7-7-2020

Pattern Recognition Using Spiking Neural Networks

Amir Javid Talaei
University of Windsor

Pattern Recognition
Using
Spiking Neural Networks

By

Amir Javid Talaei

A Thesis

Submitted to the Faculty of Graduate Studies

through the Department of Electrical and Computer Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Applied Science

at the University of Windsor

Windsor, Ontario, Canada

2020

Pattern Recognition
Using
Spiking Neural Networks

By

Amir Javid Talaei

APPROVED BY:

A. Edrisy

Department of Mechanical, Automotive & Materials Engineering

E. Abdel-Raheem

Department of Electrical & Computer Engineering

M. Ahmadi, Advisor

Department of Electrical & Computer Engineering

May 13, 2020

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Deep learning believed to be a promising approach for solving specific problems in the field of artificial intelligence whenever a large amount of data and computation is available. However, tasks that require immediate yet robust decisions in the presence of small data are not suited for such an approach. The superior performance of the human brain in specific tasks like pattern recognition in comparison to traditional neural networks convinced neuroscientists to introduce a biologically plausible model of the neuron, which is known as spiking neurons. In opposition to conventional neuron, spiking neurons use a short electrical pulse known as a spike to transfer the information. The complexity and dynamic of these neurons allow them to perform complex computational tasks. However, training a spiking neural network does not follow the rule of conventional ANN, and we need to devise new methods of training that are compatible with the unsupervised nature of these networks. This thesis aims to investigate the unsupervised approaches of training spiking networks using spike time-dependent plasticity (STDP) and assess their performance on real-world machine learning applications like handwritten digit recognition.

to

My Parents

for Their Unconditional Love

and Support

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my advisor, Dr. Majid Ahamdi. His continuous support, encouragement, and patience have been instrumental to this thesis.

I am also inclined to express my profound appreciation to my committee members, Dr. Esam Abdel-Raheem of the Department of Electrical and Computer Engineering and Dr. Afsane Edrisy of the Department of Mechanical, Automotive & Materials Engineering for their insightful comments during the development of this thesis.

Special gratitude belongs to my family, my parents, and my brother, for providing me with incredible support and encouragement throughout the years of my study. This work would not have been possible without their sacrifice and continued care.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
EPSP	Excitatory Postsynaptic Potential
GA	Genetic Algorithm
IF	Integrate And Fire Neuron
IPSP	Inhibitory Postsynaptic Potential
LIF	Leaky Integrate And Fire Neuron
LTD	Long Term Depression
LTP	Long Term Potentiation
MLP	Multilayer Perceptron
PSP	Postsynaptic Potential
SNN	Spiking Neural Network
SOM	Self Organizing Map
SRM	Spike Response Model
STDP	Spike Timing Dependent Plasticity
WTA	Winner Takes All Strategy

Chapter 1

INTRODUCTION

1.1 Introduction

Recent advances in the field of artificial intelligence and machine learning affected various aspects of human life. With the emergence of self-driving cars, the proliferation of virtual assistants, and highly intelligent search engines, we feel the presence of AI in our life more than ever.

One of the milestones in the history of artificial intelligence happened when Hinton and Osindero [20] published their work about deep neural networks. In their paper, they proposed a method to pre-train deep neural networks one layer at a time and lay the foundation for the field of deep learning, which finds its way for widespread industrial use.

In comparison with the first generation of neural network with linear activation function, the second generation (deep networks) consist of neurons with non-linear activation function (sigmoid function) which can perform much more complicated machine

learning tasks than the previous generation.

However, comparing the performance of the best deep neural networks with human-level performance highlights the need for a more biologically plausible model of a neuron, which known as spiking neuron. Spiking neural networks are a class of neuron models developed to mimic the behavioral dynamics of the biological neurons. The ability to exhibit the dynamics of one or more variable states enabled them to capture phenomena not seen in artificial neural networks. These spiking neurons communicate by sending short period, high amplitude pulse of activity referred to as spikes [1].

In the spiking neurons, the output of activation function computed by each neuron not only depends on the value of its inputs but also on the timing of input arrival. Such temporal coding allows a spiking neuron to surpass its sigmoidal counterpart in terms of flexibility of computational tasks. The dynamics of a spiking neuron influenced by the incoming spikes determines the condition and time for sending spikes.

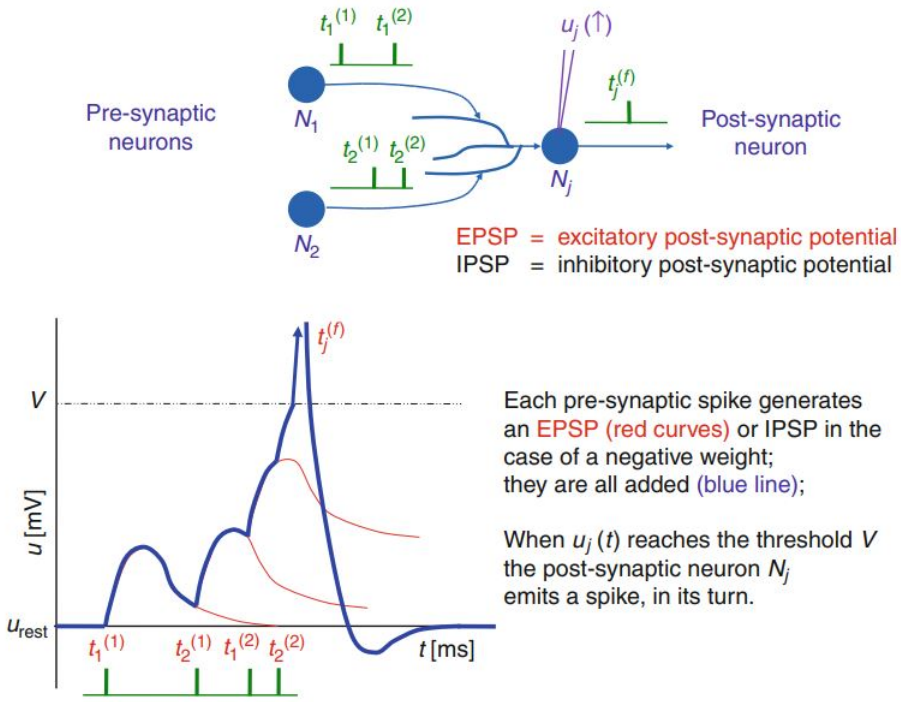


Figure 1.1: A model of Spiking Neuron [37].

Figure 1.1 illustrates a model of spiking neuron; neuron N_j generates an action potential (spike) whenever the weighted sum of incoming PSP (postsynaptic potential) approaches the threshold value. The diagram below shows the changes in membrane potentials of N_j in response to four incoming spike.

There are different types of dynamical SNN models with varying levels of sophistication. However, training a spiking neural network is not as straightforward as the conventional neural networks with the backpropagation algorithm, and the question of training a spiking neural network is still wide open. Recent studies have shown that precise spike encoding broadly used by the brain, providing a higher transformation speed and metabolic efficiency [46].

1.1.1 Why SNN?

SNN hold exceptional qualities that make them superior in a few aspects in comparison with traditional machine learning techniques [25]:

- Effective modeling of processes that include various time scales
- Event prediction
- Parallel information processing
- Compact information processing
- Low energy consumption on neuromorphic hardware

Dharmendra Modha manager and lead researcher of the Cognitive Computing group at IBM highlights the significance of brain-inspired computing as follow:

"The goal of brain-inspired computing is to deliver a scalable neural network substrate while approaching fundamental limits of time, space, and energy".

Table 1.1 illustrates a comprehensive comparison of the spiking neural networks with other machine learning methods over different inclinations.

Method/Features	**Statistical Method**	**ANN**	**SNN**
Information	Scalars	Scalars	Spike sequences
Data Representation	Scalars, vectors	Scalars, vectors	Whole TSTD patterns
Learning	Statistical, limited	Hebbian rule	STDP
Dealing with TSTD	Limited	Moderate	Excellent
Parallel Computation	Limited	Moderate	Massive
Hardware Support	Standard	VLSI	Neuromorphic VLSI

Table 1.1: Comparison of the SNN and other machine learning techniques [25].

1.2 Dynamic of a Biological Neuron

Although neurons are only one of many brain cells, they have attracted more attention than other brain cells because of their fundamental role in computational operations. The fundamental function of a neuron is simple: the neuron receives input signals from other neurons via connections called Synapses, and if the input signals excite them sufficiently, they will fire an action potential (spike) that propagates through synapses to other neurons.

Neurons consisted of three main parts: the dendrite, the soma, and the axon. Dendrites considered as the receiver of input signals, and neurons receive input current via their dendrites. This input current then transmitted to the main body of the cell, called the soma. When a neuron generates an action potential, it sends current down its axon, causing neurotransmitters to release at the synapses, which are connections

from a neuron's axon to the dendrites of other neurons. This neurotransmitter release causes the flow of dendritic currents in other connected neurons.

The main body of the neuron called soma. From a computational perspective, this is where all the incoming currents from dendrites integrated. The process of producing an action potential also occurs in the soma.

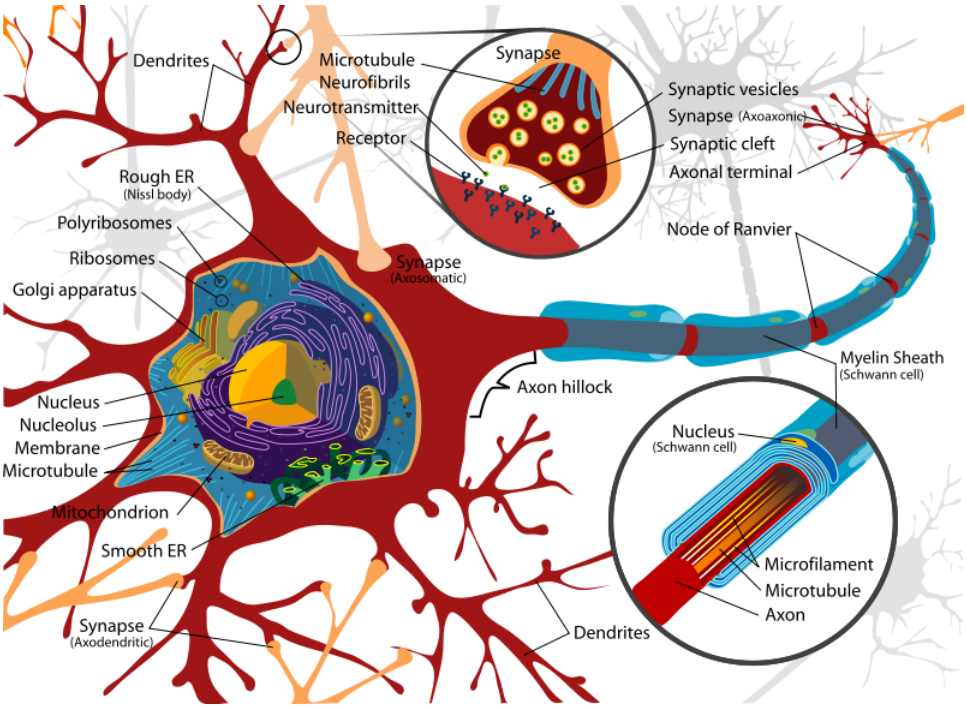


Figure 1.2: Diagram of a Neuron [10].

When a neuron is in resting state, the soma has a negative potential called the resting potential and controlled by ion pumps that maintain a particular concentration of ions (mostly sodium Na^+, potassium K^+, and calcium Ca^{2+}) inside the cell. The incoming current from dendrites causes the cell membrane to depolarize.

Figure 1.2 illustrates the neuron as a complicated information-processing unit, which receives thousands of signals from the dendrite of other neurons through synaptic

connection. The single output of this neuron is an action potential (spike) which emits whenever the membrane voltage reaches a threshold.

Whenever the potential in the soma becomes high enough, it starts to trigger sodium channels, which allow sodium ions to enter the cell and further depolarizing it. The process continues until the electrical gradient of the sodium channel opposes the chemical gradient of imbalance in sodium charge inside and outside of the cell. This process causes a considerable change in membrane potential and alters the membrane potential from a negative to a positive charge.

The considerable depolarization of the membrane potential triggers the potassium channels and let them reach out of the cell and eventually repolarize it. At the same time, the sodium channels become inactivated. The open potassium channels finally bring the cell to a potential lesser than its resting potential, which called the hyperpolarized state. Here process continues for a short while in which the neuron is not capable of generating spikes, which called the absolute refractory period.

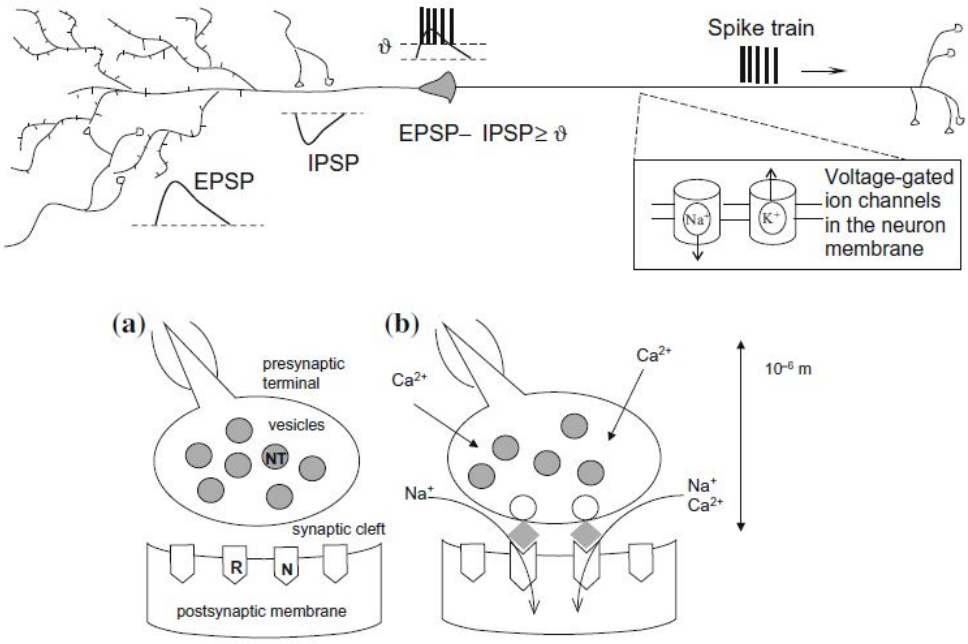


Figure 1.3: Scheme of synaptic transmission [3].

Figure 1.3 demonstrates the schematic of synaptic transmission in a neuron. In part (a), the neuron is ready to transmit a signal. Part (b) presents the sending of spike upon arrival of the spike into the terminal. Here, calcium appears as a second messenger hence triggering a cascade of biochemical responses.

The difference in ionic concentrations inside the cell membrane is considerably small during a single spike, but throughout many spikes, the ion pumps require to maintain the proper concentrations of sodium and potassium. The immediate depolarization of membrane potential triggers the sodium channel in axonal parts and generates a voltage wave that moves down the axon. Eventually, this voltage wave triggers the synaptic vesicles proximate to the ends of the axon and let them release neurotransmitters.

1.3 Models of a Single Neuron

1.3.1 McCulloch-Pitts Model

In 1943, McCulloch and Pitts published their famous model of a neuron, which known as the logic threshold unit. The computational ability of their two-state neural model presented in [31]. In such a model, the neuron can be either in an active or inactive state. Whenever the current value of the neuron surpasses a predefined value known as the threshold, neuron's state will change from inactive to active. They also used the structure of inhibitory synapses in which a neuron connected to inhibitory synapses is not able to become active by itself.

One of the great results of their works is that we can implement some of the most fundamental logical gates using their model. This property attracted much attention among computer scientists and lay the foundation for what we know as Von Neumann architecture.

However, the McCulloch and Pitts model does not represent the full functionality of an actual neuron and has its limitations. The input to this neuron model is in binary form, and inputs with real value do not apply to this neuron. In addition to this, the McCulloch and Pitts model is only able to perform linearly separable functions, and we can not implement linearly non-separable functions like XOR using this neuronal model.

Figure 1.4 demonstrates the McCulloch-Pitts neural model with a set of input x_1, x_2, ..., x_n and one output y. The output is in binary form. We can represent the function of the neuron using (Eqn.1.1) and (Eqn.1.2).

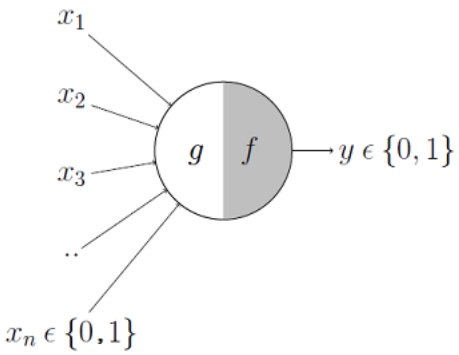


Figure 1.4: McCulloch and Pitts Model.

$$g = \sum_{i=1}^{n} x_i w_i \tag{1.1}$$

$$y = f(g) \tag{1.2}$$

Where w_1, w_2, ..., w_n are weight values normalized in range $(0, 1)$. The function f expressed as $f(x) = h(x - T)$, where h is the Heaviside step function, and T is the threshold value.

1.3.2 Hodgkin-Huxley Model

In 1952, following a comprehensive set of experiments on the giant axon of the squid, Hodgkin and Huxley presented their mathematical model for describing the dynamics of a neuron. In their model, action potentials are the result of currents that pass through ion channels. They used differential equations to describe the dynamic behavior of these ion channels. Their model immediately acknowledged as a groundbreaking achievement in neuroscience society and eventually led to the Nobel Prize

in 1963 [21].

For decades, neuroscientists successfully used this model to simulate the actual operation in the human brain. However, the computational cost is the main barrier for simulating a network consisting of a large number of neurons.

Hodgkin and Huxley described the dynamic of a neuron using three different ion channels consisted of the potassium channel, sodium channel, and a channel that handles other types of ions known as the leakage channel. The cell body acts as a semipermeable membrane and allows only specific ions to pass through it. The flow of those ions across the membrane defines the internal potential concerning the potential outside of the cell [15].

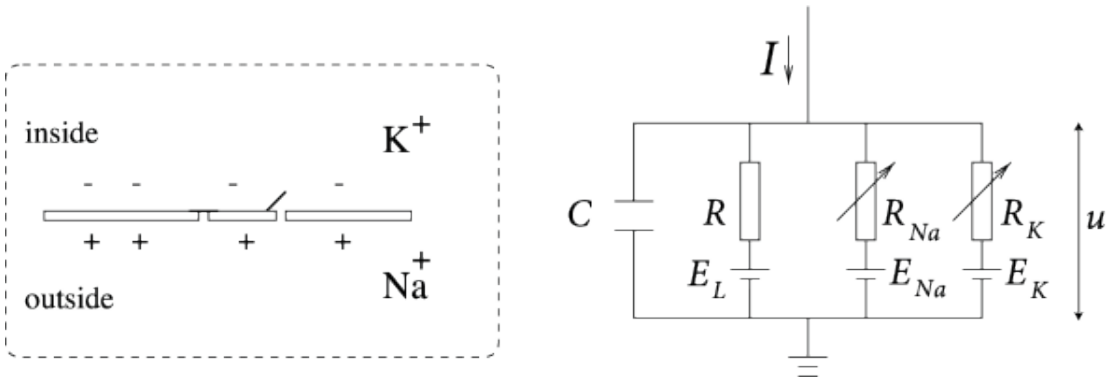


Figure 1.5: Hodgkin-Huxley Model.

