UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
INSTITUTO DE INGENIERIA
MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

A comparative analysis of Gabor filters and biologically inspired learning rules for image classification implementing Spiking Neural Networks

THESIS IN ORDER TO OBTAIN THE DEGREE:

MASTER IN SCIENCE

AUTHOR:

CARLOS ALAN GARCIA CAMPOS

SUPERVISOR:

DR. RICARDO MORALES CARBAJAL
DR. HIROTSUGU OKUNO

Mexicali, Baja California, Mexico

December, 2024

Dedication

I dedicate this work to my mother, Ciran Campos Aispuro, for her love and unconditional support throughout this work's culmination. To my aunt, Esmeralda Campos Aispuro, for her financial help since my young days on elementary school up to my master's degree fulfillment. To my Steem supervisor Dr. Ricardo Morales Carbajal for his patience and support during the whole journey and help me to understand the wonderful world of studying the brain.

Acknowledge

This endeavor would not have been possible without the help of my advisor Dr. Ricardo Morales Carbajal who guided me through the full process of the thesis's realization and introduced me to the study of understanding who we are, the study of neuroscience. I am deeply indebted to the members of the Neuromorphic Systems Lab at the Osaka Institute of Technology for polishing the research idea and main goals during my internship in Japan. Especially thanks to Dr. Hirotsugu Okuno, for supervising my research progress and guiding me through the work culmination, to Dr. Akito Morita for his help with the implementation of the toolbox (Gabor filter) used through the methodology in this work, and special thanks to Eng. Shinnosuke Touda and Eng. Yuta Tanimura for discussing the particularities of what makes a spiking neural network shine in order for a deep understanding of the brain. Many thanks also to my friend Eng. Bernardo Ramirez, for discussing the concept of my research and helping me refine some concepts of my research that were not so clear for being presented. Finally, I want to acknowledge to the "Consejo, National de Humanidades, Ciencia y Tecnologia", for financial support of the whole work realization.

Abstract

Plastic changes on the synapse drive by spike-timing have been of great interest as the main learning rule for spiking neural networks. Spike-timing-based rules are built to model the behavior of a region on the brain related to experimental data in neuroscience, therefore can lead to differences in the moment of implementing the rule within a spiking network. This work compares the performance of a pair-wise and a triplet STDP with different spike interactions to clear an MNIST classification task. A bio-inspired preprocessing stage was implemented that consisted of a Gabor filter (as a model of the simple cells mechanism orientation selectivity) and an input normalization for a homogeneous brightness level of each image. The highlights of this work are 1) The consistent improvement of the model accuracy whenever they added the Gabor filter to the inputs; 2) The input normalization to prevent the overfitting of the model; 3) The Gabor filter helps to correct decoding of some images of the dataset on the evaluation test.

Contents

Acronyms

AI Artificial Intelligence. 5

ANN Artificial Neural Network. 1

DoG Difference of Gaussians. 3

LGN Lateral Geniculate Nucleus. 3, 8

LIF Leaky-integrate-and-fire. 14

LTD Long-term Depression. 14

LTP Long-term Potentiation. 14

MNIST Modified National Institute of Standards and Technology database. 2

PSCs Postsynaptic Current. 16

SNN Spiking Neural Network. 1

SOMs Self-Organizing Maps. 17

STDP Spike-timing dependent plasticity. 14

Sym NN Symmetric Nearest-neighbor. 31

WTA Winner-takes-All. 5

Chapter 1

Introduction

Artificial neural networks (ANN) have had massive success in solving various tasks, e.g., image classification, natural language processing, financial analysis, weather forecasting, and so on. Most of the actual ANN models have their roots in the perceptron (McCulloch and Pitts, 1943) as one of the first approaches to understanding how a biological brain processes information. The perceptron considers the signals transmitted from neuron to neuron to be encoded in a rate-based manner; so considering that each neuron can transmit information in discrete pulses of information called action potentials, the frequency of those action potentials (or spikes) leads to a change in the efficacy of the synapse (medium of action potential's transmission between neurons).

In 1998, the spike timing difference between presynaptic and postsynaptic neurons on a cultured hippocampal tissue was found to produce changes in the synapse efficacy (Bi and Poo, 1998). The spike-based changes in a synaptic connection are not compatible with traditional ANN models (where only the firing rate matters for modifying the synapse). Thus, a new type of architecture is needed to cover the behavior of how a biological neural network works within the brain.

Spike-based artificial neural networks or Spiking Neural Networks (SNN) are structures that aim to close the gap between neuroscience and artificial intelligence. SNN treats the neural networks as dynamical systems to model the actual functionality of a biological brain as realistically as possible. The main idea is to evaluate the changes in the membrane potential for each single neuron on the network. Whenever the neuron membrane potential surpasses a potential limit (threshold), an action potential is fired (spike). The information encoded on an SNN is passed to the next layer

depending on the status of their actual neuron potential (sparse coding), contrary to ANN, where each iteration cycle transmits information.

SNN has not increased in popularity for artificial intelligence models compared to ANN because of the training challenge it represents and its low performance when the network is scaled (Yamazaki et al., 2022). While ANN has obtained incredible results in a variety of tasks, the overall performance of SNN stays short of the success ANN models have received. A reason behind this is the lack of compatibility with the learning paradigms used for each network. SNN mainly uses temporal coding to modify the synapse efficacy (Vreeken, 2003). At the same time, ANN is expected to use the backpropagation algorithm to reduce the error between the network output and the actual output (labeled). Another approach to training an SNN model was to use a pre-trained ANN model (trained through backpropagation) and convert it afterward to its SNN equivalent; still, it doesn't make it up to the state-of-the-art ANN model's performance (Nunes et al., 2022).

Further studies are needed to understand the global scope of how spike-based learning rules can be implemented effectively to compete with algorithms like backpropagation; indeed, in (Deng et al., 2020), it has been shown that more than an SNN with an x-learning rule would overcome whatever ANN model with a backpropagation rule, each different structure works well for their respective workloads. While most of the reason why SNN falls apart from ANN models is because of the little understanding that we have of how our brain learns, the spike-based learning rules that have been published over the past 20 years can be compared to their performance for specific tasks such as image classification.

(Diehl and Cook, 2015) proposed a framework for comparing the performance of different learning rules for a spiking neural network by promoting competition between excitatory and inhibitory neurons. They train the network to classify the MNIST (Modified National Institute of Standards and Technology) dataset by unsupervised learning. Hence, the changes in the synaptic weights (efficiency factor) only depend on the relationship between the input and the excitatory layer.

Even though spike-based learning rules are enough to sustain good performance without an input preprocessing stage for some image-based classification tasks (Qu et al., 2019; Diehl and Cook, 2015; Querlioz et al., 2013), the network can be enhanced further if we look back again on how does the human body process the environment through the retinal circuit. In the brain, a class of neurons called simple cells are well known for encoding orientation features; this means that they respond only when a specific orientation of the input image is highlighted (Hubel and Wiesel,

1962). A mathematical model that can approximate the spatial receptive field response of simple cells is the Gabor filter.

A Gabor function can be modeled as the product between a sinusoidal and a Gaussian signal; while the parameters for the sinusoidal change in the filter a preferred direction is modified. This filter has demonstrated its power for feature detections in visual processing (Moreno et al., 2005) and how it can also capture features within a spiking network (Vemuru, 2020) and increase their performance accuracy for an image classification task (Ahokainen, 2024). The Gabor filter does not consider how the processing is performed on a cellular level (it gives an approximate resemblance of simple cells based on their output response), so alternative models opt for a difference of Gaussians (DoG) as their biological resemblance to the Lateral Geniculate Nucleus (LGN) cell's receptive field and their linear sum map a representation similar to the Gabor filter (Azzopardi and Petkov, 2012; Gong et al., 2018).

Besides the implementation of the Gabor filter for discriminating the feature orientations for each of our inputs, the heterogeneity between each input class also needs to be considered. The MNIST dataset is composed of 10 classes of handwritten digits (0 to 9) of 28 x 28 pixels. The difference in shape and pixel brightness from each image provokes that some images cannot be decoded for the model evaluation (as a rate coding for the input images to spike trains, the decoding would also be rate-based, so as the structure proposed by Diehl and Cook, 2015, the winner neuron rate can fall up to 0 Hz that leads to an unwanted case).

Alternatives to solve this problem was to repeat the simulation process for that image until it can be decoded (Diehl and Cook, 2015) what it has as a byproduct is an enormous use of computation resources. In this work is proposed an input normalization feature to constrain an equal usage of the firing rate from the input images to reduce the image brightness difference.

As the introduction summary, this work's general and specific objectives are listed below:

GENERAL OBJECTIVE

- Compare the performance between different spike-based learning rules on the MNIST image classification task for different biological feature configurations on small network architectures

SPECIFIC OBJECTIVES

1. Increase the model's accuracy over 85% with the Gabor filter and input normalization within a 100 excitatory neuronal layer

2. Evaluate the Gabor filter impact for the images that are not possible to be decoded during the evaluation phase

3. Evaluate the input normalization impact for the images that are not possible to be decoded during the evaluation phase

4. Evaluate the Gabor filter and input normalization impact on the arrangement of weights after the training phase

5. Compare the network accuracy for the pairSTDP All-to-all, pairSTDP Nearest-neighbor, and Triplet STDP

6. Compare the network accuracy for Gabor filter and Input normalization indistinct of the learning rule

Chapter 2

Background

The upcoming wave of AI-based research and technological development has brought us an alternative to solve complex problems (e.g., the image classification task, robotics vision) in a systematic way using the human brain as our inspiration. Even though we have been using artificial neural network (ANN) architecture for the past ten years to solve classification problems and predict an outcome based on prior data, we still don't fully understand how the brain processes information. (e.g., from the retina, through the ganglion cells, up to the primary visual cortex (V1)).

A novel computational neural network named Spiking Neural Networks (SNN) searches to close the gap between computer sciences and neuroscience by building networks that mimic the actual way (and behavior) that neurons process information in a biological network. To understand how these networks can be modeled from biology, it is important to explain the anatomy and morphology of the neuron, mainly the importance of action potentials as the main signal to change the efficacy of the synapses from a presynaptic to a postsynaptic neuron.

The main focus of this work would be a study of competitive learning in biological neural networks with the help of the Winner-takes-all (WTA) network. A WTA architecture is a type of self-organizing map that plays a vital role at V1 in vision classification for competitive learning (Kaski and Kohonen, 1994). Excitatory neurons can be controlled with a layer of inhibitory neurons that regulate their firing rate through a recurrent connection. The main idea of a WTA circuit is to make use of the excitatory-inhibitory circuit dynamics to "*organize*" the input (mapping) into a visual representation. The visualization map would work as a guidance decoder to choose the

actual output of our model. Their biological similarity makes it a perfect object of study within a spiking neural network architecture.

To properly understand how an SNN works and its main difference from an ANN architecture, it is vital to understand some of the theories of how biological neurons propagate information and how the arrangement of their behavior ends with learning. This chapter starts by explaining the basic concepts of the anatomy of a neuron and their method of propagating information. It introduces how *Spiking Neural Networks* are modeled to build a network. Some basic concepts for the most famous temporal learning paradigm within spiking neural networks, "*Spike-timing dependent plasticity*" where they consider the exact spike timing between neurons to change the synaptic efficacy and then tend to the learning of the system. This chapter culminates with the explanation of the *Winner-Take-All* circuit and some learning considerations needed to work appropriately within an SNN architecture.

2.1 Biology of a neuron

The anatomy of a neuron is complex and difficult to categorize. It can change depending on which part of the brain it is located, its function, and its interactions with other neurons. For understanding the computations of a neuron we can take the example of a motor neuron from the spinal cord (Kandel et al., 2012).

As shown in figure 2.1, motor neurons consist of a cell body or *soma* that regulates the metabolic processes of the cell. At the surrounds of the soma, tree-like structures called *dendrites* are branched, whose main purpose is to receive the incoming signals from other neurons. From the same cell body, a more prominent branch can be observed compared to the rest of the dendrites. This branch is named *axon*. Axon branches are those in charge of propagating the *action potentials*; electric pulses that produce a liberation of neurotransmitters once they reach the communication zone at the end of the axon called *synaptic terminals*.

One of the reasons that makes the study of the brain difficult is its homogeneity. In the brain, a great variety of neurons exist, depending on their morphology, functionality, size, etc. To understand the basic structure of what a neuron could be is easier to think about motor neurons. Motors neurons are multipolar cells characterized by cells that process information from many dendrites within a single axon (Kandel et al., 2012). An easy example of their morphology is pictured in figure 2.1.

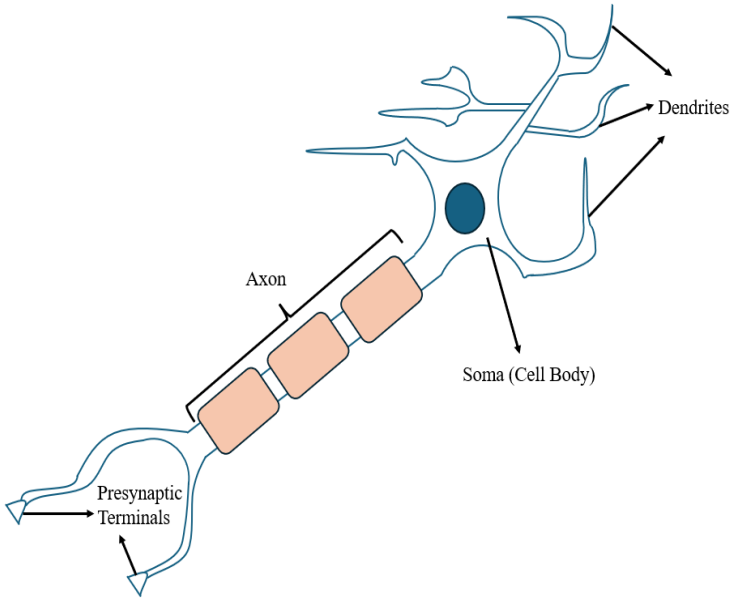


Figure 2.1: Dramatization of a common motor neuron from the spine cord pointing out the main zones for an action potential transmission and generation. The tree-like structure (dendrites) capture the incoming signals through a synaptic exchange of neurotransmitters to alter the neuronal membrane conductance. The action potential (generated at the zone between the soma and axon -usually called axon hillock) regenerates its loss of potential through the passage of the axon (an effect produced by the myelin sheath -the rectangular sections that cover the axon) until it reaches the presynaptic terminals for neurotransmitter liberation.

