mates at the Institute in Zurich, which treated me as a member of their family. My professors, who taught me what I am so proud to have learned. My supervisors, which guided me patiently and efficiently and treated me with immense respect.
The main part of my gratitude is however for my Friends and Family. The firsts were always by my side and shared so many good moments I will never forget. My Sister, which is a reference point in terms of strength. My Parents, the nicest and most supportive people I know.
At last, let me also dedicate this work to me and to my future career. In the hope that this is just the first of many achievements.

Contents

Chapter 1

Reservoir Computing

Reservoir Computing is a term introduced in 2007 [42] to indicate a computing paradigm belonging to Recurrent Neural Networks, which was developed sometime before. As a matter of fact, in the early 2000s, Jaeger [43] [44] and Maas [47] almost contemporarily defined two Network architectures, very similar one to the other, based on a recurrent pull of nodes and a linear readout from that bunch.
Later on, Steil [6] performed theoretical studies on the properties of Reservoir Computing architectures and found out about their impressive computational power, in spite of the simplicity of their operation. He also founded a learning procedure which would be classified belonging to the Reservoir Computing paradigm.
Given the promises of high computational power and low cost of training, RC developed to embrace complex architectures, such as Deep Reservoir Networks [12], to be customized for peculiar applications. State of the art RC Networks are employed for different temporal tasks, exploiting its intrinsic short-term-memory: prediction in time-varying signals, motor control in robotics, sound and speech recognition, biomedical signal processing (ECG, EMG) and many others.

1.1 Neural Networks

To give the description of Reservoir Computing a context, the concept of Neural Network [8] is introduced. Formally, it deals with a bunch of nodes, eventually called Neurons in analogy to the brain, linked with each other by connections or synapses, to recall the brain physiology. The nodes of a Network posses an activation function: they receive signals as inputs and produce an output according to a certain rule. The input signals are modulated by the so-called weight of the connection. As a consequence, an input being fed to a Neural Network spreads through it according to the behavior of the nodes and the strength and topology of the connections. The latter are adapted in order to make the Network able to perform a task: the process of alteration of the internal variables of a Network for a certain aim is said *Learning*.

Network Architectures

The pattern of connectivity between the nodes defines the structure of a Network and it is crucial in determining its computational properties. There are basically two ways to connect nodes in a Network: Feed Forward and Recurrent. Based on this two concepts, many types of different architectures can be defined.

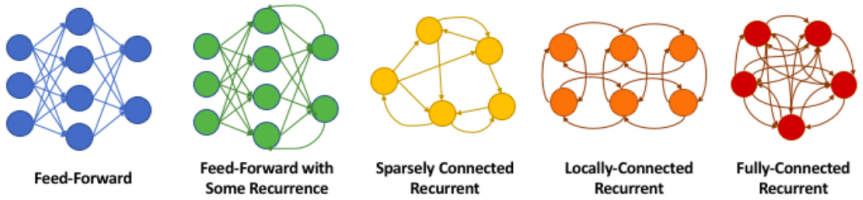


Figure 1.1: Possible types of Network topologies for NC. While the main categories are Feed-Forward, Recurrent and Fully Connected, some combination of those can be exploited for obtaining hybrid networks.

Among the Feed Forward Networks, the simplest is the case of the *Perceptron*: a Network having a single output node and several nodes as input. Due to its simplicity, the Perceptron is only able to perform a binary classification of its inputs. As a matter of fact, the output of the *Perceptron* is given by the single output neuron, implementing it peculiar activation function. For example, if the activation function is a Heaviside function with a certain threshold, it is clear why the *Perceptron* is used for binary classification of some input data. Since the inputs are directly connected to the output node, it is said that this Network has 1 only layer.
Adding more layers and thus complexity to the architecture, the field of Deep Learning was founded. The simplest example of such Networks is the *Multi-Layer Perceptron*: with respect to the simple Perceptron, it adds hidden layer(s) of nodes between Input and Output. The MLP takes advantages of supervised training techniques and is able to learn non-linearly separable inputs for classification.
Advancement in Deep Learning lead to the definition of increasingly more complex Networks. Among the many, the most important one is certainly the Convolutional Neural Network: composed of a large number of hidden layer performing convolutional operations, it excels in the field of computer vision and natural language processing.

Artificial vs Spiking

Most of the Neural Networks are implemented with Artificial Neurons, also called Digital Neurons: the units receive input signal as fixed-precision digits and compute their output through a non-linear activation function; the output is transmitted to the linked unit at the following time step, since the Network evolution in time is discretized. Some of the most common activation functions are: *Sigmoid*, *Hyperbolic tangent*, *ReLu*. These simple non-linear transformations are essential in order to both allow the Network a rich enough non-linear representation and processing of the input and also simplify the training procedure. As a matter of fact, the main supervised learning algorithm, Back-Propagation, is based on the calculation of the derivative of the error of the output with respect to the activation of the nodes in the Network [7]. Utilizing more complex activation functions and/or architectures may lead to problems in the Back Propagation procedure, such as the vanishing gradient problem. Given the low degree of non-linearity of most common activation functions, Deep Learning architectures had to evolve to highly complex structures in order to enhance computational power of the Network while allowing efficient training.

Another approach is the one of Spiking Neural Networks: inspired on biology, the nodes and connections operate in continuous time and behave in a similar fashion with respect to real Neurons and Synapses. As a consequence, nodes integrate inputs with leakages up to the point in which they emit spikes, which propagate through the Network via the Synapses. About the behavior of Neurons and Synapses, the next chapters will provide more details.
Compared to their Artificial counterparts, the nodes in SNNs encode much less information in their

output, which is conventionally represented by a delta function $\delta(t-t_s)$, t_s time of the spike. However, since they operate in continuous time, the high information content is believed to be in the timing of the spikes of all the neurons.
The major drawback in SNNs is the Training procedure: because of the peculiar input/output relation, Back-Propagation is not viable: the delta function at the output of the neurons is non-differentiable. An equivalent SNN-friendly BP algorithm has been developed, called SpikeProp, still in experimental phase. More commonly, SNNs are trained with Unsupervised techniques or exploit peculiar architectures to perform classification tasks, such as in the case of Winner-Take-All Networks.
The advantage over ANNs is that SNNs are in general more fault and noise tolerant, because of the way the input are processed by the nodes. However, the complexity of the behavior of Neurons comes with the difficulty in their control, especially for complex Networks, ruled by a large set of hyper-parameters: inspiration from biology often helps in controlling the behavior and improving the performance of SNNs.

Learning in Neural Networks

The learning procedure in NNs is crucial in determining its capabilities and performance. There a mainly 3 policies for Training (the process of the learning of the weights) in NNs:

- **Supervised Training**: learning is guided by an external agent
- **Unsupervised Training**: no external agent guides the training procedure
- **Reinforcement Training**: the output of the Network produces a reward function, which determines the correctness of the Network and guides the training of the variables of the Network itself

Sometimes a fourth policy is defined, being a combination of Supervised and Unsupervised trainings thus called Semi-Supervised.
Supervised techniques mainly deal with minimization of the error of the Network: the output is compared to the ideal result of the task and the weights are modified in order to minimize the error of the comparison. Among such techniques, Back-Propagation (BP) is by far the most common. To give a glimpse on BP, the error is minimized by means of its derivative with respect to the weights of the Network. After all, BP deals with finding the weight configuration which minimizes the error. In general, the Error function is computed with the squared difference with respect to the target value for regression problems, with categorical cross-entropy for classification problems. BP makes use of the chain-rule in order to simplify the problem of the minimization of the error: the gradient of the Cost function is thus manipulated as such.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial in_j}\frac{in_j}{\partial w_{ij}} \tag{1.1}$$

Here, E is the Error or Cost function, o_j is the activation of the j^{th} node, in is its net input, w_{ij} the weight from node i to node j. This derivative is performed in order to decrease the error of the Network with respect to the single weight w_{ij}. If one generalizes this concept to all the weights in the Network, one would compute the gradient of the Error function: this procedure is called *Gradient Descend*. Once the effect of the change of a weight has on the Cost function is computed, one has to modify that same weight in order to reduce the Error. This very procedure is called the *Back-Propagation* of the Error and it is computed, after some algebraic steps, in this way:

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} \tag{1.2}$$

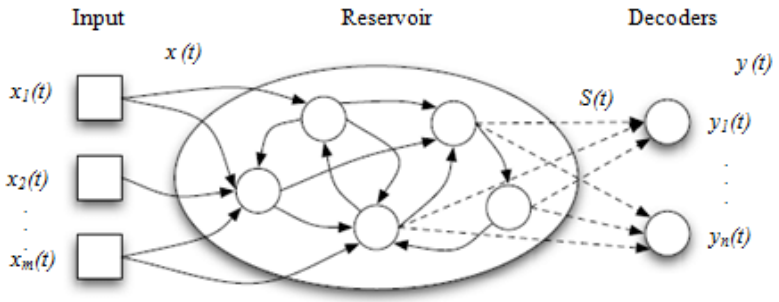


Figure 1.2: Architecture of Reservoir Computing. The 3 main elements are: Input, Reservoir and Output. Recurrent connections are present in the Reservoir only. The state of the Reservoir is linearly projected on to the Output layer.

η being the Learning Rate of the BP algorithm, controlling the how fast the gradient is reduced at each step. In general, high Learning Rates lead to faster and less accurate learning.

Unsupervised techniques often lean on powerful Clustering algorithms in order to perform classification from the behavior of the Network, or employ adaptation rules based on the dynamics of the Network itself. Needless to say, such techniques are in general less accurate that supervised training rules.

In Reinforcement Learning, instead, the focus is on the *Reward function*: the Network is seen as an agent whose actions produce an output which is evaluated by a Reward Function; the Network modifies its parameters in order to maximize the reward. If the Reward function is properly generated for the task, the agent then greatly and efficiently improves the performance.

1.2 Reservoir Computing architecture

Reservoir Computing's architecture lies in the set of Recurrent Neural Networks, but its set-up is one of the simplest both by construction and by Training procedure. A RC Network is composed of 3 elements: an Input layer, a Reservoir and an Output layer. Recurrent connections are present in the Reservoir only; the Output layer is generally connected to the Reservoir in a all-to-all fashion (every node in the Reservoir is connected to all the nodes in the Output). Among all the weights from the connections present in the Network, only the ones related to the output layer are trained in a Supervised manner. In some cases, the connections in the Reservoir are trained with unsupervised policies, but in general there are no restrictions in the connectivity of the Reservoir. As a matter of fact, these weights are often generated in a random way.

1.3 Training in RC

One may wonder why only the output connections are trained in Reservoir Computing. The answer is simple: in Recurrent Networks, performing Back-Propagation is very challenging and often suffers from convergence problem. It is hard to lower the Error through its gradient with respect to the weights when the Network is highly recurrent. The solution employed in RC is to not train the connections being involved in Recurrent microcircuits, but only the ones related to the Output. The result is that the output weights simply perform a linear projection of the state of activation of the

Reservoir on the Output layer. The role of the non-linear processing of the input is left exclusively to the Reservoir, which executes that thanks to the recurrent connections. Instead, the Output, also called Readout, simply has to linearly extract informations from the high dimensional representation of the input given by the Reservoir.
Different options are available for training the Output connections, both in *Batch-mode* and in *On-line-mode*: the first one performs the training after a batch of a dataset has been presented, the second updates the weights as soon as a new input is presented.
Among the most common and simple Batch-mode procedures for training a linear Readout are: *Linear Regression*, *Logistic Regression* and *Ridge Regression*. Instead, the main on-line training algorithm is the *p-delta rule* [10].
For what concerns the Reservoir, its weights are not modified while the training of the Output is performed, but a Pre-Training procedure may be carried out. This is done in order to adapt the dynamical representation of the Reservoir to the specific kind of input chosen for the task and in turn improve the performance of the RC machine.

1.4 Different RC implementations

While a RC, by definition, imposes constraints on the architecture of the Network, it can be realized in different manners, starting from the implementation of the nodes. Based on either ANNs or SNNs respectively, the two main versions of RC are the Echo-State-Machine and the Liquid-State-Machine. Both share the same architecture and the training procedure, but differ in the motivation behind their foundation: while ESN was developed mainly to overcome the difficulty of training in Recurrent Networks, LSM draws inspiration from biology.
As a consequence, Echo State Machines are implemented with Digital neurons and their temporal evolution is determined by discrete time-steps. Instead, Liquid State Machines evolve in continuous time and their units are implemented by biologically inspired Neuron model (mainly Integrate and Fire; see later for the description of the main Neuron models).

Echo State Networks

Echo State Machine, as reported in [43], was founded in response to the difficulty in training of Recurrent Networks. As a matter of fact, conventional gradient-descent-based approaches suffer from 4 main problems:

- Convergence is not guaranteed, due to the bifurcation phenomenon
- A single parameter update may be computationally expensive
- It is hard to train on task requiring long term memory
- In general, training algorithms require high level of expertise

ESN tackles this points with an easy architecture which is trained with simple Regression algorithms, yet is able to achieve high performance in conventional temporal tasks. The condition assuring high performance is the *Echo State* property, from which the name of the architecture: it states that the effect of a previous state $x(n)$ and a previous input $u(n)$ on a future state $x(n+k)$ should vanish gradually as time passes (i.e., $k \to \infty$), and not persist or even get amplified. Two other main properties of the ESN are the *Separation* and *Approximation* properties. The first one is related to the Reservoir capability of differentiating its internal state with respect to different inputs; the second one deals

with ability of the linear readout to extract information from the Reservoir, which is generally a noisy high dimensional representation of the input. These properties are to be evaluated when building up a RC Network: even though ESM are thought to function with no a priori constraints on the weights in the Reservoir, some measures in the formation of the Reservoir are crucial in order to assure that the Network expresses the two above mentioned properties and, in turn, is able to reach high performance for temporal tasks. For ESM, consideration on the internal dynamics of the Reservoir lead to the definition of certain rules which enhance the performance. One above all, the scaling of the Spectral Radius [42]: if the Matrix of the weights has a spectral radius lower and close to one, the Reservoir will be exhibit rich dynamics without being chaotic.
From a formal point of view, an ESN can be described by these equations:

$$y(n) = f_{out}\left[\mathbf{W_{out}}\left(u(n), x(n)\right)\right] \tag{1.3}$$

where $u(n)$ is the Input, $x(n)$ the Reservoir state, f_{out} the function implemented by the linear readout, W_{out} the output weights matrix. The state of the Reservoir evolves according to:

$$x(n+1) = \mathbf{W_{in}} \cdot u(n) + \mathbf{W_R} \cdot x(n) \tag{1.4}$$

in which W_{in} is the Input weights matrix, W_R the recurrent connections matrix.

Liquid State Machines

Defined by Maass in 2002 [47], LSM is a Spiking Neural Network based architecture which draws inspiration on biology.
LSM also has its own dedicated terminology, based on an analogy conceived by Maass: the Reservoir is called Liquid, which is perturbed by its input as the surface of the water in a lake is oscillating under the action of a stone thrown in it. The Readout is asked to get informations from the transitory perturbed states of Liquid, rather than on its equilibrium forms. As a consequence, the Liquid has to be designed in order to evolve in time and not easily fall into attractor states.
The details on the dynamics and a characterization of the LSM will be provided in one of the following chapters.

Back-Propagation-Decorrelation

Finally, the BPDC [6] [42] class of Reservoir Computing based on ANNs. The paradigm was born following an analysis of the dynamical changes happening to the weights in a Recurrent Network being trained with the Atiya–Parlos Recurrent Learning (APRL) rules. APRL is a gradient-descend based supervised training paradigm optimized for Recurrent Networks, estimating the gradient of the error with respect to the Neuron's activation instead of on the weights. Analyzing how the weights were modified by this advanced algorithm, it was shown that the hidden weights were much less updated then the connections directly linked to the output. APRL effectively decouples the Readout from the Reservoir, as said in the RC terminology. The consequence was that an even simpler learning algorithm was developed, with different policies for Reservoir and Readout weights, called the Back-Propagation-Decorrelation.
BPSC is shown to perform well with fast changing input signals, being almost unaffected by the parameters of the initialized Reservoir. However, the training paradigm easily forgets the data presented to it, when trained on new samples. This is due to the fact that the algorithm quickly adapts the Reservoir to the presented structure of the data.

Chapter 2

The structure of the Brain

Neuroscience is a concrete inspiration for mastering Liquid State Machine. As a consequence, it is helpful to get to know how the brain works in details. However, the functioning of the Brain at a high level is still to be investigated and represents one of the largest goals of Neuroscience. This is why, for example, one of the largest projects about studies on the Brain, such as the "Human Brain Project" [13], emphasizes the importance of the implementations of simplified models of neural systems for the understanding of the human Brian it-self. In that view, Neuromorphic Computing, the discipline of the implementation of Brain inspired computational model, plays a central role, offering the state-of-the-art hardware solutions employing modern electronics. The hope is that increasingly more biologically plausible and advanced models could give useful information about the functioning of Neural systems, resulting in advanced Artificial Intelligence applications and discoveries in Neuroscience.
Back to the biological nature of the Brain, the unit of such a system is the *Neuron*.

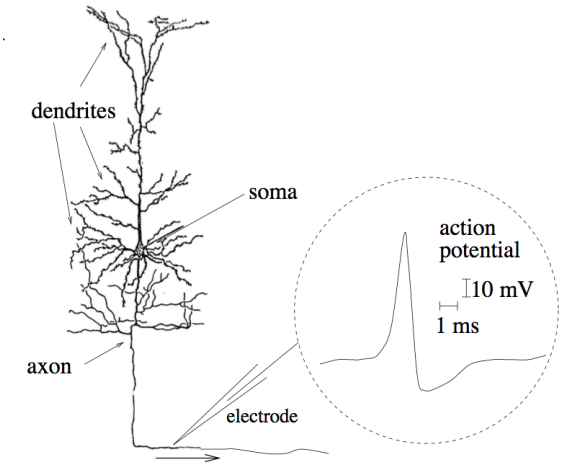


Figure 2.1: Figure of the drawing of a Neuron by Ramon Cajal

The Neuron is a complex cell basically constituted by: *Axons*, *Dendrites* and *Soma*. The last one is the "Computing Powering Unit" of the brain, being the location in which the non-linear processing of the information is performed. The Dendrites serve as the Input signal for the Soma, bringing the information from other neurons; the Axons, instead, carry the Output of Soma toward other neurons. As it can be perceived from the Figure above, which is even quite simplified in that concern, each neuron posses a high number of connections, thus forming a dense and intricate network with others neurons. It is observed that some neurons can influence - or be influenced by - up to 10^4

other neurons. Most of those lay close to the original neurons, even though connections can also be established between neurons belonging to different areas in the Brain, being several centimeters away from each other. As a consequence, the Brian, which is formed by circa 85 Billion neurons [21], is indeed a complex system: analyzing its behavior and understanding its functioning is as ambitious as it is difficult.
The Neuron is formed, as much am most of the human cells, by a membrane, which is permeable to certain molecules. In particular, the neurons' membrane is able to regulate the flux, in both directions, of some ions (Calcium, Potassium and others) thanks to the action of Ion Channels, which transport Ions in and out the membrane under a certain stimulus. Moreover, these Ions can also be exchanged with other neurons, when the information, under the form of an electrical signal, is spread through the network. These ions, with their electrical charge, vary the membrane potential: this is defined by the difference of potential between the internal and the external part of the neuron. The membrane, which is formed by a double lipidic chain, is almost a perfect insulator and is crossed by many Ion Channels, which control its permeability with respect to Ions. The difference of potential across the membrane is given by the difference in concentration of the ions, according to the Nerst equation:

$$\Delta u(x) = \frac{kT}{q} ln \left(\frac{n_1}{n_2} \right) \tag{2.1}$$

in which n_1, n_2 are the concentrations inside and outside the membrane, respectively; k is the Boltzmann constant, T the temperature and q the elementary charge. This formula is derived from thermodynamical considerations on the energy of an Ion inside the membrane, with a certain concentration in Ions n_1, and thus a consequent energy $E_{Ion}(x) = qu(x)$.
The basics of the functioning of the neuron is that when a spike, i.e. an impulse at a certain voltage from an other neuron, comes, then some ions are either introduced or expelled through the membrane of the neuron; as the concentration of the Ions inside the membrane changes, also the membrane potential does. When the membrane potential overcomes a certain threshold, the neuron emits a spike, also called *Action Potential*. This spike then serves as input for other neurons and eventually generates other spikes. The information in the Brain is constituted by the Spikes that the neurons emit in certain time t_f: since most spikes are similar to each other in their dynamics, i.e. the evolution of the membrane potential in time, it is commonly believed that the information is relegated to the timing and frequency of the spikes, rather than to its form.
Moreover, the Brain is also constituted by other types of biological systems: the most important ones are *Synapses*, which serve as a interface between a Dendrite and an Axon from different neurons. Synapses are complex biological processes bridging the gap between the Output component of a certain Presynaptic Neuron (the axon) and the Input component of a Postsynaptic one (the Dendrite). They are recognized to be a key component in the functioning of the Brain in that they regulate the magnitude of the input to the Postsynaptic neuron and are though to be of great importance in the learning processes of the Brain. As a matter of fact, synapses are plastic processes, in the fact that their biological features are adapted during the functioning of the Brain, getting stronger or weaker in conveying the signal from neuron to neuron.

2.0.1 The functioning of the Neuron

Considering a pair of neurons connected by a Synapse, when one of them fires and the spike reaches the second, the first one is said Presynaptic neuron, while the second is the Postsynaptic one. In a *Synapse*, a presynaptic spike causes the release of Neurotransmitters in synaptic cleft, a gap region which separates the membrane of the two neurons and composes the synapse. Those neurotransmitters are received by some specific receptors which, once stimulated, cause the opening of some Ion Pumps, modifying the membrane potential. As a matter of fact, Synapses can either be Excitatory

or Inhibitory, depending on the effect they have on the neurons' membrane potential, in turn due to the opening of different Ion Channels.
The Membrane potential has an equilibrium value, defined by the Nerst potential, referred to all the Ions present in the Neuron. The most important Ions which are described in most models of the neuron are Calcium (Ca^+) and Potassium (K^+), but also Sodium (Na^{2+}) and Chloride (Cl^-) play an key role. The membrane of the neuron is able to control the flow of ions in and out from the soma, depending on the relative concentration of ions and on the external stimuli the neuron receives. Moreover, when the neuron is in steady state, the membrane is able to retain a certain concentration, specific for each species of Ion, inside the neuron: for Calcium, the equilibrium concentration inside the membrane (about $60mM/l$) is generally lower than the one present outside (about $440mM/l$); instead, the opposite happens for Potassium, for which the equilibrium concentration inside the membrane is high (about $400mM/l$), while it is lower outside ($20mM/l$). Of course, the equilibrium is not static, in the sense that some Ionic currents are still present, but the positive current equals in magnitude the negative currents, for each species of Ion. This values give rise to two Nerst potentials which represent the balance point for the membrane potential: $E_{Na} = 52mV$, $E_K = -77mV$. This means that when the membrane potential is different with respect to those values, a flux of Ions will be established in order to restore equilibrium. Since the values of the ion-current changes sign around these Nerst potentials, they are also called Reverse Potentials. To give an example, the ion flux will be positive (from outside toward inside the neuron) when the membrane potential is lower than the Nerst equilibrium value. Moreover, it is experimentally found that the Resting Potential, i.e. the membrane potential of the neuron when no stimulus is applied, is of $u_{rest} = -65mV$. This means that at rest, since $E_K < u_{rest} < E_{Na}$ Potassium ions flow outside the membrane, while Sodium ions flow inside the neurons.
As Ions permeate the membrane due to external stimuli, the membrane potential changes and eventually results in a spike to be generated: formally, the spike, also called *Action Potential*, is caused by the membrane potential overcoming with positive derivative a certain threshold. The idea that the membrane potential grows until a threshold is reached, with a spiking event following the passing of that threshold, is originated in the paper from Lapique, in 1907 [16].The mathematical treatment of this process is the following:

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j \sum_f \epsilon_{ij}(t - t_j^{(f)}) + u_{rest} \tag{2.2}$$

where t_j is the firing time of the j-th neuron, while $t_j^{(f)}$ is the last firing time of the same neuron; u_{rest} is the resting potential. Instead, ϵ_{ij} is the potential in response to a spike coming from a presynaptic neuron, affecting the neuron under analysis: $u_i(t) - u_{rest} = \epsilon_{ij}$. Finally, $\eta(t - \hat{t}_i)$ is the form of the membrane potential when a spike happens.
This simplified and general model is said Spike Response Model [14] and it is the starting point for the development on any complex and accurate model of the neuron.
Once a spike has occurred in a real neuron, some Ions are released outside from the membrane and make the membrane potential decrease after a peak in a process which lasts for a certain period, called refractory period. In this time, the neuron is almost not responsive to other input spikes at all, and sees its membrane potential fall below the resting value. It is to note that in most cases the membrane potential is referred to the resting value, thus in the refractory period it has slightly negative values.
An Action Potential lasts for about $2ms$ and it reaches a voltage of about $100mV$. Its form is mostly the same for different neurons and in different moments, thus it is believed to bring no useful information in the Brain. The spike travels along the Dendrites and reaches the other neurons, through synapses, which are linked to the one who originated it. In turn, they will either excite or inhibit - according to the nature of the synapses - the membrane potential of other neurons and eventually cause other spikes to happen, spreading the information through the network. The information of the spikes

is thus to be attributed to the patters it produces in the Brain microcircuits and in their precise timing.

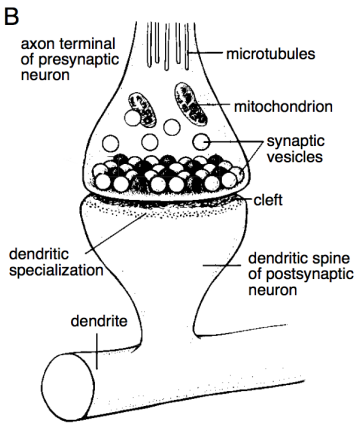


Figure 2.2: Simplistic sketch of a Synapse, from [15]

2.0.2 The functioning of the Synapse

As it can be seen in Fig 2.2, Synapses are composed by two processes, formally two extensions of the Dendrite and Axon which the synapse bridges. One is called Dendritic spine and originates from Dendrites, the other it the Axon's terminal. The latter contains mitochondria and other synaptic vesicles: those systems are stimulated when an Action Potential comes from the Presynaptic neuron, provoking a cascade of effects which results in Neurotransmitters being released from the synaptic vesicles in the synaptic cleft, being the gap between the Axonix and Dendritic processes. The surface of the Dendritic Spine is functionalized with some specific receptor for the neurotransmitters and, once stimulated, activate an electric signal which induce an Ion flow in the membrane of the Postsynaptic neuron. According to the charge of those Ions, the effect of the Ionic current can either be to depotentiate or to hyperpolarize the membrane potential. Respectively, one speaks of Excitatory and Inhibitory synapse.
To complete this simple description of the Synapse, the most important Ions involved in triggering the release of Neurotransmitters is Calcium (Ca^{2+}), which, together with the action of mitochondria, is believed to support the release of the synaptic vesicles into the cleft. Moreover, the most common type of Neurotransmitter are: glutamate, Acetylcholine, Dopamine, Norepinephrine, Serotonin and GABA. Each of them has been seen regulating physiological and behavioral properties of the brain. Details about their impact on the neuron's dynamics are still to be investigated, so they are normally not treated in detail for neuromorphic computing applications.

Other Brain constituents

Other important constituents of the Brain are the Glia cells, which are supportive cells for neurons. Their functionality is not completely understood and it is often neglected in simpler Neuronal models. Most of Glia cells are thought to serve a supportive functionality for neuron, providing nutrients, oxygen and acting as a scaffold for the neuron itself. Some other, in particular Astrocytes, are believed to participate in the functioning of the neurons and regulate some fundamental molecular mechanism having to do with Plasticity. For the rest of this work, a part for those which have to do with Plasticity mechanisms, they will not be further analyzed.

2.1 Models for the Brain

During the course of the 20^{th} century, a lot of effort has been devoted in creating a model which would explain the functioning of the neuron, especially exploiting electronics as the platform in which to implement such models. Early examples had simple equations and circuits and were characterized by spike's profile - previously denoted by $\eta(t - t_i^f)$ - based on square pulses of even delta pulses. With the progress in the understanding of the physiology of the Neuron and of the available methods for reproducing the experimental results in the form of systems of differential equations and/or electronic circuits, mode complex and accurate models were developed. It must be pointed out that the most advanced ones still have no implementation in the form of hardware for state-of-the-art Neuromorphic Computing, but represent an inspiration toward which the research is moving in order to provide the Neural Networks implemented in hardware a plausible behavior with respect to the Brain. As a matter of fact, technology sets the limit for the implementation of the latest and more advanced Neuronal model. Moreover, the challenge of biologic plausibility extends also at the systems level, since the complication introduced by detailed Neuron Model are enhanced when considering a Network in its collective behavior. More details concerning the functioning of state-of-the-art Neuromorphic hardware will be given in the following sections.

Neuron as an Electronic Circuit

As the first models of the neurons were developed as systems of differential equations, it become convenient to try to implement them in the form of electronic circuit, trying to capture the accurate dynamics of the neuron. Of course, this was no easy task, since both the theoretical models of the neuron from physiology and the possibilities of technology for what concerns electronics circuits are limited. This is why many models were developed with the aim of capturing the essence of the Neuron's functioning but keeping the level of complexity to a low level, convenient in order to transferred the model in simulations and hardware implementations.
Even though some models are more refined and better representing all the complex mechanisms of the neurons, most of them start with the same baselines, which are here reported. The membrane of the neuron is a 3-4 nm thick lipidic bilayer, being an almost perfect insulator for Ions and most charged particles. As a consequence, it is modeled as a Capacitor C_m. Following the law of capacitors, the membrane potential is changing as the charge inside the neuron varies. Moreover, modeling the possibility of the neuron of maintaining a constant voltage - different than the resting potential u_{rest} - under an external current stimulus I_e, the membrane also has an equivalent Resistance R_m. The consequence is that the membrane can be described by means of the parallel of a Capacitor and a Resistance, forming a low-pass filter which modulates the input stimuli (Action Potentials) from other neurons. Also, the multiplication of those factors gives the time constant of the membrane $\tau_m = C_m R_m$, setting the period of the fundamental variations of the membrane potential.
In addition to that, the membrane presents Ion Channels, which allow some Ions to travel through the membrane and reach the soma. From an electronic point of view, the Ions Channel abruptly diminish the equivalent resistance of the Membrane, as a variable Conductance $G_m(t)$. This quantity hides most of the complexity of the Neuron, since the dynamics of the Ions is regulated by the ionic Reverse Potential, defined for each type of Ion and function of time in general. To some extent, each Ions population behaves under the same rule, namely the Nerst equation, even though the intricate structure of the Ionic Channels makes it necessary to treat them with coupled equations. Moreover, each neuron posses several kind of different Ions Channels, whose density is of about a few hundreds per square micron of membrane, for a total of several hundreds of thousand for a single neuron: some of them are functioning for a certain type of Ion only, some other being used by more kind of Ions.
Considering all the Ions Channels and all the types of Ions, it is possible to determine a membrane

Membrane Area (A_m)	$[0.1 - 0.01]mm^2$
Membrane Thickness (t_m)	$[3 - 4]nm$
Specific Membrane Capacitance (C_m)	$10nF/mm^2$
Membrane Capacitance (c_m)	$[0.1 - 1]nF$
Specific Membrane Resistance (R_m)	$1M\Omega mm^2$
Membrane Resistance (r_m)	$[1 - 100]M\Omega$
Membrane Time Constant ($\tau_m = R_m C_m$)	$[10 - 100]ms$

Table 2.1: Main quantities in the circuital representation of the Neuron, as from [15]

current, i.e. the current flowing through the membrane due the action of Ion Channels:

$$i_m = \sum_i g_{m,i}(u(t) - E_i) \tag{2.3}$$

where the subscript i indicates the type of Ions participating to the total current and both the conductance g and the current i are expressed per unit of area. Of course, the main driving force for Ions is the voltage difference of the membrane potential with respect to the Reversal Potential E_i, characteristic of each type if Ion.
As a matter of fact, not all the conductances appear from experimental evidence to be time-dependent, so that it is convenient to separate the constant and time-dependent terms. Concerning the almost constant conductance term, that it is commonly referred to as *Leakage Current* $\bar{g}_L(u - E_L)$, where "L" refers to Lumped.
Gathering all the elements described up to now in a differential equation, one gets the general and basic rule for the functioning of a Neuron in terms of equivalent electronic circuit:

$$C_m \frac{\partial u}{\partial t} = i_m + \frac{I_E}{A} \tag{2.4}$$

where the electrode current I_E is not generally normalized by the neuron's area A and the membrane current i_m is not specified in its form, but is expresses the complication of the Ion Channel system. In principle, the membrane current term should also provide the model the ability of producing an Action Potential event as the membrane potential reaches a certain threshold voltage θ. Moreover, this term regulates the shape and duration of the Action Potential itself. Finally, a very complete model should also be able to describe the refractory period following a spike: for a certain time after an action potential occurred, the membrane potential remains below its resting value and the conductance of the Ionic Channels is reduced, so that the effect of incoming stimuli is attenuated. After the refractory period, the neuron returns to function as usual.
Expressing all this complex dynamics is an ambitious task to perform and often not even required for the specific application in which the model is employed. As a matter of fact, a complete model is more difficult to reproduce both from a computational and a technological standpoint. It is thus often required to employee a simpler model, mostly to analyze the behavior of a network of neurons. For such a case, the behavior of the network becomes too complex and chaotic is the model of the neurons is very detailed.

For the following part, the focus is on the 3 main models of the Neuron from an historically point of view, in increasing order of complexity, biologic plausibility and chronological time of development: McCulloch-Pitts, Integrate and Fire and Hodgkin and Huxley (HH). More recent models have been

developed, mainly starting from the HH one, but they have limited application in state-of-the-art neuromorphic hardware.

2.1.1 McCulloch-Pitts

It represents the most simple model for the neuron, almost neglecting all the complications of the biological Neuron and focusing on the non-linear operation of the soma with respect to its inputs. The spikes for this model are simple Dirac's delta of value 1, so the output of the Neuron is simply either 1 (firing) or 0 (not firing). The spikes are modulated by the synaptic strength, also called *weight*, being a real number generally comprised in the interval [-1,1], which allows to take inhibitory synapses into account, but also commonly restricting the interval to the case [0,1].
The neuron takes a number N of inputs, deriving from other neurons weighted by synapses, and performs a linear sum of those values: if this sum overcomes a certain threshold, then it emits a spike. This is formally defined by:

$$u_i(t) = \sum_j^N I_j \times w_{ij} \tag{2.5}$$

where u_i is the potentiation (another term for membrane potential) of the i^{th} neuron; I_j is the input from the j^{th} neuron; w_{ij} is the weight of the synapse between neurons i and j.
As it can be understood, this is an overly-simplified model, which is not considered in applications in which biological plausibility is an important requirement. Nonetheless, this model was the base-line for the development of Neural Networks, later giving rise to the field of Deep Learning. The consequent modification of the model concerned the operation of the neurons, which at this stage is a simple sum: implementing a non-linear summation of the inputs allows for a more complex dynamics of the network leading to the ability of learning more sophisticated tasks. Anyhow, this type of evolution of the McCulloch-Pitts neuron falls out from the bounds of biologically inspired research and leads to development of Artificial Neural Networks, whose aim is to obtain the best performance for its algorithms, independently of the nature of the models which constitute its Networks.