The model explained by the aid of Figure 1.5. The cell membrane separates the interior of the cell from the extracellular environment. This membrane, therefore, can think of as a capacitor in an electrical circuit. If we introduce the input current $I(t)$ into the cell, it may increase the charge of capacitors or leak into ionic channels of the cell membrane. Each ion channel outlined in the Figure 1.5 using a resistor. The resistance of sodium, potassium and leakage channel indicated by respectively

R_{Na}, R_{K}, and R. We denote the potential across this membrane by u. Three current components of their model formulated, as shown in (Eqn.1.3). The channels characterized by their respective conductance, which denoted by g_{Na}, g_{K} and g_{L}. The value of conductance for sodium and potassium is at its maximum level when those channels are open.

$$\sum_k I_k = g_{\mathrm{Na}}\, m^3 h\, (u - E_{\mathrm{Na}}) + g_{\mathrm{K}}\, n^4\, (u - E_{\mathrm{K}}) + g_L\, (u - E_L) \tag{1.3}$$

E_{Na}, E_{K}, and E_{L} are respectively, the reversal potential of sodium, potassium, and leakage channel. We can express the activation of each channel of the model in terms of voltage-dependent transition rates α and β as:

$$\begin{aligned} \dot{m} &= \alpha_m(u)\,(1 - m) - \beta_m(u)\, m \\ \dot{n} &= \alpha_n(u)\,(1 - n) - \beta_n(u)\, n \\ \dot{h} &= \alpha_h(u)\,(1 - h) - \beta_h(u)\, h \end{aligned} \tag{1.4}$$

The terms m and h are controlling variables for the sodium channel while the potassium channels constrained by the term n. Here $\dot{m}$, $\dot{n}$ and $\dot{h}$ are respectively the derivative of the m, n and h with respect to the time.

The differential equations in (Eqn.1.5) determine how the gating variables m, n and h evolving over time. This gating variable defines the probability for which a particular channel is open since most of the time; one of these channels is blocked.

$$\dot{x} = -\frac{1}{\tau_x(u)}[x - x_0(u)], \tag{1.5}$$

Hodgkin-Huxley model can successfully describe the dynamic of the squid neuron (their experimental subject); however, experiments confirm that there are other kinds of electrophysiological properties in cortical neurons of the vertebrates, which need additional channels to explain the behavior of the neuron sufficiently. Detailed models of these types of neurons developed over the years, however, the computational demand of these models made the Hodgkin-Huxley model the first choice for neuroscientific investigations.

1.3.3 Integrate-And-Fire Models

Integrate-and-Fire models represent action potentials as events in which if the voltage $u_i(t)$ (which comprises the summed effect of all inputs) reaches a threshold ϑ, the neuron fires a spike. The shape of the action potentials is not of the highest importance in this model. To describe the dynamics of the neuron, integrate and fire models use two separate components; first, an equation that defines the evolution of the membrane potential $u_i(t)$; and second, a mechanism for generating action potentials [16].

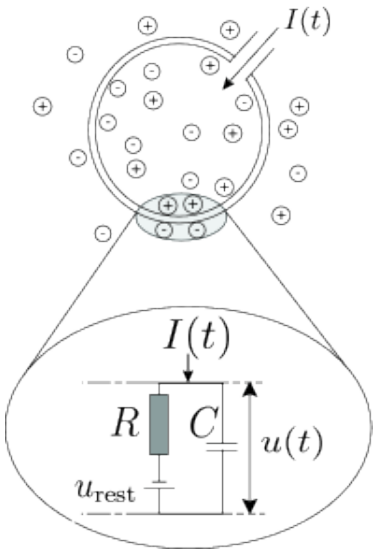


Figure 1.6: Electrical properties of the Integrate-And-Fire neuron.

The variable u_i represents the membrane potential of neuron i. Usually, the potential is at its resting value u_{rest} when there is no incoming input to the cell membrane. Whenever the neuron receives synaptic input from other neurons, the potential will be different from its resting value.

Figure 1.6 illustrates the electrical properties of the integrate-and-fire neuron. We can link the instantaneous voltage of the cell membrane to the input current from synaptic input using the elementary laws of electricity. Considering the neuron which surrounded by a cell membrane, we can think of it as a capacitor which will charge if a short current pulse $I(t)$ injected into the neuron. Since the insulator is not ideal, we have a slow leakage of potentials through the cell membrane. Finally, we can characterize the cell membrane by a finite leak resistance R.

The basic integrate-and-fire model consists of a capacitor C in parallel with a resistor R and input current I(t).

Using the law of current and splitting it into two elements we have:

$$I(t) = I_R + I_C \tag{1.6}$$

Using Ohm's law, we can rearrange (Eqn.1.6) to the equation presented below:

$$I(t) = \frac{u(t) - u_{\text{rest}}}{R} + C\,\frac{\mathrm{d}u}{\mathrm{d}t} \tag{1.7}$$

Multiplying (Eqn.1.7) by R and using the time constant $\tau_m = R\,C$ yields the standard form:

$$\tau_m\,\frac{\mathrm{d}u}{\mathrm{d}t} = -[u(t) - u_{\text{rest}}] + R\,I(t)\,. \tag{1.8}$$

The solution to this differential equation considering the initial condition $u(t_0) = u_{\text{rest}} + \Delta u$ is in form:

$$u(t) - u_{\text{rest}} = \Delta u\,\exp\left(-\frac{t - t_0}{\tau_m}\right) \quad \text{for } t > t_0\,. \tag{1.9}$$

When there is no incoming input to the membrane, the potential exponentially decay to its resting value. The membrane time constant determines the characteristic time of the decay.

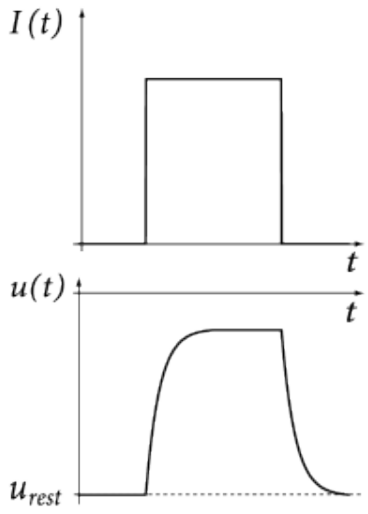


Figure 1.7: Potential of the cell membrane (bottom) when a step current (top) injected to the membrane.

Figure 1.7 demonstrates the smooth reaction of the cell membrane in response to a step input current. We can interpret the result considering the electrical diagram of the RC-circuit in Figure 1.6. Whenever the circuit enters a steady-state, the charge on the capacitor no longer increases, and all the incoming input current should pass through the resistors.

Leaky Integrate-and-fire model [14]

A specific case of integrate and fire neurons which incorporates the notion of leakage channel in membrane potential is leaky integrate and fire model. The leakage channel reflects the diffusion of ions that happens through the membrane when some equilibrium condition is not satisfied in the cell. The differential equation of the leaky integrate-and-fire model represented as:

$$\frac{dv}{dt} = -\frac{v - v_{eq}}{\tau} + I(t) \tag{1.10}$$

Where v_{eq} is the is the equilibrium potential, $\tau = RC$ is the time constant of the membrane, and v represents the membrane potential. Integrating the differential equation over an arbitrary input current I we have:

$$u(t) = \eta(t - \hat{t}) + \int_0^\infty \Theta(t - \hat{t} - s) \exp(-s/\tau)\, I(t - s) ds \tag{1.11}$$

Where $\hat{t}$ is the firing time of the last spike of the neuron, $u = v - v_{eq}$ is the potential following the reset after each spike, Θ is the Heaviside step function, and η is the spike shape function which determines the mean shape of spike and represented as:

$$\eta(t - \hat{t}) = (v_{reset} - v_{eq})\ \exp[-(t - \hat{t})/\tau] \tag{1.12}$$

The leaky integrate and fire is a simplified model of an actual neuron, and it misses several characteristics which neuroscientists have observed when they study neurons in the living brain. However, this model considered as a reliable model for generating spikes since it is surprisingly precise for simulating timed events phenomena [16].

The study suggests that after each spike neurons enter a refractory period during which they are incapable of generating new spikes. In addition to refractoriness, neurons show adaptation, which constitutes over hundreds of milliseconds. We can increase the accuracy of the simple leaky integrate-and-fire model by adding adaptation and refractoriness to a much higher degree. A simple method to satisfy this property is to consider a dynamic threshold for the neuron model. The neuron threshold will increase by constant value θ after each spike and will approach to its stable

value in the quiescent period. Using the delta function, we can formulate this idea:

$$\tau_{\text{adapt}} \frac{\mathrm{d}}{\mathrm{d}t} \vartheta(t) = -[\vartheta(t) - \vartheta_0] + \theta \sum_f \delta(t - t^{(f)}) \tag{1.13}$$

Where $t^{(f)} = t^{(1)}, t^{(2)}, t^{(3)}...$ are firing time of the neuron, and τ_{adapt} is the time constant for adaptation. $\vartheta(t)$ and ϑ_0 are respectively, membrane and resting potential of the neuron.

1.3.4 Izhikevich Model

The Izhikevich neuron model is a simplified model of the biological neuron which can describe the dynamic properties of the cell membrane using two distinct differential equations. This model of the neuron is capable of generating various kinds of action potentials observed in biological neurons, which include regular spiking, intrinsically bursting, chattering, and many others [23].

Differential equations in (Eqn.1.14) and (Eqn.1.15) used to describe the Izhikevich model. Here, v is the membrane potential, and u is the membrane recovery variables, which provides negative feedback to v. These two variables are dimensionless. The threshold value set to 30 mV and if the voltage v is larger than the threshold, v and u will reset as (Eqn.1.16). There are four additional dimensionless parameters which are a, b, c, and d. The parameter a represents the time scale on which the membrane potential u operates and parameter b represents the sensitivity of u to fluctuations in v. The parameter c used to define the reset potential of v after a spike and parameter d specifies the reset potential of the variable u after producing an action potential.

$$v' = 0.04\,v^2 + 5v + 140 - u + I \tag{1.14}$$

$$u' = a(bv - u) \tag{1.15}$$

$$\text{if } v \geq 30\,\text{mV, then } \begin{cases} v \longleftarrow c \\ u \longleftarrow u + d \end{cases} \tag{1.16}$$

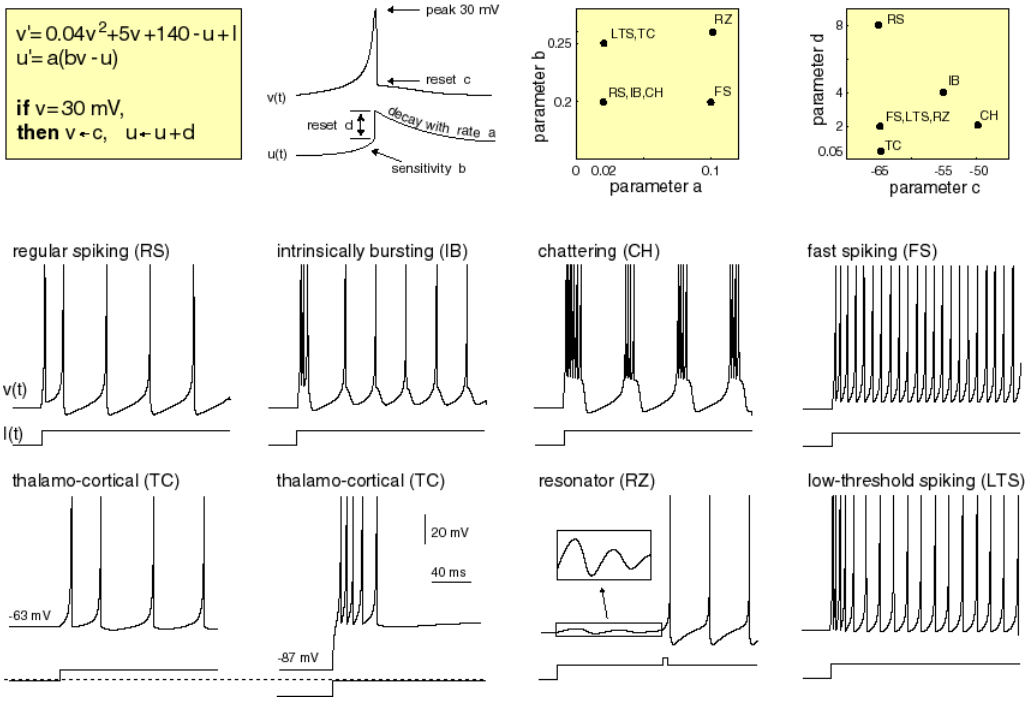


Figure 1.8: Izhikevich neuron model.

The input current to the system denoted using parameter I. By adjusting the model parameters a, b, c, and d, we can present different kinds of spiking patterns. Figure

1.8 demonstrates the ability of the Izhikevich neuron to produce various kinds of action potentials, including intrinsically bursting (IB), chattering (CH), thalamocortical (TC), and resonator (RZ).

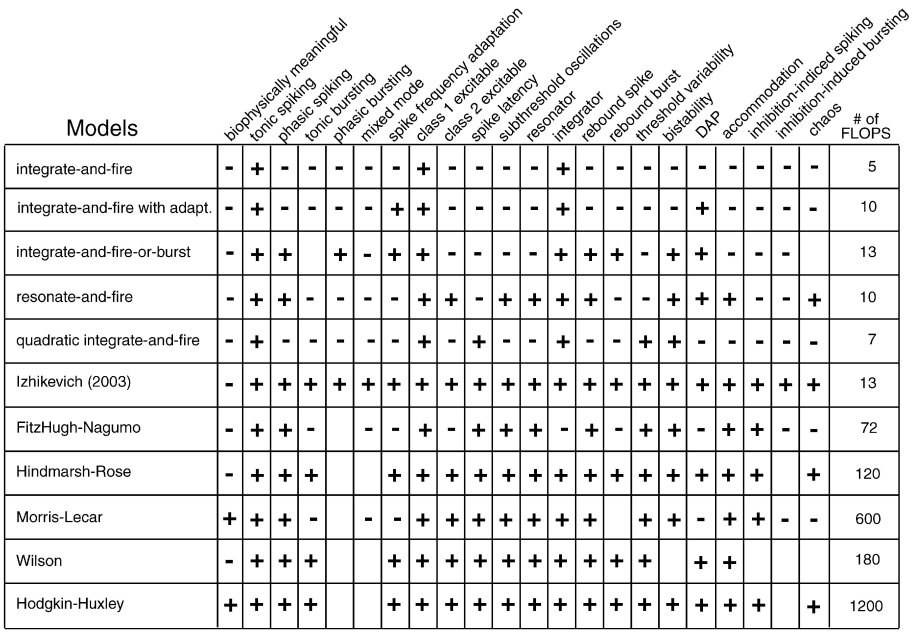
As a consequence of simplification, the Izhikevich model does not reflect the refractory period after generating spikes, which may lead to unrealistic behavior of the neuron under specific situations. A solution to this problem proposed in [44].

To incorporate a refractory period, we need to interrupt the dynamic equation in (Eqn.1.16) whenever v reaches the threshold value at time t_f. Therefore we need to modify the (Eqn.1.16) adding a new constraint as shown in (Eqn.1.17) and (Eqn.1.18)

$$\text{if } v \geq 30\,\text{mV, and } t - t_{prev} \geq \Delta(abs) \begin{cases} v \longleftarrow c \\ u \longleftarrow u + d \end{cases} \tag{1.17}$$

$$\text{else if } v \geq 30 \text{ then} v \longleftarrow 30 \tag{1.18}$$

Possessing a strong biological characteristics and regarding the computational efficiency, the Izhikevich neuron is a liable candidate for big network simulations. Figure 1.9 is a comparison between different neuronal model which presented in [24].



Models	biophysically meaningful	tonic spiking	phasic spiking	tonic bursting	phasic bursting	mixed mode	spike frequency adaptation	class 1 excitable	class 2 excitable	spike latency	subthreshold oscillations	resonator	integrator	rebound spike	rebound burst	threshold variability	bistability	DAP	accommodation	inhibition-induced spiking	inhibition-induced bursting	chaos	# of FLOPS
integrate-and-fire	-	+	-	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	-	5
integrate-and-fire with adapt.	-	+	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	10
integrate-and-fire-or-burst	-	+	+		+	-	+	+	-	-	-	-	+	+	+	-	+	+	-	-	-		13
resonate-and-fire	-	+	+	-	-	-	-	+	+	-	+	+	+	+	-	-	+	+	+	-	-	+	10
quadratic integrate-and-fire	-	+	-	-	-	-	-	+	-	+	-	-	+	-	-	+	+	-	-	-	-	-	7
Izhikevich (2003)	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	13
FitzHugh-Nagumo	-	+	+	-		-	-	+	-	+	+	+	-	+	-	+	+	-	+	+	-	-	72
Hindmarsh-Rose	-	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	120
Morris-Lecar	+	+	+	-		-	-	+	+	+	+	+	+	+		+	+	-	+	+	-	-	600
Wilson	-	+	+	+			+	+	+	+	+	+	+	+	+	+		+	+				180
Hodgkin-Huxley	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	1200

Figure 1.9: Comparison of different neuron models [24].

1.3.5 Spike Response Model(SRM)

There is another approach rather than using the system of differential equations to describe the behavior of the cell membrane, and that is replacing parameters of the model by a (parametric) function of time called filters. Spike response model is a generalized version of the integrate-and-fire model formulated using filters. In contrast to the leaky integrate-and-fire model, this model incorporates the notion of the refractoriness [17].

In the spike response model, the membrane potential denoted by u, which is an essential factor for determining the state of the neuron. In the absence of the input cuurent, the membrane potential is at its resting state u_{rest}. Adding a short current pulse into the cell membrane will change its potentials, and it takes a while before u

return to its resting state.

Because of the linear characteristics of the membrane potential below the threshold value, we can express the voltage response h of the membrane to a time-dependent simulating current by (Eqn.1.19).

$$h(t) = \int_0^\infty \kappa(s)\, I^{\text{ext}}(t-s)\, \mathrm{d}s \tag{1.19}$$

Here the function $\kappa(s)$ defines the time scale of the voltage response to a short current pulse at time $s = 0$. Since many ion channels are still open immediately after the spike, the membrane time constant is shorter during that brief period.