The neuron that transmits the action potential is called the *presynaptic cell*. Meanwhile, the cell that receives the neurotransmitters to increase the flow of ions to the neuron is regarded as *postsynaptic neuron*. The neurotransmitters released by the presynaptic neuron connect to the ion channels from the postsynaptic neuron to let ions flow inside the postsynaptic neuron. This would lead to a change in the membrane conductance at the postsynaptic neuron, making the neuron more likely to fire an action potential.

An action potential has a shape like in figure 2.2. The membrane potential of the neuron increases until it reaches a *potential threshold* value, which generates, in consequence, an abrupt increment of the membrane potential. Once it reaches the maximum potential value (spike), the membrane decays to its *resting potential*. Usually, action potentials are not generated one after another, so neurons have a cooldown time called *refractory period* where the neurons would find it more difficult to change the membrane potential even with the same amount of energy passed

through the presynaptic cell.

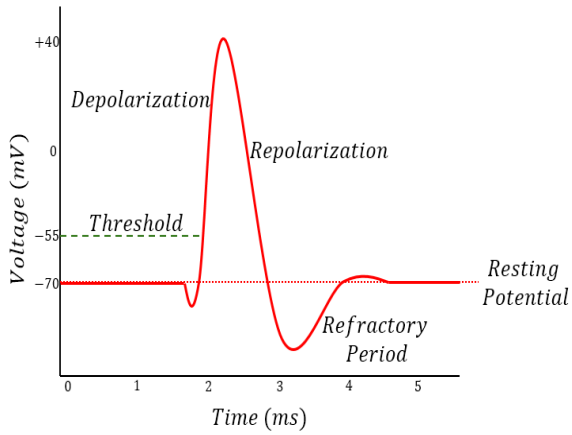


Figure 2.2: Waveform of a motor neuron action potential. At normal conditions, the membrane potential keeps on the resting potential until an input triggers the neuron to go up to the voltage threshold. The neuron depolarizes, reaching a "spike" peak (usually $40mV$), then the membrane potential decays below the resting potential (repolarization) for their refractory period. After this, the membrane potential returns again to its resting potential.

2.2 Visual Pathway

Before entering in the biological modeling of a neuron, an explanation of the biological tools as well as definitions that would be used throughout the whole document is needed. In a human brain, the light seen through our eyes is captured within rods and cones photoreceptor cells and converted into electrical signals that are passed up to the retinal ganglion cells through bipolar cells. Ganglion cells are the output cells from the retina and project a sequence of action potentials to a small structure in the thalamus called lateral geniculate nucleus (LGN).

An important feature of ganglion cells are the ON and OFF responses to light stimuli. Some ganglion cells are tuned to fire only when light is presented as a stimulus, whereas another group of ganglion cells fire when the light source is diminished. A spatial region can be mapped based on the light stimuli response of ganglion cells within the visual field called *receptive field*. Retinal ganglion and LGN cells have similar receptive fields where they respond better to circular spots surrounded by either light or dark regions (center-surround receptive field). LGN cell's receptive field contrasts with the receptive field of simple cells in V1 (which receive their input directly from LGN cells) as

they respond best to elongated light or dark bars surrounded by dark or light regions respectively. As we can rotate the bar, we can obtain information regarding the orientation of the object. The orientation selectivity feature of simple cells can be abstracted up to the MNIST dataset as would be further explained in chapter 3.

2.2.1 Retinal Structure

There is no exaggeration to say that the eye is the window to our world. The eye is the main organ in charge of projecting the visual scene through the retina's photoreceptors so our nerve cells can interpret it. In figure 2.3 it can be seen a simple diagram of the eye anatomy. The light that enters to the eye is refracted through the cornea and focused afterward by the lens until it reaches its focal point named *fovea*, so, the photoreceptors have direct access to the stimulus. Not all the light is captured by the photoreceptors in the eye, the remnants are also captured and diffused by the pigment epithelium as it also surrounds the photoreceptors.

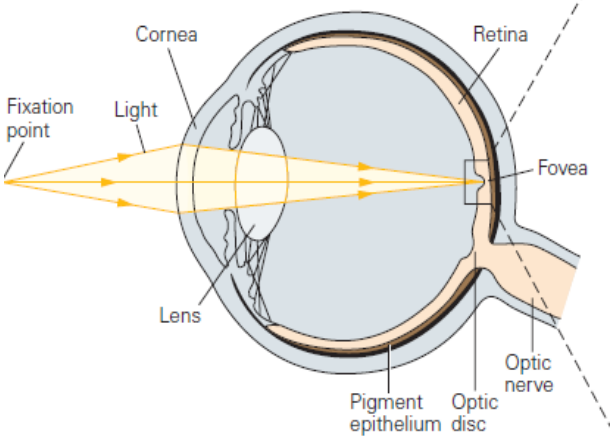


Figure 2.3: Anatomy of the eye seeing a fixed point in the space. The eye is refracted and concentrated through the first layers of the eye (cornea and lens). The light stimuli are concentrated on the fovea and passed by to the retina for coding the image into action potentials. The optic nerve is where the retinal ganglion axons project their action potential to the brain as they pass through the optic disc, a small blind spot of the eye. (Image recuperated from Kandel et al., 2012)

The retina has a layered structure that starts from the photoreceptors up to the ganglion cells whose axons project their action potential to the brain (figure 2.4). There exist two principal types of photoreceptors called rods and cones. Each photoreceptor differs in function, such as rods absorb

better photons under dim illumination, whereas cones are less sensitive to light but they respond to a variety of wavelengths, being in charge of color information. The rods and cones produce an electric signal that is passed through the bipolar cell up to the retinal ganglion cells. Before arriving to the ganglion cells, the signal in the bipolar cells is hyperpolarized either by a source or in the absence of light (depending on the type of bipolar cell). Within the retinal layers, two types of cells synapse horizontally within photoreceptors and ganglion cells (horizontal and amacrine cells) for grouping into subunits based on their functionality. At the end of the retina layer, the ganglion cells produce the sequence of action potential to be carried to the primary visual cortex.

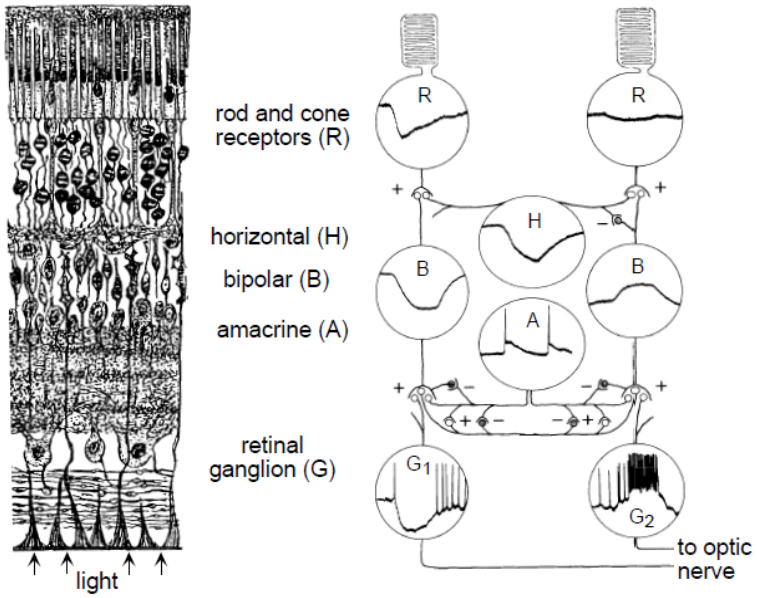


Figure 2.4: Anatomical diagram of the circuitry of a retina. Rods and cones capture the photons and produce an electric signal that is hyperpolarized within the bipolar cell. The retinal ganglion cells generated an action potential sequence that matched the input obtained from the bipolar cells. (Image recuperated from Dayan and Abbott, 2001)

2.2.2 Ganglion and LGN cells receptive field

After leaving the retina through the optic nerve, the response of ganglion cells depends on the variation of the light stimulus. This is called the ON-OFF response of ganglion cells, where being labeled either ON or OFF is restricted to an increment in a firing rate for light stimuli (ON response) or for the absence of light (OFF response). An easy way to visualize the boundaries of the ON-

OFF responses from the ganglion cell within the visual field can be done through a map called the receptive field. A receptive field captures the range within the visual field where a cell responds the most. For example, a ganglion cell's receptive field is characterized by being divided into a *center* and *surround* regions with a clear contrast response between them (figure 2.5). Ganglion cells synapse with LGN cells before arriving at the primary visual cortex, which is mainly the reason why both ganglion and LGN cell's receptive fields are similar (center-surround).

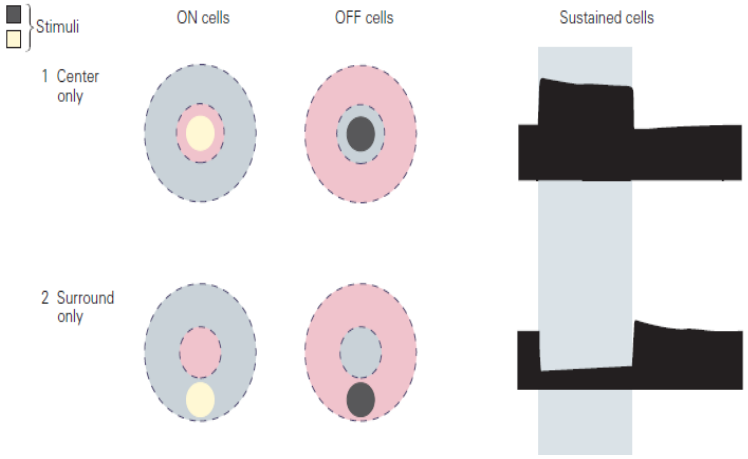


Figure 2.5: Center-surround receptive field of ganglion cells for different ON-OFF stimulus patterns. The plot of sustained cells captures the increment in firing rate based on the stimulus presented within each case on the ganglion cell's receptive field. Light stimulus (yellow) within an ON area (pink) produces the highest firing rate as it is in the other case around for a dark stimulus (black) within an OFF area (grey). (Image recuperated from Kandel et al., 2012)

In figure 2.6 we can visualize the visual circuitry starting from the retinal layered structure up to the primary visual cortex (V1). The ganglion cell's axon extends until reaching the LGN. There's a section when the ganglion cell's axon from both eyes crosses over (optic chiasm) to let the LGN distinguish between left and right, so the LGN receives a connection from both eyes ganglion cells. LGN cells then relay the retinal input to the primary visual cortex, where it is processed individually.

2.2.3 Simple Cells receptive field

Once the connections received from the LGN cells arrive at V1, some cells would experience changes in the shape of their receptive field. Cells in V1 are known for responding more to stimuli of the

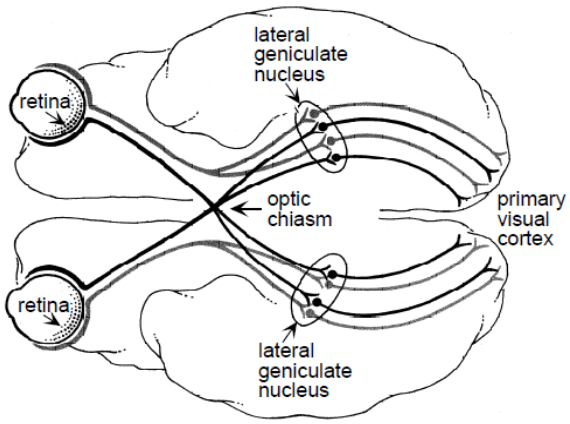
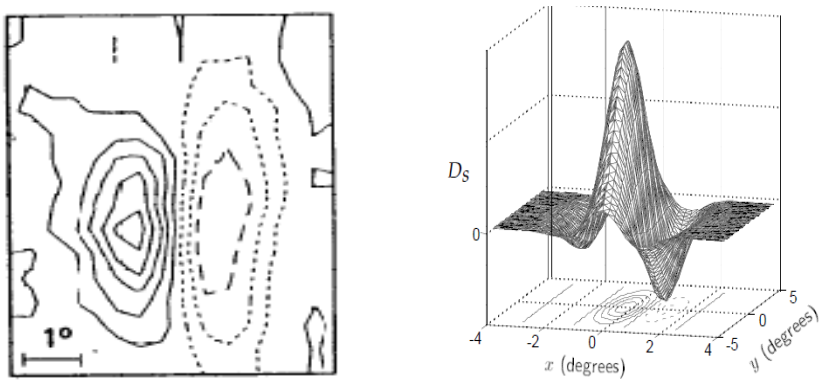


Figure 2.6: Visual processing pathway. Retinal ganglion cells cross over the optic chiasm, in this way, the lateral geniculate nucleus receives input from both eye's hemispheres to represent correctly each left and right side from both eyes. (Image recuperated from Dayan and Abbott, 2001

movement of a bar, edges, and elongated light or dark objects in contraposition to the center-surround receptive field held by the retinal ganglion and LGN cells. There exist two principal classes of cells in V1 if their receptive field can (*simple cells*) or cannot (*complex cells*) be mapped with defined excitatory and inhibitory boundaries (the ON-OFF regions can be clearly seen within its receptive field). So, abstracting the feature of simple cells for ON-OFF defined regions makes it possible to obtain the orientation of a visual scene. The receptive field of a simple cell can be seen in figure 2.7. The ON-OFF regions can be abstracted either on a 2D or 3D spatial representation. The orientation selectivity of simple cells is an important tool that can be applied within the digital image processing field. In chapter 3 will further be explain in detail the importance of simple cell's receptive field as a tool for preprocessing images.

2.3 Spiking Neural Networks

AI-based research has been having a huge impact since the last decade. Important improvements in the models and the manufacturing of more powerful processors gave birth to models such as transformers (Lin et al., 2022), Convolutional Neural Networks (Li et al., 2022), and Recurrent Neural Networks (Grossberg, 2013). Still, those models are part of the perceptron (McCulloch and Pitts, 1943), enclosed in the classical artificial neural network category.