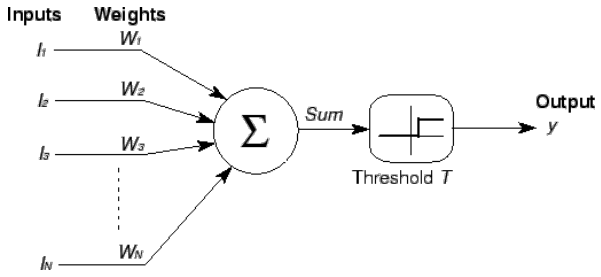


Figure 2.3: Graphs representation of an McCulloch-Pitts neuron, from [14]

2.1.2 Integrate and Fire

All the models which describe the dynamics of the membrane potential, independently on their accuracy, with a differential equation based on Eq.2.4 and leave the description of the formation and dynamics of the Action Potential to an additional part of the algorithm describing the neuron, coming in action when the threshold voltage is passed, fall in the category of Integrate and Fire neurons. They are also the most popular choice in Neuromorphic Computing application because of their versatility and simplicity: the electronic circuit required to implement the neuron is relatively simple and the

computational power required for simulating Networks composed by such model is modest.
As a matter of fact, the complex dynamics of the Neuron is often relegated to the membrane conductance, especially when describing the Action Potential formation and the Refractory Period. In this models, the shape and duration of the spikes and of the refractory period are forced by the intervention of a part of the algorithm describing the neuron: the way this is implemented is straight-forward for the case of simulation, while in the case of hardware versions of the neuron this is performed by additional circuits which take over the control of the membrane potential in some specific conditions.
Many examples of Integrate-and-Fire neurons have been developed, each of which differentiating for the level of rigor with respect to biology of the sub-threshold dynamics and for the form and duration of Action Potential and Refractory period: a very simple case, the Leaky-Integrate-and-Fire, gained popularity for application in Neuromorphic Computing. This model treats the conductances of the membrane as a single time-independent term $i_m = \bar{g}_L(u(t) - E_L)$. As a matter of fact, it is observed that the membrane conductances are almost constant for small fluctuations around the resting potential, even though the Leaky-Integrate-and-Fire takes this assumption as valid for the whole sub-threshold range. With this approximation, the resulting electronic circuit describing the sub-threshold dynamics is a simple parallel of a Capacitor and a Resistance.
Developing the membrane current term and multiplying the whole Eq.2.4 by the membrane resistance R_m, the fundamental equation for the Leaky-Integrate-and-Fire becomes:

$$\tau_m \frac{\partial u}{\partial t} = (u(t) - E_L) + I_E R_m \tag{2.6}$$

where E_L represents the Resting Potential which is commonly set to zero. The parameter $\tau_m = R_m C_m$ still represents the time constant of the membrane potential, R_m is the membrane resistance and C_m the membrane capacitance. The model is said "Leaky" since, in absence of external stimuli I_E, the equation yields the membrane potential to decrease in an exponential fashion with time constant τ_m.
As the membrane potential overcomes the threshold, which is commonly around -50/-55 mV [15], an Action Potential is activated having the shape $\eta(t - t_i^f)$, t_i^f being the firing time of the i^{th} neuron. After the spike event is concluded, the Neuron is inactive for a certain period T_{refr}, the refractory period.
By knowing the duration of the spike, the refractory period and the time constant of the neuron, it is possible to calculate analytically the firing rate of the neuron ν under a constant external current I_E:

$$\nu = \left[T_{refr} + \tau_m ln \left(\frac{RI_E}{RI_E - \theta} \right) \right]^{-1} \tag{2.7}$$

where θ is the threshold for the membrane potential.
As a matter of fact, the input current of a Neuron is coming from other Neuron and depending on the interaction at the Synapses between the presynaptic and postsynaptic neurons. This interaction is not trivial and is a process which has to be described in a biologically inspired model of neuron. If for the simple McCulloch-Pitts neuron the synapse are modeled by a simple real number w_{ij}, the weight, modulating the signal from the presynaptic spike, Integrate-and-Fire neuron often include better description of the physiology of the Synapse: the general model for the synaptic current is reported below.

$$I_{syn,i} = \sum_j w_{ij} \sum_f \alpha(t - t_j^f) \tag{2.8}$$

That states that the input of every neuron is due to the output of every other neuron to which that is linked, modulated by a weight and by a function α which hides the complex biological nature of the Synapse. This latter term is treated in the same way as the membrane conductance, since the Synapse can be seen as a variable resistance which allows a certain amount of current to pass under certain stimuli. An important feature of Synapse of that their conductance is a function of the membrane

potential, which leads to the form of synaptic current term as:

$$\alpha(t - t_j^f) = -g(t - t_j^f)\,[u(t) - E_syn] \tag{2.9}$$

where E_syn is the Reversal Potential of the synapse. This parameter allows to distinguish excitatory synapse - for which this value is above the resting potential $E_syn > u_{rest}$ -form inhibitory synapse - for which the opposite holds $E_syn < u_{rest}$.
It is also to be noticed that if the Synaptic Reversal Potential is about the value of the threshold $E_syn \approx \theta$, the contributions from the presynaptic neuron vanish as the membrane potential is increased. The risk is that this effect, called Synaptic current saturation, may not allow the neuron to ever reach the threshold and emit a spike. For this reason, the condition $E_syn > \theta$ must be verified.
The following figure sums up the model of Integrate-and-Fire neuron in its circuit representation, in the case of a delta spike being at the input.

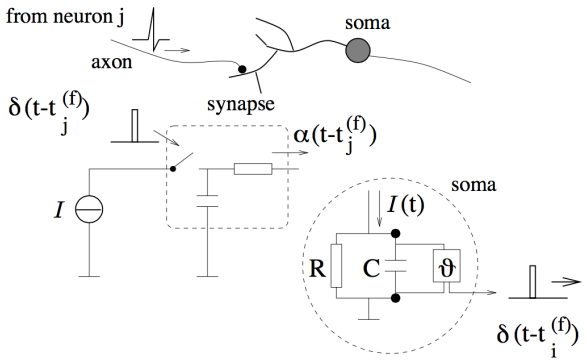


Figure 2.4: Circuital representation of an Integrate-and-Fire neuron, from [14]

At last, extended versions of the Integrate-and-Fire model are to be mentioned: they all concern the development of the non-linearities related to the conductances of the membrane potential and synapses. The general equation for the non-linear model becomes:

$$\tau_m \frac{\partial u}{\partial t} = F(u) + G(u)I_E \tag{2.10}$$

where $G(u)$ is a voltage dependent input resistance, while $-F(u)/(u - u_{rest})$ is a voltage dependent decay constant.
A case of non-linear Integrate-and-Fire neuron model is the *quadratic* model, whose equation is:

$$\tau_m \frac{\partial u}{\partial t} = -a_o(u - u_{rest}(u - u_c)) + G(u)I_E \tag{2.11}$$

with $a_o > 0$ and $u_c > u_{rest}$. This model results, as it can be easily guessed, in a quadratic variation of the membrane potential under the effect of an input current, opposed to the usual linear summation of the inputs of the standard Integrate and Fire neuron.

2.1.3 Hodgkin-Huxley model

In 1952, the Nobel prize winners Hodgkin and Huxley published a work about the analysis of the Giant Axon of a Squid [17] [18] [19] [20]. This was one of the very first studies able to capture the dynamics of Ions in the functioning of the neuron. The developed model analyses the behavior of 3 Ion channels: Sodium (Na^+), Potassium (K^+) and a third Channel which is generally called *Leak*

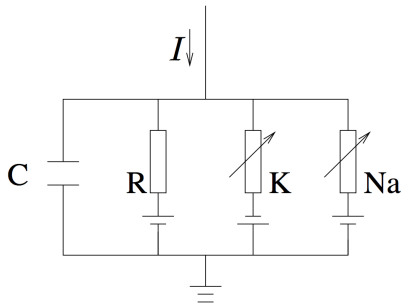


Figure 2.5: Equivalent Circuit of the Hodgkin Huxley model for the Neuron, from [14]

Channel and is thought to mainly describe the dynamics of Chloride (Cl^-). Actually, the leak current derived from it takes care of all the Ionic Channels which are not explicitly described in the model. As a consequence, one can extend the accuracy of the representation by adding one term to the base equation referred to a specific Ionic Channel. This equation, which fully characterizes the model, without the need of external intervention - as instead it happens for the Integrate-and-Fire - for the description of feature of the neuron is based on the general equation of the circuital representation of the neuron 2.6 and it specifies the terms related to the conductances of the Ion Channels.

$$C\frac{\partial u}{\partial t} = -\sum_k I_k(t) + I(t) \tag{2.12}$$

$$\sum_i I_k(t) = g_{Na}m^3h(u(t) - E_{Na}) + g_K n^4(u(t) - E_K) + g_L(u(t) - E_L) \tag{2.13}$$

In this model the variable conductances are represented by some values g_{Na} and g_K which set the upper limits, being modulated by some factors ranging from 0 to 1, thus often referred to as *Gating Probabilities* or *Gating Variables*. Instead, the conductance of the Leakage Channel is constant in time and set as g_L. The dynamics of the conductances are governed by the behavior of the parameters $m(t), h(t), n(t)$: the first two of which are operating on the Sodium Channel, while $n(t)$ acts on Potassium. All of them are characterized by a differential equation in which two functions α and β appear, being empirically modeled to describe the squid's giant axon dynamics.
The differential equations regulating the gating variables are reported below:

$$\dot{m} = \alpha_m(u)(1 - m) - \beta_m(u)m \tag{2.14}$$

$$\dot{h} = \alpha_h(u)(1 - h) - \beta_h(u)h \tag{2.15}$$

$$\dot{n} = \alpha_n(u)(1 - n) - \beta_n(u)n \tag{2.16}$$

the behavior of this function is visualized by the following figures.

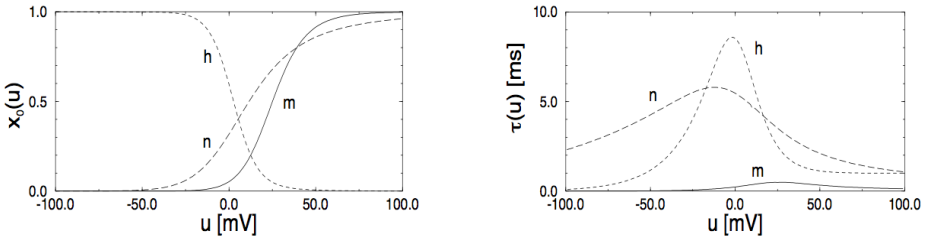


Figure 2.6: Plots of the dynamic behavior of the Hudgkin-Huxley gating functions. m and h are referred to the Sodium channel, h to the Potassium channel. Figure from [14]

As it is clear, m and n are increasing with positive membrane potential, while h is decreasing: the result is that the conductances of the Sodium and Potassium channels are increased when the membrane potential is depotentiated by an excitatory presynaptic spikes and, if those inputs are strong enough, a positive feedback establishes. Anyhow, the time constant of m is lower than that of h, where the first values tends to open the conductance channel while the latter closes it. The result is that if the membrane potential is not raised strongly and quickly enough then the positive feedback does not hold and the h gating function shuts the Sodium channel off. The behavior of the Potassium channel is instead simpler, being governed by one only gating function, which increases its value as the membrane potential is depotentiated.
With the values and function developed by Hodgkin and Huxley, the Action Potential is of about $100mV$ and lasts for about $1ms$. The following figure gives some information about the kernel of the HH-action-potential and of the refractory period of the neuron.

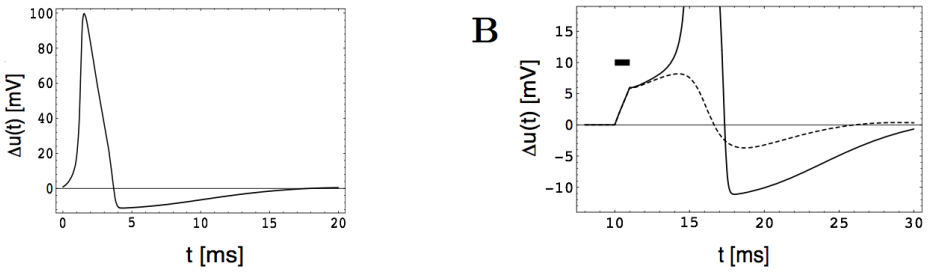


Figure 2.7: Plots of the Action Potential of the Hudgkin-Huxley neuron. On the left, the neuron is stimulated by a strong but short current pulse, able to generate the action potential and observe the refractory period under no external influence. On the right, a comparison of the behavior of the neuron under a current pulse of 1ms of either $7\mu A/cm^2$ and $6.9\mu A/cm^2$: the first is able to generate the Action Potential, while the latter is not. This is due to the Threshold Effect. Figure from [14]

As it can be understood from the Fig.2.7, the amplitude of the input current is a key parameter in determining whether the neuron will produce a spike or not. The plot on the right of Fig.2.7, despite having two different scaling for the y-axis, shows that an apparent subtle different in magnitude is able to determine the output of the neuron. In a real situation however, a single input from a presynaptic neuron is not able to cause the Action Potential to happen at the postsynaptic neuron. Moreover, since the conductance is generally reduced as the membrane potential get closer to the ideal threshold - remember that no formal condition on the threshold is imposed for the HH neuron, instead the firing condition is encoded in its base differential equation -, more and more input spikes are required to generate an Action Potential. It has been measured [15] that an input spike is at about 1mV and that the Action Potential is triggered for a depotentiation of $20 - 30mV$ (of the membrane potential with respect to its resting value). As a result, depending on the frequency and on the magnitude of the incoming spikes, the number of input events necessary to generate an Action Potential is between 20 and 50.

Ion Channel	Reversal Potential: E_x	Maximum Conductance: g_x
Na	50mV	$120mS/cm^2$
K	-77mV	$36mS/cm^2$
L	-39mV	$0.3mS/cm^2$

Table 2.2: Values of the Reversal Potential and Maximum Conductances related to Eq.2.13

Gating Function	$\alpha_x(u/mV)$	$\beta_x(u/mV)$
n	$(0.1-0.01u)/[exp(1-0.1u)-1]$	$0.125exp(-u/80)$
m	$(2.5-0.1u)/[exp(2.5-0.1u)-1]$	$4exp(-u/18)$
h	$0.07exp(-u/20)$	$1/[exp(3-0.1u)+1]$

Table 2.3: Form of the functions related to Eq.2.16. The form and values of the parameters were fitted by Hudgkin and Huxley to describe the Squid's Giant Axon.

2.1.4 More Advanced Models

As already said, an advanced and complex model finds no applications in neuromorphic hardware, neither in Computer simulation for complex Networks, so there is no interest in discussing them in details. Nonetheless, advancements in technology is expected to include in neuromorphic hardware increasingly biologically plausible models, so they represent an inspiration for the development of the next generation of chips.
On one side, models are developed starting from the Hodgkin Huxley one, improving the fidelity of the variable conductances to the biology of the neuron in order to account for most of the small events happening in neuron not described by the Sodium and Potassium channels in the HH model.
One other approach is even more effective in aiming at fully describe the neuron: to expand the model from a lumped circuit model to a distributed level. As a matter of fact, the neuron is a 3 Dimensional systems and the length of some of its components, related to their width, is such that a description with Cable theory can be beneficial. In particular, Dendrites and Axons are often treated with this formalism. Such models fall in the set of Compartment (or multi-compartments) Models, whereas the ones analyzed before are labelled as Single-Compartment Models. The following figure illustrates an example of a model of a Dendrite: reporting the equations related to such a system goes beyond the scope of such a section of the Thesis, but the interested reader is invited to get all the details about it in the reference [14].

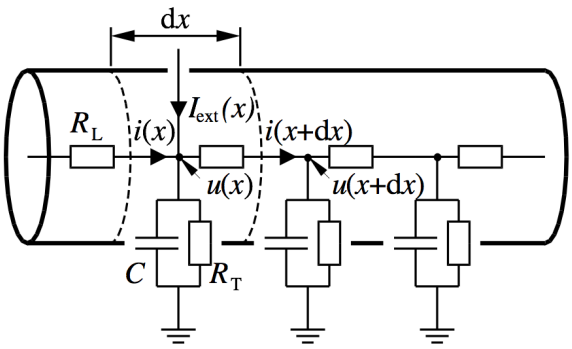


Figure 2.8: Figure of the drawing of a Distributed Dendrite, from [14]

2.2 Plasticity Mechanisms: Learning in biological systems

State variables of Neurons and Synapses are not fixed in time but vary depending to certain rules, which are said Plasticity mechanisms. Those are of fundamental importance at the behavioral level in that they are associated to the learning and memorization capabilities of the Brain. As a matter of fact, Learning as a process in the Brain is attributed to every change of the Synaptic and/or Neuronal state variables in order to map a certain Input, coming from the sensory receptors, to a certain Output. Recalling the previous Section about Neural Networks, this is the same principle regulating Learning in those algorithms. The difference of the learning process in Neural Networks with respect to biological neural systems is that the learning paradigms are greatly simplified, in order to be conveniently computed by PCs and be able to learn "quickly".
Modeling the Plasticity mechanisms is as challenging as it is for the dynamics of Neurons and Synapses, so simplified models have been developed in order to result more efficient when employing them in simulation of neural system. Most of them also had to deal with the fact that the state-of-the-art hardware for the implementation of neuronal systems struggles in updating its internal variables efficiently, thus making Plasticity algorithms limited both for what concerns speed and consumed energy of the chip. This is because Plasticity mechanisms are not about reassigning a new values to the state variables of the system from scratches, but updates are chosen based on the past states of the network and on some peculiar criteria of the specific neuron/synapse on which they act. It results that the information required to compute and reassign the values of the internal variables of the neuron/synapse must often be stored in a memory element which is accessed by and external processing unit. This leads to the well known problem of the Von Neumann-bottleneck, for which the temporary storage and computation of those variable away from where the processing of the data is performed makes the system less efficient from the energetic point of view and sets the limit for the speed of the device. This argument will be treated more in detail in the section devoted to Neuromorphic Computing.

2.2.1 Hebbian Plasticity

The theoretical base for most of the model about Plasticity is the Postulate of D.Hebb, which in his book from 1949 [22] stated that:

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.

The idea behind this postulate is that there had to be correlation between spike events of two neurons in order to produce a change of synaptic weight or neuron's internal variable. Hebb thought that when the same input were repeatedly stimulating a certain area in the Brain, the connection between neurons become stronger in order to maintain a similar firing patter in each repetition, meaning that a function was mapped starting from the persistent input. As it has been stated, the change happens when the presynaptic neuron participates to the firing of the postsynaptic one, but that represents only half of the up-to-date statement which is the fundamental rule for Plasticity:

Synapses change in proportion to the correlation or covariance of the activities of the pre- and postsynaptic neurons.

In that sense, connections between neuron are not just potentiated by Plasticity but can also be weakened by events in which neurons spike in a particular order. Two types of Plasticities are defined in turn: Long term Potentiation (LTP), occurring when the connections between neurons are persistently strengthened; Long term Depression (LTD) when the connections are weakened. It can be easily understood that any model trying to capture the behavior of such mechanisms must avoid the problem of synaptic weight divergence: the weight of the synapse cannot always increase by the same amount in the case the same input is displayed repeatedly, since that would be inconsistent with the functioning of a real synapse. Either some constraints to the strength of the connection are embedded in the model or they must be hard coded for the specific case. Back to the dynamics of Plasticity mechanisms, the timing of the spikes determines whether the connections between neuron should be potentiated or weakened. In references [15] and [14] one can find in their Bibliographies many examples of studies in which the generalized Hebbian Postulate was found to be shaping the strength of the synapses and the internal Neuron's state. This Plasticity mechanism was not the only one found in physiology studies - all the other phenomena are referred to as *Non-Hebbian Plasticity* mechanisms -, but that certainly is the one for which most research was conducted and the most accurate experimental evidences were obtained. The reason being that this paradigm appears to be the most important in relation to the learning in the Brain. As a consequence, this was also the most analyzed and employed Plasticity mechanism in Neural System simulation and Neuromorphic Computing, and also the one for which the most biologically plausible mathematical models were developed. There are many models for Hebbian Plasticity developed with different levels of accuracy to biology and simplicity in hardware or software implementation, but there is also a common form for the base equation regulating the phenomenon.
In this formalism, the Activity of the presynaptic neuron is ν_j and the one of the post-synaptic neuron is ν_i.

$$\frac{\partial w_{ij}}{\partial t} = F(w_{ij}, \nu_i, \nu_j) \tag{2.17}$$

The function F expresses the correlation between the activity of the Pre- and Post-synaptic neuron and also depends on the previous values of the synaptic weight. A requirement for such function is that it should avoid the positive feedback which establishes when similar input are repeatedly displayed to the Network, stimulating the firing of the same group of neurons. The consequence is that most of the synapses get clamped to their upper bound and are not able to carry information and contribute to the learning process. The solution for such a problem is to employe coefficient which modulate the weight update strength as a function of the upper and lower limits.
Moreover, this general form does not specify the temporal relation of the spikes which generates the change in synaptic weight. In order to do so, the F function is commonly expanded around $\nu_i, \nu_j = 0$ and some of the terms are discarded as an approximation.

$$\frac{\partial w_{ij}}{\partial t} = c_0(w_{ij}) + c_1^{post}(w_{ij})\nu_i + c_1^{pre}(w_{ij})\nu_j + c_2^{post}(w_{ij})\nu_i^2 + c_2^{pre}(w_{ij})\nu_j^2 + c_2^{corr}(w_{ij})\nu_i\nu_j \quad (2.18)$$

where the coefficient c_2^{corr} expresses the correlation between the Pre- and Post-synaptic Neuron. The simplest formulation of Hebbian Plasticity only implies this term and thus is:

$$\frac{\partial w_{ij}}{\partial t} = c_2^{corr}(w_{ij})\nu_i\nu_j \quad (2.19)$$

This simple equation states that the weight of the synapse varies only when the Pre- and Post-synaptic neuron are active mostly at the same time. In that sense, a positive coefficient c_2^{corr} makes the weight stronger and represents the case of Hebbian Plasticity. Instead, if the same coefficient is negative, the equations describes a Anti-Hebbian mechanism in that the weight is weakened even if the two neuron are active simultaneously. Moreover, if a first order term is included in the equation, then it is the case of a Non-Hebbian Plasticity mechanism, while more complex Hebbian case can be developed by including the remaining second order terms.
As already said, a problem of such a model is Synaptic Saturation: when a large portion of the Network is stimulated, most of the synapses are reinforced, leading to a positive feedback mechanism which clamps most of the synapses to the upper bound. The same may happen but with respect to the lower bound, normally fixed at 0: in this case one speaks of vanishing of the synaptic weights. A possible form for the coefficients in order to solve this issue is the following: $c_2^{corr}(w_{ij}) = \gamma_2(1 - w_{ij})$ for the upper bound; $c_0(w_{ij}) = .\gamma_0 w_{ij}$ for the lower bound. Including the zero order term in Eq. 2.19 one get a more explicit form for an Hebbian Plasticity mechanism:

$$\frac{\partial w_{ij}}{\partial t} = \gamma_2(1 - w_{ij})\nu_i\nu_j + \gamma_0 w_{ij} \quad (2.20)$$

For a mention, Plasticity models based on the terms of the expansion in Eq.2.18 of the first order are not strictly Hebbian model. As a matter of fact, in this cases the synaptic weight is modified even if only either the Pre-synaptic or Post-synaptic is active. The same happens for the quadratic terms, for which the dynamic is even more complex. Typically, this mechanisms are not employed alone but constitute additional terms in a more complete equation describing a peculiar quasi-Hebbian plasticity. Since they are often omitted in simple implementation in simulations and neuromorphic hardware, they are not further analyzed.

2.2.2 Spiking Time Dependent Plasticity

It represents the most common form of Plasticity in simulations and implementations of neuromorphic hardware, since it can be developed as a simplified version of Hebbian Plasticity conserving a certain level of fidelity with respect to the biologic behavior in the Brain. The key feature of such models is that the weight change depends on a variable $s = t_j^f - t_i^f$ being the delay of the Post-synaptic neuron's firing event. Note that if $s < 0$ means that the Post-synaptic neuron has fired before the Pre-synaptic one.
The reinforcement/weakening is generally composed by two contributions: a non-Hebbian term both in positive and negative form depending on the Pre- and Post-synaptic activity alone; an Hebbian term, also in both positive and negative form depending on the correlation between the activity of the two neurons. This means that a change of weight occurs in any case of firing event of either of the two considered neurons, not just requiring a correlation between the two firing events. The mathematica

formulation of such statements is:

$$\frac{\partial w_{ij}}{\partial t} = a_0 + S_j(t)\left[a_1^{pre} + \int_0^{\inf} a_2^{pre,post}(s)S(s-t)ds\right] + \\ +S_i(t)\left[a_1^{post} + \int_0^{\inf} a_2^{post,pre}(s)S(s-t)ds\right] \tag{2.21}$$

the parameters a_1^{pre}, a_1^{post} compose the non-Hebbian term, while $a_2^{pre,post}(s), a_2^{post,pre}(s)$ are the kernel referred to the Hebbian one. Moreover, the functions S are time-dependent counters for the firing events of the Pre- and Post-synaptic spikes: $S_k(t) = \sum_f \delta(t - t_k^f)$, for $k = i, j$.
According to the already mentioned requirement to avoid synaptic vanishing and saturation, the form of the coefficients a_1 and of the Kernels a_2 is often a function of the current synaptic weight and of the weight constraints. In this way, it is assured that the weight values remains within their bonds.
In simplified explicit model derived from Eq.2.21 the non-Hebbian terms are commonly dropped and the kernels $a_2^{pre,post}(s), a_2^{post,pre}(s)$ are respectively positive and negative and act for $s < 0 and s > 0$.
A typical choice for the form of the STDP function is the so-called Exponential Learning Window, in which the updates on the synaptic weights follows a piece-wise exponential function changing sign around $s = 0$:

$$\Delta w_{ij}(s) = \begin{cases} A_+ exp[s/\tau_+] & \text{if } s < 0 \\ A_- exp[-s/\tau_-] & \text{if } s > 0 \end{cases} \tag{2.22}$$

This represents the most simplified and common form of Plasticity in simulation and Hardware implementation of Neural Systems, and is the main rule allowing for Unsupervised learning, usually followed by Principal Component Analysis for the readout of the Network. As a matter of fact, STDP is the main learning rule allowing a learning process in Unsupervised training, thus the parameters which regulates it must be carefully tuned so to maximize the accuracy of the system.
In the following, the parameters may be expressed with a different formalism, but the general rule is that $A_+ > 0$, $A_- < 0$. As always, those variables can be expressed in a way to avoid synaptic vanishing and saturation: $A_+ = (1 - w_{ij})a_+$, $A_- = w_{ij}a_-$.

2.2.3 Synaptic Normalization

Hebbian rules, despite being the main contribution to synaptic change in order to allow learning to happen, are not the only Plasticity mechanism present in the Brain and, to some extent, one could state that without the other accessory feature of Plasticity learning would not be as efficient, if not even possible. As a mater of fact, STDP, the most common expression of Hebbian learning paradigm, leads to major issues in the dynamics of the Network. Even with the safe policy for the coefficient to avoid synaptic saturation and vanishing, the possibility of storing information in the network may be limited in particular cases by STDP it-self. For example, consider a neuron with many more inputs than average: as the information is spread through the Network, this neuron is more likely to produce frequent spiking. By doing so and thanks to STDP, its input synapses are more likely to get stronger, generating a positive feedback which leads to a situation in which most of the input synapse of such a neuron are close to the upper bound level. As this neuron also has outputs, that situation may spread through the Network, causing most of the synapse to be close to upper bound level. In than situation, the Network looses ability to store more information in the synaptic weights, decreasing the learning potential.
Actually, it has been shown that, during the dynamics of the Brain, most of the synapses are close to the bound levels, either up or down state, in an almost binary configuration. Still, the distribution of

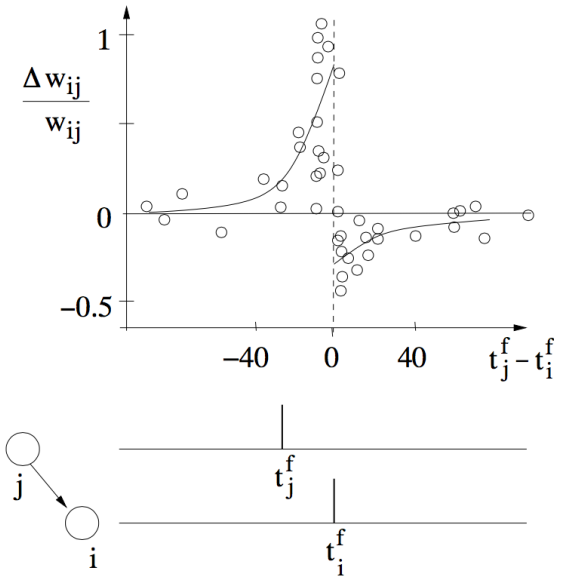


Figure 2.9: Functioning of the STDP Plasticity mechanism, with dots representing experimental data and the solid lines being the interpolated function for the STDP model. One understand why the Exponential Learning Window version of STDP is so popular in applications. Figure from [14]

these two quasi-states is far from being unimodal, which would inhibit learning capabilities.
The missing piece is a further Plasticity mechanism which regulates the weight distribution of different synapses, not allowing an entire group to be clamped at either of the two bounds. This mechanism is Synaptic Normalization.
It is not still clear how this mechanism acts at a cellular level, but models are developed from the evidence of the synaptic weights distribution at the long run. As a consequence, two approaches are possible: to act globally in the network: if the weights are all non negative, one imposes a constraint on the sum of the weights; valid also in case some of the weights being negative, the constraint can be applied to the sum of the square values of the weights. An example of both cases is given.

Subtractive Normalization

Give a network of N_u synapses, $\mathbf{n}$ being a N_u dimensional vector in which each entry is equal to 1, the equation governing this Plasticity mechanism is:

$$\tau_w \frac{\partial \mathbf{w}}{\partial t} = \nu_i \nu_\mathbf{j} - \frac{\nu_i(\mathbf{n} \cdot \nu_\mathbf{j})\mathbf{n}}{N_u} \tag{2.23}$$

The constraint imposed by such rule is that the synaptic the vector $\mathbf{n} \cdot \mathbf{w}$ is kept constant in time. This is easily verified by the next equation:

$$\tau_w \frac{\partial \mathbf{n} \cdot \mathbf{w}}{\partial t} = \nu_i(\mathbf{n} \cdot \nu_\mathbf{j}) \left(1 - \frac{\mathbf{n} \cdot \mathbf{n}}{N_u}\right) = 0 \tag{2.24}$$

where the last equality follows from $\mathbf{n} \cdot \mathbf{n} = N_u$. The main characteristics of Subtractive normalization is that the relative variation of weight applied to those of lower values is greater than the ones with higher values. As a consequence, some constrain have to be imposed in oder to avoid some weights to vanish or to become negative. Moreover, the process is non-local, since the it takes to know the values of all the weight related to a certain Post-synaptic neuron in order to perform the normalization.
The usual result of this policy, when combined with STDP, is that, due to high competitiveness of

the process, most of the weights are negligible and a few of them is close to the upper bound.

Multiplicative Normalization

It represents an example of normalization with respect to the square of the weights $\sum_i w_i^2$. In this case, the base equation, also called Oja rule, reads:

$$\tau_w \frac{\partial \mathbf{w}^2}{\partial t} = \nu_i \nu_{\mathbf{j}} - \alpha \nu_i^2 \mathbf{w} \tag{2.25}$$

where α is positive. Unlike Subtractive Normalization, this rule is local and it is called multiplicative because of the way the weight term w appears in the right part of the equation.
A similar analysis as the one done for the previous case of normalization yields to verifying the stability of such a rule:

$$\tau_w \frac{\partial |\mathbf{w}^2|}{\partial t} = 2\nu_i^2 \nu_{\mathbf{j}} (1 - \alpha \mathbf{w}) \tag{2.26}$$

from which it is clear that the vector of weights $|\mathbf{w}|^2$ relaxes on the long run to the value $1/\alpha$, which also prevents the weight from growing or decreasing without bounds. Moreover, the competitiveness of the process is preserved.

2.2.4 Intrinsic Plasticity

Plasticity is not a property of synapses only, but Neuron, and in particular Dendrites, have been shown to adapt their internal state, shape and structure in order to either maximize information transfer of a population of neurons or to minimize energy consumption. It is trivial to understand that in response to a slowly varying and almost constant input to a Network of neuron, the response of the Network which maximizes the information of the input is a well spread out distribution from a firing rate of zero to the maximum firing rate of the Neuron. Restating that with concept of statistics, maximum information is transferred/stored when the Probability Density Function of the firing rate of the neurons is an Uniform Distribution from zero to the neuron's upper bound. It is reminded that a neuron's firing rate is bounded between zero and the inverse of the sum of the Action Potential's time and the Refractory time. In a simpler form, if the Action Potential and the refractory period last long, the neuron is then unable to fire many times in a given interval of time.
On the other hand, Energy is also an important quantity to care of when dealing with neuronal systems, which are among the most efficient computation devices. In that view, the best energy management is obtained by the Network when the distribution of the firing rates of the neurons is more concentrated toward low firing rates. Again, in Statistical terms, this corresponds to an exponential distribution. This PDF is observed in neural recordings across multiple organisms such as that from the macaque inferior temporal cortex [24].
These results are not to be attributed solely to synaptic plasticity but are thought to be the consequence of neuronal adaptation to external stimuli and synaptic changes. However, the level of comprehension about such mechanisms at a biological level is lower with respect to the case of the synapse, so that a unified formal mathematica treatment of the phenomena has not been developed. Instead, the general approach is to retrieve the results of experimental data such as [24] with algorithms based on the architecture of the Neuron employed in the specific case. As a matter of fact, depending on the employed neuron model it is possible to developed algorithms which make the parameters of such models adapt according to a determined aim or result. The consequence is that in the literature one finds examples of Intrinsic Plasticity which differ from each other in the way they act on the neuron, but are able to achieve similar results, often aiming to either maximize information

storage or minimizing energy consumption.
The only constraint that are imposed to such algorithms to be plausible with respect to biology is that the changes of neuronal parameters must be based on the state of the neuron under analysis only, as it is observed to happen in biology. From this property of such algorithms the name of Intrinsic Plasticity
A last mention about Plasticity mechanisms is that they are believed to function together in order to allow the Brain to achieve its impressive performance. A single Plasticity mechanism is not enough to allow the Brain to acquire its learning capabilities, nor to function with such high energetic efficiency. Instead, when all the Plasticity mechanisms are acting in the same time, they are able to conveniently combine in order to maximize the perfomances of the Network. In the development of the Thesis, it will be shown that not only this fact is found experiment on biological systems, but it is also supported by simulations of Spiking Neural Networks.

Chapter 3

Neuromorphic Computing

In this chapter, a particular emphasis will be given to the theoretical aspects and hardware implementations that are developed and found in the Laboratory in which I had the opportunity to carry out my Thesis, the Institute of NeuroInformatics, ETH and UZH. Other institutes and groups employ other approaches and solutions, but those will only be cited for completeness at most.
Before diving into the world of Neuromorphic Computing (NC), it is useful to analyze its history and motivation for developing such a subject. NC is born from the conjunction of the need for more efficient hardware for Machine Learning applications and the desire to implement in hardware the founding from Neuroscience for a better understanding of the functioning of the Brain.
The therm Neuromorphic Computing was coined in 1990 by Carver Mead, which originally intended it for indicating all forms of Very Large Scale Integration (VLSI) electronics inspired by the functioning of the Brain. During history, NC has assumed new characteristics and aims, being influenced and inspired by other subjects. As already said, the main groundwork for the development of NC has been Neuroscience, Machine Learning and Electronics (for what concerns the hardware implementation). The first one gave NC the inspiration for the models of Network and components employed in NC machines; the second one offered solutions for practical implementation of learning algorithms able to reach remarkable results; the third one provided the means for the implementation of algorithms with efficient hardware solutions, competitive with conventional electronics and also paved the way for new possibilities in the field of computation.
Moreover, the difficulty in following the path of Moore's law and in turn the slower and more expensive progress in conventional computing systems turn out to be an additional motivation for conceiving new computing paradigms. One of the main obstacles in conventional computers is related to the management of the computed data. In a standard Computing Power Unit (CPU), the processing of the information is performed in a different position respect to where the data - either processed or to be processed - are stored. As a consequence, when a large amount of data is to be computed by the CPU, that has to travel back and forth in the Chip to be processed and saved. This movement of data comes with two main costs: Energy has to be spent in order to send data across the chip, where it is required; time is wasted when the data have to be processed and are not yet available to the Arithmetic Logic Unit (ALU), which performs the logic operations, resulting in limitation of the frequency of operation of the chip. This constraints in the computation are known as "Von-Neumann Bottleneck" and are common to all well-established computing machines, such as personal computers, smartphones, servers, and so on. However, incrementally increasing the rate at which the operations are performed in CPUs, allowed to increase the performance of Integrated Chip (IC) despite their architectural limitations. This process, allowed by the scaling of the transistors, seems to approach its conclusion and other computing paradigms are explored to overcome, for example, the Von-Neumann bottleneck.