We can represent the evolution of u in regards to incoming spike trains using equation in (Eqn.1.20). Here, the function η, represents various forms of the action potentials, including depolarizing, hyperpolarizing, and resonating spike-after potential [2].

$$u(t) = \sum_f \eta(t - t^{(f)}) + \int_0^\infty \kappa(s)\, I^{\text{ext}}(t-s)\, \mathrm{d}s + u_{\text{rest}} \tag{1.20}$$

We can rearrange the sum over all past firing times to a convolutional form as in (Eqn.1.21)

$$u(t) \int_0^\infty \eta(s) S(t-s) \mathrm{d}s + \int_0^\infty \kappa(s)\, I^{\text{ext}}(t-s)\, \mathrm{d}s + u_{\text{rest}} \tag{1.21}$$

It is essential to specify that in the SRM model, the threshold value is a time-dependent variable, which differs from the leaky integrate-and-fire model with a fixed threshold (Eqn.1.22). Here we observe an increase in the threshold voltage shortly after the spike. The threshold will decay to its resting state afterward.

$$\vartheta \quad \longrightarrow \quad \vartheta(t)\,. \tag{1.22}$$

Whenever the membrane potential u approaches the dynamic threshold $\vartheta(t)$ from the below, the neuron will generate an action potential (Eqn.1.23).

$$t = t^{(f)} \quad \Leftrightarrow \quad u(t) = \vartheta(t) \text{ and } \frac{\mathrm{d}[u(t) - \vartheta(t)]}{\mathrm{d}t} > 0\,. \tag{1.23}$$

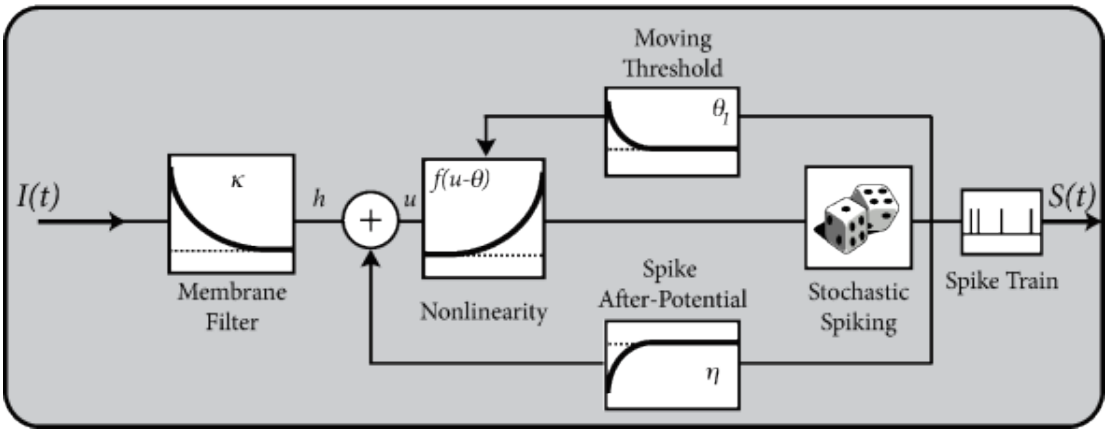


Figure 1.10: Spike Response Model [16].

Figure 1.10 describes the spike response model (SRM). Input current passes through the filter $\kappa(s)$ and creates the potential h. If the membrane potential reaches the threshold ϑ, the neuron generates an action potential. After each spike, the membrane threshold increases by θ_1. Furthermore, each spike produces a voltage contribution η to the membrane potential.

The Spike Response Model is incapable of explaining the following properties of the biological neuron:

- In comparison with the Hodgkin and Huxley model, the spike response model is incapable of describing the biophysics of membrane potential explicitly and therefore performs poorly on the prediction of the individual ion channel. This model considers the effect of several ion channels and predict their spike shape using function η and filter κ. For explaining the behavior of an individual ion channel, Hodgkin and Huxley is a more reliable choice [14].

- As observed in the biological neuron, the action potential delay differs according to the amplitude of the input pulse. We can not see this property in the spike response model. There are another type of neurons, such as the quadratic [28] or exponential integrate-and-fire model [6], which can represent this characteristic.

1.4 Neural Coding

Spiking neural networks employ precise timing of spikes for transferring information, which is significantly different from what we saw in conventional neural networks. Therefore, a different approach for presenting input stimuli to the network required. Various procedures for converting input data to an understandable stimuli for SNN proposed, here we discuss different techniques of neural coding.

1.4.1 Rate vs Temporal Coding

The rate coding refers to encoding the input to a stimulus in terms of firing rate or frequency of action potentials. Contraction of the muscle which is in accordance with the number of spike per time unit, considered as an example of the rate coding in the nervous system [33].

However, studies suggest that the human brain employs a different procedure for interpreting visual stimuli considering the response time of the visual receptors to these stimuli, which is remarkably short, and no time will remain for ascertaining the average firing rate by the neural system [15]. Though this is not the case in temporal coding, and the timing of individual spikes is equivalently important. Role of precise spike timing for localization of sound in the auditory system is an example of temporal coding [7].

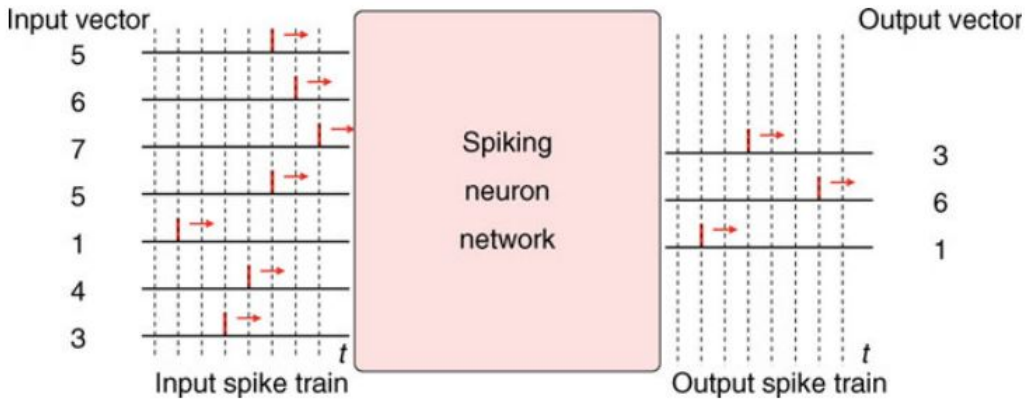


Figure 1.11: Illustration of the temporal coding principle [37].

Figure 1.11 Illustrates the temporal coding procedure for encoding and decoding of real vectors into spike trains. The network supplied by serial of n-dimensional input vectors $X = (x_1, x_2, ..., x_n)$ which translates to the train of spikes within the successive temporal window. In each time window, a pattern X is temporally coded relative to the timing of the spike emission of the neuron.

To understand the limitation of the current definition of the rate and temporal coding, we should consider the case of two neurons with the same firing rate but different timing for generating spikes. The first neuron generates all its spikes at the beginning of the period and is silent afterward. The second neuron generates its spikes in an

evenly distributed time during the given period.

To differentiate between these neurons, we should consider the notion of the instantaneous firing rate. It seems clear that the instantaneous firing rate is higher for the first neuron at the beginning, however, the second neuron has a fixed instantaneous firing rate for the whole period [12]. If a rapid change in the instantaneous firing rate observed, which contains essential information about input stimuli, the method used called temporal coding.

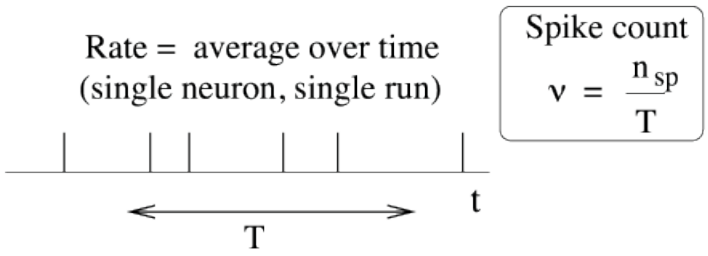


Figure 1.12: Definition of mean firing rate by temporal average.

Figure 1.12 presents the definition of the mean firing rate ν based on the number of spikes n_{sp} in a period T. Generally, Using the temporal code, neurons have a fluctuating firing rate in response to constant stimulus, whereas using rate code, the neurons exhibit the same firing pattern during the entire given period.

1.4.2 Population Coding

In opposition to rate and temporal coding, which consider the firing rate of an individual neuron, population coding illustrates the encoding behavior of a population of neurons. Regarding a large population of neurons that all execute the same code constitutes a high degree of redundancy between neurons. In population coding, we

have different groups of neurons that respond to a specific pattern of stimulus. For instance, if the aim is to identify the direction of an arrow in a picture, there are neurons that represent the direction fully turned to left, others to the right and a group of neurons that are specific to centered direction.

There is a direction for which a neuron has the highest firing rate. We call this direction the preferred direction of the neuron. Generally, neurons respond to the inputs which are close to their preferred direction, and they infrequently fire for the direction which is not similar to their preferred course.

1.4.3 Sine Wave Encoding

There is a specific type of encoding for supervised learning in spiking neural networks, known as sine wave encoding. In this method, the amplitude of the sine wave is proportional to the normalized feature of the raw input. We present this sine wave input for some portion of the simulation time. This method is quite similar to what we have seen in conventional neural networks as here, the amplitude of the sine wave is equivalent to the intensity of the input [41].

1.4.4 Spike Density Code

Spike density code is a specific form of population coding which consider the number of firing neuron during a given period. The aim is to set up a population of neurons such that the number of firing neurons is proportional to the input size. Therefore, the input information encoded as the density of the spikes produced by the population of the neurons [37].

The problem associated with this method appears when we use a large population of neurons to encode a relatively small input. The increased number of neurons and consequently increase in synaptic connection add more computational complexity to the system.

1.5 Synaptic Plasticity

In conventional neural networks, we optimize the performance of the network, modifying the connection weight w_{ij} between neurons i and j. The procedure of weight modification referred to as learning rule. A well-known method used to modify the connection weights in spiking neural networks is base on the work done by Konorski in 1948 [27] and Donald Hebb in 1949 [19].

Experiments confirm that the amplitude response of a postsynaptic neuron is not fixed and changes over time. In neuroscience, this change of the synaptic strength referred to as synaptic plasticity. In the presence of a proper stimulation paradigm, we can observe a persistence change in postsynaptic response, which may last for several hours.

If a persistent strengthening of synapses observed, the effect described as long-term potentiation of synapses (LTP). In opposition to long-term potentiation is long-term depression when we witness a reduction in the efficacy of neuronal synapses.

Hebb and Konorski described the change procedure in connection from presynaptic neuron A to a postsynaptic neuron B.

If an axon of the neuron A, which is in the proximity of the neuron B, persistently contributes to firing it, a rapid metabolic change occurs in both neurons such that

the synaptic efficiency of their connection increase. An oversimplified summary of their law presented in the following sentence [19].

"Neurons that fire together wire together."

However, this sentence does not precisely describe the Hebbian rule as we know the presynaptic neuron has to be active just before the latter one.

1.5.1 Mathematical Formulation of Hebb's Rule

Considering a single synapse with efficacy denoted by w_{ij}, we can present a mathematically formulated learning rule based on Hebb postulate. The synapse transmits electrical pulses from the presynaptic neuron i to the postsynaptic neuron j. Here, ν_i and ν_j represent the activity of the presynaptic and postsynaptic neurons in terms of the mean firing rate.

In Hebbian postulate, the changes of synaptic efficacy only depend on local variables and not on the activity of other neurons. Employing this characteristic, we can write a general formula (Eqn.1.24) for synaptic efficacy, having variables like pre and postsynaptic firing rates and the actual value for synaptic efficacy.

$$\frac{\mathrm{d}}{\mathrm{d}t}w_{ij} = F(w_{ij}; \nu_i, \nu_j) \tag{1.24}$$

Here, $\frac{\mathrm{d}}{\mathrm{d}t}w_{ij}$ denote the rate of change in synaptic strength, and F is a function that describes the synaptic change based on the local variable.

Other features of Hebb's postulate indicate that the change in synaptic weight happens when we have both pre and postsynaptic neurons active simultaneously.

Assuming that F is a well-behaved function we can use Taylor series to expand the function (Eqn.1.25)

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}w_{ij} =& c_0(w_{ij}) + c_1^{\mathrm{pre}}(w_{ij})\,\nu_j + c_1^{\mathrm{post}}(w_{ij})\nu_i + c_2^{\mathrm{pre}}(w_{ij})\,\nu_j^2 \\ &+ c_2^{\mathrm{post}}(w_{ij})\,\nu_i^2 + c_{11}^{\mathrm{corr}}(w_{ij})\,\nu_i\,\nu_j + \mathcal{O}(\nu^3)\,.\end{aligned} \tag{1.25}$$

In general, the Hebbian learning rule requires either the bilinear term $c_{11}^{\mathrm{corr}}(w_{ij})\,\nu_i\,\nu_j$ or higher-order term $(c_{21}(w_{ij})\,\nu_i^2\,\nu_j)$ that includes both pre and postsynaptic activity. If we disclude these terms form the equation (Eqn.1.25), we would have a non-Hebbian learning rule.

1.5.2 Pair-based Models of STDP

Employing a spike description for synaptic plasticity, we can present a pair-based update rule for synaptic strength. Assume that t_{pre} and t_{post} are respectively the time in which pre and postsynaptic spike happen. The change in synaptic weight is a function of temporal difference $|\Delta t| = |t_{\mathrm{post}} - t_{\mathrm{pre}}|$. a simple pair based update rule presented in (Eqn.1.26).

$$\begin{aligned}\Delta w_+ &= A_+(w)\cdot\exp(-\left|\Delta t\right|/\tau_+) \text{ at } t_{\mathrm{post}} \quad \text{for } t_{\mathrm{pre}} < t_{\mathrm{post}} \\ \Delta w_- &= A_-(w)\cdot\exp(-\left|\Delta t\right|/\tau_-) \text{ at } t_{\mathrm{pre}} \quad \text{for } t_{\mathrm{pre}} > t_{\mathrm{post}}\end{aligned} \tag{1.26}$$

Where $A_\pm(w)$ represents the update dependency on the current value of the synaptic weight. $A_+(w)$ and $A_-(w)$ normally have a positive and negative value respectively. Whenever a presynaptic and postsynaptic spike happens(respectively at time t_{pre} and t_{post}), we need to update the synaptic weight.

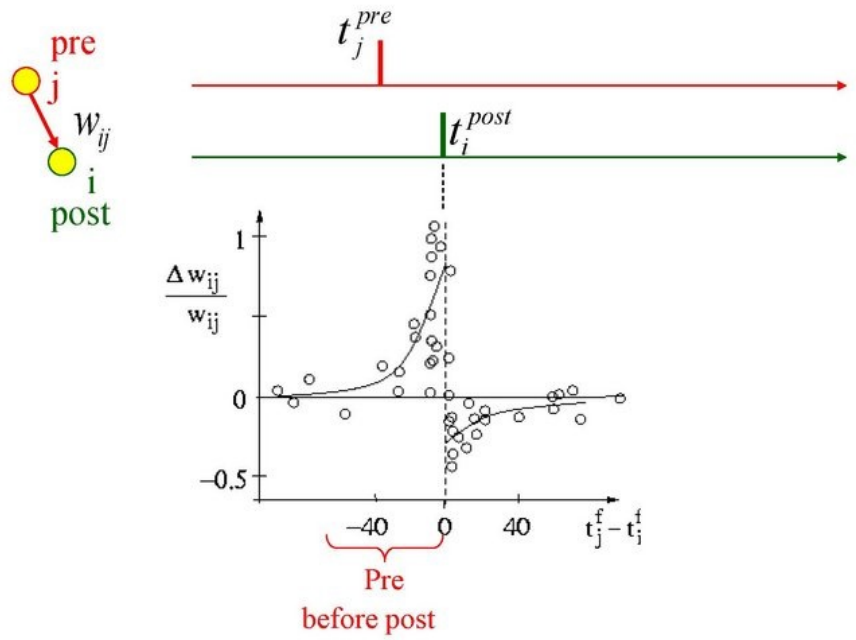


Figure 1.13: Spike-Timing Dependent Plasticity (schematic) [42].

Figure 1.13 presents the diagram of spike-timing-dependent plasticity. The STDP rule describes the changes in synaptic weights as a function of timing of pre and postsynaptic spikes.

Generally, we can specify a pair-based model by:

- the weight-dependence parameters $A_+(w)$ and $A_-(w)$
- The choice of pairs which have to take into consideration for performing the update.

The reason not to consider all possible pair of neurons is that neurons which are far

apart rarely participate in each others firing because of fast exponential decay of the update amplitude [43]. A reasonable choice is to consider pre and postsynaptic spike which are in proximity with each other.

Considering $S_j = \sum_f \delta(t - t_j^{(f)})$ and $S_j = \sum_f \delta(t - t_j^{(f)})$ as pre and postsynaptic spike trains, we can represent the update rule as follow:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} w_{ij}(t) = S_j(t) \left[a_1^{\text{pre}} + \int_0^\infty A_-(w_{ij}) W_-(s)\, S_i(t-s)\, \mathrm{d}s \right] \\ + S_i(t) \left[a_1^{\text{post}} + \int_0^\infty A_+(w_{ij}) W_+(s)\, S_j(t-s)\, \mathrm{d}s \right] \end{aligned} \tag{1.27}$$

Here a_1^{pre} and a_1^{post} are non-Hebbian parameters and and $W_\pm(s)$ represent the time scale of the learning window [26].

In the standard pair-based STDP rule, we can write:

$W_\pm(s) = \exp(-s/\tau_\pm)$ and $a_1^{\text{pre}} = a_1^{\text{post}} = 0$

Studies suggest that the pair-based STDP rule can not provide a satisfactory description of experimental results with synaptic plasticity protocols. One of the principal deficiencies of the pair-based model is its inability to produce dependence of plasticity on the spike frequency [16].

To address the issues associated with the pair-based model, Pfister and Gerstner introduced the triplet rule of STDP, which can resolve the problem of the frequency dependence [38].

1.5.3 Triplet model of STDP [38]

In the triplet model, every spike in the presynaptic neuron, j participates in the generation of a trace x_j. Denoting the firing time of presynaptic neuron with t_j^f, we can implement the triplet model of STDP model as (Eqn.1.28):

$$\frac{\mathrm{d}x_j}{\mathrm{d}t} = \frac{x_j}{\tau_+} + \sum_{t_j^f} \delta\left(t - t_j^f\right) , \tag{1.28}$$

Using this model, we need a combination of three spikes (one presynaptic and two postsynaptic) for simulating LTP.

$$\frac{\mathrm{d}y_{i,1}}{\mathrm{d}t} = -\frac{y_{i,1}}{\tau_1} + \sum_f \delta(t - t_i^f) \tag{1.29}$$

$$\frac{\mathrm{d}y_{i,2}}{\mathrm{d}t} = -\frac{y_{i,2}}{\tau_2} + \sum_f \delta(t - t_i^f) \tag{1.30}$$

Here, there are two different traces $y_{i,1}$ and $y_{i,2}$ for individual postsynaptic spike j. The time scale of these two traces are different, and we have $\tau_1 < \tau_2$ [38].

Now, we can represent the weight change rule based on the presynaptic trace x_j immediately after a postsynaptic spike and also postsynaptic trace $y_{i,2}$ from the previous postsynaptic firing.

$$\Delta w_{ij}^+\left(t_i^f\right) = A_+\left(w_{ij}\right)\, x_j\left(t_i^f\right) y_{i,2}\left(t_i^{f-}\right) \tag{1.31}$$

The term t_i^{f-} implies that we should calculate $y_{i,2}$ before its increase due to the effect of the postsynaptic spike at t_i^f. In triplet STDP, LTD is analogous to what we have

seen in the pair-based model [16].

The experiments confirm the full compatibility of this model with explicit triplet experiments [36].

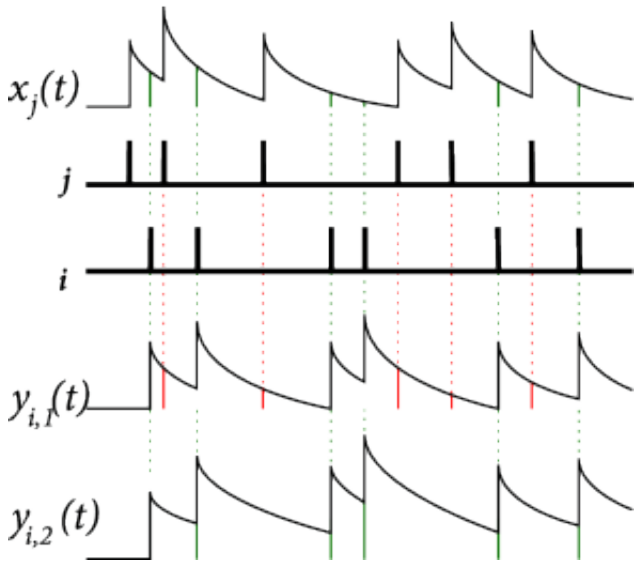


Figure 1.14: The triplet STDP rule with local variables [16].