(a) 2D Receptive Field (b) 3D Receptive Field

Figure 2.7: Spatial receptive field approximation of a simple cell. A) is a topographic representation with clear delimitation within the ON-OFF regions. B) the excitatory-inhibitory boundaries of the receptive field are mapped within a high and low hill on a 3D space. (Image recuperated from Dayan and Abbott, 2001)

A Spiking Neural Network (SNN) is an architecture that mimics more closely how the brain processes information. Unlike their predecessors (ANN-based), their information is coded sparsely; this means that all neurons do not contribute to the computations per cycle. A network in SNN architectures can be treated as a dynamic system, where the neuron information contribution is regulated at the time when they fire an action potential (spike). So, only the neurons that fire a spike within the presentation time of the stimuli (input) affect the overall behavior of the network and contribute to their information processing.

There are a variety of different neuron models to simulate an SNN with a high biological plausibility (Izhikevich, 2003; Hodgkin and Huxley, 1952). Still, in this work, we would focus on keeping the computational complexity as low as possible while preserving the biological details indispensable for the SNN network operation. The "Leaky-integrate-and-fire" neuron model captures the basic behavior of a neuron, where we keep track of the membrane potential of each neuron on the network. Once a neuron reaches the voltage threshold, a spike is fired, and the membrane potential is reset. The communication channels between neurons in the network work through the model of neurotransmission release, in other words, through a synapse model. While their models usually vary depending on the ion channel and time lapse it takes to release the neurotransmitters, is possible to greatly simplify them as a single decaying exponential function (Gerstner et al., 2014). The inclusion of the synapse model in a spiking network is the main link between the neuron model

and the learning rule. SNN learning rules depend on the temporal information given between two neurons, so how the spike transmission would behave is mainly controlled through the synapse model.

The learning paradigm is another major feature that divides classical ANN architectures from SNN. Action potentials usually are expressed as a delta function in the exact timing it is evoked ($\delta(t_{spike})$) for an SNN architecture, so backpropagation-based rules are not possible for their implementation since $\delta(t)$ is not a differentiable function. SNN then opts to rely on rules observed experimentally in a biological network. One of the most famous procedures to tune the synapse for learning uses the spike-timing correlation between synapses. The Spike-timing-dependent plasticity (STDP) is a Hebbian-type learning rule that considers the difference between pre and postsynaptic spike times to increase (Long-term potentiation -LTP-) or decrease (Long-term depression -LTD-) the efficacy on the synaptic connection (Bi and Poo, 1998). The pairwise STDP has been found to have the feature of stabilizing a network through the regulation of their firing rates for competitive learning networks (Dayan and Abbott, 2001; Song et al., 2000; Gerstner et al., 2014) requiring fewer hyperparameters to fine-tune for keeping the network in their optimal conditions. Further details of its working principles will be explained in section 2.3.3.

2.3.1 Neuron Model

This work used a conductance-based Leaky-integrate-and-fire (LIF) neuron model as described on (Gerstner et al., 2014) for its computations. The LIF model is limited regarding its biological plausibility with others more profitable as Hodgkin-Huxley or Izhikevich models (Yamazaki et al., 2022), but it manages to capture the basic behavior of action potential generation requiring fewer operations to compute. From figure 2.8 it is possible to describe the membrane potential of a neuron of the LIF model as

$$C_m \frac{dV}{dt} = -g_L(V(t) - V_{rest}) + I_{syn} + I_{ext} \tag{2.1}$$

Where V_{rest} is the resting membrane potential of the neuron; C_m is the membrane capacitance; g_L describes the leaky conductance from the neuron's membrane; I_{syn} is the presynaptic current that drives the neuron (see section 2.2.2); and I_{ext} acts as an external current from the neuron's environment (this work does not make explicit usage of I_{ext} but it was added as a description of the

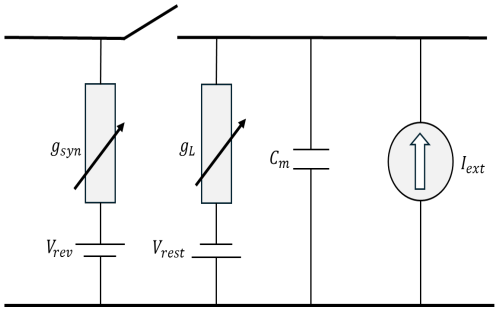


Figure 2.8: Membrane potential of a neuron driven by a presynaptic current I_{syn}. The current I_{syn} was labeled as the current passing between the synaptic conductance g_{syn} and the equilibrium (or reversal) potential V_{rev}. The membrane capacitance C_m is charged by I_{syn} and I_{ext} while a portion of the potential is "leak" through g_L.

whole model). The neuron membrane would change its potential driven by the current I_{syn} until it reaches a voltage threshold V_{thr}. Once the neuron's membrane potential surpasses the threshold, a spike is fired, and its membrane potential is reset to its resting value V_{rest} as the conditions described on (2.2).

$$V(t) = \begin{cases} V_{rest} & \text{if} \quad V(t) \geq V_{thr} \\ V(t) & \text{otherwise} \end{cases} \tag{2.2}$$

Even if the firing rate of the neuron is driven by the amount of current injected into it, in biological systems, neurons do not fire spikes consequently because of the refractory period. Once a neuron fires a spike, it enters the refractory period t_{ref}, where it is not possible to generate another spike after the period of time has passed. Usually, t_{ref} lasts around a few milliseconds after the neuron spikes (Dayan and Abbott, 2001).

2.3.2 Synapse Model

As the neuron's model, there exist different ways to set how a presynaptic neuron transfers a spike train to a postsynaptic neuron, which is expressed as the efficacy between the connection of the two neurons. For an excitatory-inhibitory balanced network, we can set different expressions for excitatory and inhibitory synapses regarding their Ion channels model (Gerstner et al., 2014). In this work is opt for the use of a single-exponential model where we can adapt for each neuron type with a simple modification of the parameters, such as done in (Diehl and Cook, 2015; Song et al.,

2000). The synaptic input from the presynaptic to the postsynaptic neuron is defined as

$$I_{syn} = g_{syn}(V_{rev} - V(t)) \tag{2.3}$$

Where g_{syn} has the following behavior

$$\tau \frac{dg_{syn}}{dt} = -g_{syn} \tag{2.4}$$

The amount of synaptic current I_{syn} for a presynaptic neuron is set to be the difference between the synaptic reversal (or equilibrium) potential V_{rev} and the membrane potential of the neuron i at time t modulated by the conductance of the synapse g_{syn}. The difference between the reversal and the membrane potential is regarded as the postsynaptic current (PSCs). A synapse would be called either excitatory or inhibitory depending on which direction flows the PSC (positive PSC for excitatory synapses and negative PSC for inhibitory). For equations (2.3) and (2.4), is possible to define a synapse either excitatory or inhibitory changing τ and V_{rev} to work for each respective condition. The parameters chosen for simulating the neuron and synapse behavior are shown in table B.1 from the appendix B.

2.3.3 Learning

Training and learning paradigms are some of the features that change the most compared to ANNs architectures. SNN relies on experimental discoveries from neuroscience to formulate a proper definition to modify the synapse efficacy and tend to the system's learning. The main learning principle in a biological organism was formulated by Donald Hebb in 1949 as

> *Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. ...When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cell firing B, is increased.* (Hebb, 1949)

This learning behavior is usually shortened as "*Neurons that fire together wire together*" (Löwel and Singer, 1992), and all the learning types that depend on the persistent repetition for modulating the efficiencies of the connectivity between two neurons were enclosed as Hebbian learning.

Hebbian learning can be called to those whose increment and decrement of the efficacy is mod-

ulated by the difference between the timing of presynaptic and postsynaptic neurons (Bi and Poo, 1998). The Spike-timing-dependent plasticity (STDP) has a great regulation power for competitive learning networks (Song et al., 2000). From (Sjöström and Gerstner, 2010) we can propose two spike trains from a presynaptic and a postsynaptic neuron with timing as $t_i^{s_{post}}$ and $t_j^{s_{pre}}$. So s_{pre} and s_{post} are evaluated as the number of spikes produced for each neuron up to N_i and N_j. A synaptic increment in the synapse can be overall expressed as in equation (2.5). See table B.2 in Appendix B for the parameters chosen to run the simulation within the reported results.

$$\Delta w_{ij} = \sum_{s_{pre}=1}^{N_i} \sum_{s_{post}=1}^{N_j} W(t_i^{s_{post}} - t_j^{s_{pre}}) \tag{2.5}$$

Where Δw_{ij} describes a synaptic efficiency increment for a post-pre spike correlation (the decrement can be also described if the spike train relationship is flipped). The function $W(t)$ describes the learning window (see figure 2.9). A common choice for $W(x)$ used for competitive networks (Song et al., 2000)

$$W(x) = \begin{cases} A_+ e^{-x/\tau_+} & \text{for} \quad x > 0 \\ A_- e^{x/\tau_-} & \text{for} \quad x < 0 \end{cases} \tag{2.6}$$

We are able to see how much it affects the STDP to a particular connection by plotting their learning window as shown in figure 2.9. Each blue dot represents either an increment or decrement of the synaptic weight at the time difference between spike trains during the network simulation.

2.4 Self-Organizing Maps

In unsupervised learning models, the Self-Organizing Maps (SOMs) is an artificial neural network that can generate a representation ("Map") and regulate their response based on the input data (Miljkovic, 2017). SOM techniques have strong motivation from brain mapping techniques (e.g. functional magnetic resonance imaging, positron emission tomography) for creating a visualization of how a specific region of the brain (or the brain network as a whole) reacts to known stimuli (Toga and Mazziotta, 2002).

We will discuss a SOM structure called Winner-takes-all (WTA), well studied in biological visual systems (Yi and Arisaka, 2020; Zhaoping, 2005) for promoting competition between neurons

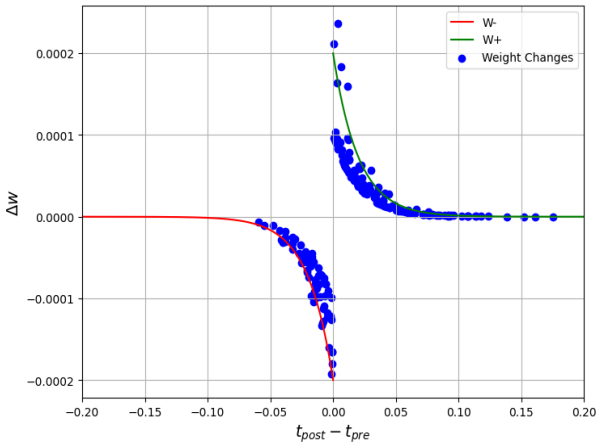


Figure 2.9: Weight changes represent the difference between the spike timing of pre and postsynaptic neurons. The area under the green curve is the increment of the efficacy while on the red curve is the decrement that has experimented with the connection during a total simulation time T.

usually through a balanced excitatory-inhibitory neural connectivity (Sanchez-Garcia et al., 2023). The main idea of a WTA circuit is to let the excitatory neurons gradually increase the firing rate based on the input being fed. A 2nd layer, at the same time, would control their firing rate inhibiting the response of the neighbor excitatory neurons. The dynamic balance between excitatory - and inhibitory neurons would tune the network to a group of neurons to respond to specific inputs. The neuron (or group of neurons) with the highest response is regarded as the "winner." The input mapped to that neuron can be taken as the actual output of our circuit.

It is a difficult task to balance the connections between excitatory and inhibitory neuron layers in the WTA architecture. Some principles were published to ensure the competition of neurons using STDP (Song et al., 2000); another approach relies on homeostatic processes such as adaptative thresholding of neurons in the excitatory layer (Diehl and Cook, 2015; Querlioz et al., 2013). Here in this work is set an adaptative threshold as a decaying exponential function similar to the one used on (Diehl and Cook, 2015), where each spike fired by the neuron increases its firing threshold to keep a homogeneous usage of neurons through the evolution of the network.

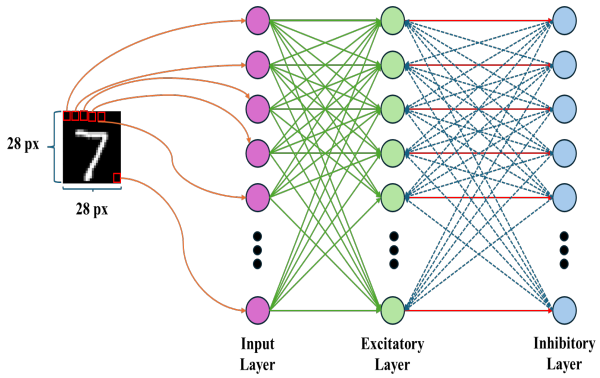


Figure 2.10: Classical *Winner-takes-all* circuit for a MNIST classification task. Each pixel is encoded as a Poisson spike train based on the intensity for each pixel (e.g. 255 of intensity is represented as a spike train of 255 Hz within the duration of the stimulus presentation) and its feed to the network in a fashion all-to-all to the excitatory layer.

2.4.1 Winner-takes-All Network

A simple architecture to generate competition between layers of excitatory and inhibitory neurons has been studied through the Winner-takes-all (WTA) circuit. WTA connectivity has been defined as a network able to elicit an active response at the cell that receives the highest external input (Kaski and Kohonen, 1994), meaning that is possible to train the network and make the cells represent a respective input pattern. Proper tuning of the dynamic behavior of WTA circuits can lead to interesting results such as an increment of the input's spiking patterns that the network can learn (Chen, 2017).

A WTA circuit usually consists of two processing layers (excitatory and inhibitory) as shown in figure 2.10, where the excitatory layer receives the inputs from the network while the inhibitory layer through a recurrent connection regulates the firing of the excitatory neurons until the system reach their stable state. Different works have been published with modifications on the WTA connectivity and statistic analyses (Chen, 2017; Lynch et al., 2019) improving the overall performance of the circuit. Still, this thesis work would use the basic WTA circuitry used on (Diehl and Cook, 2015) to exchange their performance through an input preprocessing stage.

The usual structure for generating the WTA phenomenon's competitive environment is one-to-one connectivity for each neuron from the excitatory layer to the inhibitory layer. This arrangement would make each spike generated from an excitatory neuron elicit a spike to fire on their subsequent

inhibitory neuron. At the same time, the neurons from the inhibitory layer have a recurrent connection to the excitatory layer except for that excitatory neuron that receives its initial spike. This connectivity from the inhibitory layer receives the name of *lateral inhibition* (Xie et al., 2000), and it is the main mechanism that generates competition through neurons for a hard-winner WTA.