Non-Von-Neumann architectures are made possible by the utilization of novel memory concepts. In order to overcome the VN bottleneck, memories have to be embedded with the computational units: advancements in technology allow to conceive new electronic circuits in which memories are co-located with more conventional electronic devices, such as transistors.

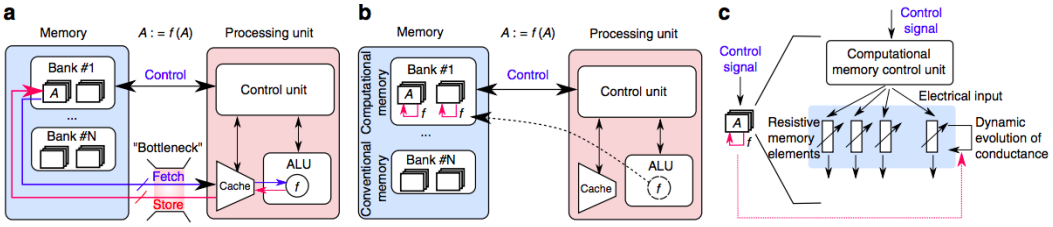


Figure 3.1: a) Schematics of conventional Von Neumann architecture, where data is repetitively sent back and forth from the CPU to the storing memory; b) On the center, a possible example of Non-Von-Neumann architecture: the processing of the data occurs in the same location in which the data are stored in the chip. In this case, by means of the function $f(A)$ which directly acts on the memory unit. c) More common schematics of the computing in NC employing memristive devices: resistances are controlled by a Control Unit and their value can be read and processed on chip

Exploiting parallel computing opens up to multiple advantages with respect to conventional Von Neumann architectures: anyhow, since the speed of computation of the established machines has reached astonishing performance even in commercial devices, it would be reckless to think that Non-Von-Neumann computer could outperform conventional machine in any task. As a matter of fact, Von Neumann computers have become, thanks to the developments in Material Science, Chip Design and Computer Science, very fast in computation in series, where data are processed one next to the other, synchronously. This is particularly convenient when the data to be processed is of small dimension, so that during operation only a small portion of memory has to be written and sent back to the CPU. In that case, series computation is expected to achieve maximum speed.
Anyhow, in other cases the data to be processed may be of larger dimension and the algorithm may require that most of the information of the data has to be updated at every cycle. For this occasions, series computation is not efficient since a huge amount of data should be stored in memories and read by the CPU to be processed every time-step. Locating the memory available to the computing unit in the same place as the processing units solves this problem.
Examples of such cases are most of the Neural Network algorithms, in which the information of a Network - connections, weights, activation of neurons and more - are to be updated every cycle. Of course, implementing a physical model of the Network in a chip seems to be the most straightforward solution, but that presents many challenges in the design process of the Integrated Circuit.
Neuromorphic Computing, as one of its aims, proposes a novel computing paradigm based on parallelism. Moreover, NC comes with additional advantages that derive from either its biological inspiration or the customization for Neural Network algorithms: Energy Consumption, Real-Time operation, Small Footprint in hardware implementation.
At last, a mention is required for AI Accelerators. In the field of Machine Learning and especially for Artificial Neural Networks, algorithms evolve their variables in parallel at every time step. This workflow is not matched by common processing units, which can only compute in series. For this reason, CPUs need to store, write and access a large quantity of data with the limit of the already mentioned Von-Neumann bottleneck, making the computation of the algorithm intrinsically slow. The fact that the update of the variables is generally simple makes the GPU a valid alternative in terms of computational power, exploiting the multi-core architecture and highly hierarchical memory structure. GPUs require however high power for achieving the required performances, thus industries have

developed new dedicated chips for running ANNs algorithms. This Chips, called AI Accelerators, despite being based on parallel Non-Von-Neumann architectures, are not in general belonging to the class of Neuromorphic Computing Chips. Most Deep Learning algorithms are based on Digital Neurons, thus requiring low bit precision for the computation. Such algorithms can be implemented in digital hardware, exploiting conventional Chip Design rules and production paradigms. Such highly customized chips achieve high performances and result to be very energetically efficient. Anyhow, that is not the NC approach, which exploits the notions of Neuroscience to implement biologically plausible neural systems.

Miniaturization

NC is mainly developed with Analog electronic circuits - or Mixed Signal electronics -, contrary to most ICs, so that its design and hardware realizations are challenging. Nonetheless, exploiting memristive crossbar arrays promises to allow to implement large memories in conventional form factors. More conventional implementation of equivalent memories require larger chip dimensions, as for the case of Graphical Processing Units (GPUs).

Energy Consumption

Energy Consumption is probably the main advantage which makes NC approach interesting in many applications which require an analysis of large sets of Data in real time: due to its biologically inspired nature, NC may in future produce powerful machine able to operate at ultra low energy. As a matter of fact, it is estimated that the Brain only consumes some 20W in average, while being able to process data from all its sensory stimuli and performing complex tasks. Being the Networks in NC simplified version of real neural systems, the energy consumption of the electronic circuits with which they are implemented can be optimized.

Real Time operation

In conventional computers, simulating Neural Network is inefficient due to the already mentioned Von Neumann Bottleneck. That is why in simulations one often rely on GPUs, which have a so-called "Near-Memory" architecture reducing the problem of storing and processing large amount of data. Nonetheless, parallel computing is not achieved in GPUs, which require huge amount of power to provide acceptable performance. As a consequence, even though GPUs may provide good enough performance for Real Time application in simplified Neural Networks, they exceed in Energy Consumption and Chip Integration, subsequently being unsuited for In-Sensor operations. NC offers the possibility of achieving the same processing power of GPUs but at a much lower Energetic cost and in Integrated Circuits form-factor. That paves the way for integration of Sensory and Processing units in the same chip, with relevant application in Edge Computing, in the framework of the *Internet of Things*.

3.1 A NC machine: how it is made

Even though NC machines are very different one with respect to the other, most of them are based on the paradigm of *Spiking Neural Networks* (SNNs): this represents a subset of Neural Networks with enhanced biological plausibility in that the operations of the neuron occur in continuous time;

In order to perform such a complex time-continuos - and thus asynchronous - dynamics, NC mainly employs analog or mixed-signal electronics: in particular, it is common practice to utilize conventional VlSI devices, such as CMOSs in sub-threshold design.
For completeness, not all NC are based on asynchronous operations: an example of such an exception is **Loihi** [40], the first Neuromorphic Chip by Intel, exploiting a fully digital architecture and synchronous operations which emulate the dynamics of SNNs.

3.1.1 Neuron and Synapse models

Computation in neural systems can become heavy considering the fact that each neuron implies the calculation of coupled differential equations whose results influence other neuron's state. The result is a complex non-linear system that evolves in parallel: a prohibitive task for simulation in conventional computers, from which the need for dedicated hardware is evident. Moreover, from the point of view of Machine Learning it emerges the complexity in controlling such a system in order to obtain the required results: while units and links in Artificial Neural Networks - such as the ones used in Deep Learning - are regulated by few parameters and low precision digital values, Neuron Models are analog and governed by a high number of parameters. As a consequence, it is challenging to implement effective algorithms for learning. A compromise has to be found between biological plausibility, the computational cost of simulations and control over the learning algorithm by means of the variables of the units.
Of course, there is no uniform choice among the Neuromorphic Community, which is driven by different motivations. Scientists with a strong background and interest in Neuroscience are prone to retain high biological plausibility, despite losing computational efficiency and potential in learning tasks: Networks developed with such a philosophy are mainly used to study the functioning of the Brain in simulations from a dynamical and physiological standpoint, ignoring the learning capabilities and potential. The models of choice are mainly Hodgkin-Huxley or even Multicompartments-based ones and the Networks are composed of a low number of units.
Engineers with a background in Machine Learning or Electronics are more prone to a simplified representation of the Brain, which allows a higher degree of control and reasonable computational efficiency. The choice of the neuron model in these cases often relies on the Integrate-and-Fire, in its multiple realizations. Network architecture is in this case usually more complex and inspired by the concepts of Machine Learning: despite the complications of a continuous-time operation and the non-trivial neuron's dynamics, results in terms of learning capabilities are improving with time and already at remarkable level for simple datasets (TIDIGITs, MNIST).
The approach used in this work is closer to the second one, with Integrate and Fire neurons and complex, recurrent architecture.

Neurons: Leaky Integrate and Fire

It constitute the simpler neuron model in the family of Integrate-and-Fire, being governed by one simple differential equation in the state variable of the Membrane Potential. Current from external stimuli is integrated up to the point in which the Membrane potential reaches a threshold and the neuron fires a spike.

$$\frac{\partial v(t)}{\partial t} = \frac{(v_0 - v)}{\tau} \tag{3.1}$$

In simulations, one also has to specify the values of the Spike, the Refractory period, the Reset potential and the Reference potential v_0.

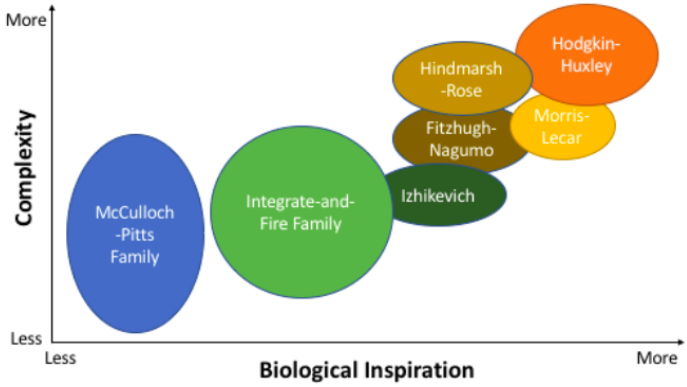


Figure 3.2: Representation of the complexity and quality of different Neuron models with respect to their computational complexity/efficiency, from [25]

Izhikevich

This model represents probably the optimum compromise between biological plausibility and computational complexity in simulations. As a matter of fact, it is able to reproduce advanced behaviors, such as spiking, bursting and adaptation, so that it is not so far from the completeness of Hudgkin-Huxley model in term of reproduced dynamics. However, HH model is much more complex to reproduce both in electronic hardware and to be simulated by computers, making the Izhikevic model a valid alternative for applications. The model comprises two equations, the main of which is of the second order, and 4 variables a, b, c, d heavily affecting the behavior:

$$\frac{\partial v(t)}{\partial t} = 0.4v(t)^2 + 5v(t) + 140 - u + I \tag{3.2}$$

$$\frac{\partial u(t)}{\partial t} = a(bv - u) \tag{3.3}$$

$$\text{if} v > 30mV \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3.4}$$

The 4 variables have different purpose, which can be summarized in this way:

- a: time-constant of the variable $u(t)$. Typical value: 0.02
- b: sensitivity of the variable $u(t)$ to sub-threshold fluctuations. Typical value: 0.02
- c: reset value for the potential $v(t)$. Typical value: -65mV
- d: reset value for the variable $u(t)$. Typical value: 2

Moving the discussion over Synapses, there is less variation on the models employed for the connections. The complex dynamics occurring in synaptic cleft during the release of neurotransmitter in mostly neglected and the models are characterized by ideally analog values - often stored in memories in 32 bit format - which modulate the magnitude of the spikes traveling from the pre-synaptic toward the post-synaptic neuron, resulting in post-synaptic currents. This value, called weight, is ideally not changing sign over time and not bounded in absolute value, nonetheless, in order to avoid synaptic saturation, their mechanisms of plasticity may include upper or lower bounds. The weight is not else than the conductance of the synapse which, ones received the pre-synaptic spike, converts it to a post-synaptic current to feed the post-synaptic neuron. If one were to emulate the mechanism in

detail, the complex dynamics formed by the synaptic vesicles release, Neurotransmitters diffusion in the cleft and acceptance in the post-synaptic membrane should be accounted. Instead, since this process is not know completely and mainly since it is of high complexity, the dynamics of synapse is often highly simplified for applications in NC. Before describing some of the main models employed in simulation, two are the concepts for describing synapse: CUBA and COBA. The first one is a more simplified version, based on current and weights, resembling the linking methods of Machine Learning Neural Networks. CUBA, instead, is a more complex class of synapses which implies the model of the conductances as a function of time: despite the high potential biological plausibility, this often leads to too complex dynamics for implementation both in simulation and in hardware realization.
For what concerns the more complex COBA models, they are applicable in the cases in which both Excitatory and Inhibitory synapses are present in this way:

$$I_{syn} = g_{exc}^{syn}(t)(V_m - E_{exc}^{rev}) + g_{inh}^{syn}(t)(V_m - E_{inh}^{rev}) \tag{3.5}$$

where $E_{exc,inh}^{rev}$ are the excitatory and inhibitory reversing potentials. This equation is then to be inserted in the System comprehending the differential equation regulating the membrane potential of the Neuron. Of course, the complexity of the model lies in the conductances $g_{exc}^{syn}(t)$ and $g_{inh}^{syn}(t)$.
Instead, CUBA models are simpler and more effective in simulating the synapses: as an action potential reaches the synapse, this is integrated and generates a current which is modulated by the weights of the synapse, the main variable parameter of the model.

$$I_{syn} = I_0 w_{i,j} \Theta(t - t_s) \mathcal{K}(t) \tag{3.6}$$

where I_0 is the standard produced current of the synapse to be modulated by the weight $w_{i,j}$ when an action potential comes at the synapse at t_s. Moreover, $\mathcal{K}(t)$ is a temporal kernel which modulated the behavior in time of the synaptic current. Example of those kernel will be shown in the following. As a matter of fact, due to its simplicity, the CUBA is often chosen in NC system both for what concerns the simulations and the hardware implementations.

Synapses: Exponential Kernel

When an action potential reaches the synapse, the synaptic vesicles release their neurotransmitter in the synaptic cleft and those molecules reach the post-synaptic membrane by diffusion. The exponential kernel that the synapse implements aims at modeling the behavior of the neurotransmitter which reach the post-synaptic membrane in time. Of course, being that a simple decaying exponential function, the biological plausibility is still not as coherent as in Multi-compartment models, but those require a much higher computational effort for simulation and hardware implementation.

$$\frac{\partial I_{syn}(t)}{\partial t} = -\frac{I_{syn}(t)}{\tau_{syn}} \tag{3.7}$$

Despite not being the best available model in terms of fidelity to real synapses, the implementation of the exponential kernel offers a minimum degree of biological plausibility and light computation cost. Moreover, it is also flexible as the time constant of the synapse can be adjusted and constitutes a single parameters of the Network, enriching the possibilities of dynamic behaviors at the cost of just one additional tuning parameter.
From the physiology stand-point, this type of Synapse is suited to simulate the AMPA neurotransmitter dynamics, which has a steep increase of concentration in the synaptic cleft during an action potential event and then a slow decrease.

Synapses: Alpha kernel

The main defect of the Exponential Kernel is the non differential point which happens as the synapse is activated by an incoming action potential. This Dirac's Delta is convenient in terms of computational costs but is non-plausible in biology. For that reason, a simple solution to this problem is to modify the behavior of the reaction of the Synaptic Current to the action potential, reshaping it to be a smoother function. The solution is found applying an Alpha function, reported in this equation for the Synaptic current:

$$I_{syn,i}(t) = I_0 \frac{t - t_0}{\tau_{lin}} e^{-\frac{t-t_0}{\tau_{exp}}} \tag{3.8}$$

This equation refers to the case of a single action potential occurring at $t = t_0$: the function is composed by two kernels, which act in an opposed way in the first moment after the input action potential is reached by the synapse. As a matter of fact, for $t \sim t_0$ the behavior is dominated by the linear increase in time which modulates the delta of current, while for $t >> t_0$ the exponential is predominant. As a consequence, still the model involves a Dirac's delta as the action potential arrives at the synapse, but that is modulated so to become a smooth function, which results in an increased biological fidelity.
Moreover, this kernel adds one parameter with respect to the Exponential case, since the linear part is controlled by a time constant τ_{lin}. Even if that may seem an advantage in term of flexibility of the model, it makes it more complicated to be controlled in structured network, so that one often takes $\tau_{lin} = \tau_{exp}$ in order to make the tuning procedure easier.

Synapses: Difference of two exponentials

From the desire of a better description of the dynamics of the GABA and NDMA neurotransmitters, a model of a double exponential function was proposed, such as the following one:

$$I_{syn,i}(t) = I_0 \left(e^{-\frac{t-t_0}{\tau_{decay}}} - e^{-\frac{t-t_0}{\tau_{rise}}} \right) \tag{3.9}$$

As it is clear, the model is composed by two exponentials: one for the Rise of the current and one for the Decay. Also in this case, the Rise exponential aims at smoothening the delta of current occurring at the action potential input. This slow increase of post synaptic current describes the slow diffusion of the Neurotransmitters of the type GABA and NMDA in the synaptic cleft. As per the Alpha function case, this model is characterized by two independent time constant, so that it is difficult to implement in Spiking Neural Network due to the additional tuning parameter. For this reason, it rarely makes it way in complex architecture for Spiking Neural Network aimed at performing a task with high accuracy, even though it is important for the cases in which high fidelity to biology is required.

3.1.2 Learning algorithms

Short Term Depression and Potentiation

The class of Short Term Plasticity mechanisms deals with changes of synaptic features occurring at the order of millisecond to second in time. Considering the time constants of neurons, this can be considered immediate events and thus are not thought to participate in the formation of memory, but are key features for the correct behavior of neural systems. In order to understand them, one has to rely on the physiology of a Synapse to some extent: what happens when an action potential occurs is that the incoming increase in calcium density $[Ca^{2+}]$ triggers the release of synaptic vesicles, retaining

Neurotransmitter, which then are received by the post-synaptic membrane and cause a flow of Ions toward the post-synaptic neuron.
The crucial phenomenon in terms of Short Term Plasticity is the release of the synaptic vesicles: their number is limited as it is also the concentration of Neurotransmitter in the synapse. Once an action potential occurs, most of the vesicles are attracted toward the membrane in order to release their neurotransmitter charge. Nonetheless, of course, only some of them happen to make it through the membrane, depending on the synaptic weight. Still, if the next action potential occurs within a small time window, in which the equilibrium condition is still restoring in the synapse, the vesicles are more easily reversing their charge of Neurotransmitter in the synaptic cleft, thus enhancing the weight of the synapse. This determines a Short Term Potentiation of the synapse, since if the second action potential takes a longer time to occur, then equilibrium is restored in the synapse and the weight of the Neurotransmitter transfer is not modified.
At the same time, if more and more action potentials happen in a small period, the synapse may get short in Neurotransmitter concentration and available vesicles: in this case, the lasts action potentials will generate a lower Neurotransmitter transfer toward the postsynaptic membrane, thus exhibiting lower weight. For such cases, the Plasticity is said Short Term Depression.
Both of the mechanisms are often neglected since they do not fit with simpler synaptic models and their role in learning and memory in the Brain is not well identified, if not evident yet. As a consequence, it is mostly only employed in NC machines aimed at analyzing the Brain rather than at performing well in learning tasks. As a matter of fact though, a simple model for STP and STD exists and it focuses on the probability of a vesicle to transmit its charge of Neurotransmitters through the synaptic cleft $p_{rel}(t)$. The most used model for such a treatment is the Abbott equation, which yields:

$$\frac{\partial p_{rel}(t)}{\partial t} = \frac{p_0 - p(t)}{\tau_{rel}} \tag{3.10}$$

where p_0 is the equilibrium probability of release of a vesicle and τ_{rel} is the relaxation time constant of the probability of release. By imposing a condition on p_{rel} both STP and STD can be achieved:

- $p_{rel} = p_{rel} + f_D(1 - p_{rel})$, $0 < f_D < 1$ for the case of Potentiation
- $p_{rel} = f_E p_{rel}$, $0 < f_E < 1$ for the case of Depression

It is thus clear that, since this model is specific for the case of a single synaptic vesicle, it is not straightforward to integrate it in, for example, the case of I&F neuron with Exponential Kernel synapse. For this reason, this Plasticity mechanism is often not included in simpler neural models.
This discussion may leave the reader with an open question: how do Neurotransmitters end up in a synapse in the case in which flow is mainly mono-directional from a neuron to another? How is the concentration of Neurotransmitters restored?
As already mentioned in the beginning of this paragraph, the Brain posses some cells which have different supporting functions and, despite not participating in the information processing of neural system, allow the components - both Neurons and Synapses - to maintain a correct behavior in time. This cells are called *Astrocytes*, being 20% to 40% of the total of the Glia cells. One of the functions of such cells is thought to be the exchange of Neurotransmitter and Ions with Neurons and Synapse in order to restore equilibrium. As it may be clear, modeling such components of neural system often goes beyond the scope of NC, even though research is ongoing in the more biologically plausible machines in order to include them in the system.

Long Term Plasticity

It is a subset of Plasticity acting over a longer time-scale. The main mechanism in this class is Spiking-Time-Dependent-Plasticity: this plasticity feature is considered the main one occurring in neural systems and it is believed to give the Brain the properties of Memory and the ability of Learning. Moreover, it is also computationally quite efficient, so that it is a wide spread feature to be implemented in the Spiking Neural Network community. For this reason, other mechanisms are often neglected in order not to increase the complexity of the dynamics and the computational efficiency. Other examples of Long Term Plasticity are reported in the previous section about Neural Physiology. Focusing on STDP, the common feature of most of its implementation is that the behavior of the potentiation and depression as a function of the difference in the times of the spikes of two connected neurons is often exponential. In some very simplified models it can also be linear or just a squared function, but since the exponential kernel is simply implemented, the focus is on that case. For what concerns the coefficients which multiply the exponential kernel, there is more diversification, despite the most used one is the simple constant values A_+, A_-.

$$\begin{cases} A_+ \, e^{-\frac{s}{\tau_+}} & \text{for } s > 0 \\ A_- \, e^{\frac{s}{\tau_-}} & \text{for } s < 0 \end{cases} \tag{3.11}$$

in which $s = t_i - t_j$ is the variable encoding the difference in timing of the spikes from the two neurons. This model is simple to be implemented in simulation, less in hardware. However, since this plasticity mechanism is so common and effective, most team working with Neuromorphic Computing hardware implementations have developed Integrated circuit which simulate the behavior of STDP.

3.2 Hardware

NC does not only limits to simulations and software implementation, but it rather also focuses on physical realization of the model with VLSI circuit which allow to conveniently and efficiently run the developed algorithms. This results in the design of dedicated hardware solutions which implement the model in an explicit way, to some extent. One approach is to employ digital electronics to encode the dynamics of simple models, exploiting the established production processes and design rules. In this way it is however harder to capture the dynamics of biological neurons in details, despite the many advantages in terms of flexibility of the Networks for connections possibility, complexity of architectures and compatibility with conventional digital electronics. This approach, is, as already said, closer to the one of AI Accelerator chips, whose aim is parallel computation of Artificial Neural Networks algorithms. Instead, most typical NC machines retain a higher level of biological plausibility and are implemented by Analog and Asynchronous electronics in order to allow for more advanced Neuron and Synapse models to be implemented in the Integrated Circuit. This comes to the expense that such electronics, being less commonly employed, is harder to design and to build in hardware. The analysis in this section will be restricted to such Analog electronics, leaving the Digital cases for NC machines a mention when listing the state-of-the-art NC machines in the following section.

When building a Neural System, one has to select the model of Neuron and Synapse and implement them on an architecture which allows the connections between different neurons in the Network. For what concerns the connections between neurons and also most of the gating variables which control the dynamics of the Neurons and Synapse, those are controlled by conventional digital electronics. This is why NC machines are often in the class of Mixed-Signal electronic system: the neural dynamics evolves thanks to Analog electronics, while the connections, gating variables and external signals are controlled and stored in a conventional digital manner. The interesting part of NC is thus in the

Analog circuits implementing Neurons and Synapse, which is described in this section.
In agreement with the physiology of the Brain briefly described in the preceding section, components in Neural Systems must posses the following functions:

- Synapse block: receives the spikes from other neurons, integrate them and convert the result into current which feed the Membrane potential; moreover, those circuit may comprise some mechanism for implementing Plasticity and/or Adaptation feature. It is generally composed of Integrator and Plasticity sub-blocks.
- Soma block: integrates the input, according to a certain model, emit spikes when the Membrane potential is overcome either in an Analog or Digital (delta-like spikes) fashion, stops its activity for a certain Refractory Period and eventually adapts its excitability according to its activity. In general, it is composed by 4 sub-blocks: Integration, Spike generation, Refractoriness and Adaptation.

Some other block could be added if one were to consider Multi-compartments models, so to implement the extension of dendrites and axons. Nonetheless, those complications of the circuit are neglected in most NC machines. For this reason, they will be neglected in this section.

Design paradigms

For achieving the description of the dynamics of Neurons many strategies can be used: from fully Digital to Mixed-Signal, employing different techniques in electronics. While Digital systems mainly employ strong inversion CMOS and/or Switching Capacitors, Analog circuit make use of sub-threshold MOS device. This comes to be particularly useful both for the power management in NC machines and for biological plausibility: as a matter of fact, the flow of Ions in Neural Systems is mainly due to Diffusion currents, which is the same transport operation regulating sub-threshold CMOS devices. This, together with the well known exponential behavior of the Source-Drain current as a function of the Gate voltage, opens up to the possibility of implementing CMOSs for non-linear behavior blocks and achieve high biological fidelity.

3.2.1 Silicon Neuron and Synapse

In this section, circuits modeling Neurons and Synapses developed at the Institute of NeuroInformatics are reviewed. Both for the case of the Neuron and Synapse, the circuits are developed around the concept of the Double Pair Integrator, based on sub-threshold CMOSs. This DPI circuit, which also gives the name to the whole models, is the input section taking the external currents and integrating them on a Capacitor. Additional circuits take care of the implementation of the spikes, for the case of the Neuron, or the generation of the output current, for the case of the Synapse.
In other institutes, different solutions have been adopted in order to model the constituents of the circuit of the Neuron and Synapse: for a complete analysis of different circuits employed in Neuromorphic Computing, the reader is invited to consider reference [31].

DPI Neuron

The DPI is a generalized Integrate and Fire model of Neuron, meaning that, despite retaining the limitations of a point-like model which does not account for multiple gated Ion Channel dynamics, it is a more biologically plausible model than most other example of I&F neurons. As a matter of fact, the

DPI Neuron is composed of 4 blocks which implement various functions: Integrating the input current from Synapses, Generating Spikes Event with a positive feedback, Resetting the membrane voltage to the Reset value after a spike and Adaptation with respect to the past activity. With reference to Fig.3.3, this blocks are respectively colored in Yellow, Red, Blue and Green.

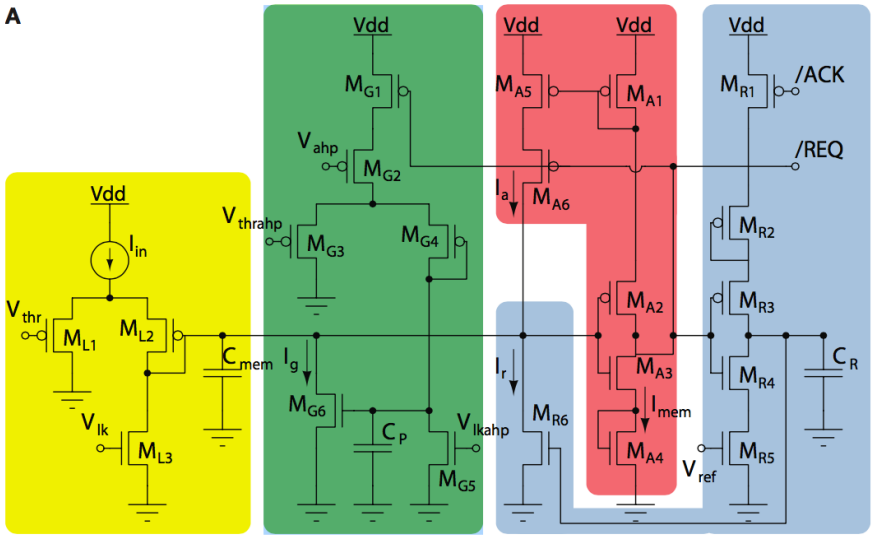


Figure 3.3: Schematics of the Double Pair Integrator based neuron, from [31]. The Yellow block is the DPI, the Red one is for the positive feedback producing the spikes, the Blue one is aimed at resetting the membrane voltage to equilibrium after a spike event, the Green one accounts for the Calcium current and Adaptation.

From a qualitative point of view, the behavior of the circuit can be described in this way, as from [33]. The input current I_{in} is summed to a constant background current (set by V_{ik}) which can be used to model spontaneous activity. Input currents are integrated by the DPI, increasing the membrane voltage V_{mem} over the membrane capacitance C_{mem}. As V_{mem} approaches the switching voltage of the inverting amplifier $M_{A2} - M_{A3}$, the feedback current I_a starts to flow through $M_{A5} - M_{A6}$, increasing V_{mem} more sharply. This positive feedback has the effect of making the amplifier $M_{R3} - M_{R4}$ switch very rapidly, reducing dramatically its power dissipation. This is because when the first inverter output voltage engages the second inverter, the refractory part of the DPI is acts in order to rapidly bring the membrane voltage V_{mem} to ground, by means of the activation of M_{R6}. The refractory period, in which M_{R6} maintains V_{mem} to ground, is defined by means of the capacitor C_R. Earlier, during the spike emission period (occurring when terminal REQ exceeds the voltage threshold), a current with amplitude set by V_{ahp} is sourced into the adaptation section of the DPI neuron, with a gain set by the gate bias voltage V_{thrahp}, and a time constant set by V_{ikahp}. The adaptation current I_g increases with every spike, following the same first-order dynamics of I_{mem}. As a consequence, given the negative-feedback property of I_g, the neuron's response to a step input current is characterized by an initial output firing rate proportional to the input current, which gradually decreases until an equilibrium is reached, thus reproducing the spike-frequency adaptation behavior observed in real neurons.
Following the approach of the *Translinear Principle*, one can get the equation regulating the whole circuit in its sub-threshold dynamics. The Translinear Principle is a rule valid for sub-threshold transistors, stating that:
Depending on the position of a transistor in a closed loop, it can be considered a Clock-Wise or Counter-Clock-Wise element. In a closed loop of Translinear Elements, the multiplication of the Clock-Wise currents equals the multiplication of the Counter-Clock-Wise currents That means that, before the

positive feedback is activated, generating the spikes, the circuit behaves according to the following equation. The derivation of the equation thanks to the Translinear Principle is not performed.

$$\tau \frac{dI_{mem}}{dt} = -I_{mem}\left(1 + \frac{I_g}{I_\tau}\right) + I_{mem,\infty} + f(I_{mem}) \tag{3.12}$$

$$\tau_g \frac{dI_g}{dt} = -I_g + I_{g,max} r(t) \tag{3.13}$$

In this equation, the non linear term $f(I_{mem}) = \frac{I_a}{I_\tau}(I_{mem} + I_{th})$ depends on both the membrane current and the positive feedback current. Instead, the two time-constants τ, τ_g and the two currents $I_{mem,\infty}$,$I_{g,max}$ are defined accordingly:

$$\tau = \frac{CV_t}{\kappa I_\tau} \qquad \tau_g = \frac{CV_t}{\kappa I_\tau}$$

$$I_\tau = I_0 e^{\frac{\kappa V_{ik}}{V_t}} \qquad I_\tau = I_0 e^{\frac{\kappa V_{ik}}{V_{ikaph}}}$$

$$I_{mem,\infty} = \frac{I_{in}}{I_\tau} e^{\frac{\kappa V_{thr}}{V_t}} \qquad I_{g,max} = \frac{I_{MG2}}{I_{\tau,g}} e^{\frac{\kappa V_{thrap}}{V_t}}$$

Of course, such a rich dynamics is required for reproducing the behavior of the Neuron in an accurate manner: this neuron has been shown to be able to reproduce successfully not only the adaptation properties it implements but, by means of the tuning of the time-constants τ and τ_g, the refractory period and the adaptation parameters, it is possible to differentiate its functioning from regular spiking to burst activity. Such flexibility makes it a reference in Neuromorphic Computing, being appealing both for more biologically plausible applications and for the ease of tuning for Machine Learning tasks. Finally, a table with some interesting parameters from the DPI Neuron is reported. This values are taken form the reference [33] in which the transistors are assumed to be in the $0.35\mu m$ node, with a power supply voltage of 3.3V. This represents an older generation technological node, still producing interesting results in terms of power consumption, yet with some limitations in terms of integrability in ICs.

DPI Neuron parameters	
Membrane Capacitance, Area	100 μm^2
Membrane Capacitance (F)	0.5 pF
Neuron layout, Area	913 μm^2
Supply Voltage	3.3 V
Power Consumption/spike (300ns pulse)	7 pJ
Power Consumption/spike (100ms , including integration phase)	267 pJ

Table 3.1: Main parameters of the DPI Neuron

Of course, more modern technological processes and advancement in the implementation of Integrated circuits allows for better performance both for the case of Energy consumption and Chip Area. For example, this is not the technological node with which the DYNAPs chip was based on and the actual transistor size selected for that case allows to improve the number listed above, which were produced so to prove the feasibility and success of such a Circuit even with outdated technology. In the following, the parameters related to the DYNAPs circuit will be presented when analyzing NC Networks.

DPI Synapse

The Double Pair Integrator has application in Synapses also. The same chip DYNAPs employees synaptic circuit of this kind in order to integrate the voltage spikes coming from the output of Neurons. The conversion from voltage spike to output current is modulated by additional circuit which implement the weight that the biological synapses apply to their input in order to produce the output stimuli. This weight is not fixed in time in most cases, it rather varies with the rules of Plasticity: it is possible to reproduce those rules by means of additional electronics circuits.
As for the DPI Neuron, the Synaptic counterpart is an example of high biologically inspired circuit, possessing peculiar features which are not common in Integrate-and-Fire based Networks. For example, the DPI Synapse exhibits Short Term Plasticity as well as Spiking Time Dependent Plasticity and also, to some extent, NMDA voltage gated channel features.
The following figure represents the main building block of the DPI synapse, even though the circuit implementing STDP is not shown for simplicity. As a matter of fact, that specific circuit is more complicated than the rest and is composed of multiple subparts which accounts for various features in modulation of the Plasticity mechanism. The interested reader is invited in consulting [32] for more details about that circuit.

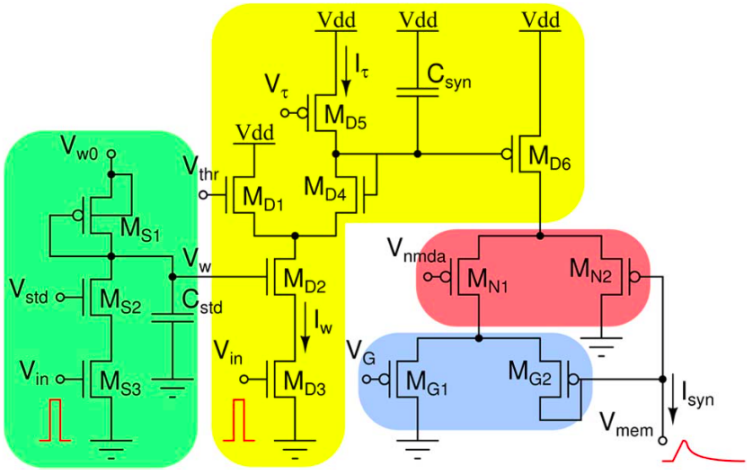


Figure 3.4: Schematics of the Double Pair Integrator based excitatory Synapse, from [32]. The DPI is shown in Yellow, while the Short Term Plasticity is implemented by the Green block. The Red and Blue block implement the NMDA and Conductance-based plasticity features.