Figure 1.14 describes the triplet STDP rule using local variables. $x_j(t)$ is the spike trace of presynaptic neuron j, while $y_{i,1}(t)$, and $y_{i,2}(t)$ are respectively fast and slow spike trace of postsynaptic neuron j. The update of the weight w_{ij} at the moment of a presynaptic spike is similar to pair based model of STDP.

1.6 Unsupervised Learning [15]

In contrast to supervised learning where the network parameters optimized for every input stimuli to achieve the least error, unsupervised learning refers to the change of synaptic connection according to the statistics of the input stimuli. Assume that we

have N input neurons with index of $1 \leq j \leq N$. We denote the firing rate of these neurons by ν_j. These firing rates belong to a set of P different firing rate pattern with the index of $1 \leq \mu \leq P$.

In a static pattern scenario, we present an input pattern like $\boldsymbol{\xi}^\mu = (\xi_1^\mu, \ldots, \xi_N^\mu)$ to the network for a predefined period Δt. Note that here, the firing rates of the neurons in the input layer are $\nu_j = \xi_j^\mu$.

Using the Hebbian learning rule of the form (Eqn.1.32), we can take advantage of competitive learning.

$$\frac{\mathrm{d}}{\mathrm{d}t} w_{ij} = \gamma \, \nu_i \left[\nu_j - \nu_\theta(w_{ij}) \right], \tag{1.32}$$

Here, γ is a positive constant, and ν_θ is the firing rate reference, which depends on the current value of the synaptic weight.

Considering a group of active neurons that connected to the postsynaptic neuron j, we will observe an increase in the strength of the synaptic connection. The firing of the postsynaptic neuron leads to long term potentiation(LTP). At the same time, the firing of the postsynaptic neuron causes the long term depression(LTD) on inactive synaptic pathways. The procedure here is similar to the "winner takes all" strategy in competitive learning.

1.6.1 Rate model learning

For a rate-based learning model, we can use a simple Hebbian rule to illustrate the joint activity of pre and postsynaptic neurons which leads to change in synaptic weights as (Eqn.1.33):

$$\Delta w_i = \gamma \, \nu^{\text{post}} \, \nu_i^{\text{pre}} \, . \tag{1.33}$$

Where γ called learning rate and has the range of $0 < \gamma \ll 1$. In the general form of the rate-based model, we can represent the postsynaptic firing rate as a function of the input stimuli (Eqn.1.34).

$$\nu^{\text{post}} = g\left(\sum_j w_j \, \nu_j^{\text{pre}}\right) \; ; \tag{1.34}$$

To gain a better understanding of the learning equation in (Eqn.1.33), we consider a simple linear model for the firing rate of the postsynaptic neuron. Suppose that g is a linear function, we can write (Eqn.1.34) as:

$$\nu^{\text{post}} = \sum_j w_j \, \nu_j^{\text{pre}} = \mathbf{w} \cdot \boldsymbol{\nu}^{\text{pre}} \, , \tag{1.35}$$

Note that using a linear function, we can represent the firing rate of the postsynaptic neuron as the projection of the input vector onto the weight vector. Coupling the learning rule in (Eqn.1.33) and linear rate model of (Eqn.1.35) we have:

$$\Delta w_i = \gamma \sum_j w_j \, \nu_j^{\text{pre}} \, \nu_i^{\text{pre}} = \gamma \sum_j w_j \, \xi_j^{\mu} \, \xi_i^{\mu} \, . \tag{1.36}$$

Then, we can illustrate the change in the weight vector after each iteration by equation (Eqn.1.37):

$$w_i(n+1) = w_i(n) + \gamma \sum_j w_j \, \xi_j^{\mu_n} \, \xi_i^{\mu_n} \, , \tag{1.37}$$

Using the equation (Eqn.1.37), we update the synaptic weight after presentation of each input pattern. For this reason, the method called the online learning rule of the rate-based model, which is in contrast to presenting a large number of the input pattern to the network.

We can consider the situation when we present all input patterns to the network before an update happens.

$$w_i(n+1) = w_i(n) + \tilde{\gamma} \sum_j w_j \sum_{\mu=1}^{P} \xi_j^\mu \, \xi_i^\mu \tag{1.38}$$

Here $\tilde{\gamma} = \gamma/P$ is the new learning rate called the batch learning rate. We can rearrange equation (Eqn.1.38) as:

$$w_i(n+1) = w_i(n) + \gamma \sum_j C_{ij} \, w_j(n) \,, \tag{1.39}$$

Where C_{ij} is the correlation matrix of the form:

$$C_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \, \xi_j^\mu = \langle \xi_i^\mu \, \xi_j^\mu \rangle_\mu \,. \tag{1.40}$$

It is clear now that the change in synaptic weights determined by the correlation of input vector.

1.6.2 STDP Learning Equations

In a pair-based STDP, we can use a Poisson model to generate the output spike train at the postsynaptic neuron. The firing rate of this Poisson group determined by:

$$\nu_i(u_i) = [\alpha\, u_i - \nu_0]_+ \tag{1.41}$$

Here u is the membrane potential, α is the scaling factor, and ν_0 is the threshold value. Note that the positive sign at the right side of the equation, denotes a piece-wise linear function in which: $[x]_+ = x$ for $x > 0$

Supposing all input spike trains as Poisson group of the firing rate ν_j, we can represent the expected firing rate of the postsynaptic neuron as (Eqn.1.42)

$$\langle \nu_i \rangle = -\nu_0 + \alpha\bar{\epsilon} \sum_j w_{ij}\, \nu_j\,, \tag{1.42}$$

Where $\bar{\epsilon}$ is the area under the postsynaptic potential of an excitatory neuron denoted by $\bar{\epsilon} = \int \epsilon(s)\mathrm{d}s$.

Finally, the correlation between pre and postsynaptic spike train estimated as:

$$\begin{aligned} \langle \dot{w_{ij}} \rangle = \nu_j \langle \nu_i \rangle [-A_-(w_{ij})\tau_- + A_+(w_{ij})\,\tau_+] \\ + \alpha w_{ij}\nu_j\, A_+(w_{ij}) \int W_+(s)\epsilon(s)\mathrm{d}s \end{aligned} \tag{1.43}$$

1.7 Summary

Neurons use a short voltage pulse called action potential (spike) to communicate with each other. These pulses distributes to several postsynaptic neurons where they excite postsynaptic potentials. If a sufficient number of spikes reaches to the postsynaptic neuron, its membrane potential exceeds a critical voltage (threshold), and neuron generates an action potential (fire a spike). This spike consider as the output signal of the neuron and transmits to other neurons.

There are different models of neurons with various levels of sophistication. The Hodgkin-Huxley model explains the generation of action potentials using three ion channels and ion current flow. The model stands paramount in describing the dynamic behavior of the biological neuron and incorporates most of the fundamental properties of an actual neuron. However, the complexity of the Hodgkin-Huxley neuron convinced neuroscientists to seek a more simplistic but computationally efficient model of neurons.

A simple model of a spiking neuron is the leaky integrate-and-fire (LIF) model, which applies a linear differential equation to represent how input currents integrated and converted into a membrane voltage $u(t)$. The simple model does not incorporate the notion of refractoriness. Including the mechanism of adaptation, the model can successfully predict spike times of cortical neurons.

Experiments demonstrated that the relative timing of the pre and postsynaptic spike plays an essential rule in determining the amplitude and direction of change in synaptic efficacy. To demonstrate the spike timing effects, standard pair-based models of STDP (synaptic time-dependent plasticity) formulated, which consider a learning window for modification of synaptic weights. If the presynaptic spike occurs before a

postsynaptic one, the synaptic weight will increase. In the case when a presynaptic spike arrives after a postsynaptic one, synaptic efficacy decrease. Nevertheless, classical pair-based STDP models ignore the frequency and voltage dependence of synaptic plasticity. Modern variants of STDP like triplet rule proposed to fix deficiencies of the pair-based model.

1.8 Outline of the thesis

This thesis has organized into five chapter. **Chapter 2** presents a brief review of supervised and unsupervised learning in spiking neural networks.
A computationally efficient SNN for classification of images of handwritten digit has proposed in **Chapter 3**. **Chapter 4** belongs to an unsupervised SNN for recognition of the MNIST dataset. Conclusion and potential future work delivered in **Chapter 5**.

Chapter 2

SURVEY OF RELEVANT LITERATURE

2.1 A Review of Supervised Learning in SNN

The first supervised algorithm which used a gradient-based technique to transfer information in the timing of the single spike was SpikeProp [4]. In this model, each neuron can produce at most one action potential during the spike interval. If the neuron fires more than one spike during the period, the algorithm only considers the first spike as the exact firing time. The model comprised of the connections with different synaptic delays and weights, which enable them to solve linearly inseparable problems(like XOR function) and attain high-grade results on the problem with a small dataset. However, having multiple connection weights per synapse and adopting a single spike optimization procedure restricted its application to the problems with small datasets.

McKennoch, Liu, and Bushnell [32] proposed the method to enhance the convergence rate of the SpikeProp, though their approach was not expandable to large datasets. An alternative method to SpikeProp proposed in [34] which specially designed for non-

leaky integrate and fire models. The model replaced the multi delay elements of the SpikeProp model with an exponential connection between each pair of neurons. The model replaced the multi delay elements of the SpikeProp model with an exponential synaptic connection between each pair of neurons. The single and two-layer model of the proposed network achieved the test error of 2.45% and 2.86%, respectively. The main associated problems with the proposed method is a dropout since most of the regularization techniques do not apply to the network and some times prevents neurons from firing.

Stromatias and Marsland [45] used a different approach than the previous works and employed the genetic algorithm to optimize multiple spikes of each neuron instead of considering only the first spike. However, this method only applies to small networks with less than ten neurons in the hidden layer. One of the main reasons is the limitation of the genetic algorithm for scaling to problems with so many parameters.

Lee, Delbruck, and Pfeiffer [30] proposed a different method for optimizing multiple spikes of the neuron, assuming the output of the neuron as a linear function of its input. This simplification allows them to train the network in the forward direction and still can perform backpropagation in the backward direction. The method ignores the refractory period following the generation of a spike and use the property of lateral inhibition to enhance the performance of the network. Despite all the simplification, the model still able to achieve good results on the MNIST dataset, obtaining a test error of 1.30% using stochastic gradient descent.

2.2 Unsupervised Learning in SNN

During recent years, various strategies for unsupervised learning in spiking neural networks developed, which often based on variants of the Hebbian method. Inspired by the Hopfield's idea, Natschläger and Ruf [35] introduced an unsupervised clustering method in spiking neural networks. Their approach is analogous to the radial basis function (RBF) except the input, which is in terms of spike timing.

A winner-takes-all learning rule used to adjust the synaptic weights between the source neuron and the first firing neuron in the target layer. If the start of the postsynaptic potential occurs immediately before the spike in the target neuron, the weights of the synapse will increase. On the other hand, the synaptic weights of the earlier and later synapses will decline, which indicates their negligible impact on the firing of the target neuron. Employing this learning procedure, we can encode input patterns into synaptic weights in such a way, the spike timing of the output neurons indicates the difference between the evaluated pattern and the learned input pattern, which is quite similar to unsupervised learning in RBF neuron.

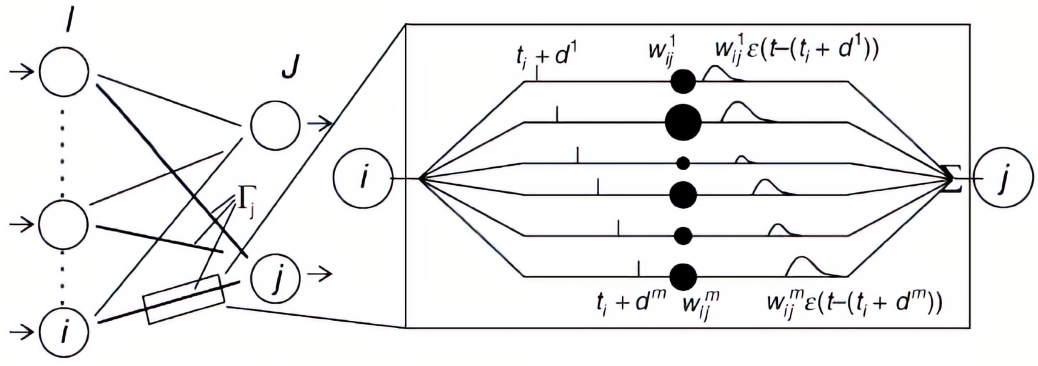


Figure 2.1: Unsupervised learning rule in SNN proposed in [35].

To improve accuracy and expand the clustering capacity of the Natschläger and Ruf

network, Bohte [5] applied a temporal version of population coding. He applied multiple receptive fields to encode the input data into temporal spike-time patterns. Bohte proved using such an encoding technique, spiking neural networks are capable of performing efficient clustering tasks. Figure 2.1 presents the unsupervised SNN proposed by Natschläger and Ruf in which individual connection considered as multisynaptic. The weights are random and a set of increasing delays introduced to facilitate unsupervised learning of input patterns.

Querlioz and his colleagues [39] introduced a simplified and customized spike time-dependent plasticity (STDP) scheme for unsupervised learning in memristive devices.

Their network comprised of an unsupervised layer that extracts features of the inputs images utilizing a rectangular shape of STDP and achieves the accuracy comparable to traditional supervised learning models with the same number of parameters. They imployed homeostasis and lateral inhibition to encourage competition among neurons. The neuron used in their network is a current based leaky integrate and fire model with the equation presented in (Eqn.2.1)

$$\tau \frac{dX}{dt} + gX = \gamma I_{\text{input}} \tag{2.1}$$

Where τ is the time constant of the leakage, and I_{input} describe the flowing current through the crossbar lines connected to the neuron. g and γ are also other constants of the equation which describe the dynamic of the neuron. They illustrated the high adaptivity of their systems to various environments, which can lay the foundation for circuit design with compact and low power consumption.

Figure 2.2 displays the crossbar layout of the network proposed by Querlioz [39] in which neurons are CMOS silicon devices, and their associated synaptic connections

are the dots. The synapses serve as adaptive resistors. Employing the crossbar layout, the output calculated directly as the sum of the currents passing through the synapses.

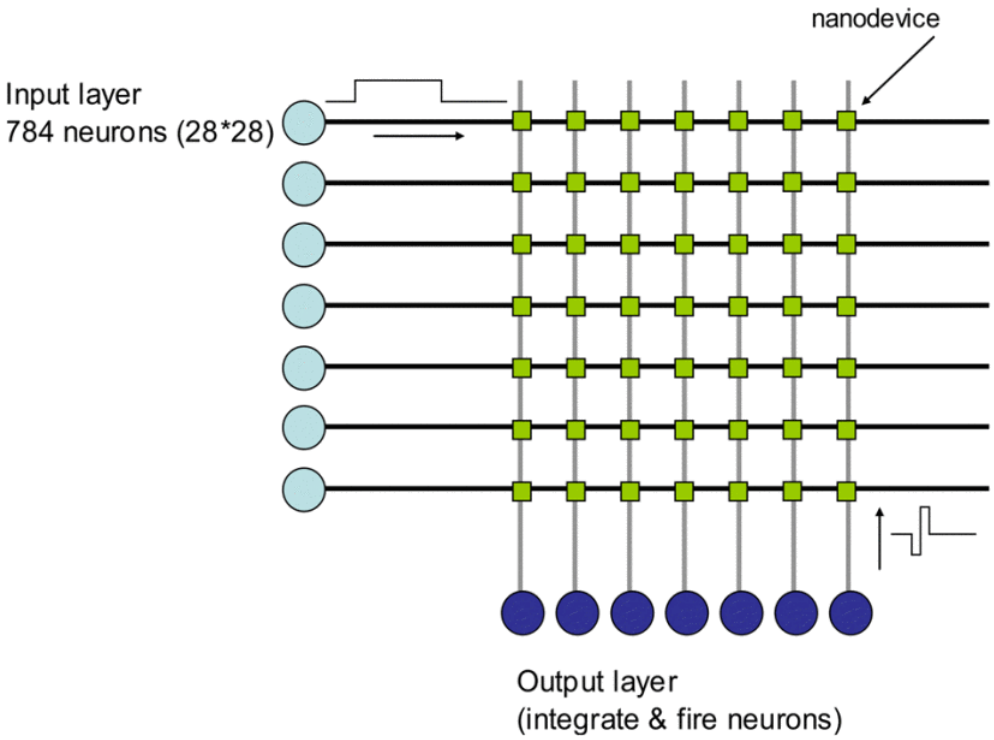


Figure 2.2: Architecture of the memristor-based SNN proposed in [39].

Diehel and Cook [13] proposed an unsupervised method for digit recognition using a conductance-based model of leaky integrate and fire neuron. They introduced an adaptive threshold method which prevents a neuron from dominating the response to the input pattern and facilitate the competition among neurons. Using 3600 excitatory neurons, they obtained an accuracy of 95% on the handwritten digits of the MNIST dataset. Their model consists of the same number of inhibitory neurons in the output layer. The neurons in the excitatory layer are connected in a one to one fashion to the corresponding inhibitory neuron in the output layer. The neurons in the inhibitory layer connect to all the other neurons in the excitatory layer except

their corresponding neuron in the excitatory layer (See Figure 2.3). This architecture allows them to use the property of lateral inhibition in which the first firing neuron inhibits all the other neurons in the output layer plus their corresponding excitatory neuron. The lateral inhibition enables the neuron to adapt its weights according to the input pattern.

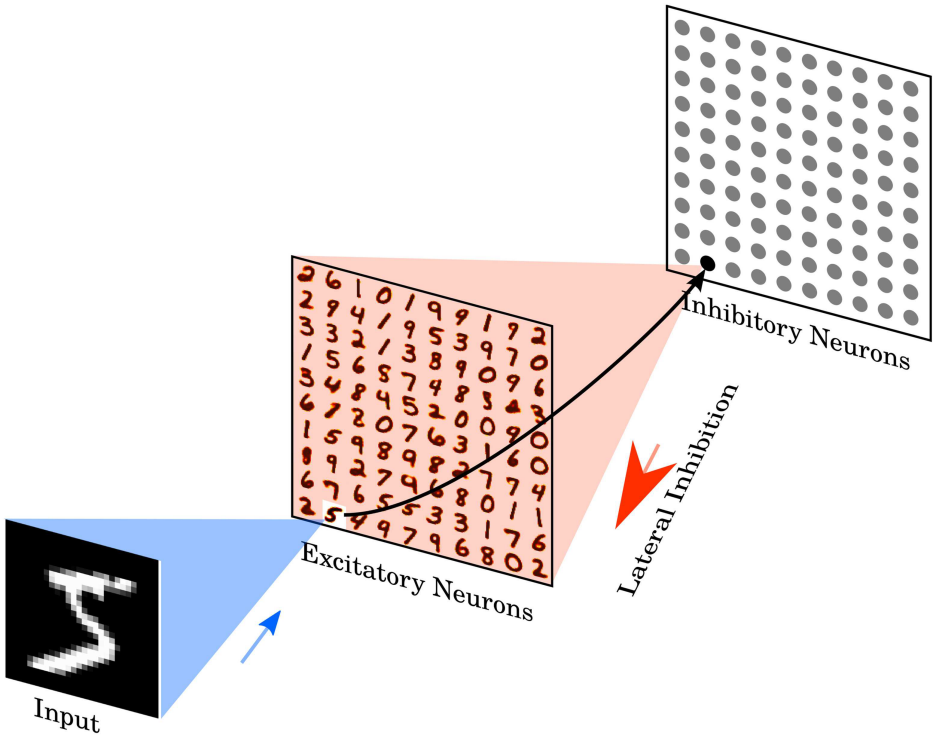


Figure 2.3: Architecture of the Diehel & Cook network [13].