2.4.2 Neuronal Balance

We know that even if each neuron within the excitatory layer maps and learns the input pattern through STDP, the dynamics of the network can drift the overall excitatory layer firing rate, becoming over-excited or over-inhibited, leading to an unstable regime of the population activity that would not produce the competitive computation we are aiming with the Winner-takes-all. A response regulation process is necessary to prevent a single excitatory neuron holds all the input patterns mapped within it.

A neuronal property known as *Intrinsic plasticity* (Watt and Desai, 2010; Zhang and Linden, 2003) has been studied and coupled with the STDP to regulate the electrical excitability of individual neurons. We can consider a neuronal threshold trace $V_{thr}(t)$ that decays over time exponentially when each time a neuron produces a spike $\delta(t)$ it increases our neuron's threshold by a fixed value θ as shown in equation (2.7). This implementation of intrinsic plasticity as an adaptative threshold has been used in other works such as (Querlioz et al., 2013; Diehl and Cook, 2015).

$$\tau \frac{dV_{thr}}{dt} = -V_{thr} \qquad Where \quad if \quad \delta(t_{spike}) \rightarrow V_{thr} += \theta \tag{2.7}$$

2.4.3 STDP Weight Normalization

A network trained by STDP can modify its synapse locally, depending only on the pre- and post-synaptic neuronal trace correlation response. Even though, as the case of the balance between excitation and inhibition neurons, the hyperactive effect of a couple of neurons can be seen if we do not constrain how the weight changes are performed within the synapses where the STDP is working. As it is the STDP's working nature, a pre-post correlation of spikes would increase the synaptic efficiency of that synapse. A positive feedback connection can arise if those changes are absolute within that single synaptic connection.

For every Hebbian-based learning rule such as the additive STDP (Song et al., 2000; Sjöström

and Gerstner, 2010) presented in equation (2.5) it is needed some constraint to limit the harsh increment from the STDP behavior. The usual form is to constrain the weights with divisive enforcement (Goodhill and Barrow, 1994) to assure an equal usage of neurons through the training neuronal layer while preventing the uncontrolled increments of the norm of the weights by the STDP. The update of weights with this normalization is given by:

$$w_{ij} = \frac{w_{factor}}{\sum_{i=1}^{N} w_{ij}} w_{ij} \tag{2.8}$$

Here equation (2.8) prevents an abrupt increment of each synaptic weight through a linear summation of the surrounding synaptic weights between the input and excitatory layer. The weight update becomes dependent on the updates from the peripheric synapses connected to a single neuron. This phenomenon helps to preserve the features that each neuron must be able to build a map through the training phase (A simplified description of the algorithm in python can be found in section A.1).

Chapter 3

Methodology

The main advantage that makes the difference between SNN and ANNs is the usage of the membrane potential of a neuron to encode and decode the information through spike trains. This point makes the simulation time mainly important, as each spike's timing plays a significant role in producing a change in the synaptic weight. Also, it is vital to highlight the fact that in comparison with SNNs, the network evolves through the passage of time not only the synaptic connection between nodes. This fact makes it possible to treat the whole network as a simple dynamic system (in correlation with knowing that a real brain is a dynamic system).

The following chapter is divided into four sections that highlight the implementation of our model. This work mainly compares the architecture's biological arrangements and input normalization method for a classification task over the MNIST dataset. So, in this chapter, we will extend the tools explained in the prior chapter. We begin with a detailed description of the network architecture with and without the biological input normalization, followed by the preprocessing applied to each input.

Each pixel of each MNIST image is converted to a spike train by a Homogeneous Poisson process so that we can pass it through a Gabor filter inspired by how simple cells in the visual cortex produce ON-OFF stimuli to contrast in the illumination of different scenic images. It is crucial to consider the heterogeneity arises from the difference in the shape of each digit on the MNIST dataset. This shape difference can provoke that along the testing phase; some images do not evoke a spike on the excitatory layer, causing, at the same time, a decrease in the total accuracy of the model. An

input normalization method was introduced to restrict the number of spikes generated through the Poisson process to an expected value for each digit label of the dataset.

For the learning rule, as an addition to the pair-wise STDP explained in the prior chapter, it includes the arrangement of the frequency effect pre-post-post spikes modeled through a Triplet STDP. Experimentally, it has been observed that an increment in the frequency of pre-post pairs would have, as a consequence, an average increment in the "Long-term Potentiation" (LTP) of the network. This effect cannot be modeled with the common pair-wise STDP, so the focus of this study is on a comparison of how the use of the Gabor filter affects the overall accuracy of both learning rules. We end this chapter by presenting how we would benchmark the results produced within this work over the latest research of pattern recognition models with Spiking Neural Networks.

3.1 Network Architecture

The network circuitry can be divided into two stages: the input processing and the core network (Winner-takes-All) to perform the actual training of the model (figure 3.1). As was mentioned in the last chapter, the WTA network promotes competition between neurons through cyclic connectivity between excitatory and inhibitory neuronal groups. The excitatory neurons transmit the spikes in the inhibitory group in a one-to-one fashion; in consequence, each inhibitory neuron produces a spike train to inhibit all the excitatory neurons except that excitatory neuron that the inhibitory neuron receives a connection.

In the next section, it is explained in detail how the input preprocess stage was performed. While the norm operation shown in figure 3.1 is done by divisive enforcement, a biological normalization approach can be applied if it regulates the output spike trains from the Poisson distribution through an extra inhibitory layer. This chapter, it would be explain the connectivity of a Poisson output regulation and compare their performance against the divisive firing rate enforcement.

3.2 Input Processing

The Winner-takes-All (WTA) circuit trained with SNN and STDP-based rules has already produced good results for the MNIST classification task without the necessity of an input processing stage (Diehl and Cook, 2015). So, adding a further preprocessing step could lead to better overall

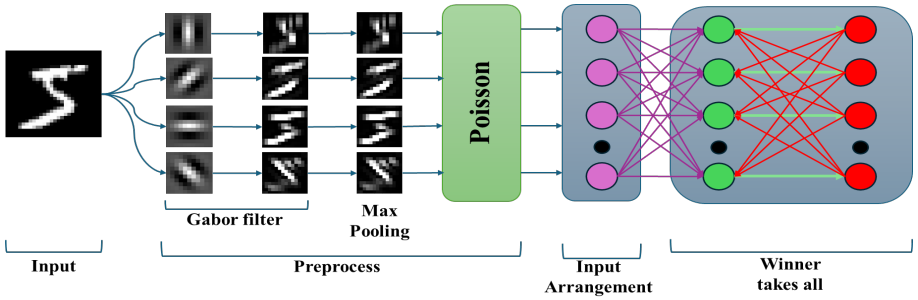


Figure 3.1: Network processing flow for the MNIST image classification task. The input preprocess stage covers an angle discrimination for each input image based on the behavior of simple cells within the visual processing pathways; furthermore, a norm operation was applied to reduce the heterogeneity between each input image and then covert each pixel into a spike train with a Poisson distribution. In the Winner-takes-All stage, the blue and red dots represent the excitatory and inhibitory groups correspondingly. Each pixel spike train is connected to each excitatory neuron through a synapse adapted with the proper learning rule to be evaluated. The output after learning is decoded within the spikes produced by the excitatory neuronal layer.

performance for the WTA network. This work presents a biological input processing stage with angle discrimination using Gabor filters, which have been proven to model the behavior of simple cells in the visual retinal pathways (Dayan and Abbott, 2001; Kandel et al., 2012). Further input process operators were applied to the network to be "trainable" and match the network structure without increasing the network complexity; such operations were max pooled (to match the original 784-pixel input of the neural network) and input normalization (to reduce heterogeneity between images).

3.2.1 Poisson Representation

One of the main differences between classical ANN and SNN-based architectures is the transformation of input encoding and decoding processes. A spiking network needs a way to represent the input into a spike scheme so the network can process the information. As has been mentioned before, this work will focus on the MNIST classification task trained by an SNN architecture.

The sequence of action potentials generated in the brain has a stochastic nature. Even if a human neural network has dynamical and deterministic foundations, the number of variables makes it seem random when a neuron fires an action potential. A Poisson process can capture a good approximation for the stochastic neuronal firing (Dayan and Abbott, 2001). Considering a

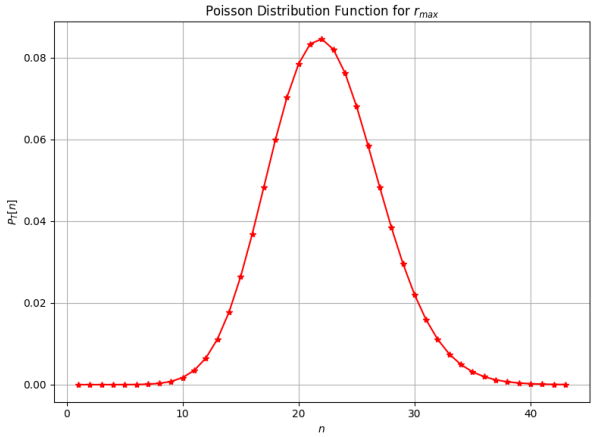


Figure 3.2: Homogeneous Poisson point process based on the maximum firing rate set for the MNIST dataset input images.

firing rate constant over time and that the spike times are ordered up to the stimulus duration T, the Homogeneous Poisson process can be defined as

$$P_T[n] = \frac{(rT)^n}{n!} e^{-rT} \tag{3.1}$$

The Poisson distribution described in (3.1) can show the probability of generating an n amount of spikes for an average firing rate r. The factor rT predicted the peak value of each possible generated spikes based on the firing frequency rate r and the duration of stimulus T. So, in an MNIST classification, it can be assigned the value of each pixel to be the firing rate we want for each pixel to produce a spike sequence. Considering we want to work within biological margins for the neuron's firing rate, we reduce the maximum brightness of each image within the dataset. The maximum pixel brightness value was chosen to be of 63.75 (or a $r_{max} = 63.75Hz$ for the firing rate equivalent). The Poisson distribution for generating a spike sequence for each pixel by a stimulus presentation of $T = 0.35s$ is shown in figure 3.2. It is important to point out that for generating the spike train for computing processing (as we are working with digital images), the discrete probability for building the spike train, n was set to a unit value. The probability of generating a spike would turn from 0 to 1 for each simulation timestep. This process is repeated up to the stimulus length. The discrete usage of the Poisson process receives the name of the Poisson spike generator.

Now that we have a proper probability function $P_T[n]$ that can generate spikes from a uniformly distributed population, a rule is needed to create the random sequence within the stimulus duration. In a computer system, for every Δt, the program generates a random value X_{rand} taken from the described Poisson distribution. If X_{rand} is uniformly distributed over the range between 0 and 1, the negative of its logarithm is exponentially distributed (Dayan and Abbott, 2001). By this convention, we can build the spike times based on the formula $t_{t+1} = t_i - ln(X_{rand})/r$.

3.2.2 Gabor Filter

A spatial receptive field of simple cells can be mapped approximately with a Gabor function. The spatiotemporal property of simple cells has been well studied since the novel Hubel and Wiesel work for cat receptive fields (Hubel and Wiesel, 1962) and even now Gabor filters continue as a good approximation of their behavior (Rouag et al., 2023; Ruslim et al., 2023). Gabor function results from the relationship between a Gaussian function and a sinusoidal to represent the frequency and orientation of a selected input. In (Dayan and Abbott, 2001), we can relate a 2D Gabor filter for modeling the simple cells as described in equation (3.2).

$$D_s(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right)\cos(kx - \phi) \tag{3.2}$$

The x and y values are the coordinates for the initial position of the 2D Gabor filter. The size of the receptive field for the x and y direction is determined by σ_x and σ_y through the respective preferred frequency k and a preferred spatial phase ϕ to set the limits of the boundaries for the ON-OFF receptive response. A spatial aspect ratio γ can modify the Gabor filter for the relationship $\sigma_y = \sigma/\gamma$. By the following relation, it can be assumed an equivalent increment on the receptive field for both x and y directions, leading to $\sigma_x = \sigma$. The contour surface of the 2D Gabor filter can be observed in figure 3.3 where the upward and downward hills delimit the ON-OFF responses of our spatial receptive field.

The easiness of implementing Gabor filters on images make it an easy approach for our MNIST classification task. Once the image pixel intensity has been transformed into spike trains with a Poisson function, we can proceed to apply the 2D Gabor filter. In equation (3.2) consider, $x = x'cos(\theta) + y'sin(\theta)$ and $y = -x'sin(\theta) + y'cos(\theta)$, we can set $\theta = [0, 45, 90, 135]$ to get a 4 angle discrimination by rotating filter. The ganglion cells modeled as a Gabor filter can be observed in

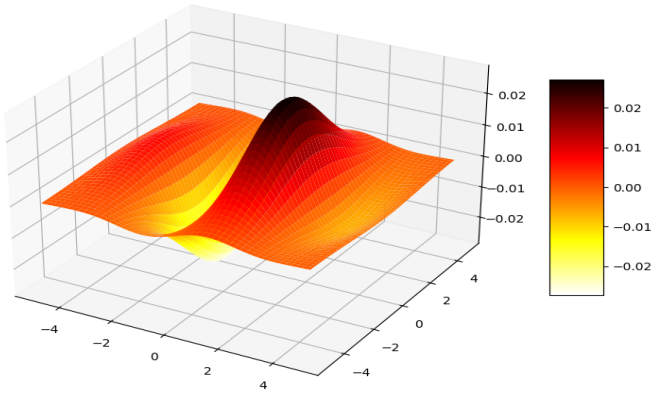


Figure 3.3: Spatial receptive field for delimited ON-OFF responses regions. The downside bright region represents the OFF response contour for a high absence of brightness stimuli (or dark stimulus). The upper side has a strong response to illumination stimuli (ON response) and is usually measured as the responsive degree from the difference between the peak points of the OFF and ON hills.

figure 3.4. Notice that by rotating to a specific angle the illumination of the input change to a strong signal at the selected orientation.