Since most of the transistor operate in sub-threshold, it is still possible to make use of the Translinear Principle in order to study this circuit. The detailed analysis will not be reported, but focusing on the Short Term Plasticity circuit and restricting the operation to the case of the bias V_{thr} of transistor MD_1 determining the condition $I_{th} >> I_{syn}$, then the equation regulating the Synaptic current is:

$$\tau \frac{dI_{syn}}{dt} + \frac{I_{syn}^2}{I_{th}} - I_{syn}\left(\frac{I_w}{I_\tau} + 1\right) = 0 \tag{3.14}$$

which can be further simplified to yield:

$$\tau \frac{dI_{syn}}{dt} = I_{syn}\left(\frac{I_w}{I_\tau} + 1\right) \tag{3.15}$$

This equation states that the response of the Synaptic current increases every time an input arrives, meaning that, as I_{syn} get larger, the every synaptic current response also becomes larger as long as the condition $I_{th} >> I_{syn}$ is satisfied. This mechanism is peculiar of Short Term Facilitation.
Instead, Short Term Depression is implemented by the Green Block ($M_{S1} - M_{S3}$): the input spikes allow a current to flow in transistors $M_{S2} - M_{S3}$, controlled by the bias V_{std}. This current lowers the potential V_w determining a lower synaptic current response as a lot of spikes come in the Synapse in a short period. On the other hand, if no spike arrives for a long time, the voltage V_w is restored to its original value V_{w0} by transistor M_{S1}.

3.2.2 NC Networks

In NC machines Neurons and Synapse are vaguely arranged in a biologically inspired manner: the Brain is however composed of many sub-systems - often said lobes - in which the architectures of the Networks are different and serve particular purposes. For example, the Visual Cortex is thought to be composed of different layers which are aimed at detecting particular patterns, forming an hierarchy of levels which have inspired the creation of the concept of Convolutional Neural Network. Other areas, such as the Cerebellum, exhibit more recurrent connections, from which the similarity to Liquid State Machines [34].
As it is clear, the requirement for general purpose NC machines is that they have to allow the configuration of many different kinds of architectures, maintaining the performances of the components, Neurons and Synapses, unaltered. That is why the connections are often realized by means of conventional - and reliable - digital electronics, which offers fast, low noise connections. That is why despite the Neurons and Synapse being implemented in fully Analog electronics such NC machines are said to belong to the Mixed-Signal class.
In order to allow for flexibility in the connection capabilities, Synapses are commonly arranged in matrices while neurons are placed at the end of each row of such matrices. Since the output signal of the Neurons can be brought to input to all the columns of the matrix, and since the Neurons can in principle gather the signal coming from all the synapses in their row, then full connectivity can be achieved. Of course, such method allows also sparser Networks to be built. The connections are controlled by CMOS in saturation operation, either switched On or Off. An example of a single row of such Matrices is shown in the figure below.

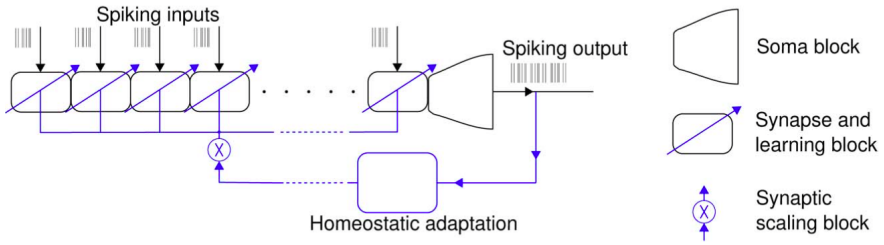


Figure 3.5: Schematics of the input , from [33]. This scheme shows the input coming from other neurons in the network being processed by synapses and fed into a single Neuron. A similar treatment applies for all the other neurons in the network

Not shown in the image, the output of the Neuron is led on the top of the matrix, where a CMOS transistor determines whether the signal is sent to the column of the matrix. Moreover, each synapse has an input transistor for completing the possibility to select the individual connection between two neurons.

3.2.3 Problems of Analog VLSI: Mismatch, Von-Neumann bottleneck

One of the main problems in Analog Integrated Circuits, in which CMOSs are operating in sub-threshold, is the device-to-device mismatch: each transistor in the Chip is heavily impacted by small changes in its size, bias and temperature of operation. As a consequence, it is possible to have different neurons behaving differently when stimulated with the same input. Despite it is believed that also biological neurons exhibit slightly different characteristics and variability between each other, mismatch is a detrimental feature in Integrated Circuit: it means that the yet complex non-linear systems created by large networks of Neurons behave in an unpredictable manner due to the non-ideality of their components. Implementing learning algorithms in such systems is a prohibitive task and that represents one of the main limitations of Analog devices for Neuromorphic Computing; this problem is not present in AI accelerators implementing Machine Learning algorithms carried out in digital reliable hardware VLSI devices, such as GPUs. This gap in reliability of the implementation of the algorithm is to be filled in two ways: one can either try to improve the hardware through design paradigms which reduce the effect of Mismatch or the alternative is to exploit biologically inspired features of the Brain which are believed to potentially not only limit the effect of variability across the Neuron population, but also bring additional benefits in the computation capabilities of the machines. The electronic design approach is commonly not employed in that it has been demonstrated to often lead to a larger area occupied by the optimized Neuron circuits, meaning that the area of the chip has to be enlarged in order to contain the same number of neurons. Instead, there is a rising interest in developing algorithms for the implementation of biological features which reduce the effect of mismatch. For example, Adaptation in Neurons is a local form of plasticity which allows each neuron to change its excitability to decrease their firing rate when highly stimulated, thus favoring the power management in the whole chip. Moreover, more advanced Plasticity mechanisms are to be developed to achieve performance in energy consumption and robustness to variability in the Networks comparable to those observed in the Brain. This Thesis, for example, goes in that direction: Intrinsic Plasticity and Synaptic Normalization are combined with a more conventional form of Synaptic Plasticity such as STDP.
The other great limitation of conventional electronic ICs is the Von-Neumann bottleneck. In NC machines this is caused by the parameters and biases of the Neurons and Synapses being stored in memories which are located away from the position of the element them-self. As a consequence, whenever those biases have to be modified, because of a local learning algorithm, for example, the update has to involve a transfer of information from the register containing the value of the bias to the Neuron or Synapse circuit. This results in additional consumed power.

Memristors: solution to both problems

Fortunately, the solution seems to be common to both the problems. Advancements in Non-volatile memories allowed to develop new electronic devices which function as programmable resistors, an thus memories. These devices are in the family of Memristors and may be realized with different methods, material and production processed. The main classes of Memristors are Phase Change Memories (PCM), Valence Change Memories (VCM), Electro-Chemical Memories (ECM) and Spin Torque transfer Memories (STT-MRAM). Research is ongoing in this field of study so that new concepts are generated very often. Most of these devices are easily integrated into ICs and can coexist with the circuits of Neurons and Synapses presented before. Their most useful application is when they are used in order to modulate the biases of the Neuron and Synapse circuits. Utilizing 1T1R structures (one transistor operating a programmable resistance), the Memristors can locally modify the value of the bias: ideally, it is then possible to operate on the biases at the Neuron level in a very energetically efficient manner. The operation of Memristors can then be programmed to overcome the

problem of mismatch or to implement local learning algorithms.

3.3 State-of-the-art NC machines

Many different implementations of NC have been developed following diverse design paradigms and architecture at the chip level. Most of them aim at being a general-purpose platform for Spiking Neural Network algorithms, even though constraints may arise from design rules and limitations in the electronics. For example, in the previous section it has been underlined the problem of Mismatch, affecting most of the Analog designed circuits: that problem imposes certain limitations in the construction of the Network which has to tolerate deviations from ideal behavior of its components. Instead, for Digital systems, the challenge is to simulate time-continuous processes with clock-based electronics.
Indeed, each of the machines presents some advantages and disadvantages: describing all such systems in detail is interesting but beyond the scope of this section. In the following, the NC machine available at the Institute of Neuroinformatics, the DYNAPs will be described and other NC implementations will be mentioned.

DYNAP

The DYNAP is a family of chips designed by the team headed by Giacomo Indiveri, INI (ETH and UZH), on which the main part of the research in that group is based. Moreover, the chips are further developed and customized for the application by Indiveri's Startup "aiCTX" [41]: DYNAP-CNN, DYNAP-SE2, DYNAP-SEL and others. The idea behind such a family of Integrated Circuit is to implement highly biologically plausible systems for the simulation of neural systems in benchmarks and standard Deep Learning applications, in the field of Spiking Neural Network. As a matter of fact, DYNAP chips differentiate with respect to conventional Deep Learning Accelerator by the fact that the processing involves trains of spikes and their relative timing, instead of clock-based floating-point computation. For this reason, this processors are to be coupled with Event-Driven sensor or conventional sensor with and Encoder: in the example of a camera, the input to be presented to the DYNAP is the change of illumination of each pixel, triggering a spike after a certain change in illumination has occurred, rather than the clock-based evolution of its values in time; a similar approach is to be applied to any source of data.

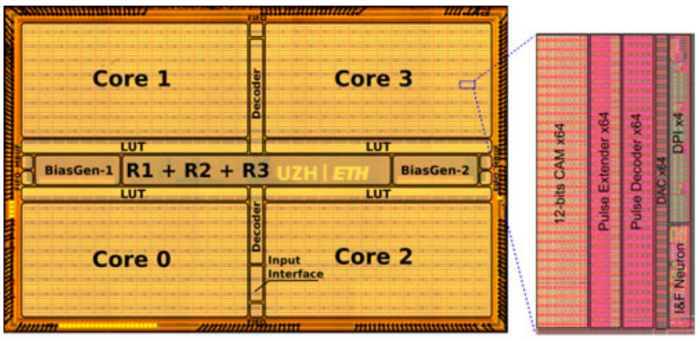


Figure 3.6: Image representing the basic Architecture of the DYNAP chip family: 1024 Neurons and 64k Synapses are distributed in 4 cores which are controlled by a Routing System (R1,R2,R3). Within each core the Biases are shared, those being stored in the Bias-Generator registers. Image from [37]

For what concerns the nodes and connections implemented in the Chip, they have been presented in the previous section, being of the DPI family: this Neurons and Synapses assure high fidelity to biological counterparts and are conveniently implemented in hardware both for what concerns Area of integration and consumed power during functioning. Nonetheless, the sub-threshold operation of the transistors in the circuit suffers from the Mismatch problem, to be solved by the Design of the Network for the particular application case.
DYNAPs chips are also innovative for what concerns the management of connections and routing of the spike events: delivering the output spikes of a certain neuron to the correct output neuron, possibly placed far away in the chip, without generating long delays and with an efficient memory management for the connection addresses represents a critical operation in the functioning of any Asynchronous system. For this task, and Address-Event-Representation (AER) routing system is required: any node of the network is assigned an address which is used to send and receive the spikes between neurons. If this system is not efficient enough, the neuron loses the ability to send outputs to other nodes and the design of the Network gets constrained by such hardware limitations, possibly resulting in worse-performing algorithms. The most basic AER system would assign each neuron a tag from 1 to N (number of neurons), encoded in $log_2(N)$ bits, for a total of $FNlog_2(N)$ bits for the connectivity memory, in the case of average Fan-out of F. Not only this is very wasteful in terms of memory, but, being the routing memory often implemented by an external SRAM module, this method encounters the limitation of the Von-Neumann bottleneck. DYNAP family, instead, was designed with the intention of making this routing process more efficient by dividing the Neurons into Clusters with number of neuron C, each of which holds the same tags shared by all the clusters (neurons tagged from 1 to K). This allows to break the routing of the spikes into a two-phases process: first, the cluster is selected, then the tag of each neuron determines whether that neuron will receive the input of not. With this method, the total memory allocated for the routing operation is:

$$MEM_{tot} = MEM_{source} + MEM_{target} \tag{3.16}$$
$$= \frac{F}{M} log_2\left(\frac{KN}{C}\right) + \frac{KM}{C} log_2(K) \tag{3.17}$$

Optimizing this value with respect to the number of Fan-out clusters M, one gets the optimal memory per neuron to be:

$$MEM_i = 2\sqrt{Flog_2(C)log_2(N)} \tag{3.18}$$
$$M_{opt} = \frac{\sqrt{Flog_2(\alpha N)}}{\alpha log_2(\alpha C)} \tag{3.19}$$

where $\alpha = K/C$.
In the chip, the routing system is hierarchically divided into three parts: R1 takes care of local connections within each of the 4 main cores; when events relate neurons belonging to different cores, R2 is called into action; finally R3 regulates the highest-level communication paths. Moreover, R1 is responsible for the storing of the *Source Memory* in SRAM cells, while the *Target Memory* is located in CAM cells.
One of the major drawbacks of the DYNAP family is the distribution of the Biases over the 1024 Neurons. The chips have 4 cores, each with an even number of Neurons and Synapses: each Neuron and Synapse in a core share the same biases. Because of the problem of Mismatch affecting the ICs of the Neurons and Synapses, and since the biases cannot be tailored individually in order to overcome this problem, the units in DYNAP chips suffer from Mismatch. Besides, it is not possible to operate a learning algorithm that involves the local operation of such biases. For example, the Plasticity mechanism concerning Adaptation or the Input section of the Neuron cannot be implemented.

Further details related to the main parameters of the implementation of the DYNAP chip family are gathered in the table 3.2 .

DPI Technological parameters	
Process Technology	0.18 μm 1P6M
Supply Voltage - core	[1.3 - 1.8] V
Die Size	43.79 mm^2
Number of Neurons	1k
Number of Synapses	64k
Total Memory	64k CAM + 4k SRAM

Table 3.2: Parameters related to the technological implementation of the chip DYNAPSe, as from [37]

SpiNNaker

The University of Manchester [38] has realized its own Neuromorphic Computing machines employing conventional VLSI digital electronics. The machine is actually composed of an ensemble of chips, each of which is an ARM9 core with its dedicated memory. Despite the presence of some local memories in the machine, there is no global memory accessible by all the processing units, thus making this computational paradigm fall in the Near-Memory computing class. Units in the machine communicate with high bandwidth fabric, optimized for small messages (max 72 bits). As a matter of fact, the communication between the chips, realized with Address-Event Representation (AER) communication protocol, is the only limitation to the size of the machines, since the developers build the systems to make modular and capable of hosting up to 576000 ARM9 cores. Routing all such processing units and managing to control the system in real-time is the challenge of such a project. Such complication in the size and connectivity of the machine come to the advantage of implementing any possible Network architecture, even at large scale. For what concerns the realization of the Neural components, they can be selected to be either Leaky-Integrate-and-Fire (LIF) or of the Izhikevich's model. Moreover, they can receive up to 1000 connections as input. The system is controlled by dedicated software, based on C++ and Python languages.
The main advantage of such a machine is its immense size compared to most NC systems, allowing for advanced architecture to run in real-time. Instead, one can debate that its hierarchical structure for the memory management does not allow the machine to overcome the well-known Von-Neumann architecture paradigm, making SpiNNaker suffer for the problem of the Von-Neumann bottleneck.

TrueNorth

TrueNorth is the NC machine built by IBM aiming to overcome Von-Neuman architecture for Spiking Neural Network. From the point of view of the electronic design paradigm, TrueNorth is realized with digital - thus saturated - 28nm CMOS process, with a fully asynchronous, event-driven method. Nonetheless, neurons are simulated with a clock of 1ms and displaced in the 4096 cores present in the machine. Each core contains 256 virtual LIF neurons - where the term "virtual" indicates the fact that one only neuron can simulate the activity of 256 other neurons - connected by 256·256 synapses. For what concerns connectivity, the routing system in TrueNorth allows each virtual-neuron to connect to up to 256 other neurons in a destination core: as a matter of fact, the options for connectivity are more restrictive than the ones in SpiNNaker, still the routing system is quite advanced. Synapses are built to take three values: Inhibitory, Weak Excitatory and Strong Excitatory.
The main problems related to this machine are that it occupies a large area - $4.3cm^2$ - and, most

importantly, it does not integrate any learning-on-chip or Plasticity mechanism. While the problem of the chip dimension could possibly be tackled by exploiting more advanced production processes and novel technologies, such as Programmables Memories crossbars arrays, the lack of Plasticity/Learning capabilities on-chip represents a choice in the phase of design. On one side, it makes the circuit required for the model of Neuron and Synapse way simpler both for the design and operation standpoints: the chip is simulated with CPU/GPU off-chip, in order to improve the performance and optimization of the network, and the resulting system is transferred to the chip. On the other side, this procedure and design choice exclude the possibility of performing on-chip learning, thus to make the chip directly operate with sensors, without a previous training session off-chip.

NeuroGrid

It represents the NC machines designed at Stanford by Prof. Boahen [39]. As much as SpiNNaker and TrueNorth, NeuroGrid is designed to run large Spiking Neural Network in real time with efficient energy management. Unlike most of the NC machines, its Neurons and Synapse are implemented by means of Analog Electronics, while the routing fabric is fully Digital, thus constituting a Mixed-Signal machine. Moreover, NeuroGrid implements advanced Integrate-and-Fire neurons and synapses, so that the dynamics of its Networks is relevant also for biologically inspired/aimed analysis. This is allowed by the exploitation of sub-threshold CMOS, which has the advantage of allowing real-time low power operation.
The main downsides of such project are the fact that the connections between neuron are constrained by the Von-Neumann bottleneck and, commonly to most SNN-based, biologically inspired devices, optimizing the training algorithm is a complex task to perform.

Loihi

As one of the leading companies in the Integrated Circuit technology solutions, Intel has developed its first NC chip [40]: it is called Loihi and it was launched in 2008. The chip is based on a 14nm FinFET process, comprehending over 2 billion transistors and 33MB of memory, all in the size of $60mmm^2$, being one of the most compact implementations of NC machines. The neurons are grouped in 128 cores and in each of those they are displayed in a tree-like manner. Also, the chip contains 3 x86 processing units devoted to routing and controlling the communication between the cores. Together with its compact size, Loihi is advanced also for what concerns energy management, consuming only $15pJ$ per synaptic operation, and the computational power, being able to perform 30 billion synaptic operations per second. The chip is a promising solution for embedded system in which NC machines are directly interfaced to sensors given also its ability to perform On-Chip learning, mainly through STDP.

	DYNAPSe	TrueNorth	NeuroGrid	SpiNNaker	Loihi
Producer	INI (ETH/UZH)	IBM	Stanford	University of Manchester	Intel
Process Technology	Mixed-Signal 180nm	Digital (ASIC) 24nm	Mixed-Signal 180nm	Digital (ARM) 130nm	Digital (ASIC) 14nm
Die Size	43.79 mm^2	4.3 cm^2 (chip size)	165 mm^2	102 mm^2 per die	60 mm^2
Number of Neurons	1024	1M	64k per core 16 cores	1k per core 1M cores	130k
Number of Synapses	64k	265M	375M	All-to-all connectivity	130M
Total Memory	64kB CAM + 4kB SRAM	12.75kB per core 4096 cores	-	-	2Mb SRAM per core 128 cores + 16MB SRAM
Energy per hop	17pJ @ Vdd = 1.3	2.3pJ @ Vdd= 0.77	14pJ	1.1nJ	15pJ

Table 3.3: Summary of the main features of the mentioned NC machines. The digital ones allow for higher scalability of Networks, but have limitations in the complexity of the units (Neurons and Synapses) and in Energy management. Mixed-Signal based devices are excellent in real-time operation with ultra low power consumption

The table above sums up the main characteristics of the NC machines analyzed up to now: as it is clear, by comparison of the number of Neuron and Synapses, Digital systems allow to integrate a larger number of components, resulting in the capability of implementing larger Neural Networks. On the other hand, this approach is less detailed in providing accurate Neural and Synaptic dynamics: as a matter of fact, Digital NC machines can be viewed as a development of conventional hardware accelerators for Machine Learning toward more biologically inspired ICs, so that their degree of biological plausibility is generally low.
Instead, Mixed-Signal electronics offer the ability to operate at ultra low power, emulating complex neural behavior: this makes them potentially more attractive to be coupled with sensors in low-power Systems-on-chip, such as wearable devices. Nonetheless, this design paradigm comes with the costs of a larger chip area per circuital element - Neuron and Synapse - leading to limitations in the complexity of allowed Networks. Moreover, their related algorithms, based on Spiking Neural Networks are more difficult to be set up and generally produce worse results than their ANN-based counterparts. Important efforts are being done in research in the NC field for improving the capabilities of Mixed-Signal devices, also from the point of view of algorithms, to exploit the capabilities of such interesting systems.

3.3.1 Softwares

Simulating Spiking Neural Network is a complex task since it involves the solution of several coupled differential equations evolving in continuous time. For this reason, some platforms have been created in order to build libraries containing pre-built models for the Neurons, Synapses and Network Architectures. Moreover, those libraries also often include routines able to calculate the complicated system of equations for the neurons, leaving the programmer the selection of the model for the components of the system and the architecture of the Network. This allowed to open up the world of NC to scientists less experienced in Computer Sciences, favoring the convergence of experts from multiple disciplines to work on Neuromorphic Computing.
One such Library is called Brian2 [35] and it is open-source: it provides the user different options in terms of Neuron and Synapse models and the possibility to write their constitutive equations in an

intuitive syntax, solving them automatically. Like most of the Machine Learning Libraries, Brian2 is written in Python and compiled with either the Numpy or Cython suit for more efficient performances. Other python packages for implementing SNNs in software are NEST, ANNarchy, NEURON and Genesis.

Chapter 4

Liquid State Machine

4.1 LSM: Definition and Characterization

In a famous paper from Maass [28] the framework of an important computing paradigm was delineated: Liquid State Machines (LSMs). To give that a context, LSMs belong to the field of Recurrent Neural Network and more specifically of Reservoir Computing (RC), which is defined in this way:

An RC system is composed of 3 main components: an Input layer, a Reservoir and an Output Layer, also called Readout. Connections in the reservoir are recurrent and the training of the weight of the connections is performed on the Readout weights only. The units of the Reservoir are non linear.

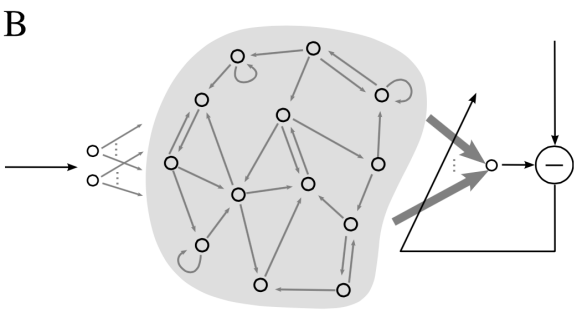


Figure 4.1: Schematics of the architecture of Liquid State Machine: an Input is projected in a Recurrently connected Reservoir which in turn projects it self on the Readout layer. The latter connections are the only ones to be eventually trained in a supervised fashion

While the Echo State Machine (ESM), the other great branch of Reservoir Computing, belongs to the framework of Artificial Neural Networks and operates in discrete time, LSMs are governed by continuos time and are implemented by Spiking Neural Networks. For this reason, ESMs are not further investigated and the analysis focuses on LSMs.
In order to map an input, function of time $u(\cdot)$, on the desired output function $y(\cdot)$, a LSM generates a "Liquid State" $x^M(t)$, representing the response of the Network to the preceding inputs in time $u(s)$, where $s < t$. The Liquid State is a continuous-time function assuming analog values, containing the information from all the units in the Dynamical System, formed by the recurrent Neuron group. It is reminded that the order of the connection in the Recurrent group is not known nor established a priori, according to the principles of Reservoir Computing.
From a mathematical point of view, a Liquid State is the output of an operator L^M that maps the

response of the Liquid to the input $u(\cdot)$, and then projects it to the output $y(\cdot)$:

$$x^M = (L^M(u))(t) \tag{4.1}$$

The operator L^M is often referred to as "Liquid Filter" or "Liquid Circuit". The other important part of a LSM is the readout, which allows extrapolating the information from the high dimensional Liquid Circuit. That is implemented by a function f^M:

$$y(t) = f^M(x^M(t)) \tag{4.2}$$

The readout function is built specifically for the requested task, and more than one readout modules can operate in parallel mapping the Liquid State. It is then clear that the Liquid State is not tuned nor influenced by external intervention, but it is a complex dynamical state which can assume time continuos analog values and has large enough dimensionality and non-linearity so that it is able to store and process the information from the input, conserving a portion of its past evolution.
The main two attributes to be evaluated in LSMs are: Separation properties (SP) and Approximation property (AP). SP deals with the ability of the Liquid State to produce different trajectories in time when stimulated by different inputs. AP, instead, refers to the readout and in particular to its capability of producing different outputs based on different states of the Liquid Circuit. Both are fundamental for the correct learning of the mapping function from input to output, making the mapping function being able to generalize with respect to different inputs and accurate in the classification or regression procedure of the readout.
Because of its generality with respect to the input type and due to the high level of customizability of the readout function, LSM is a powerful machine for computation: in [28] it is demonstrated that LSM can be considered as a class of machines that have *universal power for computations with fading memory on functions of time*, following the fact that they possess both *Fading Memory* and *Time invariance* properties.

4.2 Training in Liquid State Machines

Training is typically attributed to the readout layer of weights in Reservoir Computing: as a matter of fact, the output connections act as a classifier with respect to the Liquid State, which contains the processed information of the input. Due to the recurrent connections and the complex behavior of the nodes, the Liquid is capable to perform non-linear operations and project the input in a higher-dimensional space. Exploiting this processed representation of the input, the readout is simply given the task of classifying the different behaviors of the Liquid. There are several ways in which the output layer can be trained, depending on the type of application. In principle, both "on-line" and "Batch-mode" training can be applied: for the first case, the weights are regularly updated following learning rules based on the Liquid State; instead, Batch mode learning refers to the computation of the weights based on the whole history of the Liquid State after the presentation of a data-set.
Nonetheless, Reservoir Computing has evolved from its formal definition and it has been shown that the performance of the LSMs could be improved if some policies were applied to the randomly generated recurrent connections. First, some general rule has been shown to improve the performance, such as the control over the Spectral Radius of the weights in the Reservoir in Echo-State-Machines [42] or the generation of the connections based on the distance between nodes [47]. Also, the effect of unsupervised learning of recurrent connections has been investigated: mainly STDP has been studied as a plasticity mechanism able to improve the performance of the reservoir, but other plasticity mechanisms are potentially helpful in that perspective.
Of course, however, the main effective part of the Training of LSM deals with the output weights. A

first difficulty in performing such procedure derives from the nature of the Neurons in SNNs: while with ANNs the evaluation of the error with supervised techniques is feasible, thanks to the simple node's activation functions, and leads to the Back-Propagation training policy, applying the same procedure to SNNs is far more complex. The behavior of Neurons is way more non-linear and evolves in continuos time. Moreover, the activation function of LIF neurons is non-differentiable, being a delta function. As a consequence, standard Gradient Descend based approaches are not applicable or efficient in LSMs.
As it often happens for SNNs, one has to find some expedient in order to perform learning. One has to extract information from the spike trains produced by the neurons. Mainly, there are two policies:

- **Rate Coding:** it is based on the frequency of the Neurons, also said Firing Rate. Synaptic weights are adjusted based on the rates of the neurons rather than for particular spikes patterns or timing. It is a robust decoding of information from spike trains, especially against noise, but it neglects small temporal features. Evidence of such mechanism was first found in the nervous system of muscles [48] and lately in cortical areas in the Brain. Nonetheless, it is believed that such a neural coding paradigm lacks in computational power and misses some of the key information in spike trains. As a consequence, it is thought not to be the main mechanism carrying information in the Brain [49].
- **Time Coding:** it is based on the precise timing of each spike. Noise is thought to contain a crucial part of the information in spike trains. In that way, the time features extracted by the activity of neurons are on the ms time-scale, rather than on $100ms$ as in Rate Coding. Despite being proposed to be the main coding mechanism explaining Brain's information transfer and processing [50], it is highly challenging to implement algorithms based on Time-Coding for Neural Networks due to their high sensitivity with respect to noise.

Due to the difficulty in finding suited algorithms and high noise sensitivity of Time Coding, most SNNs are based on Rate Coding. That said, the performance of Rate Coding is heavily affected by the way the frequency of the Neuron is calculated. As a matter of fact, spike trains are all but regularly timed events, so the firing rate has to be approximated. There are two main definitions of firing rates for SNNs applications: **Spike Counts** and **Low pass filter of Spikes**.
The first definition of firing rate is based on the counts of the spikes in fixed intervals. Given an interval size, more commonly on the $100ms$ order of magnitude, the number of spikes is counted and the frequency is obtained by dividing by the interval length.

$$FR_i = \frac{\sum_{t_i}^{t_{i+1}} \delta(t - t_{spike})}{t_{i+1} - t_i} \tag{4.3}$$

The rate along time is given by the discrete succession of the spike-count values. Of course, the small temporal features at the ms-scale are lost due to the size of the time interval. One may then think to lower the binning size, however that generally lead to highly noisy signals with large fluctuations of the frequency of neurons. Noise is crucially detrimental with respect to most common rate-based learning procedures, such as Linear and Logistic Regression. In order to find a compromise between the stability of such learning algorithms and the amount of information extracted by the decoding, interval sizes of the order of $100ms$ are typically preferred. An example of such a decoding of information from a train of spikes is given in Fig.4.2. In this case, a $100ms$ is chosen to establish the frequency of the spike train. As it is clear, the time-binned frequency is able to capture the overall behavior of the Poisson distribution of spikes, but it lacks the small feature at the ms time scale. Also, since the time-interval is fixed, the available values of frequency are given by $f_k = k/t_{interval}$. As a result, the decoding is limited both to the time-scale and in the frequency conversion: nonetheless,

this compression of the information produces a signal which is favorable for more common learning algorithms, such as Linear and Logistic Regression.
In the second case, the spikes are treated as delta functions, increasing by one unit the value of a Firing Rate function defined on the continuous-time. The function implements, in general, a convolution operation of the spikes with an exponential decaying kernel.

$$FR(t) = \frac{\sum \delta(t - t_{spike})}{\tau} e^{\frac{t - t_{spike}}{\tau}} \tag{4.4}$$

In that way, the frequency is estimated by means of the time-constant of the exponential decay: with low values, the small temporal features of the firing rate are captured, leading however to a more noisy function; with high time-constant, a more stable yet less accurate representation of the firing rate is obtained.
Such method recalls the functioning of the Calcium current produced when an action potential occurs in a Neurons. Ion Channels control the flow of the Calcium Ions based on the voltage of the Membrane of the Neurons, thus called Voltage-Gated-Ion-Channels. The behavior of the Calcium Current is often modeled with Eq.4.4 by multiplying the right side by a proper constant. An application of such method is also presented in Fig.4.2: here, the time-constant of the exponential kernel is presented in three values $[50, 100, 500]ms$. Due to the fact that the increment of the Firing Rate is weighted by the inverse of the time-constant and the decay is faster for small time-constants as well, when $\tau = 50ms$ the approximation of the Frequency produces a highly noisy function, able to capture the overall behavior at the $10ms$ time scale. This highly fluctuating function is however unsuited for the use in learning algorithm, due to the high level of noise. Instead, increasing the time constant produces a function which ignores the fast fluctuations of the frequency, but better captures the average value of the Firing Rate and is less affected to noise. In general, since the signal corresponding to $\tau = 500ms$ is also much slower to react to changes in the firing of the neuron, it is particularly suited when the Firing Rate has to be evaluated representing the average activity of the Neuron in a longer time. One of such cases, is the use of Plasticity, in which not only the instantaneous value of the Firing Rate matters, but also the behavior of the Neurons in the previous time counts.
In general, the Spike Count method will be used for the learning of the output weights, while the Low-pass filter of the spikes will be employed for Intrinsic Plasticity and the evaluation of the activity of the Network.

Linear Regression

It represents one of the simplest models for interpreting the relationship between a dependent variable and an explanatory variable. In the case of a scalar explanatory variable, it is called Simple Linear Regression, otherwise it is said MultiLinear Regression. The relationship between the dependent and explanatory variables is modeled with a linear function, whose parameters are the object of the learning. Explaining LR from the standpoint of its application in this work, 3 matrices are given: $\mathbf{R}$ being the Reservoir activity or rate, obtained by performing the Spike Count method on each neuron; W_{out} the output matrix; O_{teach} the teaching Output rate matrix. The goal is to tune the output weights W_{out} in order to minimize the error of the actual Output of the Network $\mathbf{O}$ with respect to the teaching signal O_{teach}. This is performed by the following simple linear relations:

$$\mathbf{O} \approx \mathbf{W_{out}} \cdot \mathbf{R} = \mathbf{O_{teach}} \tag{4.5}$$

The Linear Regression is computed in order to minimize the Mean Squared Error of the scalar product W_{out} x $\mathbf{R}$ with respect to the teaching Output O_{teach}.
As it is clear, Linear Regression assumes a linear relation between Input, which in the case of LSM

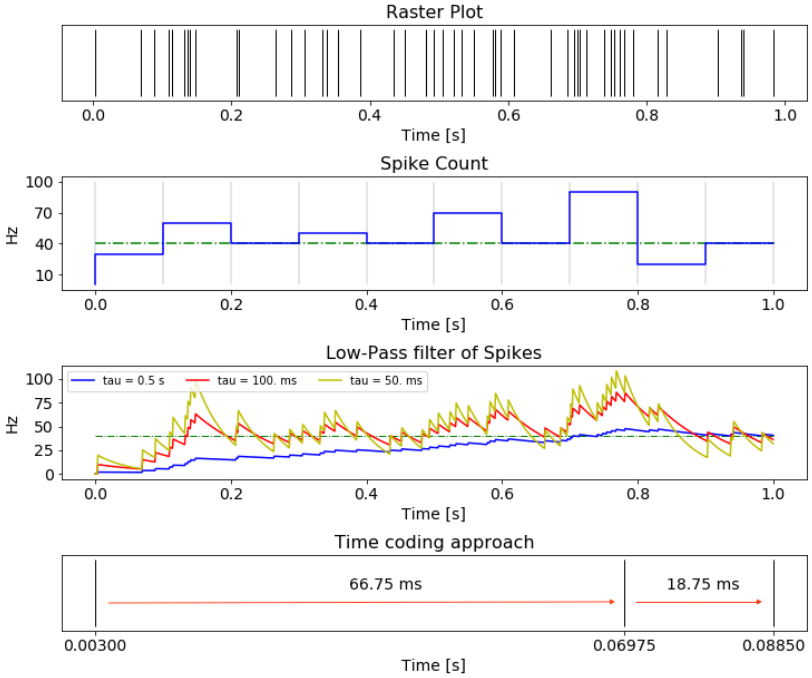


Figure 4.2: Comparison of the functioning of most common Rate and Time coding method for decoding the information from a Spike train, shown at the top, obtained from a Neuron firing with a Poisson distribution at $40Hz$. In the Spike Count method, the spikes within an interval are summed and the frequency is obtained by dividing by the interval length. The Low-Pass filter implements the Eq.4.4 in order to approximate the frequency: three cases of time-constants are plotted. Time coding, instead, deals with the precise timing of the spikes with respect to each other.

is the Liquid State, and Output, the output group. However, the dynamics of Neurons is much more complex and inherently non-linear. As a consequence, Linear Regression is not the optimal solution for training the weights in SNNs. Still, the performance of the LSM tuned by this method will be shown to be acceptable. The benefit of such a non-ideal training method is that it can be compared with on-line training methods, which are in general less accurate. Since on-chip operation is a goal of many neuromorphic systems and on-line training is the only viable option if one does not have control over the device during the whole phase of training, it is interesting to evaluate the effect of non optimized weights.