2.3 Summary

The first algorithm which performed supervised learning in spiking neural networks was SpikeProp [4]. This algorithm and other similar methods, which referred to as spike-based methods, optimize the firing time of individual neurons to reduce the

overall error of the network. A problem associated with the spike-based methods is the high nonlinearity of the problem in which a small change in the input of the neuron can push the membrane potential to its firing threshold and substantially change the neuronal output.

In contrast to supervised methods, we have unsupervised approaches that utilize the properties of the Hebbian learning rule and competitive learning for modification of synaptic weights. A biologically plausible spike-timing-dependent plasticity (STDP) rule updates the weights based on the timing of the pre and postsynaptic spikes.

Chapter 3

PROPOSED SNN FOR IMAGE CLASSIFICATION

3.1 Introduction

In this section, a python implementation of the spiking neural network applied to classify the black and white handwritten digit of the MNIST [29] dataset . The neuron model employed in this section inspired by the simplified spike response model proposed by [22]. The learning method for updating synaptic weights is the pair-based spike time-dependent plasticity (STDP). The proposed method incorporates some of the fundamental properties of the biological neuron, such as homeostasis and lateral inhibition. The later parts belong to the simulation results for classifying of the handwritten digits of the MNIST dataset and the comparison of the suggested network to some of the related works such as [22] and [8].

3.2 Network Architecture

The SNN model presented here is a two-layer feedforward network consisted of 784 neurons in the input layer (equal to the size of an MNIST image which is 28×28), and eight inhibitory neurons in output layer for classifying six different input patterns. Each neuron in the input layer connected to all the neurons in the output layer through a weighted synaptic connection (See Figure 3.1). Input neurons in the first layer require spike trains, and we should encode the input image to a train of spikes in which the frequency of the spike pattern is proportional to the intensity of the pixel in the input image. The membrane potential of the neuron updated after each time step according to the learning rule and the associated synaptic weights. First firing output neuron inhibits all the other neuron in the output layer form generating spikes and win the competition for the specific input pattern.

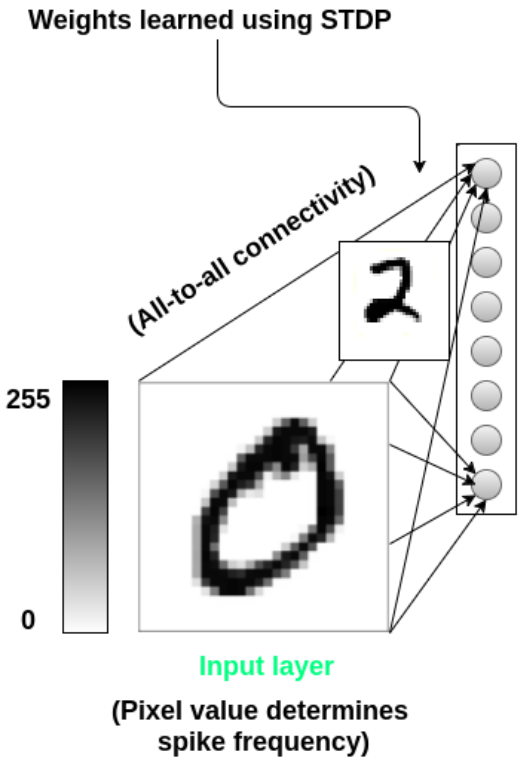


Figure 3.1: Proposed SNN architecture for image classification.

3.2.1 MNIST dataset

The MNIST database [29] (which stands for Modified National Institute of Standards and Technology database) is a big database of gray scale handwritten digits which widely adopted for training of various image processing methods.

Figure 3.2: Sample images from MNIST test set.

The database contains 60,000 training and 10,000 testing images, each of the size 28×28 pixels. The intensity of each pixel in the image represented by a number in range 0 to 255 in which higher numbers correspond to bright colors and darker shades represented by small values. Figure 3.2 illustrates sample images of the MNIST test set.

3.2.2 Input Encoding

Input images to the network are six arbitrary images of the MNIST dataset illustrated in Figure 3.3. Each image is of size 28×28 pixels and represented as a matrix in

which the value of each position is proportionate to the intensity of the corresponding pixel in the input image.

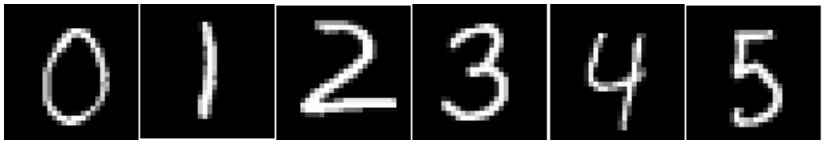


Figure 3.3: Sample images used for classification.

Since this input representation is not understandable for our spiking neural network, we should encode the image input to train of spikes in which the frequency of the spike train is proportional to the intensity of the pixel in the image. This type of encoding in which the information represented as the firing rate of the neuron called rate coding.

3.2.3 Neuron Model

Considering a spike response model (SRM), the postsynaptic neuron generates a potential (postsynaptic potential) whenever it receives a presynaptic spike. This potential is excitatory whenever the membrane potential increased and is inhibitory when it decreased. To determine the instant value of membrane potential, we need to aggregate all existing PSP at the neuron input. When the membrane potential exceeds the critical threshold value, neuron generates an action potential and enter its refractory period. During the refractory period, neuron is overpolarised and is not able to generate action potentials. After this short period, membrane potential resets to its resting value and can produce spikes again. Input neurons in the first layer require spike trains, and we should encode the input image to a train of spikes in which the frequency of the spike pattern is proportional to the intensity of the pixel

in the input image.

We can describe the postsynaptic function as follow:

$$\mathrm{PSP}(t) = e^{(-\frac{t}{\tau_m})} - e^{(-\frac{t}{\tau_s})} \tag{3.1}$$

Where τ_m and τ_s are the time constants that describe the steepness of the curve for LTP, and LTD respectively, and t is the time after the arrival of the presynaptic spike.

Denoting the threshold value by υ, we can present the refractory η function by equation

$$\eta(t) = -\upsilon e^{(\frac{t}{\tau_r})} H(t) \tag{3.2}$$

Here H is the Heaviside function, and τ_r is the time constant for the refractory period.

Considering a train of spikes $F_i = \left\{ t_i^{(g)}, \ldots, t_i^{(K)} \right\}$ arriving at a postsynaptic neuron, we can write the potential equation for j-th neuron as (Eqn.3.3)

$$P_j = \sum_{i}^{K} \sum_{t_i^{(g)} \in F_i} w_{ij} \mathrm{PSP}(\Delta t_{ij}) \sum_{t_j^{(f)} \in F_j} \eta \left(t - t_j^{(f)} \right) \tag{3.3}$$

Where Δt_{ij} is the time difference between presynaptic spike and the change in postsynaptic potential considering the delay d_{ij}, and described by equation

$$\Delta t_{ij} = t - t_i^{(g)} - d_{ij} \tag{3.4}$$

Here, F_j is the trains of spike generated by the postsynaptic neuron.

Simplified SRM Neuron [22]

Without losing the generality of the SRM model, we can consider a simplified model of spike response model as [22] in which the membrane potential of the postsynaptic neuron increases according to incoming spike trains S_{it}, $i = [1, ..., n]$. On the other hand, the membrane potential decreases by a constant value D in every time instant (considering time instants as discrete values).

The postsynaptic potential donated by (Eqn.3.5)

$$P_t = \begin{cases} P_{t-1} + \sum_{i=1}^{n} W_i S_{it} - D, & \text{if } P_{\min} < P_{t-1} < P_{\text{threshold}} \\ P_{\text{refract}} & \text{if } P_{t-1} \geq P_{\text{threshold}} \\ P_R & \text{if } P_{t-1} \leq P_{\min} < 0 \end{cases} \tag{3.5}$$

Figure 3.4 illustrates the membrane potential of the simplified neuron in response to random input spike trains for a duration of 50 time units.

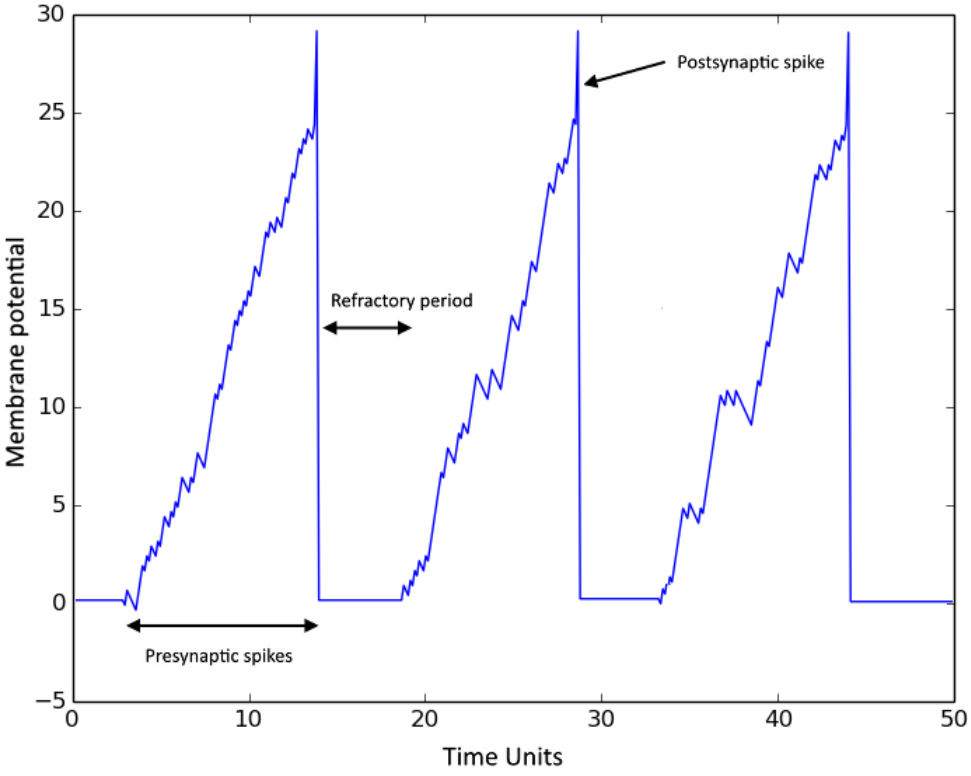


Figure 3.4: Membrane potential of the simplified neuron in response to random input spike train.

3.2.4 Learning Rule

A pair-based spike time-dependent plasticity rule employed to update synaptic weight connections. Generally speaking, we can explain the STDP rule as follow:

- All the synaptic connections which contribute in the firing of the postsynaptic neuron should strengthen, and in other words, we should increase their associated weights.
- Synapses that are not contributing to the firing of the postsynaptic neuron

should weaken, and the learning rule reduces their corresponding weights.

Here, the firing rate of the presynaptic neurons is proportional to the intensity of the input signals. The frequency of the spike train transferred to the postsynaptic neuron depends on the strength of the synaptic connection. Whenever the membrane potential of the postsynaptic neuron exceeds the threshold value, it generates an action potential. At this moment we should monitor all the presynaptic neuron which have produced spikes immediately before the postsynaptic neuron, and increase their corresponding synaptic weights.

The weight change in the synaptic connection represented in equation (Eqn.3.6). Note that this change is inversely proportional to the time difference between pre and postsynaptic firing. See Figure 3.5.

$$\mathrm{STDP}(\Delta t) = \begin{cases} A^{+} e^{-\Delta t/\tau^{+}} & \text{if } \Delta t > 0 \\ A^{-} e^{\Delta t/\tau^{-}} & \text{if } \Delta t < 0 \end{cases} \tag{3.6}$$

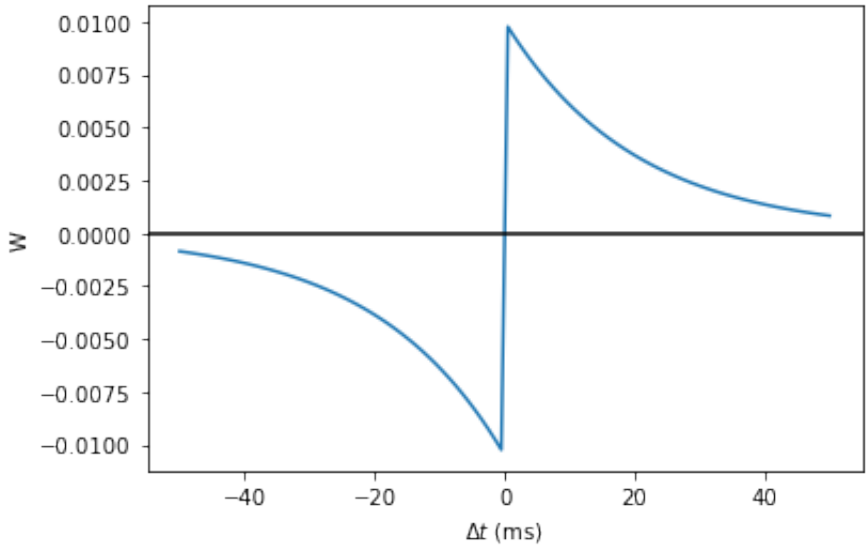


Figure 3.5: STDP curve of (Eqn.3.6).

Here, A^+ and A^- are respectively positive and negative constant of the weight change. τ^+ and τ^- are time course of the LTP and LTD, which describe the steepness of the function.

The total weight change Δw presented as:

$$\Delta w = \sum_{f=1}^{N} \sum_{n=1}^{N} \text{STDP}(t_i^n - t_j^f) \tag{3.7}$$

Where t_j^f and t_i^n are firing times of the pre and postsynaptic neuron.

And finally, the new weight obtained by (Eqn.3.8)

$$w_{\text{new}} = \begin{cases} w_{\text{old}} + \eta \Delta w (w_{\text{max}} - w_{\text{old}}) & \text{if } \Delta w > 0 \\ w_{\text{old}} + \eta \Delta w (w_{\text{old}} - w_{\text{min}}) & \text{if } \Delta w < 0 \end{cases} \tag{3.8}$$

Where η is the learning rate which controls the speed of the weight adaptation. To prevent synaptic weights to become extremely large or negative, we should consider a boundary condition for the connection weight such that $w_{\min} < w < w_{\max}$.

Lateral Inhibition

Since STDP considered as an unsupervised learning method, we can apply the characteristics of the competitive learning to our problem to build competition between neurons in the output layer. From a biological perspective, the ability of an excited neuron to degrade the activity of its neighbor called lateral inhibition. Adopting lateral inhibition, we can produce a contrast in stimulation, which leads to an increase in sensory perception. This quite similar to what we have seen as the winner takes all strategy (WTA) in unsupervised learning methods in machine learning.

Here, the winner is the neuron, which produces the first action potential in response to the input pattern. Following the generation of the first spike, it inhibits all the other neurons in the output layer from firing and fully adapts its associated synaptic weights to the input pattern (See Figure 3.6).

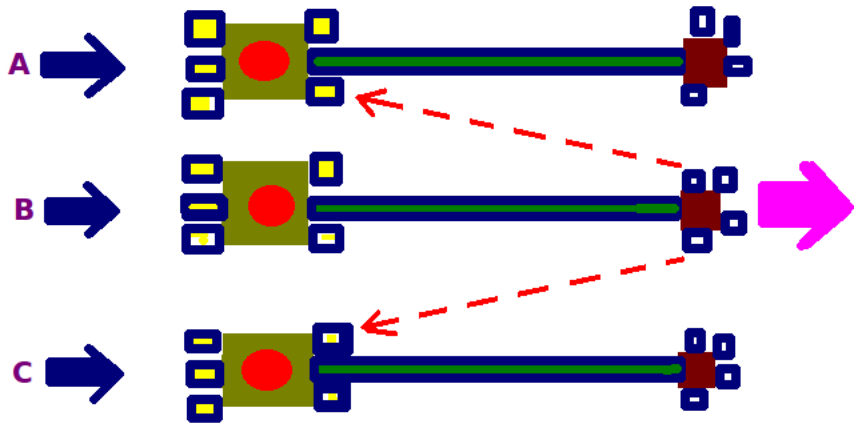


Figure 3.6: Neuron B (Winner) sends lateral signals to Neuron A and C.

Adaptive Variable Threshold

Choosing a fixed threshold value for all neurons creates a situation in which a single neuron dominates the response pattern and prevents all the other neurons from participating in the competition. To resolve the issue, we can apply an adaptive value of threshold in which we increase the threshold value of a neuron whenever it fires an action potential. we can represent this adaptive threshold as $V_{\text{th}} + \theta$, in which the value of θ increases after an action potential and decaying afterward.

Adopting the adaptive threshold, it turns out that the excited neuron requires more input spikes to fire again in the short-period following its action potential and therefore provides the opportunity for the other neurons to compete for the next input patterns. This feature in which the inhomogeneity of the input patterns, cause exci-

tatory neuron to have different firing rates called homeostasis.

3.2.5 Parameters Tuning

Some of the most significant parameters that need to be taken care of while designing a spiking neural network are:

- Learning rate
- Potential threshold
- Initialization of the synaptic weights
- Range of the weights
- Firing rate of the input neurons

Learning rate determines the speed of weight adaptation during the learning process. Selecting a higher value for learning rate, speeds up the creation of the receptive field. However, this may lead to what we know as negative learning.

Similar to what we observed in conventional neural networks, the initialization of the weight is of the highest importance, which can reduce the computational expense to a substantial level and further increase the accuracy of the network to an optimal point.

The firing rate of the input neurons determines the magnitude of change in synaptic weights. Selecting a lower value for the firing rate of the input layer causes the membrane potential not to cross the threshold, and consequently, no changing of the

weights would take place. On the other hand, choosing a higher value of firing rate leads to negative learning and the creation of a noisy receptive field.

3.3 Simulation Results

After training the network and obtaining the optimal value of the weights, we can assess the performance of the system using the learned weights. Figures 3.8 - 3.15 depict the membrane potential of the output neuron in response to the input pattern. We present the input patterns to the network for the time unit of 500 ms and asses the performance of the system by monitoring the activity of the membrane potential of the output neurons in response to input images. Presenting the images of zero to five to the network consecutively, we observe that six out of eight neurons are responding to a specific input and consequently learned that particular pattern. The neurons 3 and 6 are noisy outputs and did not learn any input pattern. These neurons generate random spikes at the beginning of any input presentation and remained silent for the rest of the period.

An essential feature of the spiking neural networks, which is highly beneficial in analyzing the training process, is the generative properties of SNN. If we properly scale all the synapses connected to an output neuron and rearrange them in the form of the input image, it reveals the specific pattern that the output neuron learned. In our case, we need to properly scale all the 784 synapses connected to the output neuron and rearrange them into an image of size 28×28.