After capturing the orientation features of the MNIST dataset, we would get four image sets of the length of the original dataset. To prevent increasing the number of input pixels to match the just generated 4 orientation images, it arises the necessity of further input preprocessing. It is possible to downsample the four input images using *max pooling* and make it match the original number of pixels, at the same time it increases the intensity of the feature orientation. In figure 3.5 is observed the input image after a max pooling operation. Notice, that the reduction of pixels was after the Gabor filter we have a 28 x 28-pixel image, and the max pooling operator downsample it to 14 x 14 pixels.

3.2.3 Input Normalization

The MNIST dataset has images from the 0 and 9 digits in a hand-written representation having differences in brightness and shape depending on each digit class. Normalizing the matrix image values is a usual approach to reducing the heterogeneity between images. A usual approach is a

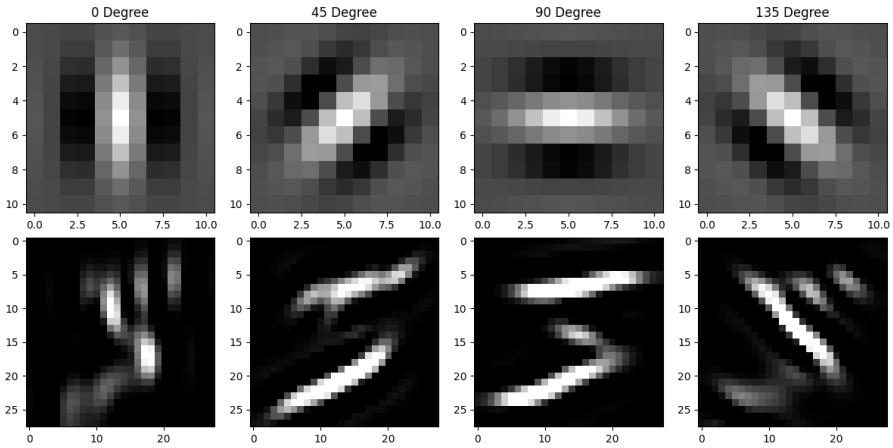


Figure 3.4: 2D Gabor filters applied to four degrees orientation ($\theta = [0, 45, 90, 135]$) to the MNIST dataset. The parameters used for the filter were: $\sigma = 2.201$, $\lambda = 5.6$, $\gamma = 1$, and we described an even phase by setting up $\phi = 0$. The four orientations of the Gabor filter help to discriminate successfully the angle features of the MNIST dataset.

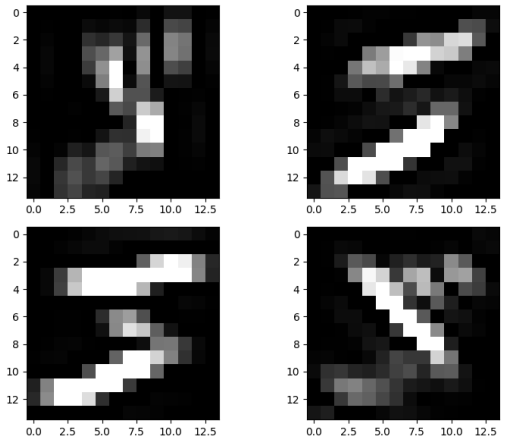


Figure 3.5: Downsample of the Gabor filter output by a max pooling operator. Each input image was reduced to half the number of pixels while maximizing the intensity of the feature orientation.

divisive enforcement described by the equation (3.3).

$$p_{ij} = \frac{p_{factor}}{\sum_{i}^{N_i}\sum_{j}^{N_j} p_{ij}} p_{ij} \tag{3.3}$$

Where the sum terms below in (3.3) represent the sum of all the pixel brightness values from the image, p_{ij} is a specific pixel within the image, and p_{factor} is a scalar value to modify the tendency to which we want to set the norm for all our input images. Note that the divisive enforcement of (3.3) is similar to the applied for the synaptic weights in (2.8).

3.3 Learning Rule Implementations

The STDP learning rule presented by (Song et al., 2000) has a great percentage of characterization from the original Hebb learning postulate. Even if it can correlate experimental data with a pair-wise STDP rule, it may fail to reproduce pair-based frequency effects from some experimental data (Sjostrom et al., 2001). There exists a broad range of different modifications possible to the classical STDP rule to make it match the experimental data. This work was developed under the following adaptations: the triplet STDP (pre-post-post) rule and the relationship between the "most important" spikes that tend to stable learning with a nearest-neighbor spike scheme.

3.3.1 Triplet STDP

In a biological system, the change in the efficacy connection between synapses depends on the exact timing of presynaptic and postsynaptic neurons based on the Hebbian statement (Hebb, 1949) described in the prior chapter. The basic STDP rule evaluates the weight changes in pairs of pre-post spikes where common experiments consider between 50-60 pairs of spikes. Some authors have confirmed that an increase in the frequency of pairs for the experiment of the STDP can increase the net potentiation of the network synaptic efficacy (Sjostrom et al., 2001).

Still, the frequency effect is not possible to reproduce with the common pair-wise STDP, where an increment in the number of pairs leads to a net depression of the synaptic connection. This problem was addressed in (Pfister and Gerstner, 2006), where they implement a Triplet version for the STDP that can account for the repetition frequency effect to a net potentiation of the network.

The synapse efficacy modification rule of a triplet-based STDP has the same working mechanisms as the pair-wise STDP presented in section 2.3.3 with the addition of another synaptic trace at a different speed. We can assume that each presynaptic and postsynaptic spike will leave two synaptic traces (fast and slow). Let the variables $pre_{fast}(t)$ and $pre_{slow}(t)$ increase for every presynaptic spike event produced at the synaptic connection with an exponential decay behavior:

$$\frac{dpre_{fast}}{dt} = \frac{pre_{fast}}{\tau_+} \quad if \quad t = t^{pre} \quad then \quad pre_{fast} \rightarrow pre_{fast} + 1 \tag{3.4}$$

$$\frac{dpre_{slow}}{dt} = \frac{pre_{slow}}{\tau_x} \quad if \quad t = t^{pre} \quad then \quad pre_{slow} \rightarrow pre_{slow} + 1 \tag{3.5}$$

Analogously to (3.4) and (3.5), the postsynaptic spike event generates for every synaptic connection Syn_{ij} to drive the variables $post_{fast}(t)$ and $post_{slow}(t)$.

$$\frac{dpost_{fast}}{dt} = \frac{post_{fast}}{\tau_-} \quad if \quad t = t^{post} \quad then \quad post_{fast} \rightarrow post_{fast} + 1 \tag{3.6}$$

$$\frac{dpost_{slow}}{dt} = \frac{post_{slow}}{\tau_y} \quad if \quad t = t^{post} \quad then \quad post_{slow} \rightarrow post_{slow} + 1 \tag{3.7}$$

With both presynaptic and postsynaptic traces defined, we can state how the synaptic efficacy $w_{ij}(t)$ would evolve during the training process.

$$w_{ij}(t) \rightarrow w_{ij}(t) - post_{fast}(t)[A_1^- + A_2^- pre_{slow}(t_{spike}^-)] \quad if \quad t_{spike} = t^{pre} \tag{3.8}$$

$$w_{ij}(t) \rightarrow w_{ij}(t) + \ pre_{fast}(t)[A_1^+ + A_2^+ post_{slow}(t_{spike}^-)] \quad if \quad t_{spike} = t^{post} \tag{3.9}$$

Note that equations (3.8) and (3.9) represent the decrement and increment of the synaptic weight regarding if it had occurred a presynaptic or postsynaptic event correspondingly. The variables $A_{1,2}^{\pm}$ are positive constants that indicate how much each trace impacts the synaptic efficacy. It is important to remark that setting up $A_2^- = A_2^+ = 0$ can model the synaptic efficacy changes as in a normal pair-wise STDP. The slow synaptic trace is meant to be evaluated one timestep before the next spike is fired (t_{spike}^-). This convention helps us to fix the weight changes to depend on a triplet or quadruple interaction of spikes.

The relationship for the change in synaptic efficacy w_{ij} is shown in figure 3.6. Every presynaptic spike event would cause the weight to be evaluated between the presynaptic trace and a fast

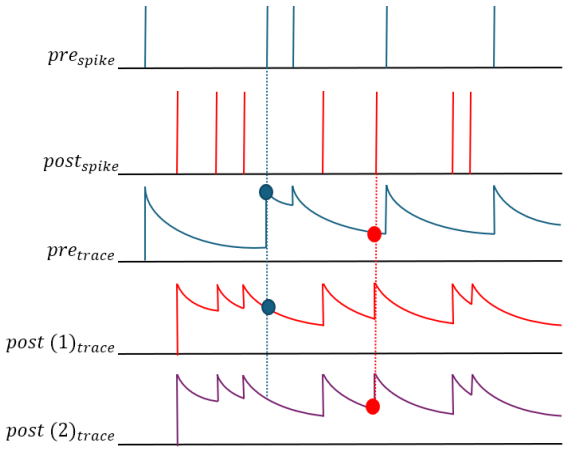


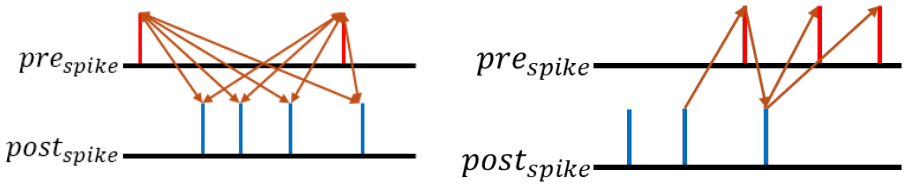
Figure 3.6: Relationship of weight updates for a triplet STDP (pre-post-post). A presynaptic spike event is described by the blue dots for pre_{trace} and $post(1)_{trace}$ to calculate the positive weight update. The postsynaptic spike event calculates their efficacy with the values of pre_{trace} and $post(2)_{trace}$ marked with the red dot. Note that in the postsynaptic trace 2, the red dot is proceeding to its increment value to capture the effect of the postsynaptic slow trace.

postsynaptic trace (as done in the pair-wise STDP to represent long-term depression of the synaptic efficacy). On the other hand, a postsynaptic spike event would elicit a change in the efficacy of the connection between the presynaptic trace and a slow postsynaptic one. This method lets us model a net potentiation effect from the whole network at the increase in the frequency of pre-post synaptic pairs. (Note that is possible to set the LTD to depend on the pair frequency effect changing how the weights interact between the synaptic traces; experimentally only the LTP frequency effect has been reported in the literature (Sjostrom et al., 2001)).

3.3.2 STDP with nearest neighbor spike paring scheme

The previous chapter gives an overview of how learning is modeled using exponential traces on a spike pair-wise-based update between presynaptic and postsynaptic spike trains as set in (2.5). Still, it is not a biological feature that neurons can keep a record of the history of past spikes that were produced, as has been said for common additive models (Song et al., 2000)

The Symmetric Nearest-Neighbor (Sym NN) scheme (Izhikevich et al., 2004; Morrison et al., 2008) has the feature to only consider those spikes that are the nearest to either the presynaptic (potentiation) or postsynaptic (depression) to update the strength of the synaptic connection.



(a) All-to-all Interaction (b) Symmetric Nearest-Neighbor

Figure 3.7: Interaction of presynaptic and postsynaptic spike trains for updating the weight of the synapse. (a) All-to-all Interaction. Consider each spike (either presynaptic or postsynaptic) to modify the synaptic efficacy w_{ij}. (b) Symmetric Nearest-Neighbor. Only the peripheric spikes would be considered to update the weight of the synapses. A peripheric spike would be considered to be those opposite spikes from either the prior or posterior neuron.

In figure 3.7 it is observed a comparison between the Symmetric NN scheme and an All-to-All interaction.

A Symmetric NN spike interaction has been demonstrated to match experimental data for synaptic evolution (Sjostrom et al., 2001). His ability to regulate which spike pairs contribute to plasticity (Izhikevich and Desai, 2003) makes it relate to the novel biological model of LTP/LTD modification. Symmetric NN can be applied either to pair-wise or triplet STDP and in this work, it would proceed to report the results with both learning rules.

3.4 Benchmarking Tools

The reliability of a neural network nowadays usually has a common comparison point to highlight the advantages and disadvantages of each model. For example, if we were to use a method for tunning parameters and hyperparameters for a model, an accuracy result comparison between the same algorithm can give us an overview of which parameters affect the overall accuracy of the model within the same algorithm. For a Spiking Neural Network, this point turns out to be particularly important. Since an SNN introduces the concept of spike coding and decoding of information, a common ground is needed to be set for the decoding of the output based on the maximum firing rate of each neuron at the excitatory layer (a method used in (Diehl and Cook, 2015)).

A slight difference in the computation method can also produce divergent results at the moment of validating the model performance. For the same reason was built our model with the Brian2 framework which has been increasing lately in popularity with the simplicity of recreating

experimental data based on the dynamical equations that govern the system behavior. A specific description is shown in the following section.

3.4.1 Model Accuracy Calculation

In order to benchmark the model with the latest research for the SNN image classification task, it is necessary to have a common start line for the validation tests. The decoding scheme implemented for the project is based on the maximum firing rate from each neuron regarding the label classes of our training dataset. We get our model reliability with the following three-step process flow.

1. **Neuronal Mapping (Train dataset)**

There are two conditions to meet in order to produce a proper mapping of the excitatory neurons to the class labels for the arrangement of synaptic weights (receptive field): fix the homeostatic rule parameters of the network and stop the learning from progressing along the dynamics of the network. After the model's training was disabled the plastic rule and passed through a single presentation of the training dataset to record the number of spikes per neuron from the excitatory layer. It is assigned a class label to each neuron based on the maximum firing rate presented for each digit of the MNIST dataset.

2. **Model Validation (Test dataset)**

As was done in the previous point, it was recorded the number of spikes fired in the excitatory layer to a single presentation stimuli of the test dataset. It is important to point out that, unlike the neuronal mapping, fixing the homeostatic rule has a more crucial impact on how the accuracy of our test would be completed. The whole Spiking neural network can be treated as a nonlinear dynamical system that continues evolving over time. The evolution of the homeostatic rule can produce subtle differences at the moment of the testing evaluation.