Logistic Regression

Logistic Regression is a tool for evaluating the probability of a certain variable to be classified in a binary fashion. In order to model the probability of the variable, a logistic function is utilized. The process of applying Logistic Regression to a classification problem then reduces to quantifying the parameters of the logistic function describing the probability function.
As much as for Linear Regression, the logistic method can be applied in the case in which a teaching signal is known and a system needs to be approximated to it. In this case, the teaching signal has to be a succession of binary values, virtually indicating the activation of one or more classes in the Output. For example, if the output of a system is composed of the letters $[a, b, c]$ and the requirement is to classify the Input as a b, the binary output will have to be $[0, 1, 0]$ to indicate that the second class is selected.
The form of the problem applied to the LSM case remains unaltered: a Reservoir activation matrix $\mathbf{R}$

is defined and an Output Weights matrix W_{out} is learned in order to minimize the error with respect to a teaching signal O_{teach}. In this case, the linear system given by $\mathbf{W_{out}} \cdot \mathbf{R}$ is approximating a Logistic function:

$$Logit\left(\mathbf{O_{teach}}\right) \approx \mathbf{W_{out}} \cdot \mathbf{R} \tag{4.6}$$

where the Logistic function is in general defined by:

$$Logit(p) = ln\frac{p}{1-p} \tag{4.7}$$

defined over $0 < p < 1$.

4.3 Dynamics of the Liquid

While Echo State Machines have been extensively characterized by the Deep Learning community and mathematicians have developed strategies in order to control the behavior of the Reservoir so to maximize performance, Liquid State Machines remain a less robust computing paradigm. Spiking Neural Networks present the difficulty of dealing with more complex equations at the node (Neuron) and connection (Synapse) levels and thus the tuning of the Reservoir for maximizing classification results requires a greater effort. Despite the simplicity of the LIF neurons and Exponentially Decaying synapses with respect to other more biologically inspired models, the amount of hyper-parameters to control is large, each having an effective impact on the dynamic state of the Liquid.
In particular, it is common practice to tune Reservoirs at the Edge of Chaos, a situation in which the dynamics of the network is neither deterministic or vanishing nor chaotic. The Edge of chaos can be viewed as a region of the hyper-parameter space in which the Lyapunov Exponential (LE), a common metric for dynamical systems, is close enough to 0. Unfortunately, calculating the LE is very time consuming, especially if it is required during the phase of tuning of the parameters of the Network, since it has to be performed by repeating long runs in simulation.
For these reasons, the Lyapunov Exponentials are not a common metrics for SNNs, for which the tuning phase remains a critical step in the setup of an experiment.

4.3.1 Tuning of the Liquid

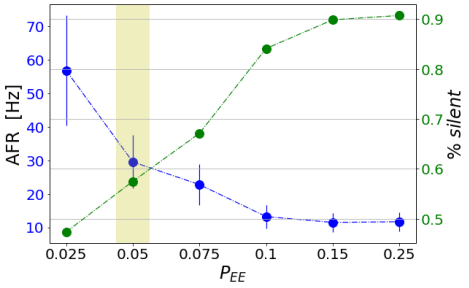
Despite the LSMs being general-purpose computation machines, their Liquid has to be tailored for the specific task it is used for in order to reach acceptable performance. Two aspects have to be taken into account when approaching the tuning of the Liquid: what kind of Input will be proposed to the Reservoir and what kind of Output the LSM is expected to produce.
When analyzing the Input, the main information it contains, given the way learning is performed, by means of Rate-Coding, is the Frequency. In general, the Input may be coming from sensors or external agents and they all code some information in the form of the frequency of the spikes. The Liquid's time-constants have to be set in order to make the Liquid able to respond to the Input in an effective way. A simple *rule of thumb* is to match the time constants of the input signal with these of the Neurons and Synapses, even though slightly faster components may help in processing the information without losing a substantial portion of the Input information. For the specific cases of study in this thesis, the Input will be a group of Poisson-firing Neurons at a frequency of $100Hz$: Neuron's and Synapse's time-constants in the range $[10 - 50]ms$ are consequently adopted.
For what concerns the Output, the LSMs will be both asked to reproduce a specific firing pattern or to perform classification. For the first case, it is important to tune the Output neurons in order to

make them able to produce the required signal. Instead, for the sake of classification, no particular constraint is imposed on the frequency of output neurons: ideally, in order to represent a certain class, a single neuron or a single group of neurons should be firing, with whatever frequency, and all the other neurons stay silent. However, this is difficult to achieve, so the policy for classification is that the Output will assume the value corresponding to the highest frequency firing Neuron. For both the cases, the same consideration on the time-constant related to the Reservoir components is valid.
In general, it is not feasible to tune each parameter of the Network due to the highly non-linear and recurrent nature of the Reservoir: the effect of each parameter is dependent on each other, so tuning a parameter at the time is useless. Because of that, a baseline is assumed for most of the parameters of the Network:

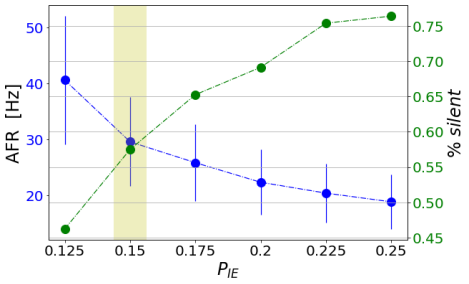
% Inhibitory	p_{II}	p_{EI}	$\tau_{syn;inh,out}$	$I_{0,syn;inh,out}$	$R_{in;inh,out}$
20 %	0 %	10 %	15 ms	0.2 nA	350 $M\Omega$

Table 4.1: Base-line parameters for the tuning procedure of the Liquid

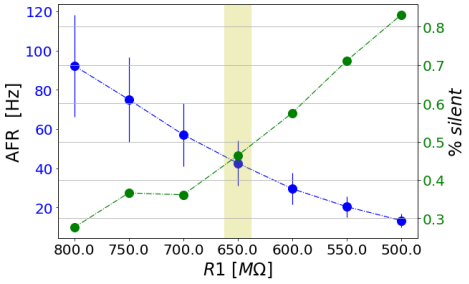
The percentage of inhibitory Neurons and the absence of recurrent connections between them have been chosen based on biological motivations and inspiration from most of LSMs implementations found in the Literature. Instead, fixing the synapses time-constant, both inhibitory and at the output, at $15ms$ was chosen based on the fact that most of the signals processed by the Liquid had to present features at the $100ms$ time-scale. Finally, the synaptic currents and the input resistances for Inhibitory and Output Neurons have been chosen in order to produce firing rates on the order of $100Hz$.
Instead, tuning is performed for the other parameters which have a stronger impact on the dynamical behavior of the Reservoir. The probability of connections between Excitatory neurons, p_{EE}, controls the homogeneity of the activity in the Reservoir: high connectivity means that the Network has a lot of synapses which distribute the activity in the Network evenly. Instead, a low degree of connectivity allows the activity to form some paths in the Reservoir and form clusters of active neurons. The degree of connectivity from Inhibitory and Excitatory neurons p_{IE} controls how strong the inhibition is to the Reservoir: in general, the inhibition is evenly spread in the Reservoir, but too large connectivity may slow down Excitatory neurons too much. The parameters at the input section of the Excitatory Neurons are also tuned: the Membrane Resistance and Capacitor play a crucial role individually and together they form the time-constant of the Neuron. Lastly, two additional parameters of the Synapses connecting Excitatory units are tuned: the synaptic unit current I_{EE} and the synaptic time-constant τ_{EE}, both of them regulating the strength of the Synapses. The figures below are obtained by varying the mentioned hyperparameters individually, thus are not representative of the phase of tuning, but showcase how difficult this phase is due to the high sensitivity of the Liquid with respect to any parameter modification.



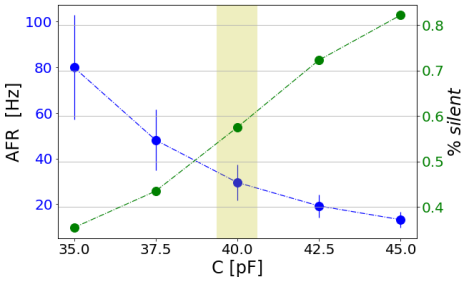
(a) AFR and % of silent Neurons vs p_{EE}



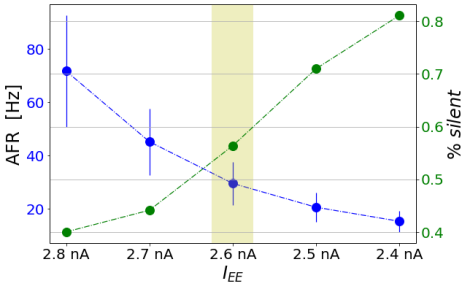
(b) AFR and % of silent Neurons vs p_{IE}



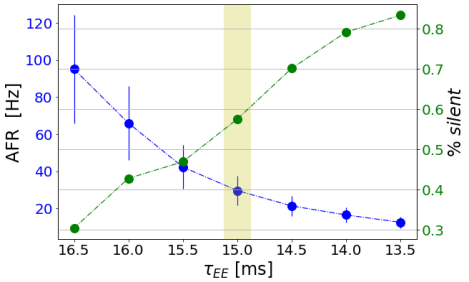
(c) AFR and % of silent Neurons vs Neurons' input resistance



(d) AFR and % of silent Neurons vs Neurons' input capacitance



(e) AFR and % of silent Neurons vs Excitatory synapses current unit



(f) AFR and % of silent Neurons vs Excitatory synapses time-constant

Figure 4.3: Effect of the tuned parameters on the Liquid. AFR is the Average Firing Rate of the Excitatory Neurons, p_{Sil} is the percentage of Excitatory neurons which are silent during the simulation time. The values selected during the tuning procedure are marked in yellow.

In these plots, each point corresponds to the average between 10 simulations of $5s$ performed by projecting different Poisson-Input spike trains at $100Hz$ of average frequency each time to a group of 10% of the Excitatory Neurons. The response of the Liquid is characterized by its Average Firing Rate (AFR) and the percentage of Silent neurons (p_{Sil}). Immediately, a feature of the plot is clear: both the quantities are plotted with their standard deviation of the 10 trials, but the p_{Sil} appears to have almost no variability. This turns to be a good thing since it means that the activity of the Liquid due to a certain Input is spread through the same group of Neurons every time, despite the noise at the Input. In turn, this is good for identifying in the response of the Liquid what Input signal has been presented.
Furthermore, the behavior of the AFR and p_{Sil} is similar for every parameter modification. In general, these quantities have a sigmoidal behavior when spanning over different values of a single parameter, but due to the high non-linearity of the Liquid, it is difficult to predict the behavior when more than one parameter is changed at one time.

The values selected during the tuning procedure are marked in yellow. The choice was based on a paradigm: the active Neurons had to fire around the Input frequency ($100Hz$) and they had to form a defined cluster of activity that made it easy to recognize the type of input and at the same time present a complex enough dynamics in order to perform, later in this work, temporal tasks. This last requirement is not easily evaluated by means of the AFR, even though the Standard Deviation of such a quantity may give an intuition that the response of the Network was not deterministic and static. While in this case the same 10% subset of the Liquid was linked to the Input, connecting the Input to different subgroups of the Reservoir is in general a way to encode information to the Liquid: each of the subgroups represent a given input signal even though the Input group is maintained the same all the time. For example, the figure above may refer all to the Input signal 1, choosing other subgroups one can feed the Liquid with signals [2,3,4..,10]. Since one aims at producing a signature pattern of activity for each of the input signals, the subgroups do not share any neurons and are independent with each other. In order to find a compromise between richness of dynamics expressed by the Liquid and classification performance with respect to each signal, it has been decided that each signal had to activate [40-60]% of the Reservoir Neurons.

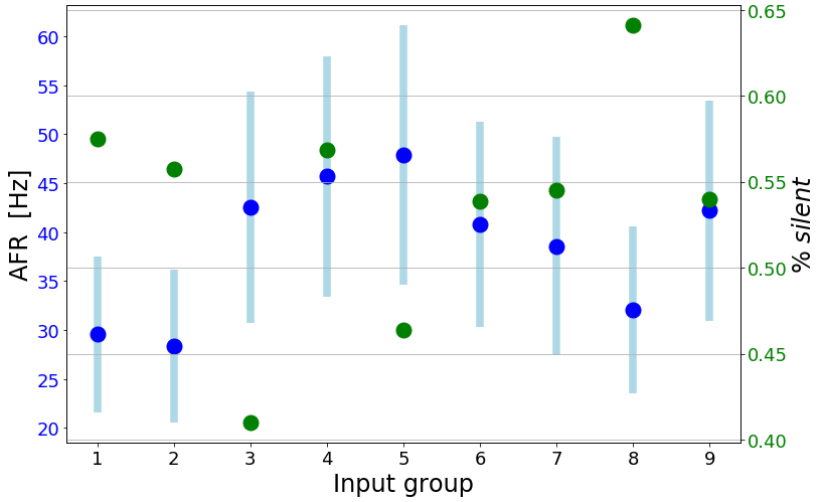


Figure 4.4: AFR and Percentage of Silent Neurons with respect to 9 different input condition. The same input group is linked to a different subgroup of the Liquid containing 10% of the total amount of Excitatory Neurons. No neuron belong to more than one group.

4.3.2 Liquid State characterization

Once that the requirement for the AFR and Percentage of silent Neurons have been verified for one single signal, the same procedure is in theory extended for all the other 9 input signals. It is then clear that the tuning procedure is long difficult. Furthermore, one more degree of freedom has to be taken into account: in Brian, the Python library used to simulated the Network, one can establish connections between groups of Neurons based on the probability of forming the synapse between two Neurons. Indeed, the effective presence of the Synapse is determined by the "seed" generating the pseudo-random variable for establishing or not the connection. When the seed is changed, different

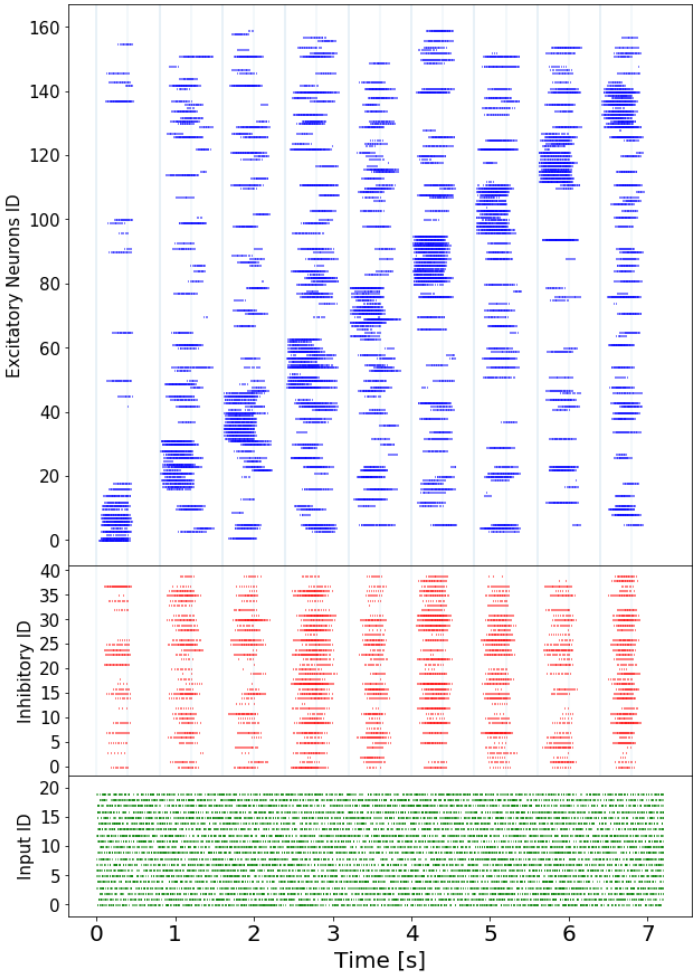


Figure 4.5: Raster Plot of 9 input signal being fed to the Liquid. Each signal lasts for $0.4s$ and there is a $0.4s$ silence from the Input between them. It is noticeable that each signal produces it signature response of the Liquid.

results are obtained, forming different patterns of connections. Changing the seed has a considerable impact on the behavior of the Liquid, so that can be considered as an additional parameter involved in the Tuning phase. In this case, after trying some different seed, it has been found a configuration able to satisfy the requirements for AFR and p_{Sil} for all the different input signals.
As for the case of the Tuning plots, each case is simulated 10 times under different input conditions and the results averaged. Clearly, each signal produces a different percentage of silent neurons, indicating that probably a defined cluster of activation has formed for each case. The average activity of the Liquid ranges from 20Hz to 60Hz which, considering the number of silent neurons, means that the active neurons are close enough to the target $100Hz$.
Still, this figure does not indicate whether each of the input signals presents its characteristic signature, i.e. its own defined cluster of active neurons in the Liquid, so to maximize classification performance. In order to analyze that, first the Raster Plot of each of the signals is presented and then a Principal Component Analysis is performed on that simulation in order to capture a quantitative metrics of the dynamic behavior of the Liquid.

In this raster plot, each signal is presented for $0.4s$ and there is a pause of $0.4s$ between each signal. This silence is needed to evaluate the eventual persistence of the dynamics generated by a signal.

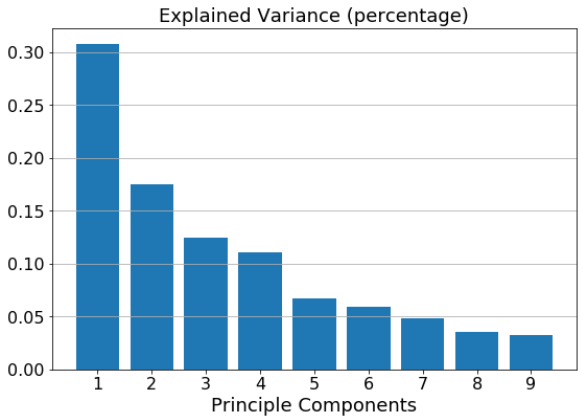


Figure 4.6: Explained Variance of the first 6 Principal Components, for a 200 Neurons Network, during a 20s simulation

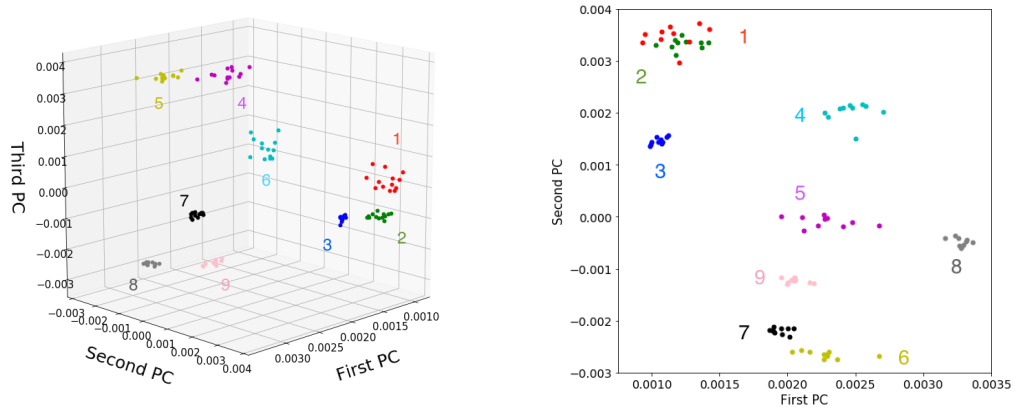
For the sake of classification, persistence activity related to a signal is detrimental as it may lead to interference when a new signal is presented. On the other hand, in order to perform temporal processing of the input, each signal has to present a complex enough dynamic behavior of the Liquid. Such dynamics cannot be evaluated with such short times each signal is presented, even though it is still visible that within the interval of each signal some neurons are activated and deactivated at different times. The important qualitative information from this plot is that each signal produces its own pattern of active neurons: despite some of the active neurons are shared between the cluster of activation related to different input signals, each signal presents its own distinctive signature on the Liquid.

It is rare to find an analysis of the dynamical properties of a Liquid in the literature, so it has been proposed to exploit a Principal Component Analysis (PCA) to extract further information from raster plots, such as the one in Fig.4.5. In this case, with the metric of the variability explained by the firsts few Principal Components, one evaluates the richness of the dynamics, while the separation property is evaluated by projecting the Liquid State for each of the input signals in the first 2 or 3 Principal Components. When performing the PCA, each neuron is treated as a dimension and its Calcium Current is the value along time of such dimensions.

In Fig. 4.6, because one has 9 input signals, it makes sense that the explained variance of the Network is spread through the first few components almost evenly. As a matter of fact, 98% of the variance expressed in the simulation is explained in the first 12 components. Ideally, if each of the clusters were totally uncorrelated with each other, the variance would be explained by at most 9 components; however, some of the Neuron belongs to the active cluster related to more than one input signal, so that the explained variance is not even in the first 9 components. Moreover, because of the dynamic behavior of the Liquid, some neurons are activated only after a certain time when a signal is presented, so that the internal representation of a signal is evolving in time. The consequence is that some of the variance of the Network is explained by additional components that encode the information about the Liquid's dynamics.

Lastly, the Liquid State is projected in the first 2 and 3 Principal Components in order to qualitatively evaluate the Separation Property of the Reservoir. Ideally, when the same sequence of signals is repeated over time, the Liquid State related to each signal is projected in the same portion of the PC space. Since the input is noisy, because of the nature of a Poisson distributed train of spikes, one cannot expect that every signal is exactly projected in the same spot but clusters will form, each one representing different input conditions. The tighter and more packed the cluster, the better the classification of the input signal. The farther the different clusters, the easier to distinguish between

different input signals for the classification. If one or more cluster over-impose on each other it will be not possible for the readout to classify them correctly.



(a) Input signal's Liquid representation in the firsts 3 Principal Components

(b) Input signal's Liquid representation in the firsts 2 Principal Components

Figure 4.7: Projection of the Liquid State when different input signal are presented to the Reservoir. Most of the signals are well separated, meaning that their corresponding Liquid State is projected in a certain region of the first Principal Components, avoiding the overlap of such regions

In Fig. 4.7 it is clear that each input signal produces a cloud of points that are of the same color, representing each of the input signals. The dimensions of the cluster are smaller than the distance between them so that the readout can in principle perform classification based on the Liquid State. One could complain about the red and green clusters being positioned in the same position in the 2D PC plot, but the 3D figure reveals that the two clouds are not actually the same position. Yet, the projection over two only components does not allow to verify that.
Some of the clouds are more compact, such as the blue, grey, black and pink ones, indicating that those signals are better encoded by the Liquid. Other signals, such as the red, violet and light blue ones, are instead more spread in the PC space. This fact can have two explanations: either the input signal is more sensitive with respect to the variability of the Input spike pattern or the dynamics related to these inputs is more chaotic and less deterministic.

Limitations

One of the main aspects which limit the performance of such a machine is the Readout, especially due to the way it is trained. As a matter of fact, training methods based on the precise timing of the spikes have been developed [51] [52] but are expensive from a computational point of view and not feasible for Recurrent Networks.
The alternative is, as already mentioned, to perform rate-coding based training: one treats the frequency of the neurons instead of its single spikes, and aims at producing a target activity on the output layer. The weights are computed by approximating the Reservoir frequency matrix to the desired output. By doing so, two errors are introduced:

- Most of the algorithms (Linear or Logistic Regression) to map the frequency of the Reservoir to the Output assume a linear relation between them. Anyhow, LIF neurons present a non-linear behavior due to the leakages

- The definition of frequency is ambiguous for asynchronous trains of spikes. The most common solution is to count the spikes in time-intervals. However, the length of the interval influences the behavior of the Frequency in time

Given these two problems, it is clear that the SNNs are not capable to offer the same level of performance as ANNs, for which training presents fewer non-idealities.

4.4 Delayed Auto-Encoder

One of the crucial properties of Recurrent Network is their inherent memory, granted by the recurrent connections and supported by the time features of the processing of the information by the nodes (Neurons and Synapses) of Spiking Neural Networks. As for the case of the richness of the dynamics, there are numerous metrics and benchmarks for the evaluation of the Short Term Memory in Artificial Neural Network based architecture. Above the many, the Memory Capacity and the NARMA test [44] [46].
Those tests, however, are of no trivial transfer in the framework of the time-continuos Spiking Neural Network so that it is proposed to apply the Liquid State Machine for a temporal task requiring memory, in order to quantify its Short Term Memory. This task is the Delayed Auto Encoder (DAE): the Liquid is fed with a certain input which is reconstructed at the Output with a certain delay, through the output weights. The weights are computed with simple Linear Regression and the error is computed by the Mean Squared difference of each Output neuron with its corresponding Input counterpart. As a consequence, the number of output neurons is the same as the input neurons. The error is called Normalized Mean Squared Error (NMSE) and the Accuracy is in turn $1 - NMSE$. The figure below represents the scheme of the DAE task.

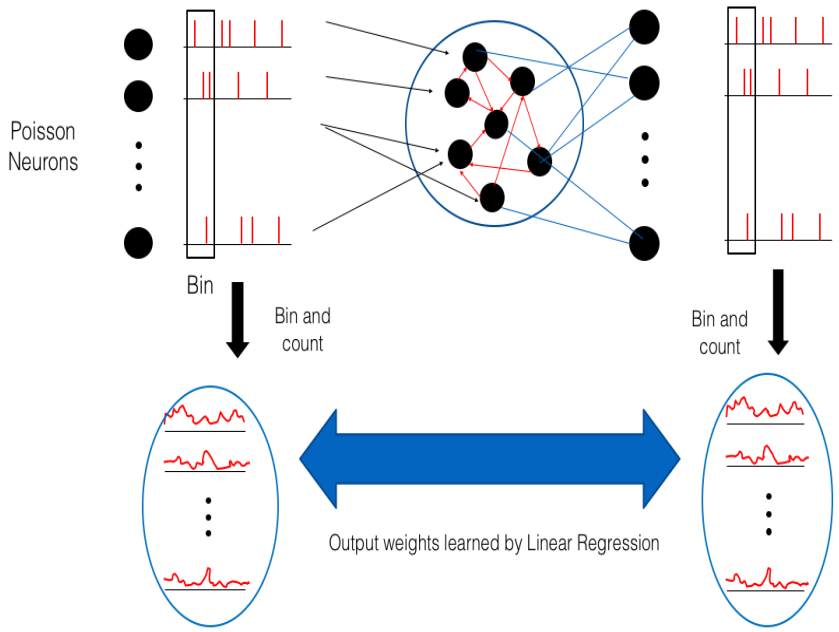
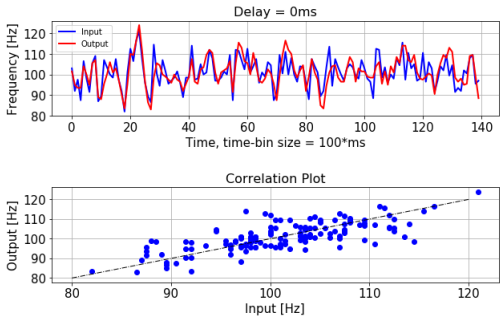


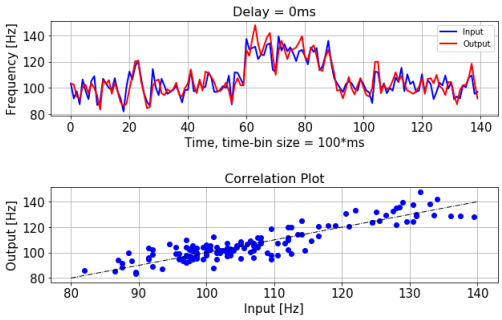
Figure 4.8: Schematics of the DAE task, performed with rate-coding

The task is performed with two different sets of input: the poisson group is either maintained at

a steady frequency of 100Hz or, at a certain moment ($10s$) the frequency is stepped up to 125Hz for 1s. The NMSE is evaluated in an interval of time of $8s$, while the spikes are time-binned in intervals of $100ms$. Examples of the Input and reproduced Output with Delay of $[0, 0.4, 1]s$ are plotted below. In all the cases below, the same input is presented, in order to better compare the quality of the reconstruction with different delays.



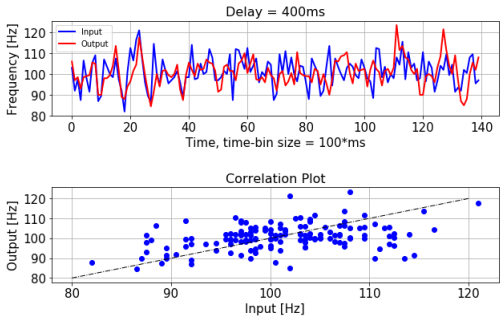
(a) Input/Output comparison, steady input



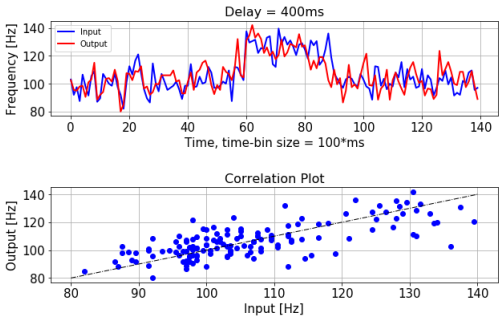
(b) Input/Output comparison, 3s step input

Figure 4.9: Reconstruction without Delay. The upper plots represents the time-binned evolution of the spike count, i.e. the frequency. The lower plots instead are obtained by displaying the Input values on the x-axis and the Output on the y-axis, in order to evaluate the correlation

When no delay is applied, the Liquid simply has to provide the output the same features in time it receives from the input. As a consequence, the task is relatively simple and, as it is shown in Fig.4.9, the accuracy of the reconstruction is high. Without introducing the step of frequency in the input, the Output is even able to follow the fast fluctuations of the frequency of the input. Also in the case of the input presenting the step of frequency, the output is able to reproduce most of the frequency fluctuation, even though the step increases the difficulty of the task and thus slightly lowers the accuracy of the output.
All these considerations are confirmed by the Correlation Plots, in which the Input is plotted on the x-axis and the Output in the y-axis: ideally, if the output would be able to perfectly follow the input, the dots would be aligned in a straight line. The closer to the ideal line, the more accurate the output.



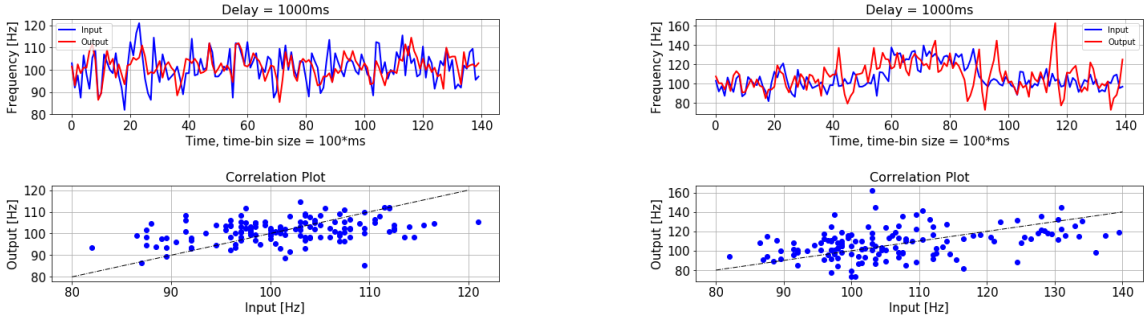
(a) Input/Output comparison, steady input



(b) Input/Output comparison, 3s step input

Figure 4.10: Reconstruction for a Delay of $400ms$. The steady frequency case exhibits good level of reproduction of the input frequency, without the same precision it had without the delay. When a step of frequency is introduced, the level of accuracy drops, since the task is harder.

The figures above are obtained by shifting the Output of the Delay which they are given, in order to judge the quality of the reconstruction. With a $400ms$ delay, the accuracy is in general worse, as expected: the Liquid is asked to provide the output information it received some time before. The ability to retrieve this information is related to the recurrent architecture of the connections, which form the short term memory. By storing information about the past inputs in the complex recurrent dynamics of the Network, the Liquid is in principle able to provide some information about the past to the Output. Of course, the quality of the reconstruction is lower with respect to the previous case, since the readout has to linearly select the information of the past inputs in the Liquid, ignoring the ones of the present.
Moreover, the complexity of the task is much enhanced by the step of the frequency of the input, for which the reconstruction is of pretty poor quality.



(a) Input/Output comparison, steady input

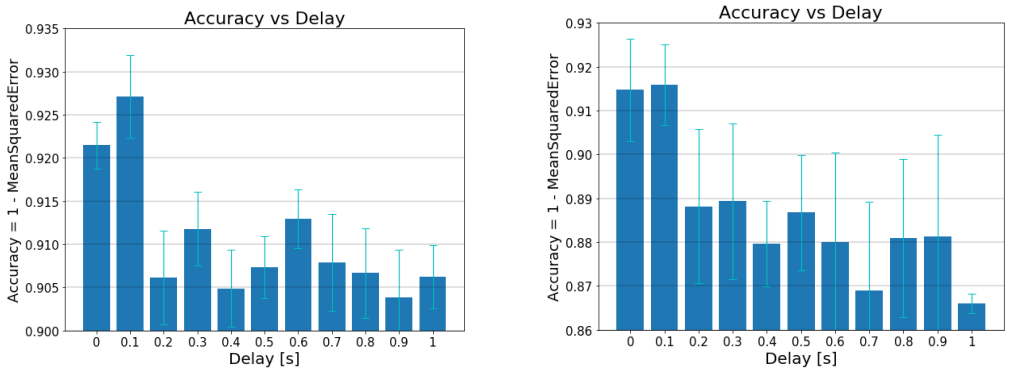
(b) Input/Output comparison, 3s step input

Figure 4.11: Reconstruction for a Delay of $1000ms$. In the case of steady input frequency, still some of the features are reproduced: the Liquid has a rich enough dynamics so that it can reproduce, to some extent, the oscillations of the input frequency. With the step of frequency, the Output only learned the average frequency of the input and forgot the step of frequency.

Increasing the Delay to $1s$, one expects to see a further decrease in accuracy. This is true for the case of the Step of frequency: the output is seen to mainly learn the average frequency, losing the ability to reproduce the fast frequency oscillations. This behavior is expected to saturate the accuracy for higher delays. As a matter of fact, the output weights can learn the average frequency anyway, also producing random oscillations due to the dynamics in the Liquid.
Instead, for the case of steady average input frequency, the accuracy has already saturated: no major difference in behavior is seen with respect to the case of $400ms$ delay. The Output is mainly reproducing a signal at the same average frequency with some oscillation, which in some cases remind the ones of the input. Some correlation with the Input is still present, but the level of accuracy of the case of no Delay is not even nearly reached.

Results of the DAE

The NMSE and Accuracy are evaluated for different Delays in order to estimate the memory of the Reservoir. In order to test the Accuracy, the DAE is performed 10 times, each of which varying the onset time of the step of frequency, if present, and the structure of the input. This means that the Poisson input group, still firing at the same average frequency, produces different frequency oscillations every time. The average NMSE and Accuracy are plotted.



(a) Accuracy of the DAE for the steady input

(b) Accuracy of the DAE for the 3s step input

Figure 4.12: Results of the Delayed-AutoEncoder for the cases of No-Step and 3-second-Step input frequency. Accuracy is stable regardless of the Delay for the case of Steady input frequency, while it decreases for the case of the 3-second-Step

In the case of the Steady Input, the reconstruction is of great quality, as also confirmed by Fig. 4.12. For small Delays, the Output is able to capture also the small fluctuations of the input frequency in time, due to the stochasticity of the Poisson Input neurons. This is confirmed by the Correlation plot, in which the points are distributed along the straight line representing the maximum correlation between Input and Output. When instead the Delay is over $200ms$, the reconstruction is of worse quality: the Output is mainly able to learn the mean input frequency and produces some fluctuations which are correlated, only to some extent, to those of the Input. This is why the behavior of the Accuracy is almost independent on the Delay, when that exceeds certain threshold values.
If the Input presents a step of Frequency, the Output is presented a more complex task, since simply learning an average output frequency will generate a larger error. As a matter of fact, the error is in general larger than in the case of steady input for every Delay. It is relevant to observe that the reconstruction is of good quality when the Delay is small, while the accuracy decreases as the Delay overcomes $200ms$. This is attributed to the Fading Memory of the Reservoir, for which the information of the Input is retrieved in the Reservoir only for small Delays. When the Delay is greater than $0.7s$, a similar situation as the case of the steady input occurs. The Output learns the average frequency and produces random oscillations, but totally forgets the step of frequency. Still, the accuracy will stabilize around 0.85, but this does not mean that the Liquid is able to remind pas input, rather than the output weights has successfully learned the mean value of frequency of the Input.
At last, a common behavior for both the cases of steady input and with the frequency feature: the accuracy is best when the Delay is of $100ms$. The interpretation of this fact is that the activity of the Input takes time to spread in the Liquid, since the input spikes are processed by Neuronss and Synapses which have time constants at the $10ms$ time scale. This means that even a single spike event takes some time in order to have an effect on the Liquid. Due to the level of connectivity (5%) and the size of the Reservoir (200 Neurons), each input spike is processed in average by a mean number of Neuron and the time it takes to produce an effect on the Liquid is certainly on the $10ms$ scale. It is then of no surprise that when a Delay of $100ms$ is present, the reconstruction is of best quality: the Liquid contains just the information it received from the input a short time before, which has spread through the Network.