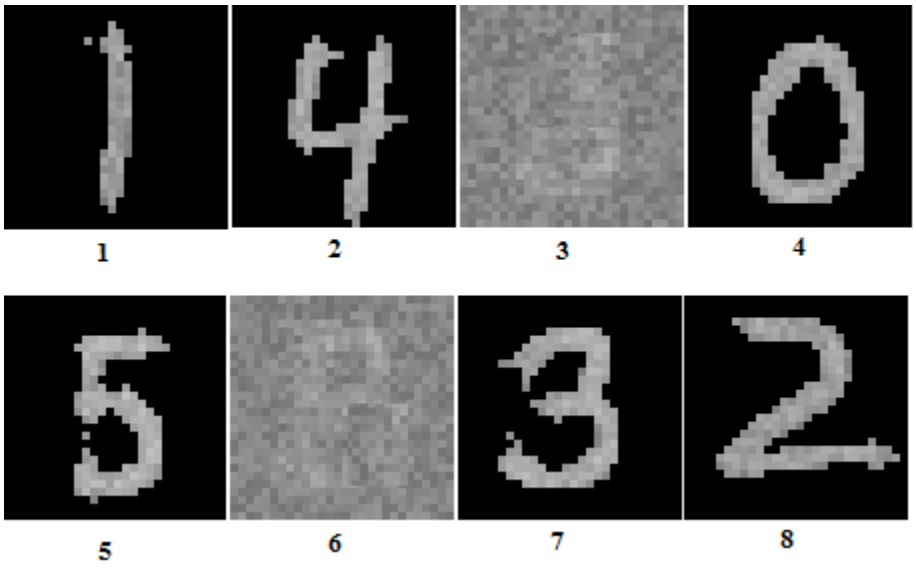


Figure 3.7: Receptive field of output neurons.

Figure 3.7 depicts the receptive field of output neurons. For instance; neuron 4 learned the pattern zero, and neuron 8 corresponds to the digit two. Neuron 3 and 6 considered as noisy output neurons and do not present any input pattern.

Some of the essential parameters for updating the synaptic connection are $A^+ = 0.8$, $A^- = 0.3$, $\tau^+ = 6$ $\tau^- = 4$, $\eta = 0.1$.

The simulation time for the classification of six digits of the MNIST dataset using eight output neurons lasted 32.25 seconds, which indicates a reduction of 78 percent in the overall simulation time in comparison with the classic SRM model (Using the same architecture and simulation time of 152.3 seconds).

The hardware used to run the simulation was intel Core i7 4747 CPU with 8 GB RAM.

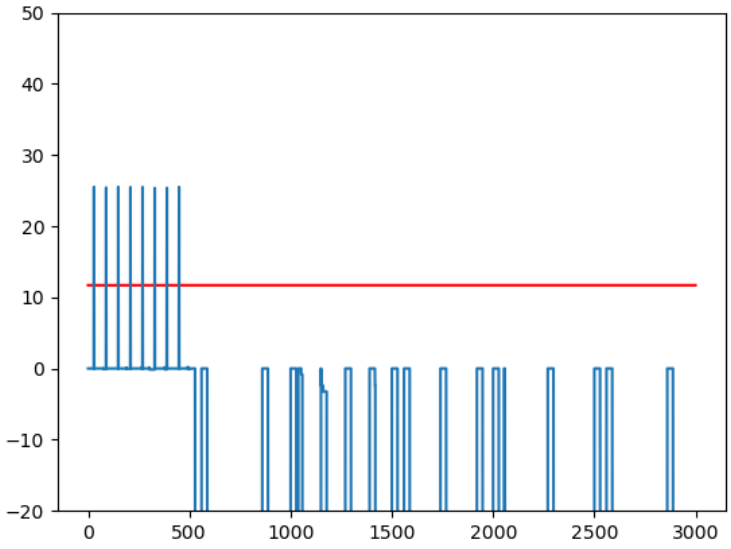


Figure 3.8: Membrane potential of the neuron 4 in response to input images.

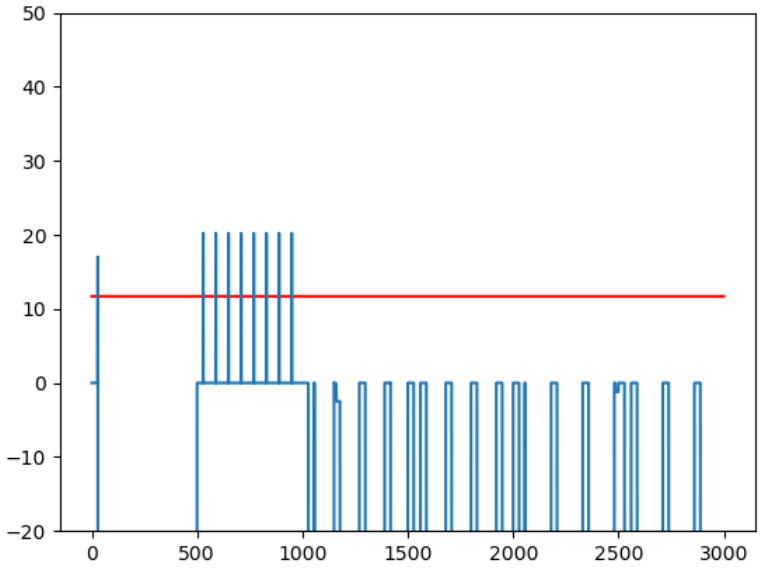


Figure 3.9: Membrane potential of the neuron 1 in response to input images.

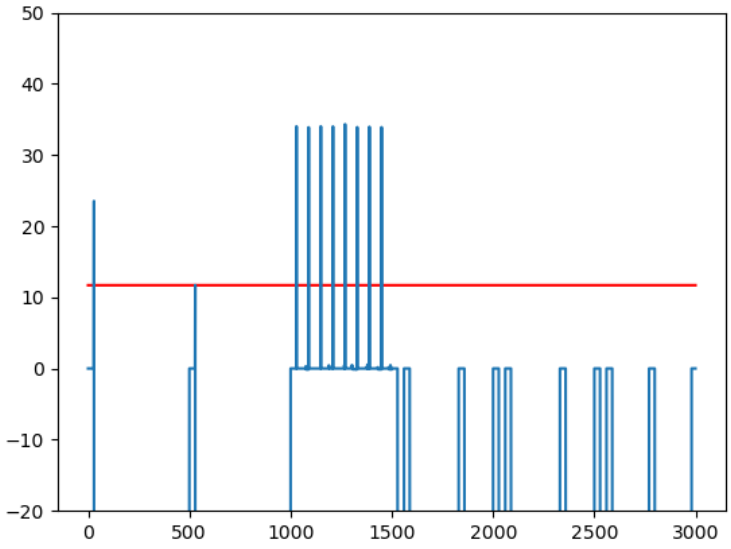


Figure 3.10: Membrane potential of the neuron 8 in response to input images.

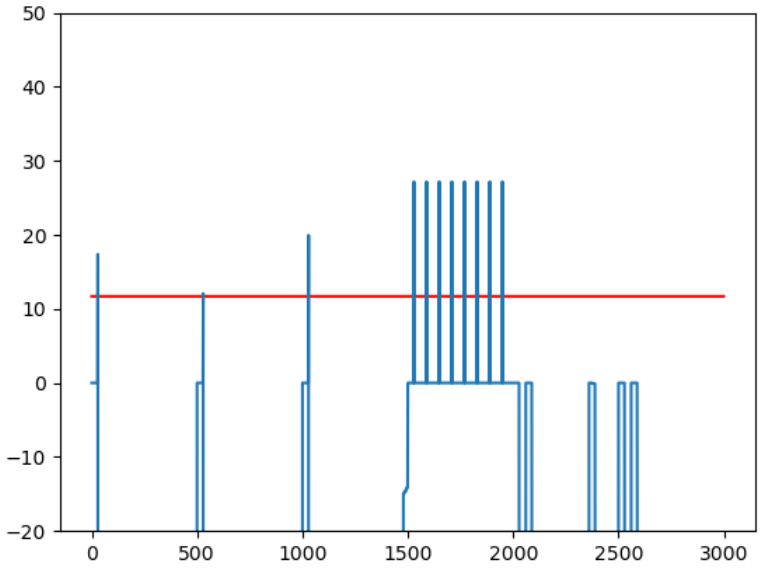


Figure 3.11: Membrane potential of the neuron 7 in response to input images.

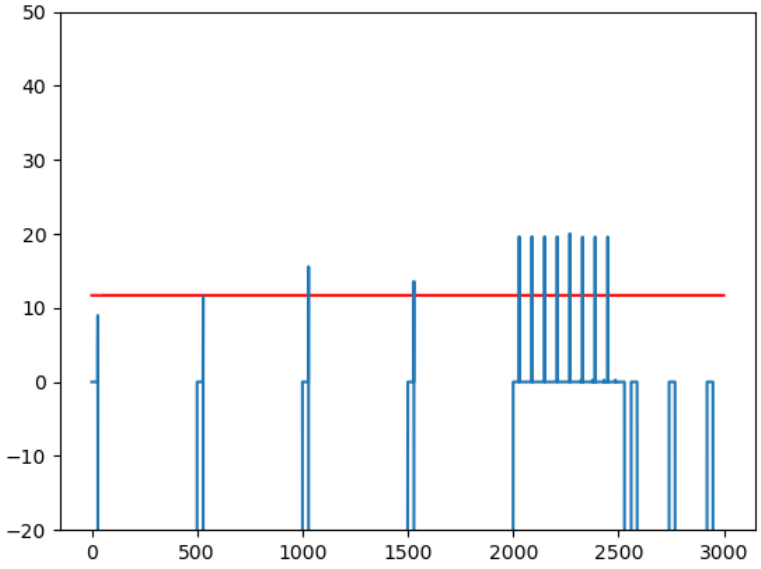


Figure 3.12: Membrane potential of the neuron 2 in response to input images.

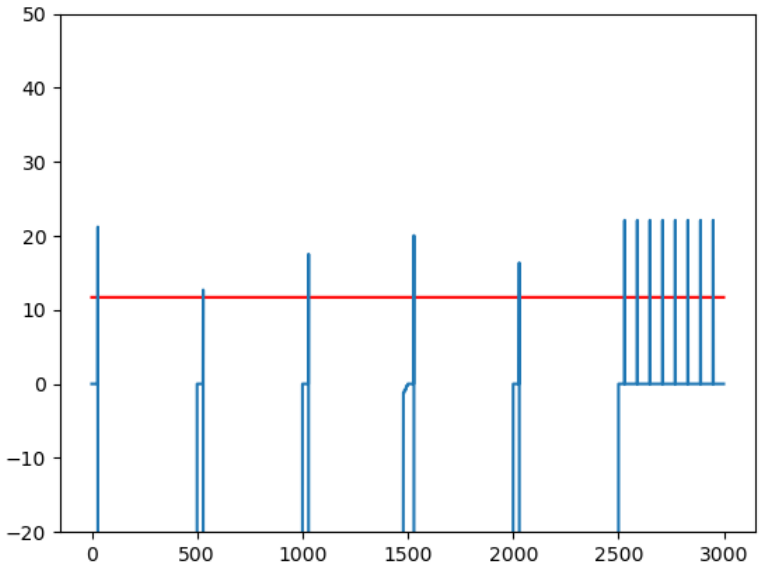


Figure 3.13: Membrane potential of the neuron 5 in response to input images.

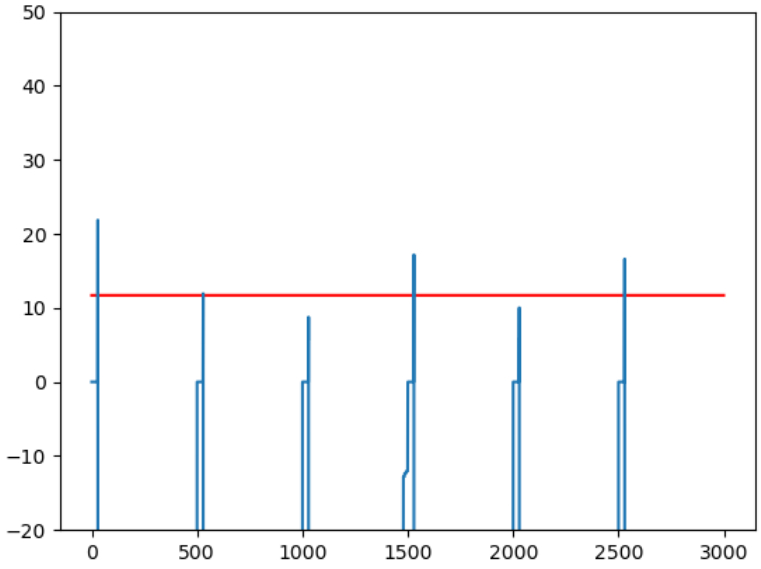


Figure 3.14: Membrane potential of the neuron 3 in response to input images.

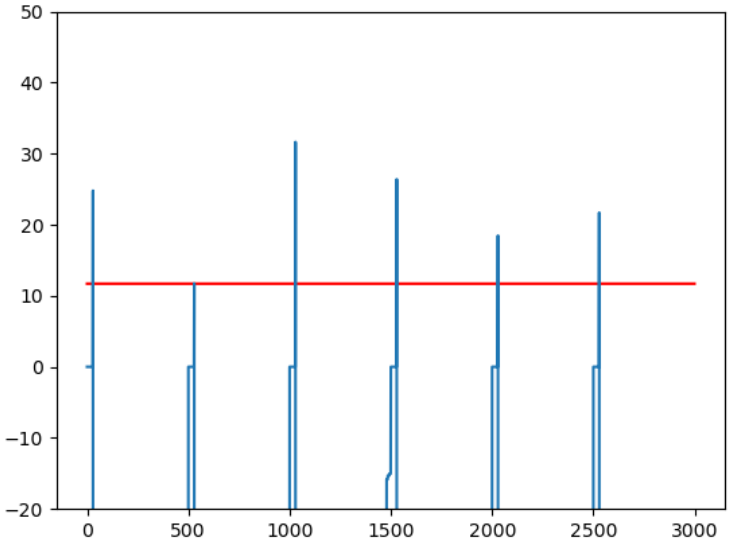


Figure 3.15: Membrane potential of the neuron 6 in response to input images.

3.4 Conclusion

The model proposed in this chapter is a computationally efficient SNN for classifying the handwritten digit images of the MNIST dataset. The neuron model is a simplified spike response model inspired by the model presented in [22]. The main distinction between our model and the model in [22] is that their model uses a convolutional filter for preprocessing of data before presenting the input to the network. They also use a modified asymmetric STDP rule to update the synaptic weights. However, the model suggested in this chapter does not perform any preprocessing of data, and the learning rule is a standard pair-based STDP.

A different architecture presented in [8], includes a hidden layer and two neurons in the output layer. The neurons in the network joined in a fully-connected fashion. One neuron in the output layer represents the input pattern, and the other one considered as a neutral neuron for handling noisy input. One of the problems associated with this architecture is that in every step of training, the network presents a single pattern and can not memorize more than one image. Another problem associated with this model is that we require different numbers of neurons in the hidden layer according to input images of various sizes. The model presented here is a multi-class classifier and is adjustable to the input images of various sizes. Table 3.1 present a basic comparison of the aforementioned network.

Architecture	**Preprocessing of Data**	**Learning Rule**	**Multi/Single Classifier**	**Correct Learning**
Simplified Neuron [22]	✓	Asymmetric STDP	Multi	100%
SNN in [8]	✗	Standard STDP	Single	80.3%
Proposed Method Here	✗	Standard STDP	Multi	100%

Table 3.1: Comparison of three SNN.

Chapter 4

HANDWRITTEN DIGIT RECOGNITION USING STDP

4.1 Introduction

In this section, an unsupervised learning approach suggested for the recognition of handwritten digits of the MNIST dataset. The neuron model is leaky integrate and fire model with conductance-based synaptic representation. An online spiking time-dependent plasticity (STDP) rule applied for modification of the weighted connections. The proposed method incorporates some of the significant properties of the biological neuron, such as lateral inhibition and homeostasis. BRIAN2 and python programming language used to simulate the spiking neural network [18].

4.2 Method

4.2.1 Neuron and Synapse Model

The model used to describe the dynamic of the network is the simple leaky integrate and fire (LIF) model. The differential equation that defines the dynamic behavior of the LIF neuron expressed as:

$$\frac{dV}{dt} = \frac{[(V_{\text{rest}} - V) + I]}{\tau} \tag{4.1}$$

Where τ is the leakage time constant, V is the membrane potentials, and V_{rest} is the membrane resting potential and I is the input current.

We can represent the input current as $I = I_e + I_i$, in which I_e and I_i are respectively excitatory and inhibitory input currents.

To define the synaptic change equation, a conductance-based model presented in which the conductance of the synapse increases by synaptic weight w whenever a presynaptic spike arrives at the synapse and will decline exponentially in other situations. The dynamic of the conductance for excitatory and inhibitory synapses presented as (Eqn.4.2) and (Eqn.4.3).

$$\tau_{g_e} \frac{dg_e}{dt} = -g_e \tag{4.2}$$

$$\tau_{g_i} \frac{dg_i}{dt} = -g_i \tag{4.3}$$

In which g_e and g_i are respectively conductance of the excitatory and inhibitory synapses. τ_{g_e} and τ_{g_i} are time constant of the postsynaptic potentials.

4.2.2 Network Architecture

Similar to the architecture proposed in [40], the network comprised of two layers that connected in a feedforward fashion. The input layer constituted of 784 neurons, equal to the size of the input image, which is 28×28. The neurons in the output layer are connected by inhibitory synapses, which allow the first firing neuron to perform lateral inhibition and prevent other neurons in the output layer to generate spike (See Figure 4.1).

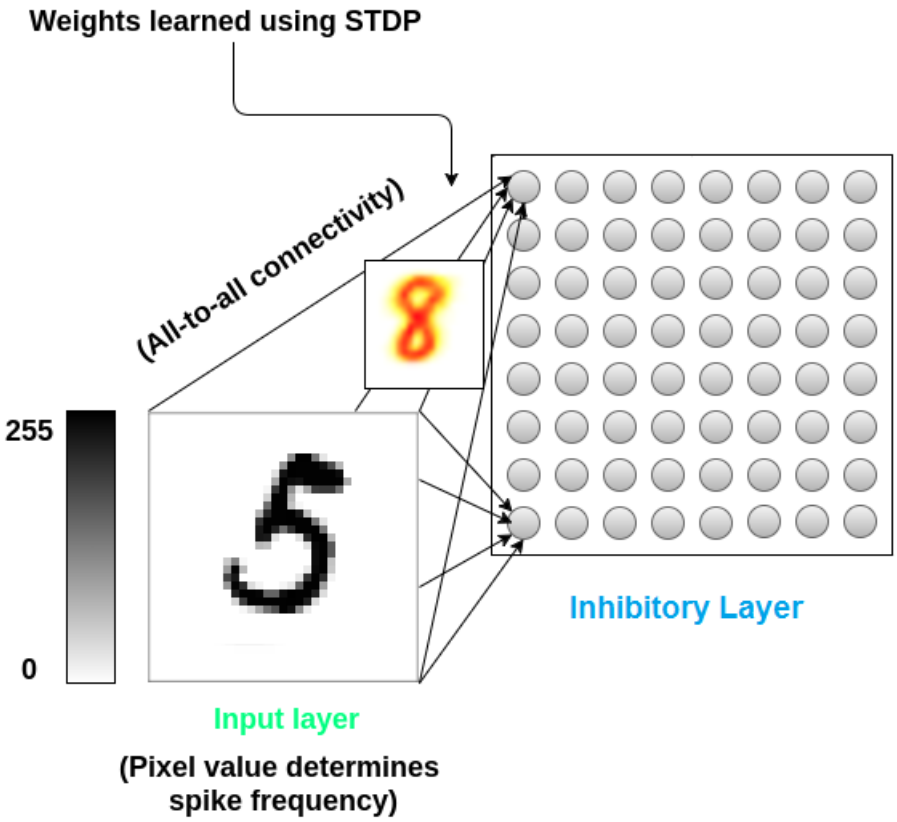


Figure 4.1: Two-layer SNN based on the architecture proposed in [40].