3. **Accuracy Calculation**

Lastly, it was gotten the neuron that produced the most number of spikes (based on the spike count recorded from point 2) to a specific image and compared the neuron label class (Neuronal map from point 1) to their true label so it can be checked whether it decodes correctly or not the input image. This decoding method (based on the neuronal firing rate) has been widely used for

the MNIST classification within a WTA architecture (Querlioz et al., 2013; Diehl and Cook, 2015), but other decoding alternatives could be interesting to study their behavior (Skatchkovsky et al., 2021). The training and validation phase workflow algorithm are shown in the figures A.1, and A.2 from Appendix A. The Python implementation of the algorithm can be also seen in sections A.1 and A.2.

3.4.2 Brian2 SNN Framework

As happens for a great variety of neural network models, a framework for building our project that can take the lead for managing the information processing for us, is a huge help to prevent us from starting from the stretch, usually leading to different results by differences in the coding structure and logic. A difference with regular artificial neural networks is that can be robustly built using frameworks such as PyTorch (Paszke et al., 2019) or Tensorflow (Abadi et al., 2016), meanwhile, Spiking Neural Networks do not have a toolbox for general purposes. Some authors have evaluated the performance (Kulkarni et al., 2021) or classified their experimental focus (Yamazaki et al., 2022) in areas such as robotics, biological or computational. So, in order to move towards a more biologically plausible model it was choosen the Brian2 framework to run our simulations.

The Brian2 framework (Stimberg et al., 2019) gives us an easy approach to creating a variety of SNN models. In contraposition, as in some SNN frameworks where the user can only select between a list of dynamical models (neurons, synapses, learning rule, etc.), Brian2 lets you describe the specific dynamics that the user wants for their particular model. For example, let it be a model as simple as the conductance-based "Leaky integrate-and-fire" described in equation (2.2). Brian2 counts with a string decoder to parse first-grade ordinal differential equations and solve them numerically. So, if it can describe the dynamics of our components within differential equations, is possible to arrange the model for match our necessities. The core of the latest neuroscience research has chosen Brian2 as its default simulator for its capability of building robust models without the need for a strong computational background. The full implementation of this project done in Brian2 can be found in a repository in GitHub shown in section A.3.

Chapter 4

Results

The analysis of this work can be subdivided into the learning behavior and feature configuration. Each learning rule characterizes different spike behavior; for example, the pair-wise STDP considers the changes in synaptic efficacy for a single pair of spikes; meanwhile, a triplet-based model can account either for net potentiation or depression based on the frequency of presynaptic and postsynaptic spike trains (a spike burst can modify the net efficacy of the synapse). Even though a burst of spikes isn't enough to compare how the learning rules behave, the pair of spikes considered for changing the efficacy can also carry important information to encode the input. We evaluate the learning rule spike interaction for a pair-wise STDP between a fully connected (all-to-all) and a nearest-neighbor (only consider those spikes near the previously fired spike) interaction.

We presented four configurations with and without the Gabor filter and input normalization for each learning rule. Does the Gabor filter affect the accuracy of the learning rule applied indistinctly? Some images cannot be decoded from the excitatory layer because No spikes are generated on the excitatory layer. Can the Gabor filter (or alternative input preprocessing) affect the intrinsic properties and behavior of the network? These are some of the questions that would be explored by measuring how a preprocessing stage can modify how a winner-takes-all network responds.

4.1 Network behavior

Before presenting the results obtained during this study, it is important to point out that a Winner-takes-all circuit within a spiking neural network has a complex dynamical behavior by the number of neurons used within the network. If it is considered even the network ease of becoming unstable because of soft modifications such as STDP parameters, the difficulty of successfully analyzing the problem will increase. In order to have a similar ground for how the learning should behave in the network, it is present the network behavior within the pair-wise STDP (with and without nearest-neighbor) and with the Triplet STDP, and how does a change in the learning rule shifts (within a defined range) the network dynamics.

4.1.1 Membrane potential

Different learning rules shine in their best scenario within slight differences in the parameters chosen for the network initialization. We can take the example of the simple pair-wise STDP, where pairs pre-post or post-pre are related to drive the network to an LTP or LTD behavior correspondingly. Experimental data and theoretical frameworks (Bi and Poo, 1998; Song et al., 2000) have shown that LTD has a slightly greater effect than LTP. This leads us to assume that to achieve a proper (or ideal) LTP effect is needed to tune the learning rate (or learning speed) to relate it to the results of the biology experiments.

As can be observed in figure 4.1, the differences between the potential behavior from an all-to-all and nearest-neighbor spike interaction are related to how the learning rates were chosen to work within. While it does not implement a proper tunning method for the hyperparameters of the network, manually tunning from it was performed. The nearest-neighbor model has the tendency to achieve greater accuracy for a slower rate; meanwhile, the all-to-all interaction was the other way around (a higher rate achieves better accuracy). This effect can also be thought of as the fact that in the nearest-neighbor model, fewer spikes impact the synaptic change, so the learning pace has a more significant impact on better learning performance than within the all-to-all spike interaction. The same reasoning can be applied to the Triplet STDP, where the performance is linked directly to how the learning rate was set.

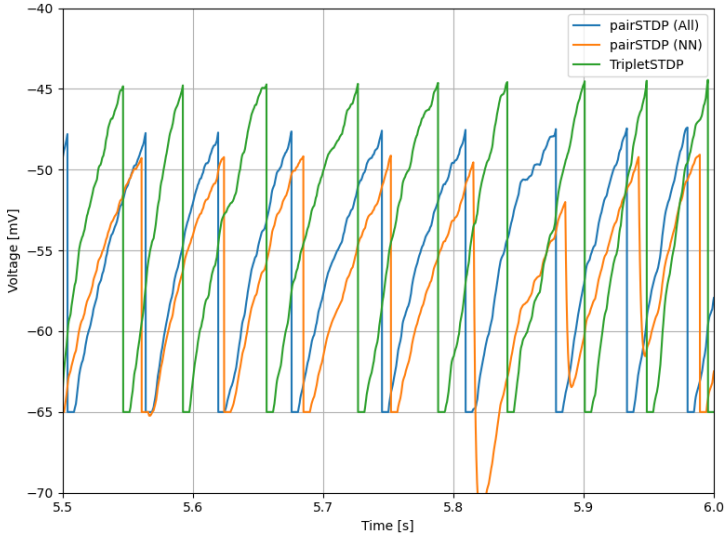


Figure 4.1: Comparison of the differences in the membrane potential per learning rule. The major difference from each rule captured is related to the speed. The learning presynaptic rate was 0.001, 0.0001, and 0.01 for the all-to-all, nearest-neighbor and Triplet STDP correspondingly. Each learning rate was modified manually for their tendency to their highest accuracy performance.

4.1.2 Spike-coding response

While it is true that our work is based on the same rate-coding system for each image from the dataset, the response (or effect) that would have within the excitatory layer of each learning rule would slightly differ. The firing behavior to 30 image iterations can be observed in the figure 4.2. The input was encoded as the flowchart from figure 3.1 considering the Gabor filter adaptation for oriented-based feature discrimination. Note that even with the slight differences in the responses from the excitatory layer for the three learning rules, the monopoly of the neural response isn't kept to a single neuron, where, over time, it is passed by to the next neuron, more likely to fire.

4.1.3 Weight changes through time

The synaptic weights for the connections between the input and excitatory layers were randomly initialized. So, to get the weight changes for each learning rule, it is vital to consider the precise spike timing within the correlation of the synaptic traces to determine how much it is affected by the changes of a specific time-lapse. The difference between the behavior of the weight changes for the

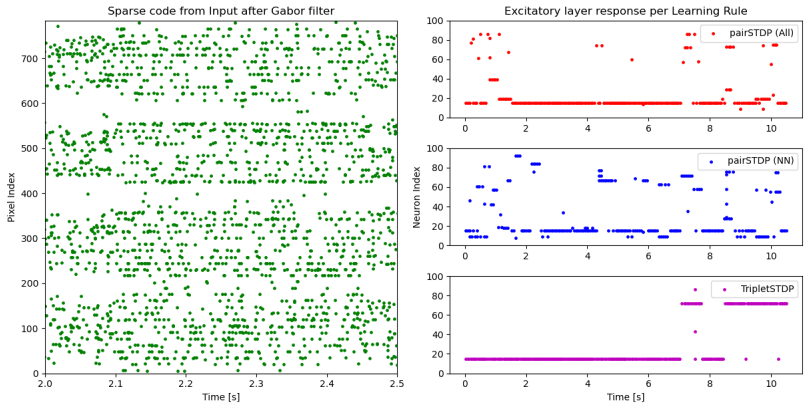


Figure 4.2: Input and excitatory layer response to the encoding of the 30-first images of the dataset. Each dot from either plot represents a spike event along the training simulation. The left side plot is the encoded output after parsing each pixel from the image using the Poisson homogeneous process. At the same time, the right side plot shows the spike behavior of the excitatory layer by being fed from the spikes of the upper plot.

three learning rules can be observed in figure 4.3. Even with the differences in the considerations of the spikes interaction for the learning rules, this does not change the overall behavior of the learning rule through time. As explained in the section for the membrane potential behavior, their major difference is the tendency speed that each learning rule has to achieve its overall best accuracy performance.

We can see this by observing the all-to-all (upper-right) and nearest-neighbor spike interaction (middle-right) and comparing their synaptic weight evolution to the Triplet STDP (bottom-right) weight changes. Also note that for each synaptic trace in figure 4.3, it is expected to have a slight decay based on the equations expressed for the model in the past chapter. The exponential decay from the synaptic traces was set to be updated at the moment that the weight change was to be evaluated. This would prevent us from needing to evaluate the evolution of all the synaptic traces, considerably increasing the computation of the model. This consideration was implemented as a function within the brian2 simulator.

4.2 Input heterogeneity regulation

The dataset called MNIST contains images depicting handwritten digits ranging from 0 to 9, with each image being 28 x 28 pixels in size. The difference in shape between digits, combined with the

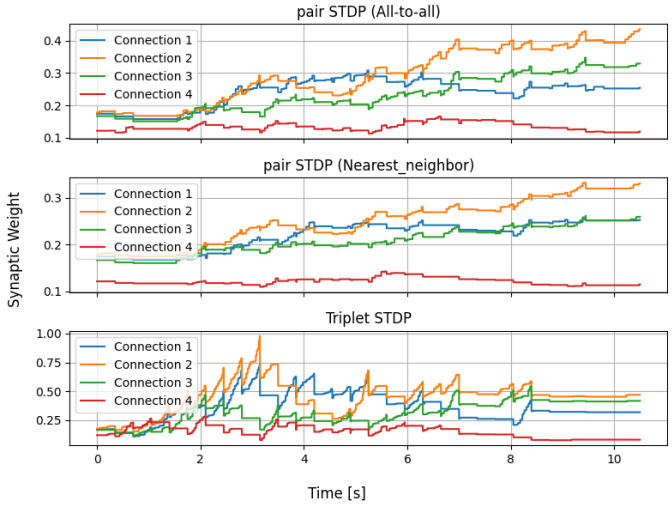


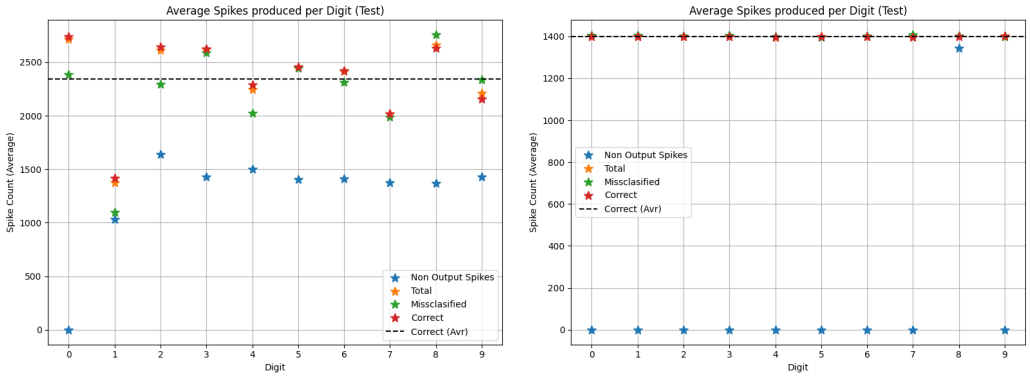
Figure 4.3: Weight changes of 4 synaptic connections between input and excitatory layers. The synaptic weight is fixed to changes based on the presynaptic and postsynaptic traces for the pair-wise STDP. At the same time, the Triplet STDP incorporates a second postsynaptic trace to include a potentiation based on the spike frequency of the postsynaptic neuron.

differences in illumination contrast between the same digits, can cause the input not to be mapped homogeneously within all the class labels. The heterogeneity of each MNIST input image can be reduced if a normalization method is applied to each input before being converted into spike trains through the Poisson point process.

The normalization effectiveness was measured by counting the number of spikes produced for each input image after being converted to spike trains, where the spike count for all the images of each class label was averaged. At the same time, the impact of the normalization method during the testing phase was evaluated. As a consequence of updating the threshold after each spike event, some neurons would need a stronger input in order to they can produce a single spike. This would result in some images with low pixel values (or an image with low illuminance contrast) not generating a single spike through the whole excitatory layer. This normalization method was proposed to make all inputs generate at least a single spike on the excitatory layer.

4.2.1 Image spike count

A comparison between using (and not using) normalization can be shown in figure 4.4 for an all-to-all pair STDP. The average spike count per class label it is represented on the horizontal axis. As



(a) Without Normalization (b) With Normalization

Figure 4.4: Pair-wise STDP with an all-to-all spike interaction comparison of spike count with and without the normalization method implemented. The blue stars represent the average of images per class that do not produce a response on the excitatory layer. The black dashed line is the average of the red stars through all the label classes.

it is observed in figure 4.4a, the class labels averages that are the farthest from the spike count an average of correctly classified images (black dashed line) do not generate a single spike within the excitatory layer (blue stars), in consequence, it cannot decode a result effectively from that specific image. This behavior is constrained by the normalization rule applied in our model (figure 4.4b).

The normalization rule assures that the pixel value is homogeneous for every image, so when it is encoded as spike trains, the average spike count would be similar for every digit. We can see in figure 4.4b that for almost every digit, the average of images that do not respond in the excitatory layer is near nonexistent. The Non-Output Spikes (blue stars) with an average spike count of 0 can be considered as if all the images within that class label generated at least a single spike, so proper decoding was possible. Similar results were obtained independent of the learning rule used within the input normalization.