Chapter 5

Intrinsic Plasticity

This chapter of the Thesis is devoted to the implementation of Intrinsic Plasticity (IP) in a Spiking Recurrent Neural Network, with the aim of controlling the dynamics of the whole system by acting on the Input resistances of the Neurons, implemented by Memristors. As a matter of fact, in Spiking Neural Networks (SNNs) with a high enough degree of plausibility with respect to biological Neural Systems, Neurons receive current pulses - resulting from Action Potentials - which are integrated by an RC group. The equivalent membrane resistance R_{mem} then has a crucial role in modulating the integration of the incoming spikes. This work proposes to act on this resistance in order to not only control the activity of the single Neuron, but also of the whole Network. Since the Membrane resistance R_{mem} is a property of the Neuron, an adaptive rule concerning that variable falls in the category of Intrinsic Plasticity. This does not mean that the real Neuron only performs Intrinsic Plasticity by varying their input resistance, but this simple concept comes with convenient implications:

- featuring a complementary circuit for the operation of the input resistance, as shown in [53], IP can be implemented for on-chip operations
- it will be shown that the switching of Memristive states is very energy efficient, thus allowing IP to operate consuming very low power
- despite the stochasticity of the Memristors, changing the input resistances of Neurons is shown to have an effective role in the dynamic behavior of the whole Network

Considering the beneficial effects observed in most experiments on Mammalians visual cortex [54], Intrinsic Plasticity promises to increase the energetic efficiency of Spiking Neural Network based system, while maximizing the volume of processed information.
All these factors make the implementation of IP a promising feature for Neuromorphic Computing able to increase the degree of fidelity to biology and possibly enhancing the performance of the Network and the efficiency of the chip, without incrementing its size.

5.1 Intrinsic Plasticity in Neuromorphic Computing

In biology, Intrinsic Plasticity is a complex phenomenon involving the adaptation of many Neural quantities over time for balancing the activity of the single neuron and the whole Micro-Circuit in which the neuron operates. Practically, Intrinsic Plasticity refers to all the persistent changes in the neuron's intrinsic electrical properties during its functioning. IP is a less investigated mechanism than Synaptic Plasticity so that its actual functioning and purposes are not retained consolidated knowledge. Still, IP is believed to act following the action of Neurotransmitters and Synaptic activity, via the weakening or strengthening of the Ion Channels responsible for the integration of the Action Potentials coming from Synapses. When the action of such Channels is facilitated, then the Neuron is more easily excited and thus more likely to increase its firing rate. The opposite happens for the cases in which the Channel is weakened.
In this way, IP can counteract the polarizing action of STDP, which tends to form clusters of active neurons, in spite of most of them being shut off. Moreover, IP can decrease the global average activity when a certain part of the Brain is stimulated for long times, thus saving energy consumption and avoiding excessive activity, eventually damaging neurons and synapses. In this view, IP is an Homeostatic phenomenon. Moreover, IP is thought to play an important role in supporting the formation of Memory in the Brain, which is conventionally attributed almost solely to the Synaptic Plasticity.
All these beneficial effects are appealing for Neuromorphic Computing, which aims to become an effective computing paradigm while remaining faithful to biology and being very energy efficient. IP promises to be a feature of great benefit for such purposes and it is increasingly become popular in the NC community [57] [58] [59], especially for Liquid State Machines application. As a matter of fact, Reservoirs Networks are the ones with the most complex dynamics and students in this field often struggle to regulate such systems, thus potentially relying on IP to improve the control over the Network. On top of that, it is also interesting to combine IP with other conventional Plasticity mechanisms, such as Homeostatic Synaptic Plasticity and Spike-Timing-Dependent-Plasticity: despite the fact that the effect of each of the mechanism is well known, it is to investigate their synergic effect. Studies on the framework of ANNs, such as [59] and [56], demonstrated that the combined effect of simple versions of different Plasticity Mechanisms can have beneficial effects that go beyond the sum of the effect of single phenomena. Such analysis is lacking in the world of Spiking Neural Networks.

5.2 Methods: Circuits and Network

The Neuron

Implementing Intrinsic Plasticity in Neuromorphic Hardware is a challenge under many points of view: Neurons have to be equipped with some variable parameters which adapt its values according to some Plasticity rule and, in order to overcome the Von-Neumann bottleneck, the additional circuitry for such a task must allow for on-line operation without relying on external memories. The most straightforward and common approach in NC and general In-Memory computing is the use of Memristors, functioning as programmable resistances. Such devices, for which research is ongoing in order to improve their properties and integrability in conventional electronics, can be programmed by Voltage pulses, thus commanded by mostly digital dedicated circuits. The usual implementations comprehend 1T1R (1 Transistor controlling 1 programmable Resistance/Memristance), with the input of the Transistor being set by the dedicated circuit, actually implementing the Plasticity mechanism. The following figure shows the Neuron employed for future analysis: it is a simplified version of the one commented in Fig. 3.3, where the input and refractory sections are modified so to host the Memristors and the Adaptation module is neglected. Not shown in this figure also the circuit dedicated to the evaluation of the Calcium Current 4.4, which is often an important parameter for the on-line

evaluation of the Firing rate. Later on, the circuits which control the behavior of the Memristors M1, M2 and M3 will be discussed.

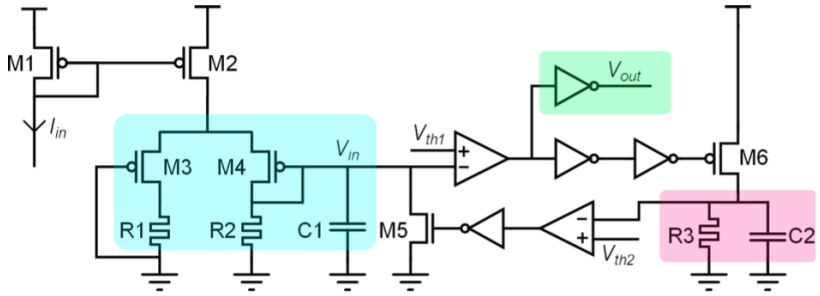
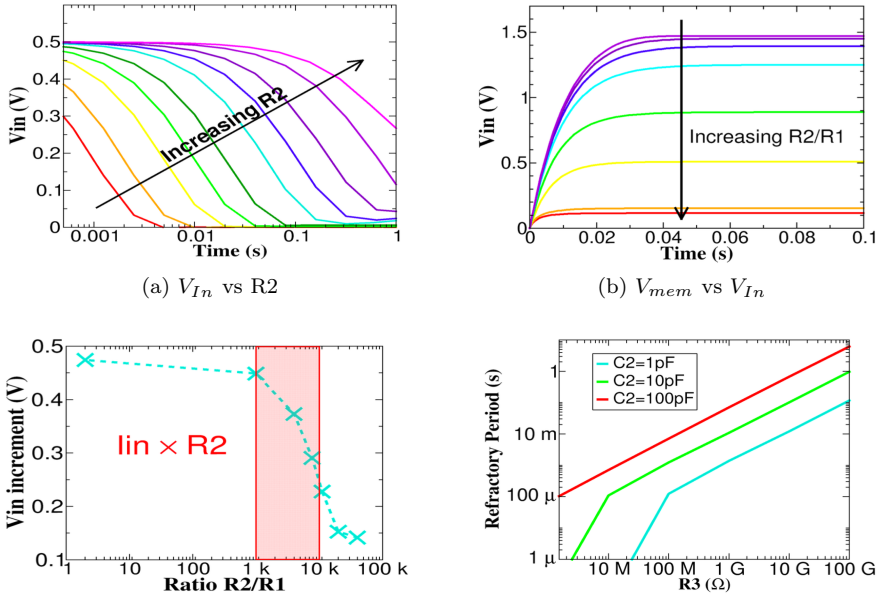


Figure 5.1: Circuital Representation of the Neuron employed for the following analysis. The circuit is based on a DPI integrator (M3,M4,C1) with a modified Input section (blue), comprising Memristors in place of R1, R2 and R3. The violet group is responsible for the refractory time, while the green operational amplifier produces the output spikes. All the additional features related to biological plausibility are omitted for clarity. From [53]

This circuit is essentially similar - and simpler - to the DPI for its functioning: the incoming current pulses from the synapses cross the input DPI pair operating in sub-threshold and are integrated by the system formed by M3, M4, R1, R2 and C1. SPICE simulations conducted in [53] show that the increase in the ratio between R2 and R1 results in an enhanced input gain. Moreover, increasing R2 results also in a slower response of the Membrane Potential with respect to the input pulses.
As the Membrane Potential increases under the effect of external stimuli, it eventually crosses the Threshold of the Neuron, thus making the output of the voltage comparator switch and inducing the formation of the output Spikes by means of the green block.
Also, as the output of the comparator is ON - close to the supply voltage - transistor M5 becomes active, thus charging Capacitor C2 with a time-constant of $\tau_{refr} = 1/R3C2$. As the voltage on C2 overcomes the second threshold value V_{th2}, it activates the other voltage comparator which switches the transistor M5 on. At this point, the integrating Capacitor C1 is grounded and quickly discharges, remaining in this state as long as the transistor M5 in on. As the R3C2 system gets discharged, M5 switches off and the integrating function of the capacitor C1 is restored. As a consequence, the effect of R3 is trivial: as it increases, the time constant of the $R3//C2$ system increases, resulting in a prolonged refractory period.

Memristors

Technology allows to conceive and realize many types of Memristive devices, each of which with its characteristics and features. To be compatible with the above-analyzed Neuron model, Memristors are required to be integrated into circuits, from a technological point of view, retaining the properties of programmability and control over their Resistive States. In reference to [60] and [61], a team in CEA Leti shows OxRAM devices integrated with a 130nm CMOS process: a TiN bottom electrode is deposited over the Cu Metal 4 layer, which has previously been planarized with Chemical Mechanical Polishing (CMP); a 10nm thick HfO_2 layer is deposited and topped by another 10nm thick Ti layer; finally, the top electrode is composed of TiN stacked on top of the Memristor. A SEM image of the device is shown in Figure 5.3.
The same team in CEA Leti, Grenoble, characterized an array of 1T1R OxRAM cells of the type described before for future use as variable resistance for replacement of conventional memories, such as SRAM and NAND Flash, for various applications including Neuromorphic Computing. The team



(a) V_{In} vs R2

(b) V_{mem} vs V_{In}

(c) V_{In} raise vs R2/R1

(d) τ_{refr} vs R3

Figure 5.2: In Figure a), the Membrane Potential V_{In} response, both in gain and time constant, increases as R2 increases, with a fixed input impulse. b) shows the response of the Membrane Potential to a DC input current with different R2/R1 ratios. c) plots the V_{In} instantaneous increment under the same input current stimulus, for different ratios of R2/R1. At last, d) is the behavior of the Refractory time-constant as a function of R3. From [53]

found out about the persistent problems related to Forming of the Low-Resistive-State (LRS) and the inherent stochasticity of the SET/RESET operation: as a matter of fact, if the Forming is carried out with too low compliance currents I_{CC}, the probability of reaching the LRS are low, while too high I_{CC} produces an almost irreversible LRS which disables the functioning of the Memory cell; on the other hand, OxRAM cells are shown to suffer from both Device-to-Device (D2D) and Cycle-to-Cycle (C2C) variability, which often results as a limiting factor for applications. For more details about the performance of the memory cells, please refer to [60], [61].

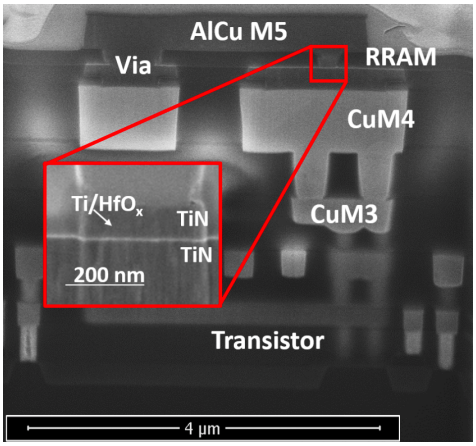
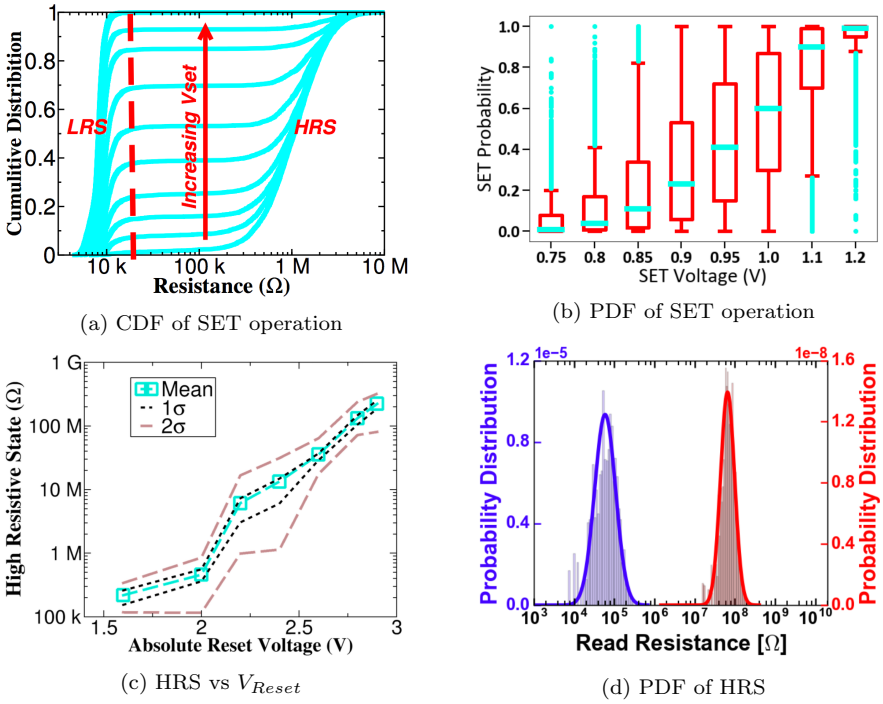


Figure 5.3: TEM images of the integration of the 1T1R cell in which the OxRAM Memristor is involved. The Actual OxRAM cell is signed by the red box and is stacked on top of the metal layer 4. From [61]

Concerning the employment of such cells as variable resistances in the Neuron model in Figure 5.2, a consideration on the Resistance values has to be done: since the time-constants of the real-world input signals, that are ideally used for analysis by System-on-Chips, are between $1\mu s$ and $100ms$, and assuming integrated Capacitances to be of the order of pF in the technological realization of the circuit, Resistances should be in the range $[100k\Omega, 1G\Omega]$ in order to match the time constants of the input signals to those of the Neurons. Such values are obtainable only at the High-Resistive-State (HRS) of the OxRAM cells. For this reason, the same team in CEA Leti characterized extensively the SET and RESET operations for the use of such a memory cell in the Neuron circuit. A 4kbit array - 4096 devices - is studied, applying SET and RESET condition multiple times to analyze both the D2D and C2C variabilities.



(a) CDF of SET operation

(b) PDF of SET operation

(c) HRS vs V_{Reset}

(d) PDF of HRS

Figure 5.4: Results of the characterization of the 1T1R OxRAM based cells to be used as R1, R2, R3 in the Neuron Model. a) and b) report the detail of the sub-threshold SET operation, which is a probabilistic process depending on V_{Set}. Images C) and d), instead, give the details about the RESET operation bringing the device in the High-Resistive-State. In particular, the OxRAM cell is shown to take values in the interval $[1M - 1G]\Omega$ with a Lognormal distribution with estimated Standard Deviation in the range [0.4-0.5]. From [55]

Operation of the OxRAM cells

As anticipated, Memristors are controlled by the two operations SET and RESET: in the first one, a current is provided to form a low-conductance Ti filament in the device resulting in a low equivalent resistance; instead, the opposite voltage is applied for the RESET operation after which the device shows low conductance. The two states characterizing the device are of course not digital, meaning that the device assumes real values of resistances in a given interval: nonetheless, in order to distinguish between LRS and HRS, a threshold is defined. During a SET operation, a device is said to be successfully be brought to LRS if its resistance is below $20k\Omega$. Since in the application of interest the SET operation is carried out in sub-threshold (with a voltage lower than the one assuring correct SET

operation), every time a voltage pulse is applied to the Memristor there is no certainty that the LRS would be reached: depending on the magnitude, duration and shape of the voltage pulse, the device has a certain probability of being SET correctly. Figure 5.4 a) shows the cumulative distribution of the SET operation: naturally, for higher voltages the LRS condition is reached more easily. On the same note, Figure b) plots the measured PDF for the same SET operation.
Once the device has been successfully SET in the LRS, it can be brought back to the HRS with a RESET operation: anyhow, the final resistance of the device is not known a priori but can be seen as randomly selected from a certain Probability Distribution Function (PDF) depending on the applied magnitude of the V_{Reset}. Images c) and d) from Fig. 5.4 report the measured results for the RESET operation, for which it has been calculated that the resulting HRS resistance is distributed with a lognormal function with a standard deviation of [0.4, 0.5]. This very feature, together with the stochasticity of the SET operation, usually described as "Cycle-to-Cycle" variability (C2C) and "Device-to-Device" variability (D2D) respectively are considered a major defect of the Memristive technology, preventing its application in conventional digital logic circuits.
Defects in memristors not only concern the operation of the single device, but also the behavior of different devices produced with the same technological process, also in the same process step. Figure 5.5 analyses the SET operation of the 4kbit array, monitoring the amount of successfully controlled devices over 100 operations. This procedure is carried out with three values of V_{Set} and shows two trends: the magnitude of the voltage has a high influence on the mean number of SET devices; each device behaves differently with respect to its neighbors. This means that despite all the devices being controlled by the same physical process - the formation of the high conductance Ti filament - the complexity of the procedure makes the results stochastic and possibly highly dependent on small details occurring in the production procedure and the Forming operation. One major concern is then to evaluate whether the D2D variability is an inherent property of the Devices or whether it could be dependent on external factors. In order to do so, the results of the SET operation of each device are tested with the SP800-22 benchmark for random number generators: this test looks for spatial correlation in randomly generated numbers in its 15 runs. From the analysis of the data from the 100 cycles of the 4096 devices, it is safe state that there is no spatial correlation between the devices, thus one cannot predict the behavior of the device based on its position in the matrix.

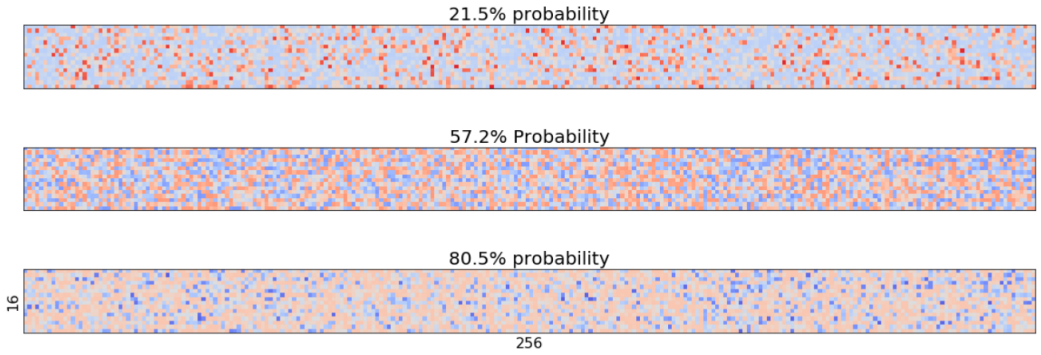


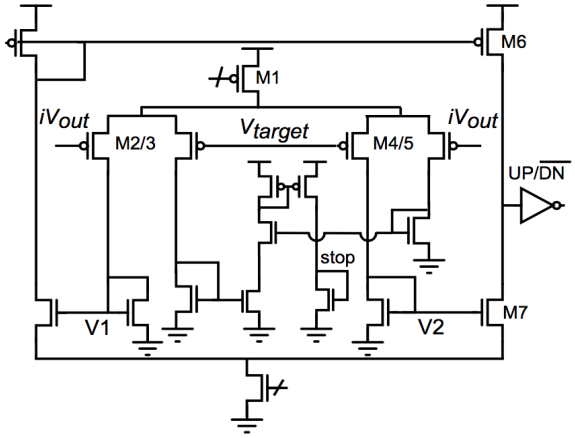
Figure 5.5: Heat-map of the SET probability of the analyzed OxRAM, for three cases of V_{Set} producing a different mean probability of SET operation: red color indicates devices over the mean probably, while blue pixels represent devices below the average. Difference in colors and brightness highlights high Device-to-Device variability, meaning that despite the global trend each device behaves differently. From [55]

Additional Circuits

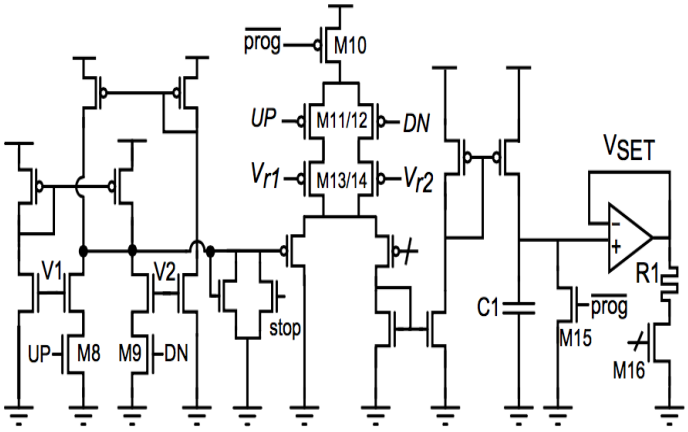
The operation of SET and RESET of the Memristors are controlled by a transistor which has to be in turn activated by additional circuits, actually implementing the algorithm of Intrinsic Plasticity. Despite this algorithm will be presented later, for now it is sufficient to consider that IP will change the resistances of the Neuron when, analyzing its activity, it will be measured firing far enough from a target frequency. As a consequence, the additional circuits have to implement two basic functions:

- to measure the activity of the Neuron and compare it to a target frequency
- to apply a voltage on the control transistor of the 1T1R cell according to the result of the measure

Because of that division, it is useful to analyze the IP circuit in two parts, reported in the following two figures. The combined action of both the circuits and their relatively simple layout confirms the integrability of Intrinsic Plasticity in usual Neuromorphic Computing platforms for on-line and on-chip operation.



(a) Circuit A: Intrinsic Plasticity



(b) Circuit B: Intrinsic Plasticity

Figure 5.6: Representation of the additional circuits Implementing IP algorithm on-line in the chip. From [53]

Circuit A is the one aimed at the comparison of the activity of the Neuron with respect to a target frequency: the activity of the Neuron (iV_{out}) is interpreted as the Calcium Current 4.4 in this

realization, thus a train of spikes integrated by a DPI circuit; the bias V_{target} instead implements the target frequency. When $iV_{out} > V_{target}$ then V_1 decreases and V_2 increases; moreover, the comparator will output DN (Down signal); the opposite happens for $iV_{out} < V_{target}$. The comparator determines whether the Neuron is firing above or below the target frequency, while $V1, V2$ encode the magnitude of the difference of activity.
This information is passed to Circuit B which actually performs the SET and RESET operation, according to the mentioned variables and some additional gates $V_{r1}, V_{r2}, prog$. The aim of this circuit is to charge a capacitor C1 and eventually emit a voltage pulse from the operational amplifier in order to perform the SET operation. After that, a RESET voltage brings the OxRAM back to the HRS.
It is to be noted that the bias V_{r1}, V_{r2} are able to control the condition of integration of the voltage on the Capacitor C1, implementing the rate of change of the voltage over C1 per unit of the difference of activity with respect to the target frequency. This results in a particular characteristics of the SET voltage with respect to the activity of the neuron, as shown in Figure 5.7:

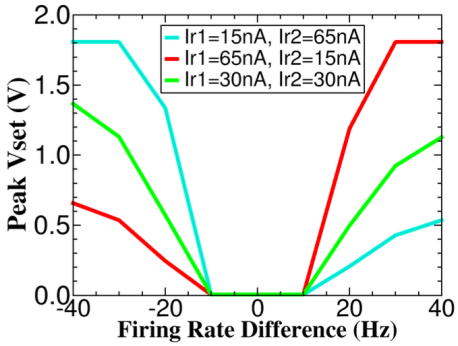


Figure 5.7: Characteristics of the SET voltage controlled by V_{r1}, V_{r2} through current I_{r1}, I_{r2} in M13/14 obtained by Circuit B. From [53]

Considering the effect of the SET voltage on the OxRAM cells, the SET operation can be modeled with the following sigmoid function:

$$y = \frac{1}{(1 + exp[(x - d) \times s])} \qquad for\ x > tol \tag{5.1}$$

where x is the absolut value of the difference of the firing rate with respect to the target, s the slope of the sigmoid, d the midpoint of the sigmoid, tol the interval around the target frequency in which the circuit does not apply any SET voltage.

5.3 IP Algorithm

In this very simple implementation of Intrinsic Plasticity, one aims at verifying the impact of the change of the input Resistances to the behavior of a whole Network. This follows the work done in [53] in which the single memristor has been equipped with IP and simulated: as expected, the change of input resistance produces a change of the firing rate and, following a certain learning rule, the author could achieve the peculiar exponential distribution of the firing rate found in experimental results from biological studies. However, controlling a single neuron is a relatively easy task to perform, while controlling the dynamics of a Network is more challenging: the non-linearity of the components - Neurons and Synapses - and the complex architecture of the Network make the dynamics of the

system difficult to control by acting on a single parameter on-line.
The Network is formed as follows:

- 35 Excitatory Neurons, as from Fig.5.1, with recurrent connections and connections to the Inhibitory ball
- 15 Inhibitory Neurons of the simple LIF model, with recurrent connections and connections to the Excitatory ball
- 15 Input Neuron, firing with a Poisson distribution, with connections to the Excitatory ball

Neuron Class	Excitatory (E)	35
	Inhibitory (I)	15
	Input (In)	15
Connections	E to E (recurrent)	15%
	E to I	75%
	I to I (recurrent)	50%
	I to E	75%
	In to E	75%

Table 5.1: Main parameters of the Network employed to test Intrinsic Plasticity

The aim of this implementation of Intrinsic Plasticity mechanism is to modify the Average Firing Rate (ARF) of the Network, specifically of the Excitatory neurons, to oscillate around a mean value - the target Frequency. It does that by controlling the firing rate of each Neuron and modifying its resistances R1 and R2 on-line, following simple learning rules. In the case the firing rate is close enough to the target frequency, a tolerance is defined in order to bypass the action of IP. As a consequence, IP can reach convergence when it succeeded in adapting the input resistances of the Neurons to the input of the Network. In terms of 'pseudo-code', the algorithm can be written in this way:

while *IP_on* **do**
  if $F_{neuron,i} > (F_{Target} + tol) \wedge P_{SET} > (sigmoid(\mu, \sigma))$ **then**
    $R_{1,2;i} = lognormal(R_{1,2} \cdot (1 - LR), SD)$;
  else if $F_{neuron,i} < (F_{Target} – tol) \wedge P_{SET} > (sigmoid(\mu, \sigma))$ **then**
    $R_{1,2;i} = lognormal(R_{1,2} \cdot (1 + LR), SD)$;
  else
    No resistance changes
  end
end

Algorithm 1: Implementation of Intrinsic Plasticity. P_{SET} is a random number uniformly distributed in the interval [0-1]. The algorithm changes the Resistances only if the Frequency of the Neurons is too far from the target value.

where P_{SET} is a random number uniformly distributed in the interval [0-1], F_{neuron} is the approximated output frequency of each neuron, *tol* is the tolerance of the algorithm around the target frequency F_{Target} and *LRs* are the Learning Rates of the resampling of the Resistances.
Since the Network evolves in virtually continuous time, defining a continuous-time evolution of the Frequency is not trivial: the problem was already introduced in section 4.2. The solution to that problem, in order to estimate the frequency of the neuron in continuous-time, is to consider the so-called Calcium Current, defined in Eq.4.4. That variable can be obtained in hardware by integrating the output spikes of the neuron on a leaky RC circuit, so that the spike train is then filtered by a

decreasing exponential kernel - given by the leakage of the capacitor. The exponential kernel allows the frequency function to be smoother and more suited for a continuous-time analysis than the train of spikes.
Given this evaluation of the frequency, it is possible to compare the output of the single Neuron to a target frequency value which is used for all the Neurons. Computing the difference of the output to the target, one can determine whether to reinforce or weaken the input section of the DPI Neuron: in the first case, the neuron integrates the incoming current with higher resistances, thus enhancing the voltage step provided by any pre-synaptic event. As a consequence, the threshold is reached with less number of input spikes and the neuron is likely to increase its output frequency. The opposite happens when the resistances are decreased in the input section.
As observed in the previous section describing the Neuron in Fig.5.1, R1 and R2 have slightly different roles in the integration of input current and leaking of Membrane voltage, so that, despite sharing the same learning rule, they are characterized by different learning rates.

5.3.1 Tuning the IP parameters

The algorithm of IP is simple, but the dynamics of Recurrent Network implemented by spiking Neurons is not. As a consequence, one has to accurately select the parameters governing IP so to maintain a healthy and homogeneous activity in the Network. IP is governed by a few parameters, so it is interesting to investigate the action of all of them to gain control over the algorithm.
The operation of the comparison between the firing rate of the neuron and the decision about whether to resample the resistances is performed on the basis of a **clock**, with a time interval dt_{IP}. This parameter is an important one when determining the performance of the IP algorithm: too fast resampling of the resistance will suffer from the fluctuation in time of the input neuron-group (firing with Poisson distribution), which has an impact on the firing rate of each neuron; on the other hand, too slow IP results in slower convergence time.
The **tolerance** of the algorithm is a fundamental parameter that allows IP to reach convergence. In the hardware implementation, the tolerance is determined by Circuit B (Fig.5.6 b)). The role of the tolerance is to fix the resistance to the Neurons whose firing rate is already in the allowed frequency window $[F_{Targte} - tol, F_{Targte} + tol]$. Because the input group is feeding the Excitatory neurons with a Poisson distributed train of spikes with mean frequency, it is tolerated that the Excitatory Neurons can vary their output frequency in time within this window in order to allow for convergence.
Circuit B also implements the slope of the probability of the SET operation respect to the distance to the target frequency, by means of the modulation of the SET voltage. For all the simulations in this work, the slope of the sigmoid SET probability function - called the d parameter - will be fixed to 0.5.
The **Standard Deviation** concerning the RESET operation determines the degree of control over the Resistances. It has to be reminded that the RESET operation is equal to sampling from a lognormal distribution with a Standard Deviation of [0.4, 0.5]. Considering the complex dynamics of the Recurrent Network architecture, this parameter is crucial for governing the whole Network. This parameter is related to the technological realization of the OxRAM cells, so that for the next tests of the IP algorithm, unless specified, the value of SD is fixed to the worst case: 0.5. Lower values result in higher control of the IP algorithm over the Resistances of the neurons and in turn better performance.
At every time-step of IP, the Resistances are resampled from a lognormal distribution whose mean is shifted in order to move the Resistance closer to its ideal value. To do that, **Learning Rates** are defined: there are different LRs depending on which Resistance they are acting and whether they have to increase or decrease its value.

Tuning Parameters	
Learning Rate R1, up ($LR1_{up}$)	percentage of increase for the mean in the RESET operation, with respect to the previous case, for R1
Learning Rate R1, down ($LR1_{dw}$)	percentage of decrease for the mean in the RESET operation, with respect to the previous case, for R1
Learning Rate R2, up ($LR2_{up}$)	percentage of increase for the mean in the RESET operation, with respect to the previous case, for R2
Learning Rate R2, down ($LR2_{dw}$)	percentage of decrease for the mean in the RESET operation, with respect to the previous case, for R2
Tolerance (tol)	difference of the neuron's firing rate within which the update of the resistance is not performed
SET voltage magnitude (d)	slope of the curve of the SET voltage with respect to the difference in firing rate

Table 5.2: Definition of the parameters involved in the process of tuning the IP mechanism

5.3.2 Metrics for the evaluation of IP

In order to quantitatively analyze the results of the Intrinsic Plasticity algorithm, some metrics are defined. The base for the evaluation of the performance of the algorithm is the definition of the frequency, which has to be defined over the continuous-time: for that purpose Eq.4.4 is the best choice, providing a smooth function which captures the behavior of the neuron on a time-scale of $500ms$, its time-constant. The metrics of IP reference to this equation when implying the frequency of the Neurons.
Taking the average of the frequencies calculated as in 4.4 for the Excitatory Neurons, one obtains the Average Firing Rate. At the same time, also the Standard Deviation of the Firing Rate of the Network is evaluated with the same frequencies. Instead, in reference to the Target frequency, the Average Distance to the Target Frequency is computed. Lastly, the Number of OxRAM switches is defined by the number of Neurons being subject to interventions of IP at every IP cycle.
The main metric is, however, the Convergence Time: namely, the amount of time the Network takes to reach convergence. To define that, one has to recall the aims of IP. IP has to control the Average Firing Rate as well the Firing Rate of all the Neurons, in turn, it has to lower the Standard Deviation of the Firing Rate. Moreover, it has to lower the number of switches of resistances of the Neurons, in order to be energetically efficient. As a consequence, Convergence is reached when the Average Firing Rate is comprised in a certain interval for all the time after the Convergence Time, the Standard Deviation is lower respect to the case of IP not applied and the Average Number of Switches after convergence is lower than a certain threshold.
From this definition, the basic metrics are summed up, with their notations:

- Convergence Time: T $[s]$
- Average Firing Rate: AFR $[Hz]$
- Standard Deviation of the Firing Rate: B $[Hz]$
- Average Distance to Target Frequency: A $[Hz]$
- Average number of OxRAM switches after convergence: C

Especially important for assessing the performance of IP are the constraints imposed to the definition of Convergence. Table 5.3 shows the values chosen for the metrics of IP:

Metrics for the evaluation of the IP algorithm	
Convergence time, T	Time after which the AFR is bounded within 30% of the Target Frequency and C is below 3 $T \in t \vert \forall t > T$, $abs(AFR - F_{target}) < 0.3 \times F_{Target}$, $C < 3$
Average Firing Rate, AFR	Average of the spiking frequency, based on the Calcium Current approximation, as in 4.4
Standard Deviation, B	Standard Deviation of the spiking frequency, based on the Calcium Current approximation, as in 4.4
Average Distance, A	Mean distance of the spiking frequency to the target value based on the Calcium Current approximation as in 4.4
Average number switches, C	Average number of OxRAM switches after the convergence time T

Table 5.3: Definition of the main quantities introduced to evaluate the efficacy of Intrinsic Plasticity

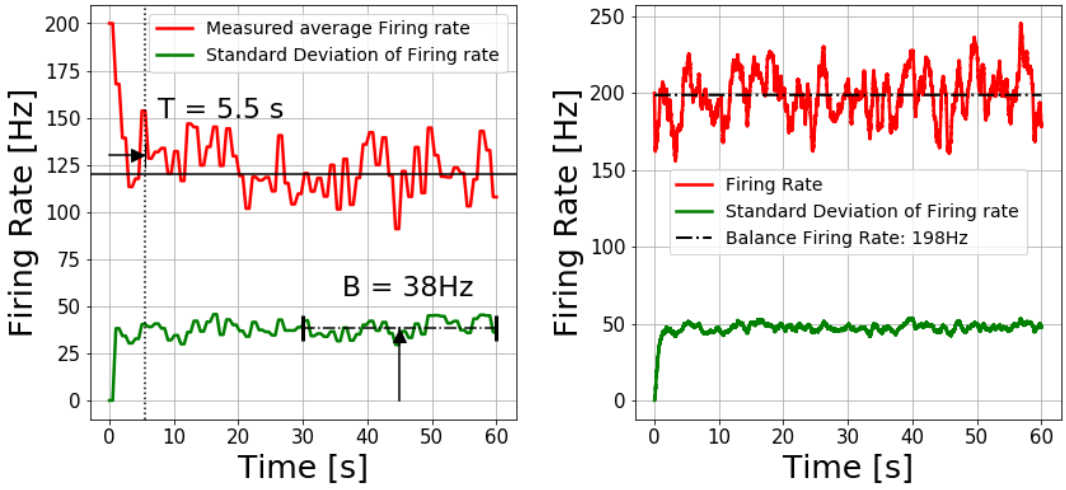
5.4 Results

Before turning IP on and trying to modulate the resistances of the Neurons in order to control the dynamics, the behavior of the Network has to be investigated for the case of IP not being applied. In that situation, the Network is simply composed of 36 quasi-LIF Neurons with exponentially decaying Synapses connecting excitatory neurons with a probability of 15%. An inhibitory population of 15 simple LIF neurons is connected to the Excitatory units with the same type of decaying Synapses. The input is present by 15 Neurons firing with Poisson distribution with fixed mean frequency and high degree of connectivity to the whole Excitatory group. The result of such an architecture is that the dynamics of the Network is controlled by the input, which forces the Excitatory units to fire at high frequency and the Inhibitory group preventing uncontrolled spiking of the excitatory group. In such a condition, one expects that the Network would reach a steady state in which the control of the input, being a Poisson train of spikes at a certain mean frequency, will determine the average frequency of firing of the Neurons. This steady-state value may be diminished by a stronger action of the inhibitory group, however, since there is no adaptation and learning for the connections, the Network is unlikely to escape from its equilibrium configuration once it is firing around a mean frequency.
This is what is observed in Fig.5.8 b), in which the Average Firing Rate is plotted as a function of time together with the Standard Deviation. Despite the Average Firing Rate (AFR) is following a chaotic oscillatory behavior, one can say that the Network stabilizes around a certain mean frequency, in that case of about 200Hz. The reasons for such behavior are to be attributed to the recurrent connections, which may give rise to oscillators, and of the input that, considering the frequency it produces in time, is a white noise around a certain mean frequency. Moreover, the information of the Standard Deviation confirms that the neurons distribute in the frequency spectrum around a certain mean. The larger the standard deviation, the higher the impact of the internal properties of the Network and the noisy nature of the input in forming the oscillation of activity.
To be considered one aim of Intrinsic Plasticity is to adapt the neurons to fire around a certain mean frequency, thus a successful implementation should not only target a certain frequency, but also reduce the Standard Deviation of the firing rate of the whole network, meaning that the neurons all fire closer to the target rate. On top of that, IP should be able to accomplish that in short times and performing a few switches of resistances to adapt the frequency of the neurons. That is why a final benchmark for IP will be to test the consumed energy of the Network when IP acts so to lower the natural balance activity compared to a Network that does not apply IP. If the energy consumed by the IP-containing Network will be shown to be lower than the one consumed by the standard case, IP can legitimately be proclaimed as a convenient feature.