During the inhibition time t_{inhibit}, the membrane potential of the non-firing neurons will reset to their resting value V_{rest}. On the other hand, the neuron enters a refractory period whenever it generates an action potential. During the refractory period t_{ref}, the neuron is incapable of producing new spikes. Adopting an equivalent value of the refractory period as inhibition time, we can provide a situation in which all neurons in the output layer have an equal chance to compete for a new pattern.

4.2.3 Learning Rule

A general form of STDP describes the synaptic weight change equation as (Eqn. 4.4)

$$\Delta w = \sum_{t_{pre}} \sum_{t_{post}} W(t_{post} - t_{pre}) \tag{4.4}$$

Here W is a function of the difference in the spike times of the pre and postsynaptic neurons. To determine the synaptic weight change, we need to calculate the sum of W for all pre and postsynaptic spike times. A common form of the function W represented as (Eqn. 4.5):

$$W(\Delta t) = \begin{cases} A_{pre} e^{-\Delta t/\tau_{pre}} & \Delta t > 0 \\ A_{post} e^{\Delta t/\tau_{post}} & \Delta t < 0 \end{cases} \tag{4.5}$$

Figure 4.2 represents the schematic of the STDP function regarding the timing of pre and postsynaptic spike. The right side of the figure belongs to the time when the presynaptic spike occurs before the postsynaptic one and is analogous to long term potentiation (LTP). The reversed timing of the pre and postsynaptic neuron denoted

on the left side of the diagram, that lead to long term depression(LTD).

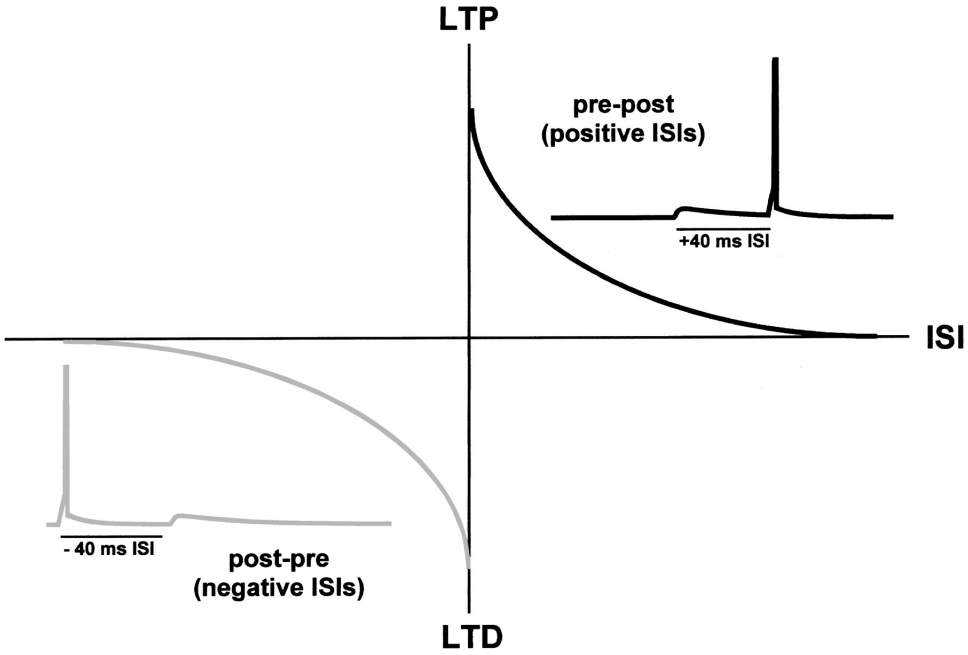


Figure 4.2: Schematic of the STDP based on equation (Eqn. 4.5) and (Eqn. 4.4).

However, using this equation does not seem to be computationally efficient since we need to sum it over each pair of spikes. On the other hand, for calculating the sum, the neuron needs to remember all its previous spike time.

Introducing two new variables, we can resolve the issue and get the same effect of (Eqn.4.4).

We represent two new variables a_{pre} and a_{post}, which stand for traces of pre- and post-synaptic activity. Theses variables described by the differential equations in (Eqn.4.6)

$$\begin{aligned} \tau_{pre}\frac{\mathrm{d}}{\mathrm{d}t}a_{pre} &= -a_{pre} \\ \tau_{post}\frac{\mathrm{d}}{\mathrm{d}t}a_{post} &= -a_{post} \end{aligned} \tag{4.6}$$

When a presynaptic spike happens, the weight modification rule expressed as

$$\begin{aligned} a_{pre} &\rightarrow a_{pre} + A_{pre} \\ w &\rightarrow w + a_{post} \end{aligned} \tag{4.7}$$

And whenever a postsynaptic spike occures we have:

$$\begin{aligned} a_{post} &\rightarrow a_{post} + A_{post} \\ w &\rightarrow w + a_{pre} \end{aligned} \tag{4.8}$$

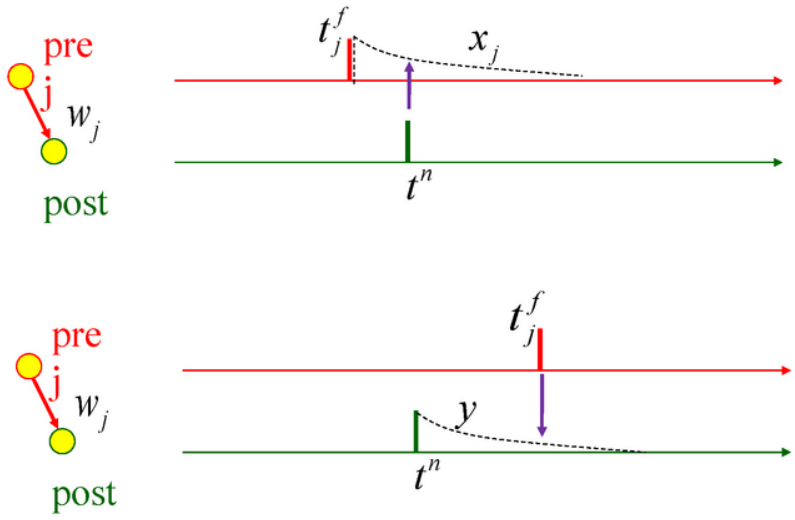


Figure 4.3: STDP weight change based on pre ans postsynaptic spike timing [43].

Figure 4.3 demonstrates the STDP rule for weight change based on the timing of pre and postsynaptic spike using the spike trace x_j.

Employing pre and postsynaptic variables, we have the formulation that only depends on differential equations and spike events.

To acknowledge that this formulation is equivalent to (Eqn.4.4), we need to check that the equations sum linearly and consider two different cases based on the timing of the pre and postsynaptic spikes. Drawing the diagram of equations in (Eqn.4.7) and (Eqn.4.8), we observe the exact curve as Figure 4.2.

From a biological perspective, a boundary condition of $w_{\min} < w_j < w_{\max}$ for synaptic weights should retain to keep the network in a stable situation.

To control the growth of the synaptic change and prevents them from becoming too large or negative, we can consider an online weight-dependent STDP rule for weight modification in which whenever a postsynaptic spike occurs we have:

$$\Delta w = \eta_{\text{post}}(a_{\text{pre}} - a_{\text{tar}})(w_{\max} - w)^{\mu} \tag{4.9}$$

Where η_{post} is the learning rate of the postsynaptic spike, a_{tar} is the target value of the presynaptic trace whenever a postsynaptic spike happens, and μ is the parameter that determines the weight dependence.

And the weight change for a presynaptic spike is:

$$\Delta w = -\eta_{\text{pre}} a_{\text{post}} w^{\mu} \tag{4.10}$$

Where η_{pre} is the learning rate for a presynaptic spike.

Since the firing rate of the postsynaptic neuron is quite low, we can limit the time

for weight modification to the moment when a postsynaptic neuron generates a spike. Applying this approach, we can reduce the computational cost to a great extent [13].

There are other types of STDP, like triplet rule [38] and symmetric rule, which are computationally more expensive for software simulation. Using a triplet STDP rule, we need to calculate the weight change of every single postsynaptic neuron for every presynaptic event.

4.2.4 Adaptive Threshold

One of the problems affiliated by selecting a fixed value for threshold appears when neurons with higher firing rates dominate the response to input patterns and prevent other neurons from participating in the competition. To exploit the full advantages of competitive learning, we have to provide the situation in which all the neurons have an equal chance to win the race for a new pattern. Having the same firing rate among neurons, we can encourage the competition among neurons and hence improve the performance of the network. An adaptive threshold value has been proposed in [47] to satisfy the same firing rate condition among neurons.

Applying the new adaptive method, we can describe the new threshold as $V_{\text{th}} + \theta$, in which the value of θ increases (by a predefined value) whenever the neuron generates a spike; otherwise, it will decay exponentially. We can describe the dynamic of θ as:

$$\tau_\theta \frac{d\theta}{dt} = -\theta \tag{4.11}$$

Where τ_θ is the time constant, which determines the steepness of the decay. The instant increase of the membrane potential causes the firing neuron to need more

input to produce a spike in the short term.

In the case when the firing rate of the excitatory neuron is less than the expected value (for instance, less than 5 spikes in 350 ms), we can increase the intensity of the input pattern until the issue resolved.

4.2.5 Training

The images presented to the network for training are from the MNIST dataset [29], which comprised of 60000 grayscale images of handwritten digit, each of the size 28×28 pixels. The intensity of each pixel in the image denote by a number between 0 to 255. To provide an acceptable input pattern for the network, we employ the Poissonson-distributed spike train in which the firing frequency of the input neuron is proportional to the intensity of the corresponding pixel in the image (see Figure 4.4).

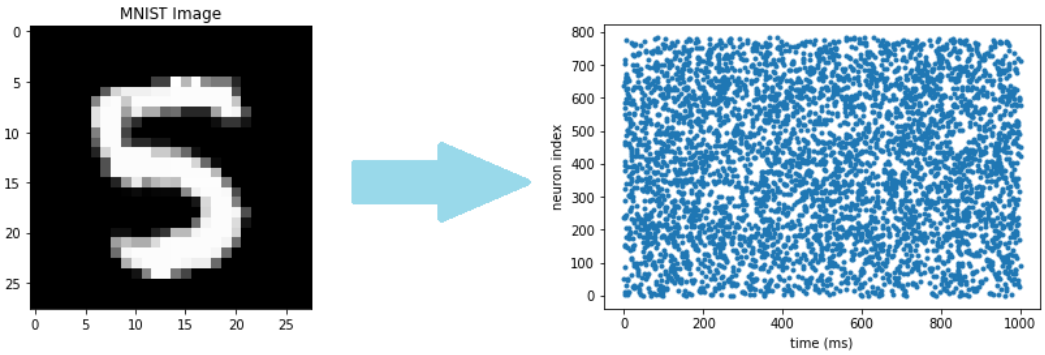


Figure 4.4: Encoding the input image to Poisson-distributed spike train.

For training the network, we present each input pattern for a duration of 350 ms and then reset the membrane potential of all neurons to their resting value before introducing the new input. We repeat this process three times for the whole 60000

MNIST training set and allow the output neurons to fully adapt their synaptic weights to the input pattern of the training set. After training, we label each of the neurons in the output layer with the digit for which it fires the most spike. To assess the performance of the system, we use another 10000 images of the test set, which we did not present to the network during the training process. During the test phase, we classify new input examples by taking the majority vote of the labels for the output neurons, which fire for a test data sample. Concerning the testing phase, we should use the fixed value of the synaptic weight (by putting the learning rate of the STDP equation equal to zero) and the threshold at the end of the training phase to classify the input images of the test set.

4.3 Results

We evaluated the performance of the network with two different numbers of output neurons. Using 400 neurons in the output layer, we obtained an accuracy of 91.35%. To enhance the performance of the system, we expanded the number of output neurons to 625 and obtained a test accuracy of the 92.7%. However, this improvement in performance comes at the price of computational cost as we increase the number of updatable parameters from 313600 to 490000.

Figure 4.5 show the raster diagram of the network with 400 output neurons. It is clear, over time, fewer neurons generate spikes in response to the input pattern, and they become selective to the specific pattern learned during training.

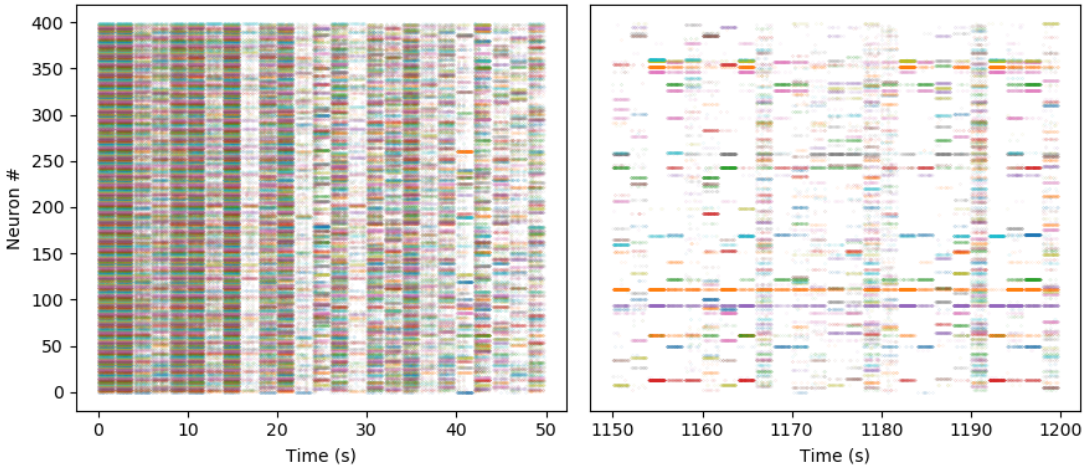


Figure 4.5: Raster diagram of SNN network with 400 output neurons.

Figure 4.6 illustrates the normalized firing rates of the output neurons in response to the input pattern after training on 40,000 images of the training set. As can be seen from the figure, most of the neurons are selective to one specific pattern they learned during the training process and generate fewer spike for other input patterns.

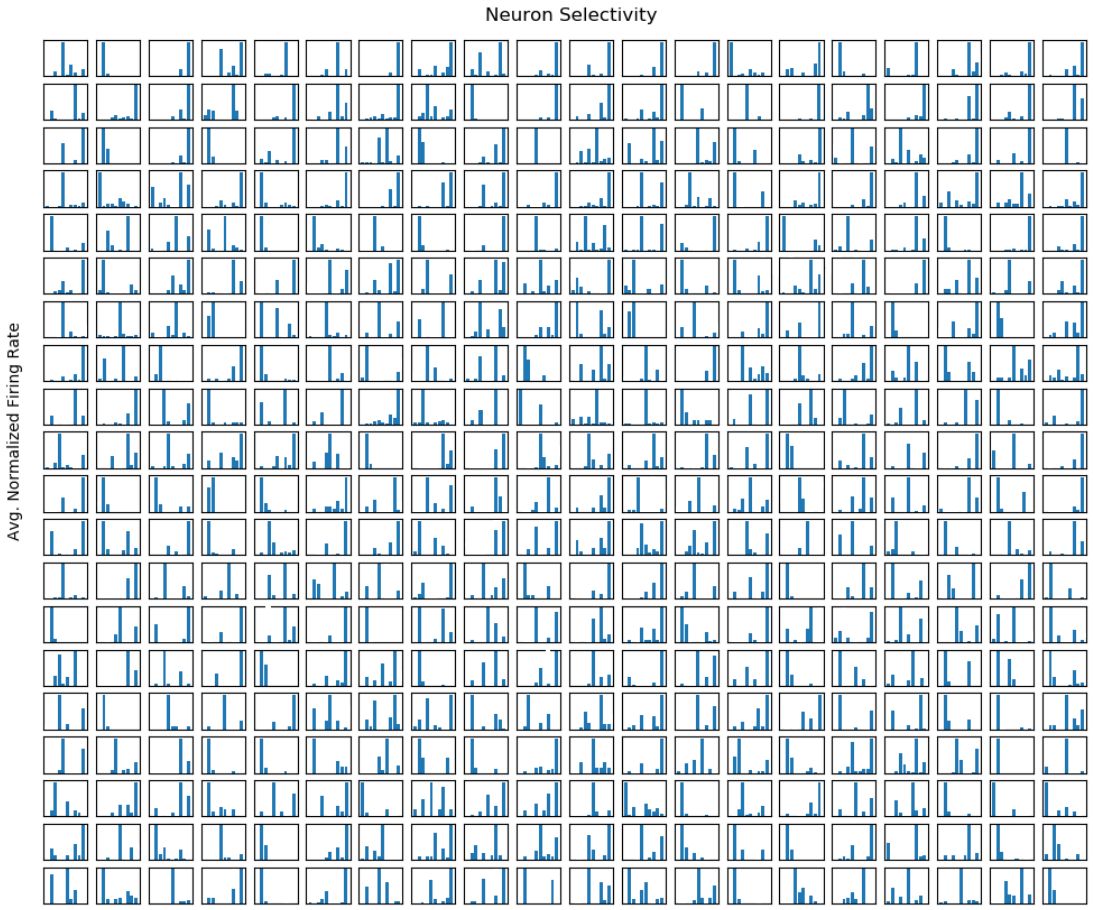


Figure 4.6: Selectivity of the neuron.

Rearranging the weight of the connections to each of the neurons in the output layer, we can construct the 2D receptive field of the network. Figures 4.7 and 4.8 show the receptive field of the respectively 625 and 400 output neurons.