4.2.2 Normalization through image iterations

While the normalization can keep a homogeneous pixel value for every image, it isn't the only factor that has an effect within the images that does not generate output spikes at the excitatory layer. In the figure 4.5a, it can be seen how the accuracy change through each image iterations for the 40,000 images of the training dataset within a single epoch. Note in figure 4.5b that, as

explained in the past section, the normalization has a direct effect on the percentages of images that do not respond from the testing dataset. This percentage increases up to a stable regime (10% for the case of the pair-wise STDP all-to-all spike interaction) as the amount of training iteration increases. It is important to check out that in the cases where the normalization isn't included in the model (yellow & blue curves), the accuracy oscillates on the 5k image iterations to then decay to a stable regime. This oscillation in the accuracy can be softened with the normalization (red & green curves) to indicate that a normalization rule can constrain the overfitting of data within the model.

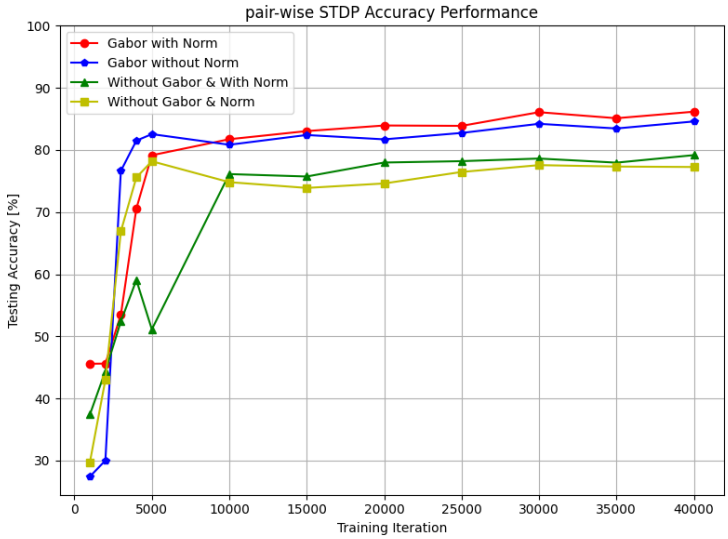
4.3 Gabor filter and model accuracy

The overall accuracy of the model was increased considerably, undistinguished by the learning rule used. While normalization could play an important factor as it increases the number of images that can be decoded effectively, the obtained results point out that this is not the case. The Gabor filter used to highlight 4 orientations of the input images has mainly a direct impact on the increase of accuracy. This can seem obvious considering that a greater number of pixels now are used to encode information, whereas, as a consequence, our model can be well-tuned to classify a wider variety of input patterns correctly.

4.3.1 Pair-wise STDP

The evolution of the model accuracy can be observed in figure 4.6 for the pair-wise STDP with nearest-neighbor spike interaction, while the all-to-all spike interaction is shown in the figure 4.5a. Note that even for both cases, the model with both the Gabor filter and the input normalization reaches a peak in the model accuracy. All the models have a quick change in accuracy up to the 5000 image iterations, where then the tunning of the model accuracy tends to be more "fine-tuned." this means the model after the 5000 iterations increases gradually but with considerably smaller changes.

The overall results from the pair-wise STDP with all-to-all spike interaction are shown in figure 4.7 for the model using Gabor filter and input normalization. The confusion matrix (figure 4.7a) highlights the percentages of the correctly (and misclassified) images per class label. The main diagonal of the matrix is the correctly classified images; meanwhile, all the other points of the



(a) Model accuracy through training images iterations

(b) Percentage of non-responsive images of the testing dataset through training image iterations

Figure 4.5: Model performance behavior through image iterations from training dataset at 4 different model conditions for the all-to-all pairwise STDP. The upper plot shows the changes in the overall accuracy model at the moment of a specific image iteration checkpoint. The bottom plot captures the percentage of images that do not produce a response on the excitatory layer at the momentary iteration checkpoint. Both figures were tested within the full testing dataset (10,000 images).

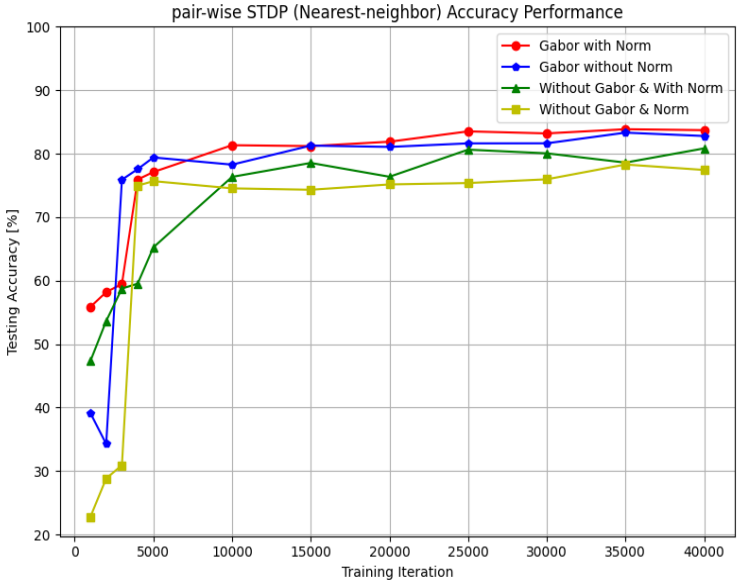
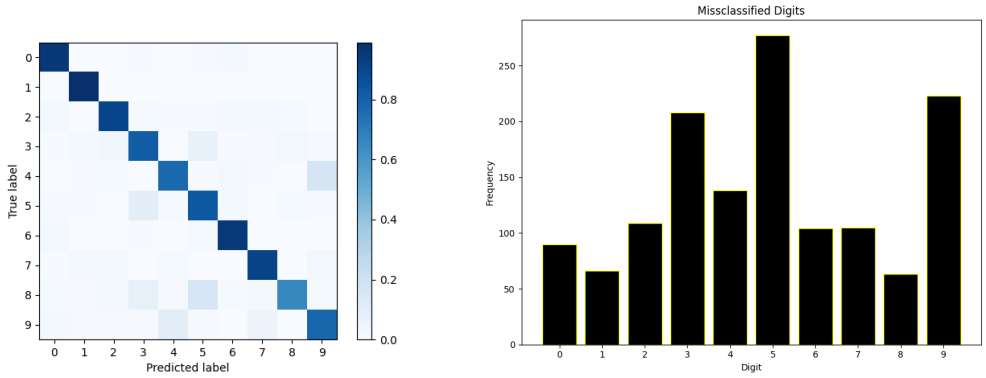


Figure 4.6: Accuracy behavior through training iteration for the pair-wise STDP with nearest-neighbor spike interaction.

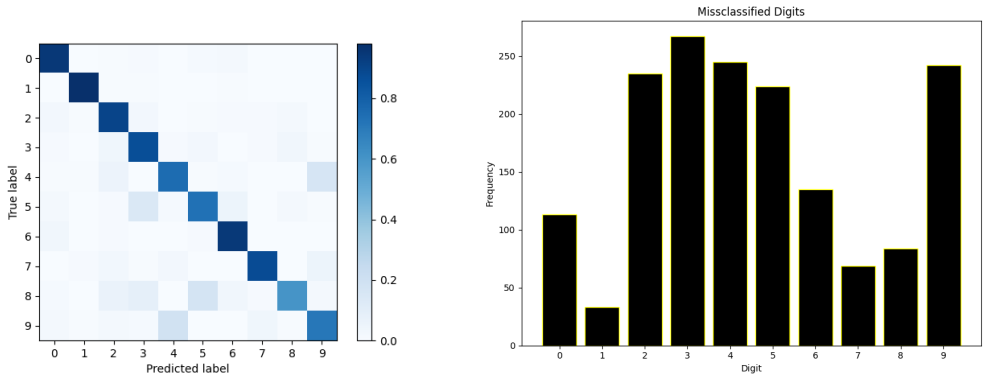
matrix besides that are considered misclassified. As we can see, there is a considerable amount of confusion between the digits 4 and 9; this can also be related to the fact that both shapes can be handwritten in a similar way, making it difficult even for humans to distinguish the difference between those two digits. Even with this fact, the class label with the higher amount of misclassified labels is the digit "5" as observed in figure 4.7b. The reason behind a higher level of confusion can be directly related a triple confusion of three digits (3, 5 & 8), where the curves of those three digits are difficult to be highlight with only the usage of the Gabor filter.

For the nearest-neighbor spike interaction, the results are observed in figure 4.8. In this case, some of the encoded information was considerably reduced by setting limits to the interaction between spikes. We would consider applying a weight change in the synaptic connection. Some of the information lost in this spike interaction provokes an increase in the misclassified frequency of the digits "2", "3" & "4". Interestingly, all the frequencies of the other digits remain almost the same, indicating that these digits have a greater influence in the spike interaction scheme selected.



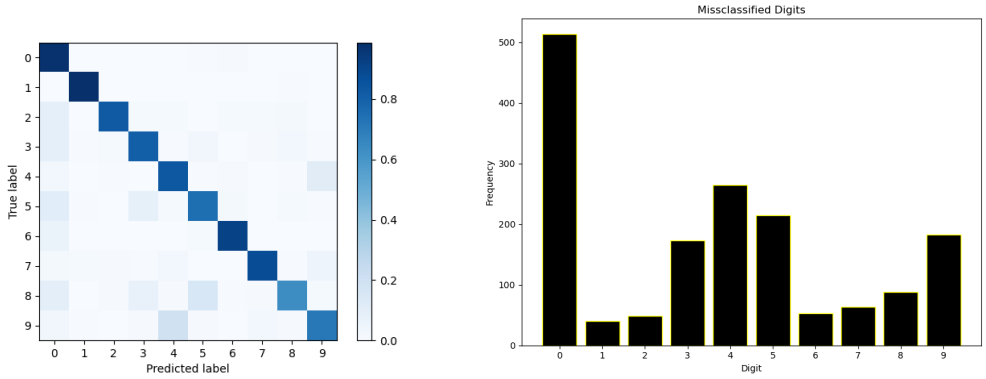
(a) Confusion Matrix (b) Misclassified Digit Count

Figure 4.7: Confusion Matrix & misclassified count of class labels for the pair-wise STDP with an all-to-all spike interaction learning rule.



(a) Confusion Matrix (b) Misclassified Digit Count

Figure 4.8: Confusion Matrix & misclassified count of class labels for the pair-wise STDP with a nearest-neighbor spike interaction learning rule.



(a) Confusion Matrix (b) Misclassified Digit Count

Figure 4.9: Confusion Matrix & misclassified count of class labels for the triplet-based STDP with a nearest-neighbor spike interaction learning rule.

4.3.2 Triplet STDP

As done for the previous pair-wise STDP with both spike interactions, figure 4.9 shows the results performance of the triplet STDP model considering a nearest-neighbor spike interaction. Notice that the frequency distribution of misclassified class labels is not different from the one shown in figures 4.7b & 4.8b with the difference of an overly misclassified images to the "0" label class. This behavior is believed to be related to the random initialization state of the model, where different random seeds do not produce the misleading fault to the "0" label class.

4.3.3 Model final accuracy

The obtained accuracy of the model per learning rule and feature configuration for a single trial are presented in the table 4.1. We can see that for all the model feature configurations, the Gabor filter has a direct impact on the accuracy changes for every learning rule. The learning rule that got the highest accuracy was the pair-wise STDP with all-to-all spike interaction with both the Gabor filter and input normalization. So, to check the overall scope of the learning rules, it was run 5 trials for each learning rule with both the Gabor filter and input normalization activated. The average accuracy for each learning rule was,

- **Pair STDP (All-to-all)**: 85.54%
- **Pair STDP (Nearest-neighbor)**: 84.32%

	Gabor with Norm	**Gabor without Norm**	**No Gabor with Norm**	**No Gabor without Norm**
Pair STDP (All-to-all)	86.16%	84.56%	79.17%	77.26%
Pair STDP (Nearest-neighbor)	83.70%	82.78%	80.83%	77.43%
Triplet STDP	83.57%	83.95%	80.80%	73.82%

Table 4.1: Model performance per variations of features (Gabor filter and input normalization) through a pair-wise STDP (all-to-all and nearest-neighbor spike interaction) and a triplet STDP learning rule.

Article	**Learning Rule**	**Preprocessing**	**Feature**	**Iterations**	**Accuracy**
Qu et al., 2019	STDP (pair-based)	No	Spike-current based homeostasis	1 (Full dataset)	88.5%
Ahokainen, 2024	STDP (pair-based)	Yes	Analytic initialization and Gabor filter for input processing	3 (Full dataset)	84%
Diehl and Cook, 2015	STDP (pair-based)	No	2-layer network architecture	0.67 (40k images)	82.5%
This work	STDP (pair-based)	Yes	Norm input response	0.67 (40k images)	86.16%

Table 4.2: Latest research performance comparison for unsupervised learning with the 2 neuronal layer circuit proposed by (Diehl and Cook, 2015). The models' reported accuracy performance of the papers in the table only contemplates an excitatory neuronal layer of 100 neurons.

- **Triplet STDP**: 85.02%

Even if the pair-wise STDP with an all-to-all spike interaction has the highest average, there is almost no difference between the learning rules. Previously published works have also shown that a difference in each learning rule can shine as the number of excitatory neurons increases. In this work, it was constrained to only using 100 neurons, getting a competitive performance for smaller architectures. In table 4.2, a comparison is shown with some of the state-of-the-art works for networks up to a layer from 100 neurons.