Standard synapses

The first results to be presented are the ones related to the Network in which the synapses connecting Excitatory neurons are of the simple Exponential type.

In Fig.5.8 a), the best example of IP has been shown so to clarify what this algorithm is capable to achieve: the target rate is achieved in a very short time of convergence of the algorithm, said $T = 5.5s$, starting from the condition of equilibrium of the Network. This fast decay is obtained by employing large learning rates for decreasing the Resistances when the frequency of the Network is above the target, $LR_down = 0.1$, meaning that the sampling of the Resistances is from a lognormal distribution centered 10% above the previous value. In this way, as confirmed by Fig.5.9 b), the resistances are quickly decreased of more than $300M\Omega$ for the case of R1. After the quick decrease, most of the Neurons approach the target rate and thus eventually require to increase back the resistances so to avoid to continue the diminishing of the spiking rate. This resetting of the resistances is controlled



(a) Average Firing rate and Standard deviation

(b) Average Firing rate and Standard deviation

Figure 5.8: Comparison of AFR and its Standard Deviation of the Network in which IP is active, a), and absent, b). In both cases, STDP is turned off. IP quickly brings the AFR close to the target value

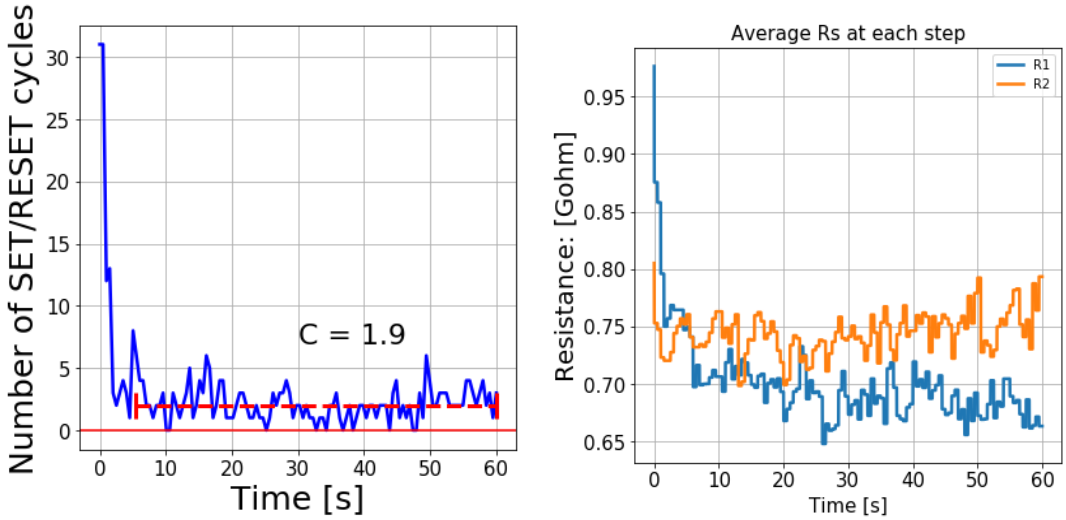
by the other parameter $LR_up = 0.2$, which is typically a quite high value: the reason for that is the shape of the lognormal distribution, for which low values of resistances could be selected in the RESET operation even though the mean of the distribution is set at very high values.

The quality of the algorithm is certified by Fig.5.9 c) which reports that the number of OxRAM cells switches carried out when the spiking frequency has settled around the target rate is very low. On average, less than 2 RESET operations per cycle - which occurs every $dt_{IP} = 500ms$ - are required so to correct the oscillations of the frequency of the Neurons. This means that the algorithm has succeeded in learning the values of resistances which make the Network fire on average at a certain target frequency. Since the convergence time is very short, 6 seconds, i.e. only 12 cycles of operation of IP, the algorithm can also be considered very effective. However, one can argue that biological Intrinsic Plasticity is a much slower process, with time-constant of minutes: this period of action can in principle be reached retuning all the parameters of the algorithm so to make the resistances of the Neurons vary their values less prominently each cycle. In this view, the main parameters to consider are the Learning Rates and the probability of SET operation.

STDP synapses

In this case, instead, the Network is an updated version of the one simulated for the last presented results, with the synapses between excitatory neurons still of the exponential type, but with the weights being ruled by Spiking-Time-Dependent-Plasticity. Under a certain point of view, the previous case of Standard synapses is totally identical to the STDP-exp - as one may call them - less for the weights which are all fixed to 1. As for the Input, this is presented in the same way as the previous case.
Despite the fact that the Network is very similar, one should not expect that the impact of the STDP-exp synapses may be negligible for the dynamics of the network: STDP is known to have a tendency in forming clusters of active and inactive Neurons. Anyhow, since IP has been shown to be so effective, one expects that a fine-tuning of the parameters of the IP algorithm may achieve success as well.

The similarity between Fig.5.8 and Fig.5.10 is a symptom that IP is almost as well performing with STPD-exp synapses than with Standard Exponentially decaying ones. This is a good feature,

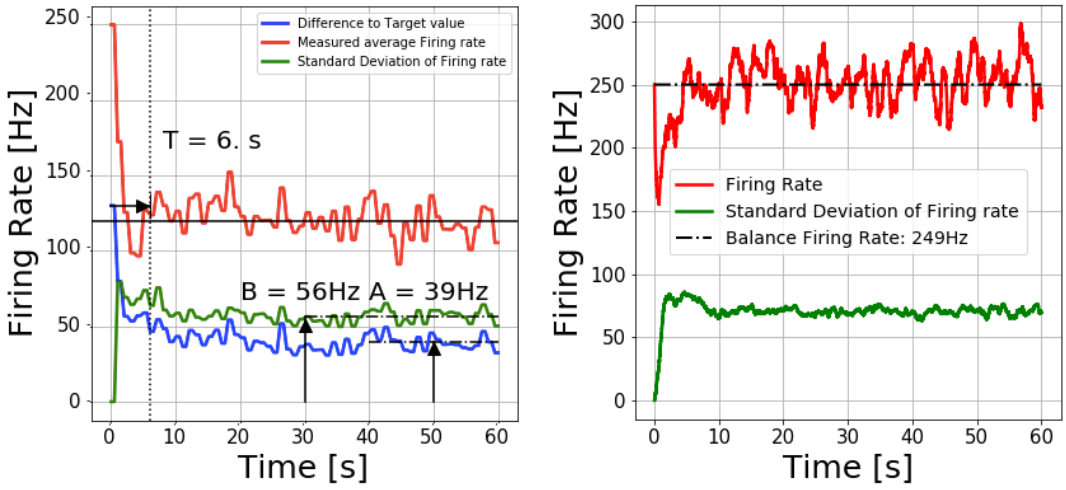


(a) Average Number of Switches after convergence

(b) Behavior of R1 and R2 in time

Figure 5.9: Results of the application of IP on the resistances of the Neurons. STDP is switched off for both cases. In a), the number of switches rapidly converges to less that 2. In b), IP is acting to decrease both R1 and R2 and then keep adjusting them to maintain the Network around the correct firing regime

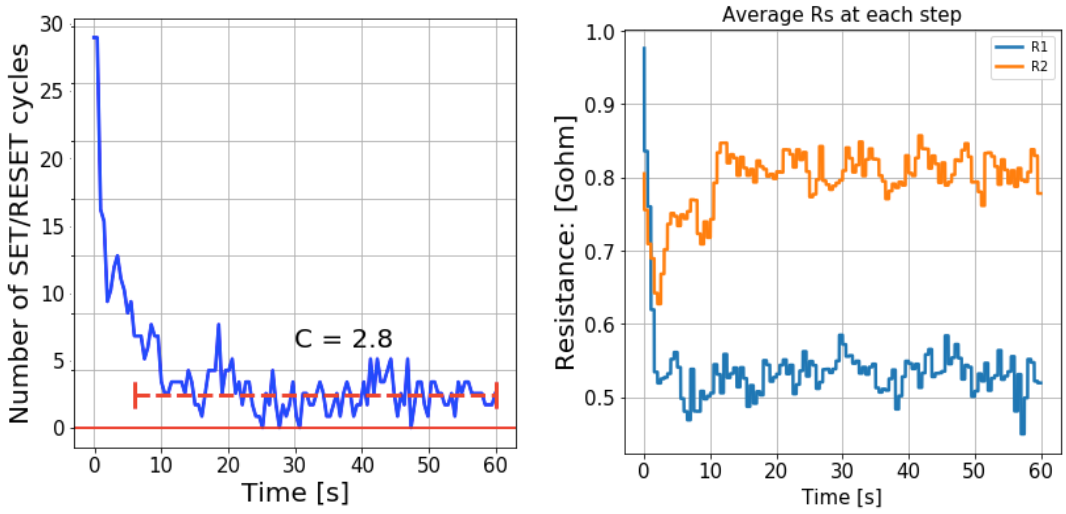
as a Plasticity algorithm should ideally be effective in different contexts, for example with different types of synapses.
Moreover, the STDP case allows for the evidence of an additional benefit of IP, which was not as clear for the case of Standard Synapses: the diminution of the Standard Deviation of Activity, the parameter B. This is a rough indicator of how much the output frequency of the Neurons is spread with respect to its mean: when it is high, Neurons are behaving differently, some of them being highly active, some other spiking rarely. IP can be said to have a high degree of control on the Network just if it is able to reduce the gap of the Average Firing Rate (AFR) with respect to the target but also if that is true for most of the Neuron in the Network, thus if B has been lowered. Evidence is that, as expected, the Standard Deviation of the Neurons is way higher when STDP is shaping the weights of synapses, respect to the case of Standard Synapses. However, the clustering effect of STDP seems to be counteracted by IP which probably increases the resistances of the Neurons whose input weights have been lowered by STDP. Of course, STDP remains a stronger agent in determining the dynamics of the Network and the B value reached with STDP active - 56Hz - is higher than the one obtained before - 38Hz - without STDP. Nonetheless, the relative variation, considering the case of IP with respect to the one without IP is more prominent for the STDP-exp synapses.
For what concerns the convergence time T, that is of the same order of magnitude of the previous case: one could probably generalize that these values are close to the best achievable with these circumstances (distribution for the RESET operation, Network operation). Anyhow, as it is the case in these complex non-linear systems, an even finer tuning of the parameters could potentially improve the performance of the IP algorithm.



(a) Average Firing rate and Standard deviation

(b) Average Firing rate and Standard deviation

Figure 5.10: Comparison of AFR and its Standard Deviation of the Network in which IP is active, a), and absent, b). In both cases, STDP is turned on. IP quickly brings the AFR close to the target value, despite the presence of STDP



(a) Average Number of Switches after convergence

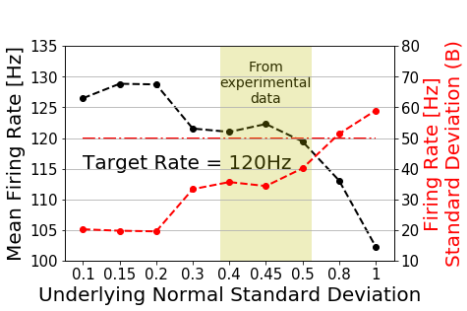
(b) Behavior of R1 and R2 in time

Figure 5.11: Results of the application of IP on the resistances of the Neurons. STDP is switched on for both cases. In a), the number of switches rapidly converges to less that 3. The presence of STDP makes the Network a little less stable and IP requires more Resistances switches to correctly operate

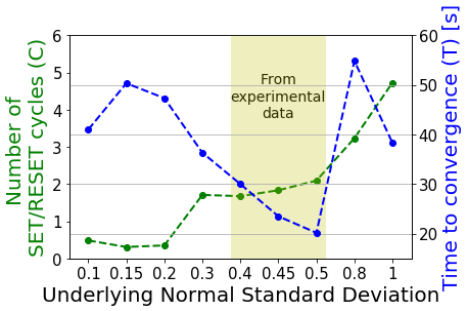
The average number of switches after convergence is increased for this case: the Neurons probably require more Resistance switches for compensating the polarizing action of STDP for the weights of the Synapses. However, this number is still to be considered very low and acceptable for defining the convergence of the algorithm. Moreover, the plot of the evolution of the resistance of the Neurons in time is very similar to the one shown in Fig.5.9, so to confirm the correct behavior of the algorithm.

Analysis of the parameters of Intrinsic Plasticity

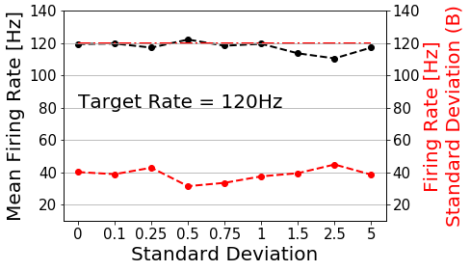
Now that it has been demonstrated that the algorithm can achieve great performance and the prefixed goals with a fine-tuning of the parameters, it is interesting to move away from optimal conditions and stress the algorithm out of its comfort zone. That means to sweep one parameter at a time in a certain range and see the effect on the results. In particular, it is interesting to analyze the parameters related to some characteristics of the device under analysis, the OxRAM cells. Since analyzing the behavior in time of every network would be very time consuming, the analysis will rely on the main figures of merit defined up to now: AFR, B, T, C. The results are collected together in Figure 5.12. Every point in the following plots will represent one of the metrics averaged from the data of three simulations run for 60s. In each simulation the only variable is the input, which is a Poisson distributed train of spikes. For what concerns the Network, that is one already presented with Standard synapses.



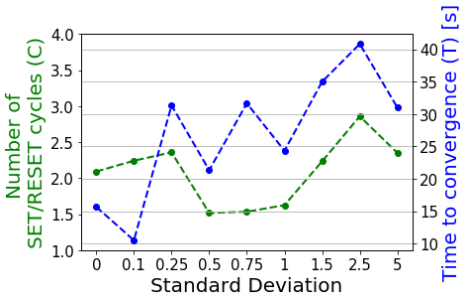
(a) AFR, SD vs RESET distribution's Standard Deviation



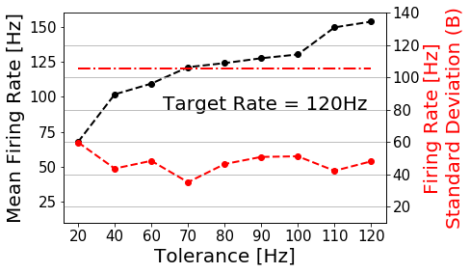
(b) # switches, T_C vs RESET distribution's Standard Deviation



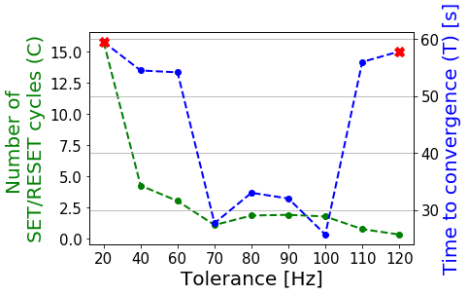
(c) AFR, SD vs D2D's Standard Deviation



(d) # switches, T_C vs D2D's Standard Deviation



(e) AFR, SD vs Tolerance



(f) # switches, T_C vs Tolerance

Figure 5.12: Effect of some of the parameters related to the IP algorithm and the OxRAM cells technology. The yellow shadows indicate the experimental parameters of the OxRAM cells.

First of all, the Standard Deviation of the distribution of the Resistance values, which is certainly the most important factor in the algorithm. Ideally, one would like to be able to control the OxRAM cells with high precision in order to select the resistance value at will, with low errors. This would imply the Standard Deviation of the probability distribution of the RESET operation to be small, ideally zero. However, limitations related to the technological process do not allow for such precision in resetting the resistance values, so that the probability distribution of the RESET operation is characterized by a standard deviation value of [0.4, 0.5] from experimental data. Advancements in technological processes may reduce this factor and it is interesting to measure what would be the advantage of such progress. For the limit of time, since tuning the parameters of the IP algorithm is a very time-consuming task, the algorithm is tested with the parameters optimized for SD = [0.4, 0.5], now in a wider interval [0, 1]. One expects that the best results come from the values of SD for which the algorithm has been properly tuned, but with some signs of improvements for lower SD values. Moreover, for too high values of SD, the algorithm does not have control over the resistances, so that, at some point, it is expected not to converge. As a matter of fact, for $SD > 0.8$ the distance from the target becomes clearly important and convergence time also larger. Instead, for lower $SD < 0.3$ the convergence time is also large and the distance to the target higher than for the case of $SD = [0.4, 0.5]$ but there is an aspect to underline: the Standard Deviation of the firing rate of the Neurons - B - is considerably the lowest. This means that the algorithm, in that case, has been able to tune the frequency of the Network to a uniform rate but, since the parameters were not optimized, that resulting frequency was not perfectly coincident with the target one. This is verified by the plot of the AFR which exceeds the target value by about $7.5Hz$.
Remaining on the analysis of the model of the device, OxRAM cells also suffer from Device-to-Device variability. This has been stated in Fig. 5.5 and affects the SET operation of the cells in an uncorrelated manner with respect to the position of the cell in the matrix: as a consequence, the SET of the OxRAM can be viewed as a probabilistic event, governed by a sigmoidal distribution as in Fig. 5.4. The midpoint of such a sigmoid function can be controlled based on the setting voltage V_{Set}. However, due to the D2D variability, the effect of the V_{Set} is not uniform across different devices, which ideally would be reset only as a function of the frequency difference with respect to the target. Such variability can be simulated by applying a gaussian noise to the midpoint of the sigmoid. The larger the Standard Deviation of such Gaussian Noise, the stronger the effect of D2D variability. In Fig.5.1 c) and d), the results of simulations in which this standard deviation is varied in the interval [0,10]*Hz are summed up. Surprisingly, despite a high variability of the sigmoid's midpoint, the effect of the D2D is marginal: that means that this problem related to the fabrication of the devices is not a major concern for the application in the IP algorithm. However, despite the previsions were much more negative, D2D is seen increasing the convergence time by more than 70%. Since the AFR at convergence is very close to target also at high D2D variability, this effect is not considered critical for the correct functioning of the algorithm.
Lastly, correlated to the previous analysis but with a different aim, the analysis of the tolerance: this parameter does not have to do with the device itself but with the magnitude of the V_{Set} in relation to the distance of the frequency of the neuron to the target rate. In turn, this allows selecting a range of frequencies around the target rate for which the algorithm does not intervene. This interval is called "tolerance". The analysis of this parameter evidences that a too low value of tolerance results in an over-correction of the values of the resistances, not allowing the Network to ever converge. A tolerance of 70Hz seems to be the best compromise for performance. Further increasing the value of tolerance, the algorithm is not able to correct the values of resistance in a strong enough manner. For both the extreme cases the effect is that the convergence time is increased up to the point of non-convergence.

Power Consumption

Lowering the average firing rate of the Network comes with the trivial advantage that the energy consumed is reduced as the total number of spikes in a given time is also reduced. As a consequence, it appears that IP, when used for lowering the activity of the neuron and thus acting as a homeostatic mechanism, automatically reduces the power consumption of the whole Network. However, one should not forget that the SET/RESET cycle imposes a penalty in energy which has to be considered when computing the total power of the system. It has thus to be calculated whether the sum of the power due to the spiking and the one due to the OxRAM cells still is an advantage respect to the case in which IP is not active. By doing so, one counts the spike events and multiplies by a fixed amount of energy, estimated by experiment. In this case, this value has been chosen to be the Energy per spike measured for the DPI neuron in the DYNAPSe chip [37], from which the neuron model employed for the simulation takes inspiration. Instead, it has been measured that on average the OxRAM cell consumes a much lower amount of energy [55] under standard programming operations (SET: $V_{set} = 2V$,$V_{gate} = 1.3V$ and RESET: $V_{reset} = 3V$, $V_{gate} = 3V$). Since the difference of energy per event is of one order of magnitude in favor of the SET/RESET cycle and since those events happen two order of magnitude less likely in time, it is clear that the power consumed to manipulate the OxRAM cells is much lower than that coming from the spikes.

$$E_{spike} = 800pJ \tag{5.2}$$

$$E_{SET/RESET} = 50pJ \tag{5.3}$$

Figure 5.13, showing the comparison between the energy consumed in average by a single Neuron with IP and without IP, clarifies the discussion about the advantage in power consumption.

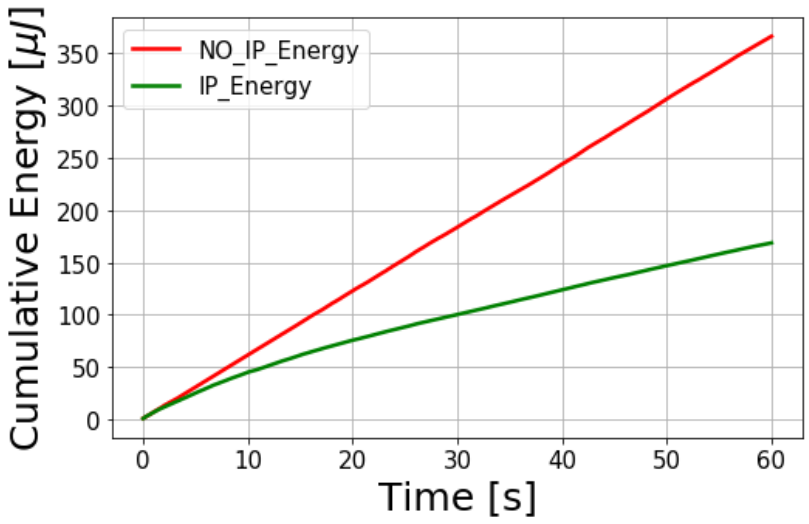


Figure 5.13: Plot of the average consumed Energy in time for a single neuron with IP, green curve, and without IP, in red. The green plot, despite an initial transient with higher slope, always lies below the red one, stating a clear lower consumed power during the run.

After all, since the energy related to the OxRAM cells is so low, IP turns out to be a clear advantage for the energy management of the system: after an initial transient phase, in which the spiking frequency of the Network is still high and the energy of the SET/RESET cycles contributes consistently to the consumed power, the low number of switches and the reduced spiking after reaching convergence makes the green curve - associated to IP - always remains well below the red one - related

to the case of no IP. As a consequence, IP can be retained to be a good candidate feature for Recurrent Spiking Neural Networks in order to reduce the power consumption and possibly enhancing the properties of the system.

Chapter 6

Self-Organized-Spiking-Network

In a well known-paper from 2009, Lazar proposed the SORN, the Self-Organized-Recurrent-Network [56]: an Echo State Machine, composed of Heaviside activated nodes, operated in discrete time, in which the Neurons in the Reservoir are equipped with 3 Plasticity mechanisms (STDP, IP, SN). Combining the action of the 3 features, the Network was shown to improve the performance in some temporal tasks. Moreover, it was experienced that in order to get the boost of performance, the 3 features had to work together, since by acting alone they were of no help in improving the performance. Since the inspiration of the 3 Plasticity mechanisms comes from Biology, it is interesting to investigate their impact on a more biologically plausible architecture, based on SNNs, namely the Liquid State Machines. The similarity with the original SORN involves the STDP and Normalization features, while IP is applied with the already mentioned algorithm, instead of the SORN's threshold manipulation. Moreover, the LIF model employed for the task is characterized by Adaptation, a further homeostatic Neural feature. All together, they are expected to boost the performance of the Network in two temporal tasks: the Counting Task (CT) and the Occluder Task (OT). The Spiking based Network with the 3 Plasticity mechanisms has been called Self-Organized-Spiking-Network (SOSN).

6.1 Structure of the Network

For this task, different Liquids have been generated in order to compare their performance. Networks of the size of 200 and 400 Neurons are tuned to respond to the incoming signals with sparse activity. Common features of all the Network configurations are the 5% of Recurrent connections between Excitatory neurons, the 10% of connectivity between Excitatory and Inhibitory and vice versa and the Input group which is composed of 10% the size of the Network. Some additional parameters are customized in order to maximize performance.
The input signals are a number N_S of different symbols which are projected to the Liquid in this way: the same Poisson input group targets different "Sensory" subgroups in the Reservoir, each of which is independent to one other, representing an input signal. Every signal is spread through the Network and finally reaches the output, which is formed by as many classes as needed for the task. The output layer is learned by Logistic Regression since the output is required to classify one signal at the time.

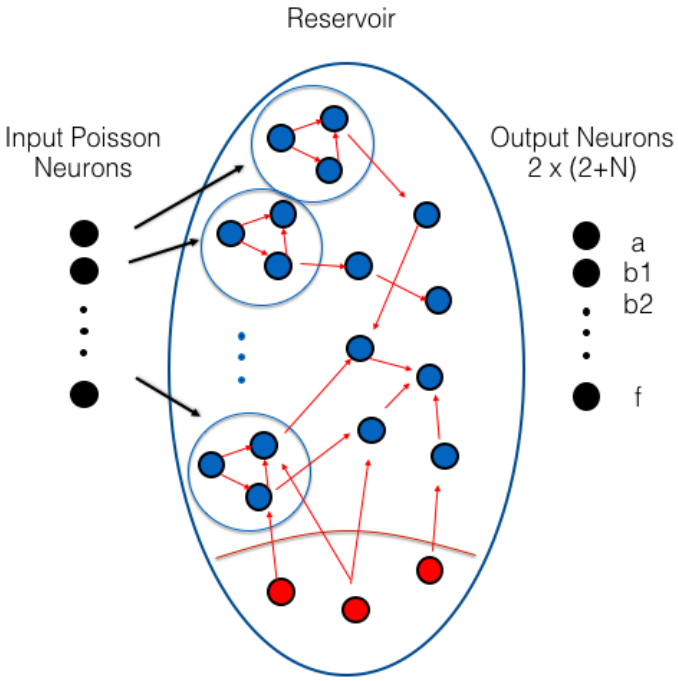


Figure 6.1: Scheme of the LSM for the Counting and Occluder Tasks. The input group fires with Poisson distribution towards different sub-groups in the Reservoir, one at the time. Each of the sub-groups representing an Input signal. The output group is composed of as many neurons as needed for the classification/prediction Task.

The Task is performed in this way: the Training and Testing phases are common for each of the configurations of the Network. Instead, in case one were to evaluate the effect of the 3 Plasticity mechanisms, a Pre-Training was performed at first. Then, the Weights and the Resistances in the Reservoir were frozen and maintained constant in the following Training and Testing phases. The Networks in which the Pre-Training was performed are called, as already said, *SOSN* (Self-Organized-Recurrent-Spiking-Networks), while in case of no Pre-Training the Network is called *Static*.

6.1.1 Plausibility with respect to Technology

When dealing with Plasticity mechanisms, in particular Intrinsic Plasticity, the requirement is to conserve a certain level of plausibility with respect to the technological realization of circuit and devices. This is because the Neuromorphic Computing approach is, as explained in previous sections, particularly slow when running on conventional computers. Then, it is interesting to develop algorithms that can be implemented on dedicated hardware. Intrinsic Plasticity's algorithm, presented in Chapter 5, certainly is in theory reproducible in Integrated Circuit. In Fig.5.6, the Circuit implementing the Neuron is presented, while Fig.5.3 depicts the realization of the OxRAM cells involved in the implementation of the algorithm.
The utilization of such a method involves a number of complications introduced in the algorithm that emulate the behavior of a real device built with the architecture and specifications of the mentioned circuits and devices. Concerning the circuit, this is easily implemented by a differential equation which describes the behavior of the Neuron, being of the Leak-Integrate-and-Fire type.

$$\frac{\partial v(t)}{\partial t} = \frac{(R_1 // R_2) \times I_{in} - v(t)}{R_2 \times C} \tag{6.1}$$

where $v(t)$ is the membrane potential, R_1, R_2 the two OxRAM cell's resistances, C the membrane capacitor and I_{in} the input current.
The complications are instead introduced by the operation of the two OxRAM cells, which have to be first successfully SET and then RESET in the High Resistive State. The SET operation is relatively easy, governed by a sigmoid probability function which implements the probability of the device to be SET, given its Firing Rate. During all the simulations, the parameters related to the sigmoid have not beeg changed, as this phase of the algorithm has been proven not to be crucial for the correct behavior of the algorithm.
Instead, the sensitive parameter of the algorithm is certainly the Standard Deviation of the lognormal distribution, modeling the RESET operation. As a matter of fact, the $SD_{lognorm}$ quantifies the degree of order with which the Resistances of the Neuron are reprogrammed, after a cycle of Intrinsic Plasticity. It is trivial to understand that high disorder in that phase results in a less controlled dynamics of the Network since the Neurons will operate with resistances which are far from the ideal values computed by the algorithm. As a result, IP will be beneficial only for certain values of $SD_{lognorm}$ lower than a certain threshold. This limit has been found to be of $SD_{lim} = 0.1$: compared to the experimental data from [60] [61] locating $SD_{lognorm}$ in the interval $[0.4, 0.5]$, this value represents the behavior of a much more precise and ideal device. OxRAM cells are in an experimental phase and are not Very-High-Scale-Integrated devices in ICs in the market. This means that research is ongoing in order to improve the technological processes required for their integration with conventional electronics. Moreover, such a value of Standard Deviation does not represent the case of an ideal device that would have $SD = 0$. The conclusion is that a degree of plausibility with respect to technology is maintained, despite the algorithm not reaching the expected results with the experimental data. Also, the following results are obtained with $SD_{lognorm} = 0.05$ which is the value which allowed to reach the best results, so that the advantages and potential of the IP, combined with the other forms of Plasticity, would be best highlighted.
In order to recap about the technological plausibility, this is the procedure that is carried out when setting up the Network for the experiments.

- Random connections are formed in the Liquid
- Resistances are evaluated at first: this happens with a RESET procedure, with $SD_{lognorm} = 0.1$, according to the technology of OxRAM cells
- IP is eventually operated as described in Chapter 2, exploiting the dedicated circuits and the mentioned SET/RESET operations. The SET phase occurs with the mentioned sigmoid function, while the RESET phase is controlled by the Standard Deviation of the lognormal distribution

In order to sum up the parameters related to the technologically realization of the plasticity mechanisms, the following table is presented:

Sigmoid (SET)			Lognormal (RESET)	
S	D2D	$tol_{sigmoid}$	$SD_lognormal$	LR
0.5	2Hz	20Hz	[0.05,0.1]	0.001

Table 6.1: Parameters related to the technological realization of the IP algorithm. The SET operation is characterized by the SET voltage, whose action is modeled with a sigmoid distribution, characterized by its skewness and Device-to-Device (D2D) variability. The RESET phase, instead, is governed by the Standard Deviation and the Learning Rate for the resampling of the new value of the equivalent resistance.

The S parameter models the slope of the sigmoid function and can be set in the interval [0,1], thanks to the flexibility of the dedicated circuits developed by Payvand et al. [53]. The Device-to-Device variability deals with the shift of the center of the sigmoid due to the fact that the probability for the SET operation is not homogeneous for all the OxRAM cells. In chapter 5 it has been shown that the IP algorithm is robust against D2D, which in this case was set at 2Hz. Concerning the RESET operation, it has been already gone through the reason why the $SD_{lognormal}$ has been rescaled to $[0.05, 0.1]$, while the Learning Rates (LR) are maintained in the order of magnitude of 0.001. This shift of resistance is obtained by properly scaling the Voltages applied during the RESET operation.

6.2 SOSN's characterization

The experiment is carried out with two different Networks, designed with the same paradigm, but with different number of Neurons. When building Networks of different sizes, the same type of dynamical behavior was aimed: the network had to be deterministic, such as the ones in the SORN paper [56], and the activity had to be modestly sparse, such that about 50% of the neurons would be active under an input signal. The requirement on the deterministic dynamics seemed to be crucial in order to avoid that the evolution of the internal dynamics of the Network would be uncorrelated with the presented input. This is crucial in order to correctly perform classification of the presented inputs. A signal of this requirement being achieved is that the Network stops its activity quickly after the input signal is removed. Moreover, that is also evident by the fact that the Liquid responds in a similar manner to the same inputs, consistently in time.
In particular, two Networks have been tuned: one with 200 Neurons, the other with 400. The main parameters related to the two configurations are presented:

Network's parameters	p_{EE}	p_{IE}	p_{EI}	tau_{EE}	tau_{IE}	tau_{EI}	C_E	I_{EE}	I_{in}
200 Neurons	0.05	0.15	0.1	15ms	15ms	15ms	40pF	2.6nA	0.2nA
400 Neurons	0.05	0.15	0.12	15.5ms	15ms	15ms	40pF	2.6nA	0.075nA

Table 6.2: Parameters related to the two main Network configurations. Both share the same philosophy in forming the Liquid, with small difference in some of the parameters in order to obtain a similar level of dynamics expressed by the Liquid.

So, as it is clear, the two configurations are basically similar with minor differences in some of the values in order to balance the dynamics and guarantee comparable operation.