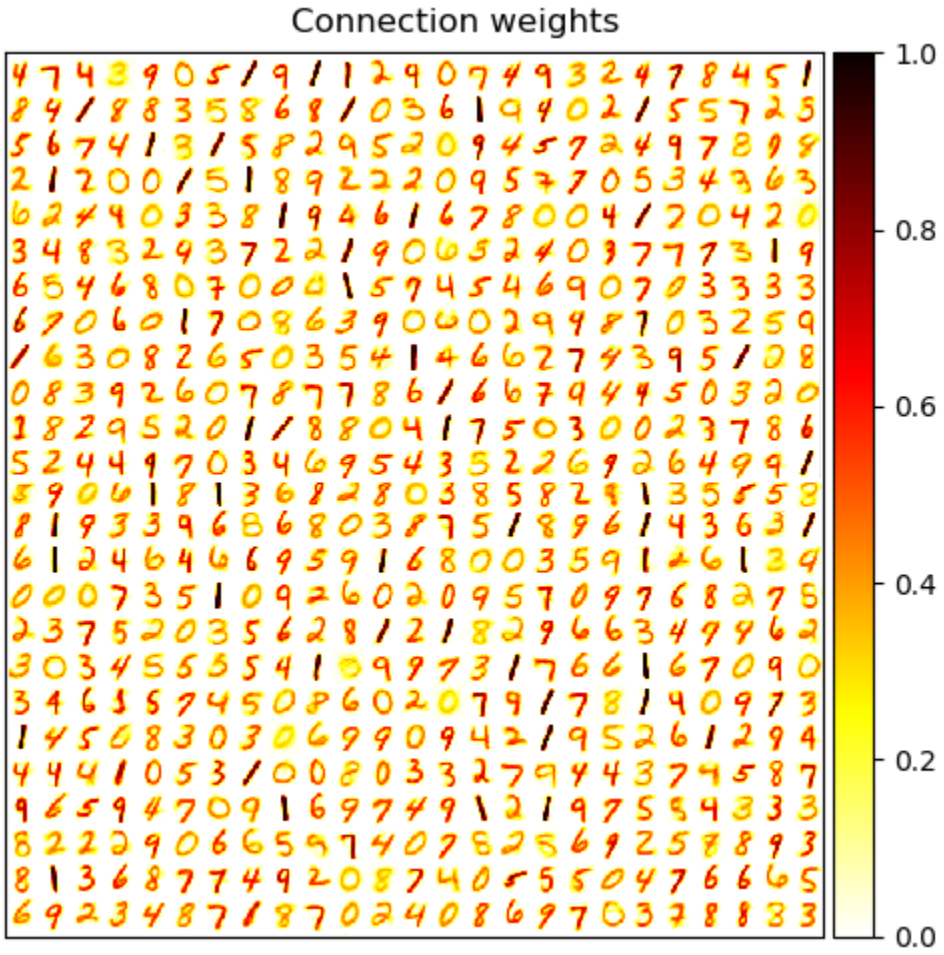


Figure 4.7: 2D receptive field of the network with 625 output neuron.

An essential factor for enhancing the performance of the system is to produce a differentiated receptive field, which implies that each neuron of the output layer learns a distinctive pattern of the training set. To achieve this goal, we used lateral inhibition and an adaptive threshold.

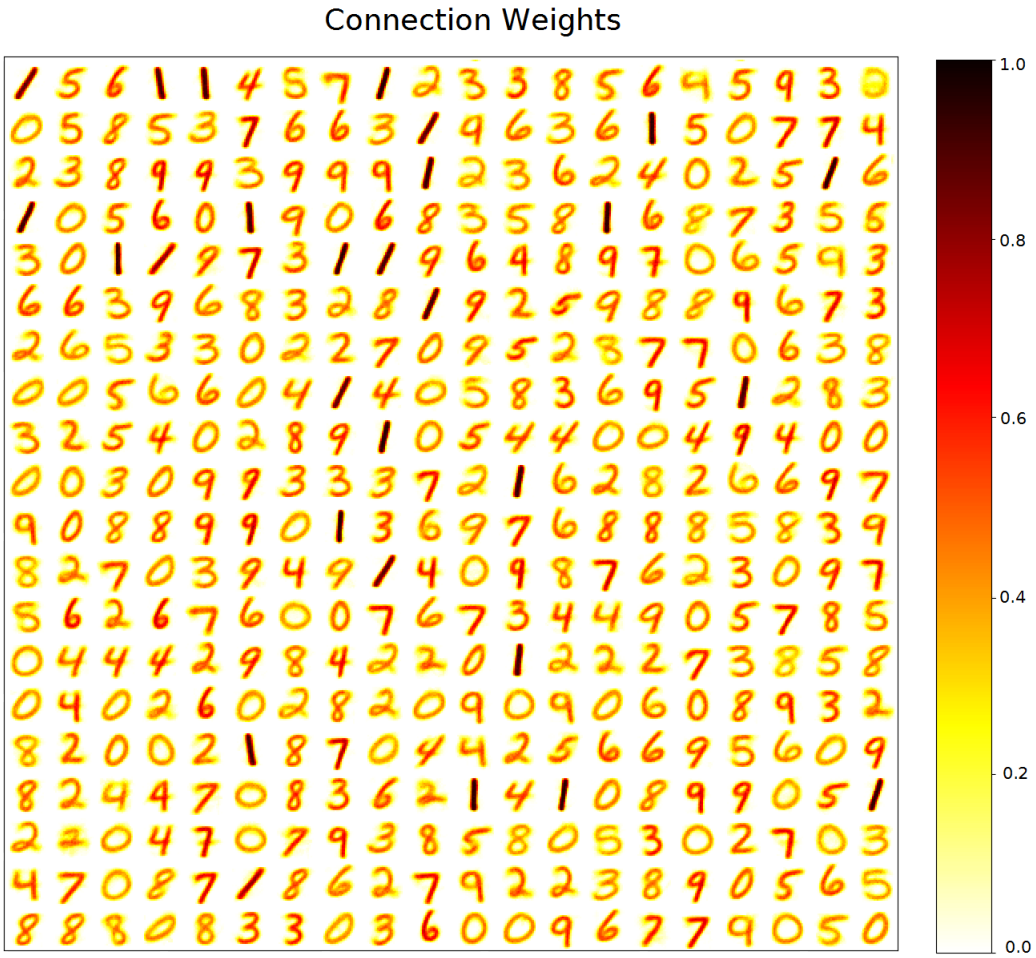


Figure 4.8: 2D receptive field of the SNN network with 400 output neuron.

Figure 4.9 represents the confusion matrix of the test results in which the high value on the identity diagonal indicates the correct classification and a high value on any other places represent the frequency of the incorrect classifying. As can be seen from the figure incorrect classifying of 4 as 9 presented with different color indicates the frequency of the faulty prediction. Incorrect classifying of 9 as 4 and 7 as 9 are other instances of frequent false recognition of the system.

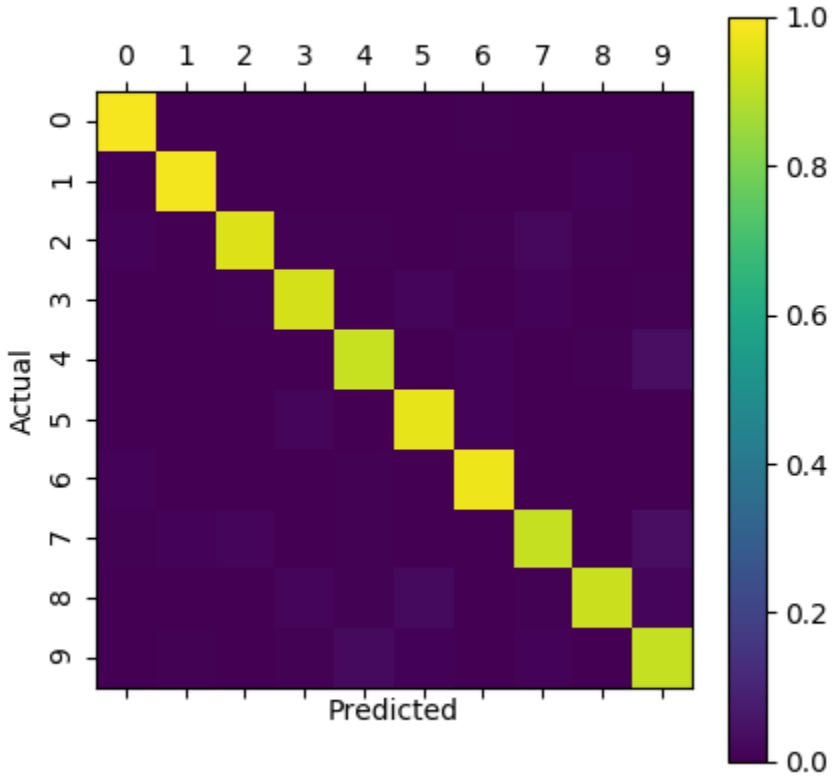


Figure 4.9: Confusion matrix of the testing results.

And finally, the table 4.1 is represents the list of parameters used to simulate the network.

Parameters	Default Value	Unit
V_{rest} (excitatory)	0	mV
V_{rest} (inhibitory)	0	mV
V_{reset} (excitatory)	0	mV
V_{reset} (inhibitory)	0	mV
V_{th}	0.6	mV
t_{ref}	10	ms
τ	100	ms
τ_{pre}	25	ms
μ	1	-
η_{pre}	1×10^{-4}	-
η_{post}	1×10^{-2}	-
w_{max}	1	-
w_{min}	1×10^{-4}	-
a_{tar}	1	-

Table 4.1: Parameters for simulation of the SNN.

4.4 Conclusion

The architecture of the network suggested here inspired by the SNN presented in [39]. The main difference between their system and the proposed work here is in the model of synapse and learning rule, which applied to adjust the wights. We use an exponential conductance-based (biologically plausible) synapse model with an online mode of power-law STDP to update synaptic weight, however, the network in [39], adopted a current-based synaptic model and simplified rectangular shape of STDP

for the weights adjusting. A different architecture for unsupervised digit recognition presented in [13] incorporates a layer of inhibitory neurons in the output layer to perform lateral inhibition. Although it is biologically acceptable to use inhibitory conductance, it takes a lot of effort to fine-tune the value of the refractory and inhibitory time constant to obtain the optimal result. The network here uses a simple but more efficient process for inhibition in which the winner neuron inhibits all the other neurons for a period t_{inhibit} and reset their membrane potential to their resting value. Applying this method, we can choose t_{ref} such that all the neurons have the same chance of firing after the refractory period and consequently improve the competition among neurons. The performance of the proposed network with 400 output neurons is comparable to the performance of a supervised fully connected neural network with the same number of neurons in the hidden unit by the test accuracy of 92.51%. A superior ANN presented in [9] obtains the accuracy of 99.7 on the MNIST dataset, though the network composed of over 12 million updatable parameters, which looks massive in comparison with 490000 parameters of our SNN with 625 output neurons. Table 4.2 illustrates a comparison between the performance of different methods on the MNIST dataset having the same number of neurons.

Method	**Learning Method**	**Biological Plausibility**	**#Neurons**	**Accuracy**
Dihel & Cook [13]	Unsupervised	✓	400	90.54%
Querlioz & Bichler [39]	Unsupervised	✗	400	93.75%
Fully-Connected ANN	Supervised	✗	400	92.51%
Proposed Method Here (400 Output Neurons)	Unsupervised	✓	400	91.35%

Table 4.2: Performance of different methods on the MNIST dataset.

As can be seen from the table 4.2, the network suggested here outperform the SNN proposed in [13] and obtained an accuracy near to a conventional fully-connected neural network comprised of 400 neurons in the hidden layer with the difference of 1.16%. However, we should take it to the consideration that the fully-connected network uses a supervised learning method for updating the weighted connection. The procedure implies that after each forward propagation, we should present the correct label of the input image to the network to adjust its wights. The method is similar to providing a feedback signal after each presentation of the input image. The learning method for the other three networks is unsupervised learning, which indicates that we should not present any additional information about the input to the system. The highest accuracy belongs to the network presented in [39], with a recognition rate of 93.75%. The main reason for the difference in the performance of the proposed method here and the SNN in [39] can address under the model of synaptic connection and learning rule for weight modification. It would be easier to fine-tune the parameters like t_{inh} and t_{ref} to obtain an optimal result using a current-based synaptic model by properties of linear summation instead of the exponential conductance-based model deployed here. However, considering a current-based synapse, we should compromise the biological plausibility of the synaptic weight change in the network.

Chapter 5

CONCLUSION

This thesis incorporates two different implementations of spiking neural networks for image classification of the handwritten digit of the MNIST dataset.

The first implementation is a computationally efficient SNN that can successfully classify differernt samples of the MNIST dataset. The network consists of a simplified spike response model similar to [22]. However, the SNN presented here employs a different form of STDP and does not perform any preprocessing of data to facilitate the learning process. To encourage competitive learning among neurons an adaptive threshold applied to guarantee the equal chance of winning the competition among neurons. The program for simulation of the network has written in python programming language.

The second SNN implements an unsupervised procedure for recognition of black and white handwritten digit of the MNIST dataset. The model incorporates some of the properties of the networks proposed in [39] and [13]. Here a leaky integrate and fire (LIF) neuron with conductance-based synaptic model used to describe the dynamic of the network. An online form of STDP with weight dependence rule used to modify

the synaptic weights. The network trained over 60,000 images of the MNIST dataset. After the training process, the performance of the system evaluated over 10,000 unseen pictures of the test set and obtained an accuracy of 92.7% using 625 neurons in the output layer. BRAIN2 used for simulation, and supplementary programs have written in the python programming language.

5.1 Future Works

A problem affiliated with the simulation of a big network in BRIAN is the slow training process, which requires several hours for one pass of the MNIST training set.(in our case 9 hours using intel core-i7 4770 CPU). One potential improvement is to reduce the number of time steps used per iteration of the training and test phases. To attain this goal, we should adjust the parameter of the equations which govern the neurons in the network to decrease the response time of the neuron to input data.

The actual ratio of 1 to 4 between inhibitory and excitatory neurons observed in mammalian neocortex, can offer essential information about the lateral inhibition of excitatory neurons [13]. A proper architecture can investigate the impact of this ratio on the overall performance of the system.

Computational complexity of the triplet rule of STDP forced us to consider other types of synaptic plasticity. However, evidence suggests the superior performance of the triplet law over different sorts of learning rules. Possessing powerful hardware, it makes more sense to use triplet rule for mimicking a biologically plausible learning procedure.

To take full advantage of spiking neural networks, a parallel computing approach re-

quired to speed up the computation and transmission of spikes between neurons. One promising field to attain this goal can be found within neuromorphic hardwares. A neuromorphic processor that scales to 8 million neurons introduced in [11], which is 10,000 times more efficient than conventional processor in terms of power consumption.

BIBLIOGRAPHY

[1] M Abeles, H Bergman, E Margalit, and E Vaadia. Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of neurophysiology*, 70(4):1629–1638, 1993.

[2] Laurent Badel, Sandrine Lefort, Romain Brette, Carl CH Petersen, Wulfram Gerstner, and Magnus JE Richardson. Dynamic iv curves are reliable predictors of naturalistic pyramidal-neuron voltage traces. *Journal of Neurophysiology*, 99(2):656–666, 2008.

[3] Lubica Benuskova and Nikola K Kasabov. *Computational neurogenetic modeling*. Springer Science & Business Media, 2010.

[4] Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[5] Sander M Bohte, Han La Poutré, and Joost N Kok. Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer rbf networks. *IEEE Transactions on neural networks*, 13(2):426–435, 2002.

[6] Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology*, 94(5):3637–3642, 2005.

[7] Steven M Chase and Eric D Young. Spike-timing codes enhance the representation of multiple simultaneous sound-localization cues in the inferior colliculus. *Journal of Neuroscience*, 26(15):3889–3898, 2006.

[8] François Christophe, Tommi Mikkonen, Vafa Andalibi, Kai Koskimies, and Teemu Laukkarinen. Pattern recognition with spiking neural networks: a simple training method. In *SPLST*, pages 296–308, 2015.

[9] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220, 2010.

[10] Wikimedia Commons. Complete neuron cell diagram, 2013.

[11] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[12] Peter Dayan and Laurence F Abbott. Theoretical neuroscience: computational and mathematical modeling of neural systems. 2001.

[13] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[14] W. Gerstner. Spike-response model. *Scholarpedia*, 3(12):1343, 2008. revision #91800.

[15] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

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

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

[18] Dan FM Goodman and Romain Brette. Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, 2:5, 2008.

[19] Donald Hebb. The organization of behavior. emphnew york, 1949.

[20] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

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

[22] Taras Iakymchuk, Alfredo Rosado-Muñoz, Juan F Guerrero-Martínez, Manuel Bataller-Mompeán, and Jose V Francés-Víllora. Simplified spiking neural network architecture and stdp learning algorithm applied to image classification. *EURASIP Journal on Image and Video Processing*, 2015(1):4, 2015.

[23] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[24] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[25] Nikola K Kasabov. *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Springer, 2019.

[26] Werner M Kistler and J Leo van Hemmen. Modeling synaptic plasticity in conjunction with the timing of pre-and postsynaptic action potentials. *Neural Computation*, 12(2):385–405, 2000.

[27] Jerzy Konorski. Conditioned reflexes and neuron organization. 1948.

[28] Peter E Latham, BJ Richmond, PG Nelson, and S Nirenberg. Intrinsic dynamics in neuronal networks. i. theory. *Journal of neurophysiology*, 83(2):808–827, 2000.

[29] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010.

[30] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[31] Warren S McCulloch. Walter pitts (1943)."a logical calculus of the ideas immanent in nervous activity". *Bulletin of mathematical biophysics*, 5(4):115–133.

[32] Sam McKennoch, Dingding Liu, and Linda G Bushnell. Fast modifications of the spikeprop algorithm. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3970–3977. IEEE, 2006.

[33] Boudjelal Meftah, Olivier Lézoray, Soni Chaturvedi, Aleefia A Khurshid, and Abdelkader Benyettou. Image processing with spiking neuron networks. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, pages 525–544. Springer, 2013.

[34] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[35] T Natschläger and B Ruf. Online clustering with spiking neurons using radial basis functions, chapter 4 in "neuromorphic systems: Engineering silicon from neurobiology"(hamilton & smith, eds), 1998.

[36] Daniel H O'Connor, Gayle M Wittenberg, and Samuel S-H Wang. Graded bidirectional synaptic plasticity is composed of switch-like unitary events. *Proceedings of the National Academy of Sciences*, 102(27):9679–9684, 2005.

[37] Hélene Paugam-Moisy and Sander M Bohte. Computing with spiking neuron networks. *Handbook of natural computing*, 1:1–47, 2012.

[38] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682, 2006.

[39] Damien Querlioz, Olivier Bichler, Philippe Dollfus, and Christian Gamrat. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology*, 12(3):288–295, 2013.

[40] Damien Querlioz, Olivier Bichler, and Christian Gamrat. Simulation of a memristor-based spiking neural network immune to device variations. In *The 2011 International Joint Conference on Neural Networks*, pages 1775–1781. IEEE, 2011.

[41] Joo-Heon Shin, David Smith, Waldemar Swiercz, Kevin Staley, J Terry Rickard, Javier Montero, Lukasz A Kurgan, and Krzysztof J Cios. Recognition of partially occluded and rotated images with a network of spiking neurons. *IEEE transactions on neural networks*, 21(11):1697–1709, 2010.

[42] J. Sjostrom and W. Gerstner. Spike-timing dependent plasticity. *Scholarpedia*, 5(2):1362, 2010. revision #184913.

[43] Haim Sompolinsky and I Kanter. Temporal association in asymmetric neural networks. *Physical review letters*, 57(22):2861, 1986.

[44] Beata Strack, Kimberle M Jacobs, and Krzysztof J Cios. Biological restraint on the izhikevich neuron model essential for seizure modeling. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 395–398. IEEE, 2013.

[45] Evangelos Stromatias and John S Marsland. Supervised learning in spiking neural networks with limited precision: Snn/lp. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[46] S Yamane, S Kaji, and K Kawano. What facial features activate face neurons in the inferotemporal cortex of the monkey? *Experimental brain research*, 73(1):209–214, 1988.

[47] Wei Zhang and David J Linden. The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4(11):885–900, 2003.

VITA AUCTORIS

NAME:	Amir Javid Talaei
PLACE OF BIRTH:	Tehran, Tehran, Iran
YEAR OF BIRTH:	1990
EDUCATION	Hamedan University of Technology, B.Sc., Computer Engineering (Hardware) Hamedan, Hamedan, Iran, 2014
	University of Windsor, M.Sc., Electrical and Computer Engineering Windsor, Ontario, Canada, 2020