6.3 Effect of Plasticity

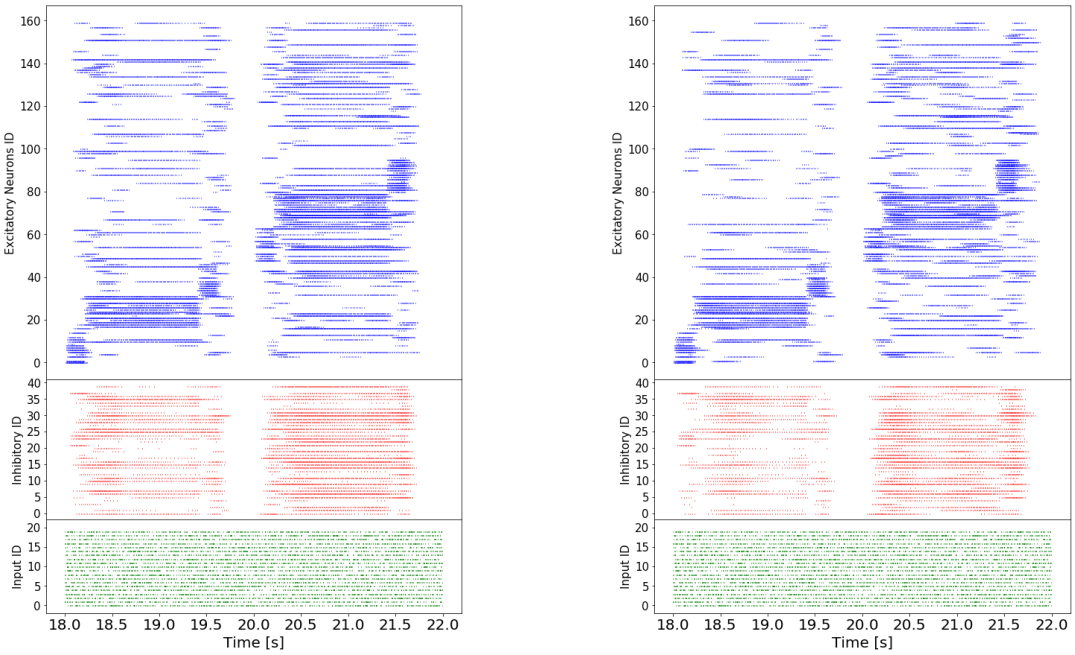
Before jumping to the results of the two temporal tasks, one has to understand why and how the Plasticity mechanisms can potentially improve the performance. Roughly speaking, it is known that STDP tends to strengthen the activity of already firing neurons and suppresses the ones which are less likely to fire. Synaptic Normalization is a break to the action of STDP, being a homeostatic mechanism. Intrinsic Plasticity should potentially increase the entropy of the Network, activating those neurons which are mostly silent and shutting off those which are too active. Combining all together, the Network should tend to increase the amount of expressed information by its neurons and the reaction of the Liquid should differentiate the SOSN to the Static cases.
The parameters governing the Plasticity mechanisms are as important as the hyper-parameters of the Network. Only a correct setting of these parameters allows to improve the quality of the Liquid and

in turn the performance of the Network in the given tasks. The process of tuning leads to these values:

STDP			Normalization		Intrinsic Plasticity			
A_{pre}	A_{post}	τ_{STDP}	dt_{SN}	Spectral Radius	LR_{up}	LR_{dw}	dt_{IP}	Target
0.001	-0.001	10ms	50ms	1	0.001	-0.001	1s	40Hz

Table 6.3: Main parameters related to the Plasticity mechanisms in SOSN. STDP is based on a symmetric exponential decay profile. Normalization is applied in order to prevent saturation of synaptic weights. IP aims at maintaining the natural firing rate of the Network, improving its dynamics.

STDP is tuned in order to slightly modify the weights every time it is involved, thanks to its low Learning Rates. Nonetheless, it still leads to synaptic normalization so that Synaptic Normalization is necessary to prevent that. SN occurs every $50ms$ and sets the weights in order to scale the Spectral Radius (SR) of the Excitatory weights matrix to 1. Setting the input weights to each Neuron normalized to 1 assures this condition. This very measure is crucial for Echo State Machines, as demonstrated by Jaeger [42], but it is unclear whether it produces similar effects on LSMs. As a matter of fact, despite setting the SR to 1, the impact of the input signal to one Neuron can still be modulated by the input resistances or the synaptic unit current. Nevertheless, this measure has been adopted in order to stay coherent with respect to SORN and allow for better reproducibility of the results.
Lastly, IP: it is the most complex Plasticity mechanisms in the lot, governed by multiple parameters. The most important consideration has to be done on the Standard Deviation of the resampling of the Resistances. According to the experimental data of the OxRAM cells used in the last chapter, that value is measured to be in the interval $[0.4 - 0.5]$. However, using this values, it has been found that the dynamics of the Network was scrambled too much: the low control on the values of the Resistances inevitably lead to the loss of the classification ability of the Network, while the continuous action of IP still allowed to maintain the target rate. However, since the performance on the tasks was compromised when using high values of Standard Deviation, it has been chosen to gain control over the resampling of the Resistance by lowering the Standard Deviation below 0.1. The results presented in the following are obtained with $SD = 0.05$. Of course, by setting $SD = 0$ one could in principle reach optimal performance, but that is of no interest: no device could potentially set its resistance with such precision.
For what concerns the other parameters of IP, it has been chosen to make it act with low frequency: after all, in biology, IP is shown to act on the minutes to hours time-scale. Lastly, the Learning Rates are on the same order of magnitude of the ones of STDP. Changes of the resistance should not be abrupt, since the input section of the neuron is crucial in determining the dynamical behavior of the single node and, in turn, of the Network.
The following figure highlights the effect of the 3 Plasticity mechanisms on the Network.



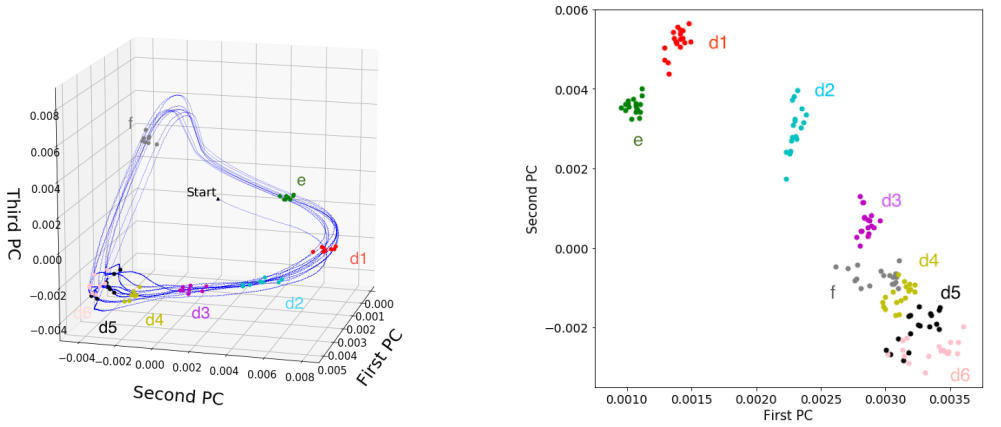
(a) Raster Plot, Static Network

(b) Raster Plot, SORSN Network

Figure 6.2: Raster plot representing the spike patter of each neuron in time, for both the Static and the SORSN Network. The richness of the dynamics is slightly increases for the SORSN

The two examples above show the Raster plot of the Liquid when two words $[e, d, d, d, d, d, d, f]$, $[a, b, b, b, b, b, b, c]$ are given to the network after an initial period of 2 seconds, in which the Reservoir forgets its initial state and enters in its operative regime. Between the two words, a space without inputs is present, so that the 'short term memory' wouldn't affect the dynamical state produced by a word based on what the network had experienced before.
Often in temporal tasks, one wants to process the input taking into consideration the time information it contains. In this case, the letters d, b are repeated many times and it is interesting to test whether the Liquid is able to count how many repetitions of the same letter have been presented. It is reminded that in order to classify the 'b's and the 'd's with different classes, the state of the Reservoir has to evolve enough during the time the same signal is shown. This means that the spiking pattern of the neuron in the Raster Plot has to present different configurations under the same input. It is thus clear that the neurons which are either constantly spiking at a fixed frequency or always silent, bring no useful information. The Static Reservoir presents some examples of those neurons. Plasticity is tuned in order to enrich the dynamics, shutting off the constantly firing neurons and activating the silent ones. In the figure on the right, representing the SOSN Network, especially on the word on the left, it is clear that the dynamics regime is more chaotic, allowing for a better classification of the same signal into different classes.
It is also clear that a too chaotic regime wouldn't allow the Network to distinguish the different inputs: the firing pattern has to be persistent over time for the same signal, meaning that each letter has to present its signature activity in the Liquid, since the Readout layer is linear thus not extremely powerful in the classification procedure. All the Plasticity mechanisms, especially IP, have to act in order to find a delicate balance between enriching the Dynamics of the Network but not bringing it into chaos.
Another point of view in the study of the effect of Plasticity is given by the Principal Component Analysis. The Liquid state is projected on to the dimensions which better explain its variance along

the time. With this technique, one can better visualize the Liquid response to the input: ideally, tight clusters are created in the 2D and 3D Principal Component space, identifying each of the input signals. The more packed the cluster, the easier for the readout to univocally classify the related input signal. Moreover, different signals should be separated, meaning that their clusters should be far apart from each other in the Principal Component space. The farther the cluster, the easier for the readout to determine what signal has been displayed by the Liquid.
In order to get a feeling of what the PCA is extracting from the Liquid, the 2D and 3D Principal Component projection of the Liquid response for the Static case is plotted. In this simulation, the same word $[e, d1, d2, d3, d4, d5, d6, f]$ is presented to the Liquid for 10 times, with silence between them. The whole word lasts $1.6s$ and each input is presented for the same amount of time.



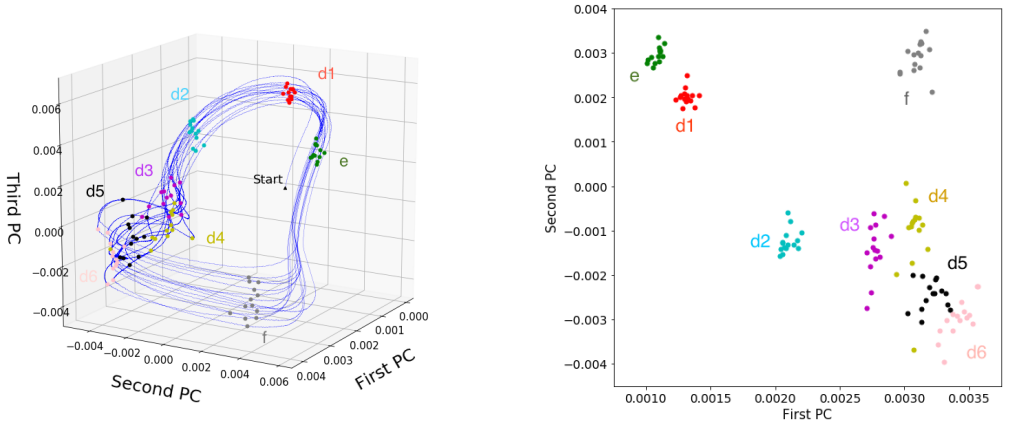
(a) Tridimensional representation of the Liquid State, with the time-evolution in blue

(b) Bidimensional representation of the Liquid State

Figure 6.3: PCA the representation in the Static-Liquid of the input word $(e, d1, d2, d3, d4, d5, d5, f)$ for the Static Network

In the 3D PCA, the trajectory in time of the Liquid State is plotted together with the signs related to the symbols, which are assumed to be the State of the Liquid at the end of each $200ms$ time interval. The trajectory is not perfectly coincident every time the word is repeated, mainly due to the fact that the input group is firing with a Poisson distribution. This uneven input spike pattern implements an inherent noise that tests the Liquid to perform also under non-ideal conditions. As a matter of fact, the trajectory is consistent in almost every phase of the input, favored by the silence imposed after every word, disabling interference between the fading activity from a past word and the rise of the new one. This interference would make the Liquid chaotic and unpredictable, producing a non persistent trajectory in the 3D PC space.
The important aspect of the analysis is the positioning of the cluster: the first consideration is that most of the clusters are compact and tight, meaning that the Network is stable and consistent in time in reacting to the slightly different inputs in the same way. The second aspect is that most of the clusters are far away from each other: this represents the separation ability of the Liquid, thus able to determine what signal has been shown in the input. In particular, 4 signals are clearly distinguished in the 3D space: the clusters related to the letters $[e, d1, d2, d3, f]$ are well separated. It was expected that the Liquid would be able to distinguish between the different letters, but it is also clear that it has a rich enough dynamics in order to separate some of the different ds input signals. Nevertheless, separating the same letter presented over time represents a complex task for the Liquid and the last ds in the sequence are all projected in a region of the PC space. This means that the Liquid response

is flattened after the same input is maintained for some time. The Liquid, being a complex dynamical system, easily falls into attractor states, which favor the classification but disable temporal information to be extracted. This is why it is important to tune the Reservoir to the Edge of Chaos, in order to find the perfect tradeoff between classification performance and encoding of temporal information. Similar information is expressed by the 2D PCA plot: however, in this case, 2 only components are not enough to encode the difference from the f letter, in grey, and the last couple of ds. Nonetheless, it is clear that the f is always well enough separated from all the other letters in the 3D plot since its grey cluster is far behind the yellow-black-pink ensemble.
In order to improve the separation performance of the Liquid, Plasticity has to act to enrich the dynamics, still conserving the ability to distinguish between different input signals. The same analysis is thus repeated for the SOSN case, in which the three Plasticity mechanisms are applied for a sequence of 20 words, to adapt the Liquid to the input signal. Then, the same word is fed to the Reservoir and the projection of the Liquid State in the Principal Components is analyzed in order to establish whether the Plasticity procedure brought an improvement to the Network.



(a) Tridimensional representation of the Liquid State, with the time-evolution in blue

(b) Bidimensional representation of the Liquid State

Figure 6.4: PCA the representation in the Static-Liquid of the input word $(e, d1, d2, d3, d4, d5, d5, f)$ for the SOSN Network

At first, the trajectory of the Liquid State along time in the 3D PC space seems to have become more complex and less linear. This indicates that the dynamic behavior of the Reservoir is less deterministic and a little more chaotic. It is particularly clear that in the region in which the letters $[d4, d5, d6]$ accumulate the behavior of the Liquid is far from deterministic: the trajectory of the Liquid State does not overlap at every cycle since the Reservoir is more unpredictable and prone to a chaotic response to the input. This criticality affects the cluster of the f letter, which is less dense and packed with respect to the case of the Static Network. However, this chaotic behavior comes with the advantage that the letters $[d4, d5, d6]$ are more separated and their respective clusters are more defined, despite being spread over a larger portion of the space. Again, since silence is imposed between the words, the initial condition is restored after every loop of the Liquid State and the initial sequence of the word is well separated in tight clusters. In that concern, it seems the SOSN Network is of no advantage with respect to the Static one. After all, the main and may only great advantage of the SOSN over the Static Reservoir is the about the behavior of the Liquid in the $[d4, d5, d6]$ region of space. Whereas the Static Liquid falls into an attractor state which makes it difficult to separate the different ds between each other, the SOSN is slightly more chaotic so that the separation between

the different letters is enhanced. This heuristic thesis is to be proven in a classification task, later introduced. If the classification performance will be effectively increased for the SOSN Network, then it will be clear that Plasticity had played a role in the performance of the Network.

6.4 Counting Task

The Counting Task is a typical benchmark in temporal tasks for Artificial Neural Networks. In this case, it is implemented for SNNs, requiring some adjustments. For example, instead of being shown for 1 time-step, the input signals are presented to the Reservoir for $200ms$; the frequency of the neurons is evaluated in time-bins of $100ms$ and, in order to determine the output, the highest frequency output neuron is considered.
The task is to predict the incoming signal when the Network is presented an alternation of 2 "words" consisting of $N_{CT} + 2$ symbols:

$$\text{[a,b,b,...,b,c] \& [e,d,d,...,d,c]}$$

where N_{CT} is the number of repetitions of the 'b's or 'd's. Moreover, the Network is not only asked to classify the incoming letter with its corresponding symbol, but also its position in the word:

$$\text{for } N_{CT} = 2,\ [a, b, b, c] \rightarrow [a, b_1, b_2, c]$$

This means that for a given N_{CT} the number of output classes is $2 \times (2 + N_{CT})$.
The accuracy of the classification is normalized in the interval $[0, 1]$ by counting how many correct predictions the Network is able to perform with respect to the total of shown signals. Of course, since the alternation of the words is random, the first signal in a word is not used for the evaluation of the accuracy, since the Network is at chance level in guessing which letter is coming next. An example of Input and Output is given below:

$$\text{Input: } [a, b, b, c; e, d, d, f] \rightarrow \text{Output: } [b_1, b_2, c; -, d_1, d_2, f; -]$$

The performances are evaluated for sequences of different lengths, where of course the longer ones were more challenging for the Network. 3 different configurations are tested: a 200 Neurons Static, a 400 Neurons Static and a 200 Neurons SORSN. The performance is evaluated by averaging 10 trials in which the Network is shown 500 sequences.

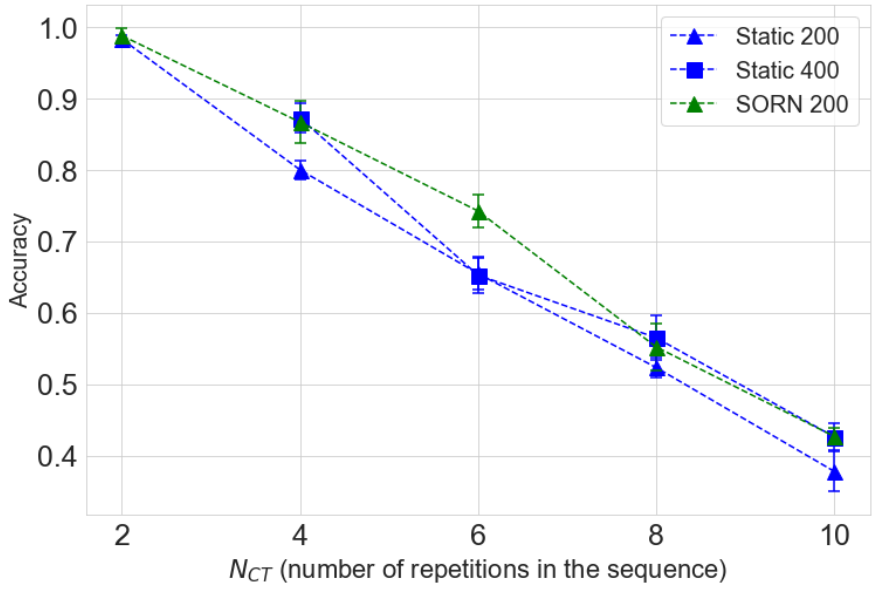


Figure 6.5: Accuracy of the prediction for the Counting Task, as a function of N

As expected, the Accuracy decreases as the length of the word increases, and it does in an almost linear way. In general, the performance is far from the one obtained by the original SORN, but a comparison is pointless since that would compare ANN with SNN. One aspect is to be highlighted: the 200 Neurons SORSN network performs as good, or even better than the 400 Neurons Static one. This means that the application of the 3 plasticity mechanisms has a similar effect on the performance than doubling the number of neurons for this task. A concrete advantage in using SORSN is clear considering that the 400 Neuron Network is much heavier to simulate in conventional computers and, even in the hypothesis of the implementation in dedicated hardware, it consumes more power than the 200 Neurons one.

6.5 Occluder Task

The Occluder Task is similar to the Counting Task in that the Network is trained to predict the incoming signal in a sequence of randomly alternated words, but presents 4 words instead of 2.

$$[1,2,3,...,\ N_{OT}],\ [N_{OT},\ N_{OT}-1,...,1]$$

$$[1,O,O,...,\ N_{OT}],\ [N_{OT},\ O,O,...,\ 1]$$

where the bottom two words include the "Occluder" (O) ,a signal that hides the ascending or descending signals, only showing the first and last symbols. The Occluder is a different signal with respect to the other N_{OT} ones and, because of the way the words are formed, it is the one to be presented the most time to the Network. This may cause problems during the pre-training, since STDP may polarize the activity of the Network based on the activation pattern formed by the Occluder.
Again, the Network is shown random alternation of words and has to predict the following signal in the sequence. In this case, not only the first number is not possible to be guessed, but also the second one in the word, since it could be the Occluder or the rest of the ascending/descending sequence. Here is an example of Input and correct Output.

$$\text{Input: } [1,2,3,4;4,O,O,1] \rightarrow \text{Output: } [-,3,4;\ -,-,O_2,1;-]$$

The task is tested for 5 sequences of 50 words, varying the length of the words.

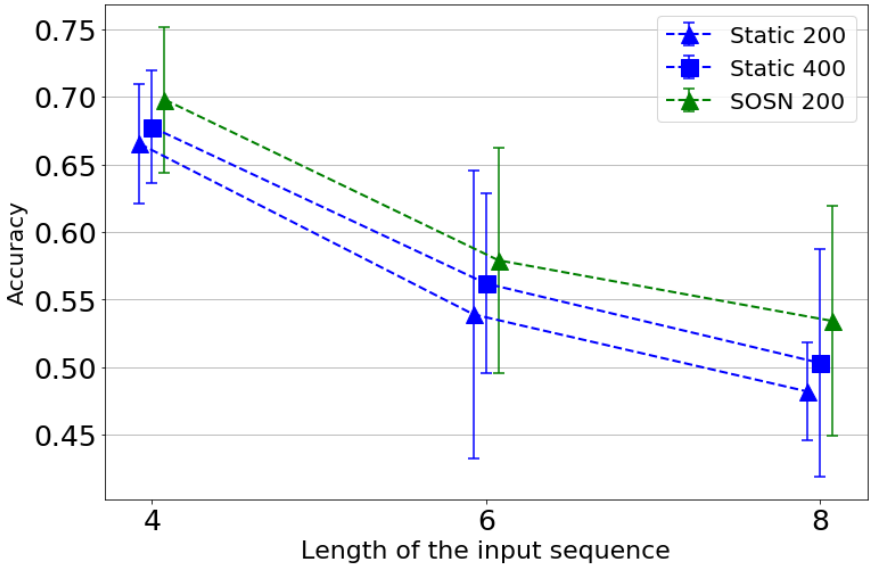


Figure 6.6: Accuracy of the prediction for the Occluder Task, as a function of N

Also in this case, the effect of the Plasticity is that the performance is increased of a considerable amount. Despite the Network has not been tested for as many words as the Counting Task, it is clear that the SORSN Network overcomes the performance of its equivalent Static counterpart but also of the 400 Static Network.

Chapter 7

Conclusion

This Thesis focuses on the simulation of Spiking Neural Networks with inspiration from biology in order to improve the performance of the chosen machine on temporal tasks. An introduction to SNNs and the paradigm of Reservoir Computing was presented at first, serving as the background for the development of the following work. In particular, the advantages of Reservoir Computing respect to Recurrent Networks training in general have been highlighted and the potential of Liquid State Machines has been explained. Supporting the motivation that led to the promotion of the Liquid State Machine as a computing paradigm, a section is dedicated to the details of the low-level functioning of the Brain, mainly focusing on the Neurons and Synapses and the way the information is processed in Neural Systems. During this part, it has been also discussed how to capture the behavior of such biologic systems into compact models, eventually employed for computation purposes. Following the Introduction, a review of the state-of-the-art of the implementations of Spiking Neural Network. It has been shown which are the most common models that are implemented with VLSI electronics and what a Neuromorphic Machine is made of. The DYNAPs, the Neuromorphic chip available to the Indiveri's group in Zurich, has been analyzed, mentioning both its qualities and drawbacks. Later, the computer paradigm of the Liquid State Machine has been studied in detail, particularly emphasizing the phase of Tuning of the Liquid and the Memory Capacity of the Network, by means of the Delayed AutoEncoder task. In this way, the limitations of such architecture have been tested and the author could build confidence in the operation of the LSM for the following tasks. Drawing inspiration from biology, an algorithm for a Plasticity mechanism (Intrinsic Plasticity) was developed, exploiting novel electronic devices, such as Memristors, in order to overcome the Von-Neumann bottleneck. This algorithm was tested in detail and has been verified to work properly: not only it was able to control the behavior of the Network in the desired manner, but it also allowed to reduce its consumed power. Lastly, Intrinsic Plasticity has been combined with STDP and Synaptic Normalization for the test of the Self-Organized-Spiking-Network. This LSM included a phase of unsupervised pre-training in which the Plasticity mechanisms adapted the Liquid to the presented input. This is shown to improve the performance on two temporal tasks: the Counting and the Occluder tasks. The benefit of the application of Plasticity is demonstrated to equalize the benefit of doubling the size of the Network.

Future works

In this work, it has been always paid attention to the hardware implementation of the algorithms. Technological plausibility is a necessity when dealing with Spiking Neural Networks since conventional computers struggle to simulate the time-continuous evolution of the Networks. So it is important to have a consolidated knowledge on Neuromorphic Computing in order to be sure that all the concepts in a Network may be successfully implemented in hardware. Regarding the circuital implementation

of the components of the Network, both the Neuron and Synapse are technologically plausible. The model of the Synapse is easily reproduced by the DPI Synapse present in the DYNAPs chips, which also include additional circuits for the STDP mechanism. For what concerns the Neurons, a new, simplified version of the DPI has been developed in order to host the three OxRAM cells. The question is then whether the models of Memristors used for the different applications fit the behavior of the real devices. In the cases of the tests of the IP algorithm and the Delayed Auto-Encoder task, the model - and in particular the Standard Deviation of the lognormal distribution in the RESET phase - is coherent with what found in the experimental characterization of the device. Instead, for the Counting and Occluder tasks, the improvements in performance were registered only when the Standard Deviation of the lognormal distribution was one order of magnitude lower than the experimental data. This may be attributed to either a defect of the tuning of the IP algorithm or to the great sensitivity of the Network with respect to any change of the parameters. Either way, both the concept of Liquid State Machine and the development of Memristors are on the first steps of their evolution, so that improvements are to be expected for the algorithm's stability on one hand, and on the technological implementation on the other hand.
Instead, concerning the benefit that biological features bring to Spiking Neural Networks and, in particular, to Liquid State Machines, it is interesting to test the accuracy of a SOSN machine on more complex tasks. The advantage of Spiking Networks over Artificial Networks is their inherent ability to process time-varying signals, due to the time-constants of the Neurons and Synapses. This ability can be exploited in complex temporal tasks, such as sound recognition. For example, one of the well-known benchmarks for speech recognition is the TIDIGITs dataset, in which digits from 0 to 9 are recorded by many different people and have to be classified by the machine. Another field of application of the SOSN is the processing of biomedical signals. Among the many, the ECG (the Electro-CardioGram) and the EMG (ElectroMyoGram). It is interesting to develop devices able to process these highly noisy signals being integrated into small footprints and consuming ultra-low power. In this perspective the results obtained for the application of Intrinsic Plasticity and its integration with other Plasticity mechanisms are promising.

Bibliography

[1] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt - *An experimental unification of reservoir computing methods*
Neural Networks 20 - 2007

[2] H. Jaeger - *The "echo state" approach to analysing and training recurrent neural networks*
GMD Report 148, German National Research Center for Information Technology, 2001

[3] H. Jaeger - *Short term memory in echo state networks*
GMD Report 152, German National Research Center for Information Technology, 2001

[4] M. Lukoševičius, H. Jaeger - *Reservoir computing approaches to recurrent neural network training*
Computer Science Review - 2009

[5] W. Maass, T. Natschläger - *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations*
Neural Computation - 2002

[6] J.J. Steil - *Backpropagation–Decorrelation: Online recurrent learn- ing with O(N) complexity*
Proceedings of IJCNN: Vol 1 - 2004

[7] D.E. Rumelhart, G.E. Hinton, R.J. Williams - *Learning representations by back-propagating errors*
Nature - 1986

[8] H. Daume - *A Course in Machine Learning, chapters 3,5,8,10,12*
2012

[9] F.Rosenblatt - *The Perceptron - a perceiving and recognizing automaton*
Cornell Aeronautical Laboratory - 1957

[10] P. Auer, H.M. Burgsteiner, W. Maass - *The p-delta rule for parallel perceptrons*
https://igi-web.tugraz.at/people/maass/psfiles/pdelta-journal.pdf - 2002

[11] B. Schrauwen, D. Verstraeten, J.V. Campenhout - *An overview of reservoir computing- theory, applications and implementations*
European Symposium on Artificial Neural Networks Bruges - 2007

[12] C. Gallicchia, L.Pedrelli - *Deep Reservoir Computing: A Critical Experimental Analysis*
Neurocomputing - 2017

[13] - *Human Brain Project* - https://www.humanbrainproject.eu/en/

[14] W. Gerstner and W. M. Kistler - *Spiking Neuron Models*
Cambridge University Press - 2002

[15] P. Dayan, L. F. Abbott - *Theoretical Neuroscience Computational and Mathematical Modeling of Neural Systems*
The MIT press - 2005

[16] Lapique - *Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation* - Journal of Phisiology

[17] A.L. Hodgkin, E. Huxley - *Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo* - Journal of Physiology- October 1951

[18] A.L. Hodgkin, E. Huxley - *Measurement of current-voltage relations in the membrane of the giant axon of Loligo* - Journal of Physiology- October 1951

[19] A.L. Hodgkin, E. Huxley - *The components of membrane conductance in the giant axon of Loligo* - Journal of Physiology- October 1951

[20] A.L. Hodgkin, E. Huxley - *A quantitative description of membrane current and its application to conduction and excitation in nerve* - Journal of Physiology- March 1952

[21] C.S. von Bartheld, J.Bahney, S. Herculano-Houzel- *The Search for True Numbers of Neurons and Glial Cells in the Human Brain* - Journal of Comparative Neurology - December 2016

[22] D. O. Hebb - *The Organization of Behavior*
McGill University - 1949

[23] M. Stemmler, C. Koch - *How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate*
Nature Neuroscience - July 1999

[24] R. Baddeley, L.F. Abbott, M.C.A. Booth, F. Sengpliez, T. Freeman, E.A. Wakeman, E.T. Rolls - *Responses of neurons in primary and inferior temporal visual cortices to natural scene*
Proceedings of the Royal Society - 1997

[25] C.D. Schuman, T.E. PotokE, R.M. Patton, J. D.Birdwell, M.E. Dean, G.S. Rose, J.S. Plank - *A Survey of Neuromorphic Computing and Neural Networks in Hardware*
IEEE - 2017

[26] G. Indiveri, S-C. Liu - *Memory and Information Processing in Neuromorphic Systems*
IEEE - 2015

[27] C.Mead - *Neuromorphic Electronic Systems*
IEEE - 1990

[28] W. Maass, H. Markram - *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations*
Neural Computation - 2002

[29] H. Jaeger, M. Lukoševiĉius - *Reservoir computing approaches to recurrent neural network training*
Science Direct - 2002

[30] M. Davis et al. - *Loihi: A Neuromorphic Manycore Processor with On-Chip Learning*
IEEE - 2018

[31] G. Indiveri, B. Linares-Barranco, T.J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S-C Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, K. Boahen - *Neuromorphic Silicon Neuron Circuits*
Frontier in Neuroscience - 2011

[32] E. Chicca, F. Stefanini, C. Bartolozzi, G. Indiveri - *Neuromorphic Electronic Circuits for Building Autonomous Cognitive Systems*
IEEE - 2014

[33] G. Indiveri, P. Livi - *A current-mode conductance-based silicon neuron for Address-Event neuromorphic systems*
IEEE - 2009

[34] T. Yamazaki, S. Tanaka - *The cerebellum as a liquid state machine* Neural Networks - 2007

[35] Link for Brian2 official webpage: `https://brian2.readthedocs.io/en/stable/index.html`

[36] D. Goodman, R. Brette - *Brian: a simulator for spiking neural networks in Python*
Frontier in Neuroscience - 2008

[37] G.Indiveri et al. - *A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)*
IEEE - 2018

[38] S.B. Furber, F. Galluppi, S. Temple, L.A. Planai - *The SpiNNaker Project*
IEEE - 2014

[39] B.V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A.R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J.V. Arthur, P.A. Merolla, Kwabena Boahen - *Neurogrid: A Mixed-Analog-Digital Multichip System for Large Scale Neural Simulations*
IEEE - 2014

[40] M.Davis et al. - *Loihi: A Neuromorphic Many-core Processor with On-Chip Learning*
IEEE - 2018

[41] Indiveri's startup - https://aictx.ai

[42] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt - *An experimental unification of reservoir computing methods*
Neural Networks 20 - 2007

[43] H. Jaeger - *The "echo state" approach to analysing and training recurrent neural networks*
GMD Report 148, German National Research Center for Information Technology, 2001

[44] H. Jaeger - *Short term memory in echo state networks*
GMD Report 152, German National Research Center for Information Technology, 2001

[45] M. Lukoševičius, H. Jaeger - *Reservoir computing approaches to recurrent neural network training*
Computer Science Review - 2009

[46] H. Jaeger - *Adaptive Nonlinear System Identification with Echo State Networks*
Internetional University of Bremen, D28759 - 2003

[47] W. Maass, T. Natschläger - *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations*
Neural Computation - 2002

[48] E.D. Adrian, Y. Zotterman - *The impulses produced by sensory nerve-endings*
Journal of Physiology - 1926

[49] R.B. Stein, E.R. Gossen, K.E. Jones - *Neuronal variability: noise or part of the signal?*
Nature Reviews Neuroscience - 2005

[50] C. Singh, W.B. Levy - *A consensus layer V pyramidal neuron can sustain interpulse-interval coding* - 2017

[51] S.M. Bohte, J.N. Kok, H. La Poutre - *Error-backpropagation in temporally encoded networks of spiking neurons*
Neurocomputing - 2002

[52] J.J. Wade, L.J. McDaid, J.A. Santos, H.M. Sayers - *SWAT: a spiking neural network training algorithm for classification problems*
IEEE Transactions on Neural Networks - 2010

[53] T. Dalgaty, M. Payvand, B. De Salvo, J. Casas, G. Lama, E. Nowak, G. Indiveri, E. Vianello - *Hybrid CMOS-RRAM Neurons with Intrinsic Plasticity*
API - 2019

[54] R. Baddeley, L.F. Abbott, M.C.A. Booth, F. Sengpiel, T. Freeman2, E.A. Wakeman, E.T. Rolls - *Responses of neurons in primary and inferior temporal visual cortices to natural scenes*
API - 2019

[55] T. Dalgaty, M. Payvand, F. Moro, Denys R.B Ly, F. Pebay-Peyroula, J. Casas, G.Indiveri, E.Vianello - *Hybrid Neuromorphic Circuits Exploiting Non-Conventional Properties of RRAM for Massively Parallel Local Plasticity Mechanisms*
API - 2019

[56] A. Lazar, G. Pipa, J. Triesch - *SORN: a self-organizing recurrent neural network*
Frontiers in Computational Neuroscience - 2009

[57] B. Schrauwena, M. Wardermanna, D. Verstraetena, J.J. Steilb, D. Stroobandta - *Improving reservoirs using intrinsic plasticity*
API - 2019

[58] W. Zhang and P. Li - *Information-Theoretic Intrinsic Plasticity for Online Unsupervised Learning in Spiking Neural Networks*
Frontiers in Neuroscience - 2019

[59] C. LI, Y. LI - *A Review on Synergistic Learning*
IEEE - 2015

[60] A. Grossi, E. Nowak, C. Zambelli, C. Pellissier, S. Bernasconi, G. Cibrario, K. El Hajjam, R. Crochemore, J.F. Nodin, P. Olivo and L. Perniola - *Fundamental Variability Limits of Filament-based RRAM*
IEEE - 2016

[61] A. Grossi , E. Vianello, C. Zambelli, P. Royer, Je.-P. Noel, B. Giraud, L. Perniola, P. Olivo, E. Nowak - *Experimental Investigation of 4-kb RRAM Arrays Programming Conditions Suitable for TCAM*
IEEE - 2018