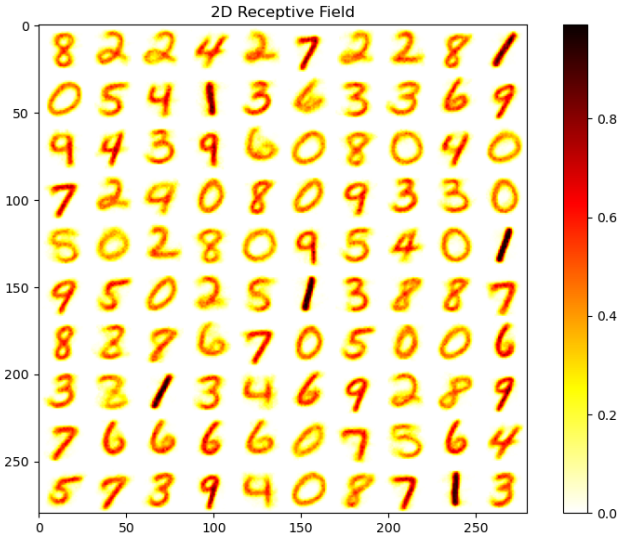
Chapter 5

Discussion

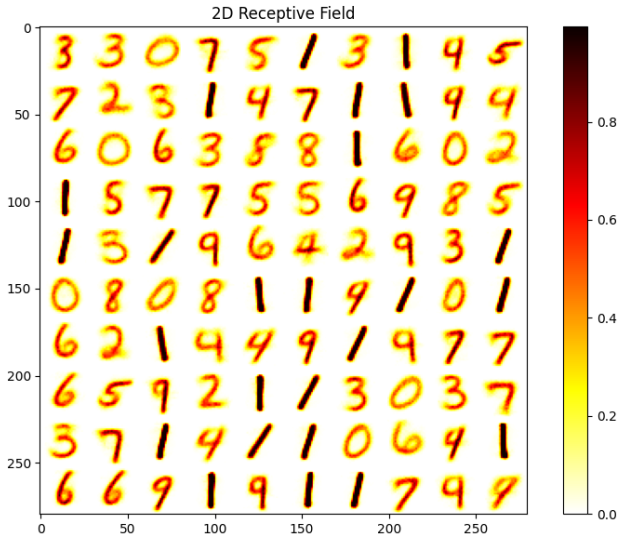
This work evaluates how a preprocessing stage can alter the intrinsic properties of the network architecture, indistinct of the learning rule, and spike interaction. In our model, using a homeostatic rule was vital as it helped us prevent a few neurons from dominating the learning, as has been demonstrated in prior works (Ahokainen, 2024; Qu et al., 2019). The MNIST dataset's varied shapes and brightness differences between inputs may make some inputs challenging to decode due to their low number of presynaptic spikes. Our logical approach to solving this problem was normalizing the input pixel values before being parsed to spike trains.

As it is shown in figure 4.4, the input normalization reduces (almost to nonexistence) the number of images that do not respond on the excitatory layers (blue stars). Interestingly, an increase in the number of images decoded from the excitatory layer does not imply a direct increment in the accuracy model. In table 4.1, while the accuracy has increased after applying the input norm to the model, the significant increment of the accuracy was only for the models with the Gabor filter. Some reasons for a non-substantial accuracy increment could be observed in the receptive field for the model with and without input norm.

Note in figure 5.1 that the receptive field, with the input normalization, was easier for it to focus on the digit "1". Further research is needed to enclose a decisive reason behind this effect. A proposed hypothesis could be related to the amount of pixels needed to encode the image. The digit "1" has the lowest pixel area along all the digits from the dataset, making it easy to focus on this digit if it reduces the brightness for all the other class images. Conversely, The Gabor filter



(a) Without Normalization



(b) With Normalization

Figure 5.1: Receptive field of the pair-wise STDP with an all-to-all spike interaction (arrangement of weights) without Gabor filter.

can also decrease the percentage of images that cannot be decoded successfully even without using the input normalization feature (figure 4.5b). The following results increase the credibility of the pixel area hypothesis and its synergy with the adaptative threshold for modifying the response of the images after training.

At the beginning of the discussion, it was stated that the preprocessing stage can alter the intrinsic properties of the network, but in what way? The input normalization affects the network indirectly as keeps a homogeneous firing rate for every input image, with the adaptative threshold developing through time. Still, the normalization does not change any inner parameters of the excitatory or inhibitory layer of the network. The Gabor filter also modifies the pixels that do not encode information to highlight a specific orientation of the image, but this does not meddle in reality with the excitatory neurons' response. The synergy between the input preprocesses and the intrinsic plasticity alters the network behavior. Further work is needed to a complete overview (advantages and disavantages) of the adaptative threshold as a viable option for the stability of the network and the development a way to measure how much does the shape (or pixel area) of an image interacts with the adaptative threshold.

Chapter 6

Conclusion

The addition of an adaptative threshold has the drawback that the membrane threshold of the neurons on the excitatory layer can be driven to a high voltage level where the inputs with a low amount of spikes would make it difficult to excite the layer. If the input decoding fails, a common approach was to increase the input pixel value and simulate the network again until at least a single spike was generated within the excitatory layer.

In this work, it was prevented the high computational cost of simulating the whole network repeatedly by normalizing the inputs to constrain a homogeneous spike count for all the class labels. Still, a successful image decoded does not imply that it would be correctly classified. The input normalization produced a slight increase in the model accuracy (2%) and was observed to prevent the overfitting of the model, as shown in figure 4.5. The oscillations from the models without norm can be cut off when the input normalization is active. We can see a more stable behavior of how the learning of the model progresses up to its stable regime.

On the other hand, the Gabor filter also demonstrates a reduction in the percentage of images that do not respond on the excitatory layer, even in the absence of the input norm feature. As it was tested the different feature configurations with varying rules of learning (pair and triplet STDP), no substantial differences were observed in the final accuracy, except that the Gabor filter significantly influences the overall final accuracy of the model. A reason for equal results on the model accuracy on every learning rule could be related to the small number of neurons used to encode the information (100 neurons). Some authors (Diehl and Cook, 2015) have shown that

increasing the excitatory layer over 400 neurons can make it easier to visualize the limitations for each learning rule and their respective spike interaction.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Ahokainen, I. (2024). Unsupervised image classification with spiking neural networks. *Tempere University - Faculty of EngineeringNatural Sciences*.

Azzopardi, G., & Petkov, N. (2012). A corf computational model of a simple cell that relies on lgn input outperforms the gabor function model. *Biological Cybernetics*, *106*(3), 177–189.

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, *18*(24), 10464–10472.

Chen, Y. (2017). Mechanisms of winner-take-all and group selection in neuronal spiking networks. *Frontiers in Computational Neuroscience*, *11*.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT Press.

Deng, L., Wu, Y., Hu, X., Liang, L., Ding, Y., Li, G., Zhao, G., Li, P., & Xie, Y. (2020). Rethinking the performance comparison between snns and anns. *Neural Networks*, *121*, 294–307.

Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, *9*.

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal dynamics: From single neuron to networks and models of cognition*. Cambridge University Press.

Gong, X.-Y., Su, H., Xu, D., Zhang, X.-T., Shen, F., & Yang, H.-B. (2018). An overview of contour detection approaches. *International Journal of Automation and Computing*, *15*, 656–672.

Goodhill, G. J., & Barrow, H. G. (1994). The role of weight normalization in competitive learning. *Neural Computation*, *6*(2), 255–269.

Grossberg, S. (2013). Recurrent neural networks [Accessed on Feb 28, 2024]. http://scholarpedia.org/article/Recurrent_neural_network

Hebb, D. O. (1949). *The organization of behavior*. Wiley Sons.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.*, *117*(4), 500–544.

Hubel, D., & Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, *160*(1), 106–154.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.

Izhikevich, E. M., & Desai, N. S. (2003). Relating stdp to bcm. *Neural Computation*, *15*, 1511–1523.

Izhikevich, E. M., Gally, J. A., & Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, *14*(8), 933–944.

Kandel, E., Schwartz, J., Jessell, T., Siegelbaum, S., & Hudspeth, A. (2012). *Principles of neural science* (5 Ed). McGraw-Hill.

Kaski, S., & Kohonen, T. (1994). Winner-take-all networks for physiological models of competitive learning. *Neural Networks*, *7*(6-7), 973–984.

Kulkarni, S. R., Parsa, M., Mitchell, J. P., & Schuman, C. D. (2021). Benchmarking the performance of neuromorphic and spiking neural network simulators. *Neurocomputing*, *447*, 145–160.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, *3*, 111–132.

Löwel, S., & Singer, W. (1992). Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science*, *255*(5041), 209–212.

Lynch, N., Musco, C., & Parter, M. (2019). Winner-take-all computation in spiking neural networks. *arXiv*.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

Miljkovic, D. (2017). Brief review of self-organizing maps. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1061–1066.

Moreno, P., Bernardino, A., & Santos-Victor, J. (2005). Gabor parameter selection for local feature detection. *2nd Iberian Conference on Pattern Recognition and Image Analysis*.

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, *98*(6), 459–478.

Nunes, J. D., Carvalho, M., Carneiro, D., & Cardoso, J. S. (2022). Spiking neural networks: A survey. *IEEE Access*, *10*, 60739–60764.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pfister, J.-P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *The Journal of Neuroscience*, *26*(38), 9673–9682.

Qu, L., Wang, L., Tian, S., Kang, Z., Li, S., & Xu, W. (2019). A novel homeostasis method to improve the learning efficiency of spiking neural networks. *2019 IEEE International Workshop on Future Computing (IWOFC)*.

Querlioz, D., Bichler, O., Dollfus, P., & Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology*, *12*(3), 288–295.

Rouag, F. E., Terki, N., & Dahmani, H. (2023). Photometric visual servoing using gabor features. *NeuroQuantology*, *21*(6), 1193–1200.

Ruslim, M. A., Burkitt, A. N., & Lian, Y. (2023). Learning spatio-temporal v1 cells from diverse lgn inputs [https://doi.org/10.1101/2023.11.30.569354].

Sanchez-Garcia, M., Chauhan, T., Cottereau, B. R., & Beyeler, M. (2023). Efficient multi-scale representation of visual objects using a biologically plausible spike-latency code and winner-take-all inhibition. *Biological Cybernetics*, *117*(1-2), 95–111.

Sjostrom, P. J., Turrigiano, G. G., & Nelson, S. B. (2001). Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, *32*(6), 1149–1164.

Sjöström, J., & Gerstner, W. (2010). Spike-timing dependent plasticity [Accessed on Feb 28, 2024]. http://www.scholarpedia.org/article/Spike-timing_dependent_plasticity

Skatchkovsky, N., Simeone, O., & Jang, H. (2021). Learning to time-decode in spiking neural networks through the information bottleneck. *35th Conference on Neural Information Processing Systems*.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926.

Stimberg, M., Brette, R., & Goodman, D. F. (2019). Brian 2, an intuitive and efficient neural simulator (F. K. Skinner, Ed.). *eLife*, *8*, e47314. https://doi.org/10.7554/eLife.47314

Toga, A. W., & Mazziotta, J. C. (2002). *Brain mapping: The methods*. Academic Press.

Vemuru, K. V. (2020). Image edge detector with gabor type filters using a spiking neural network of biologically inspired neurons. *Algorithms*, *13*(7).

Vreeken, J. (2003). Spiking neural networks, an introduction. *Technical Report*. https://api.semanticscholar.org/CorpusID:14100503

Watt, A. J., & Desai, N. S. (2010). Homeostatic plasticity and stdp: Keeping a neuron's cool in a fluctuating world. *Frontiers in Synaptic Neuroscience*, *2*.

Xie, X., Hahnloser, R. H. R., & Seung, H. S. (2000). Learning winner-take-all competition between groups of neurons in lateral inhibitory. *Advances in Neural Information Processing Systems*, *13*.

Yamazaki, K., Vo-Ho, V.-K., & Bulsara, D. (2022). Spiking neural networks and their applications: A review. *Brain Sciences*.

Yi, J. D., & Arisaka, K. (2020). Lateral phase differences in a population model of the visual cortex are sufficient for the development of rhythmic spatial sampling. *bioRxiv*.

Zhang, W., & Linden, D. J. (2003). The other side of the engram: Experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, *4*(11), 885–900.

Zhaoping, L. (2005). *The primary visual cortex creates a bottom-up saliency map*. Elsevier.

Appendix A

Simulation Workflow

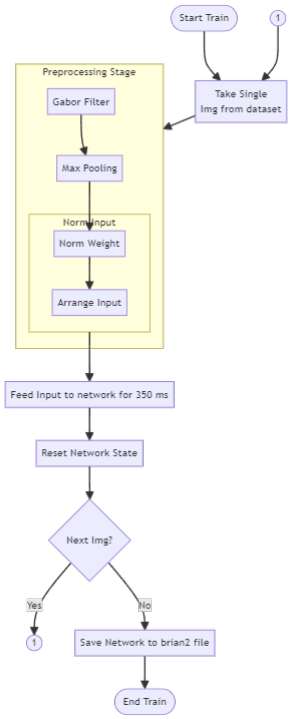


Figure A.1: Training phase algorithm for each image processed within the network. The workflow is divided into the input preprocessing and the network state evaluation. A loop is performed to iterate between the inputs until the number of epochs within the specified amount of images is reached.

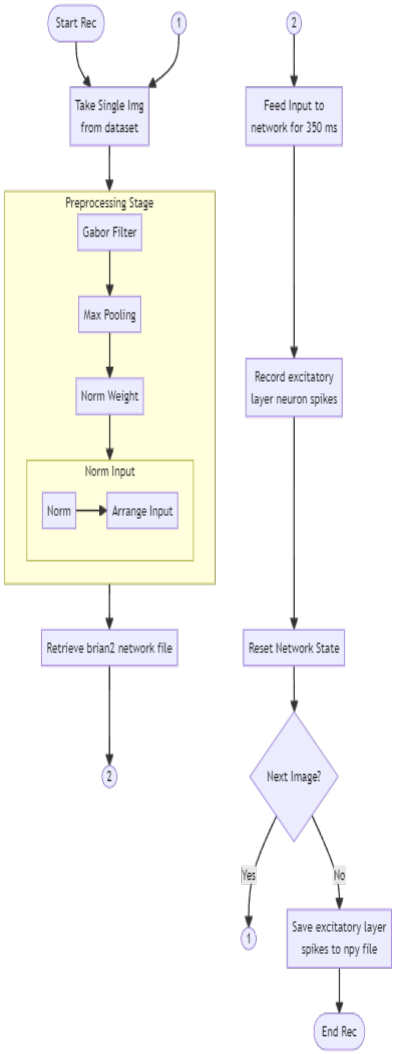


Figure A.2: Spike recording workflow for model performance validation. The algorithm is almost identical to the training phase's algorithm without the feature that, to decode the output after the training, we need to record the spikes for every image of both the training and testing datasets. The network state (after training) is loaded every time a new input is fed to the model.

A.1 Training Function

```
    # ====================== Load MNIST Dataset ========================
    (X_train, y_train), (X_test, y_test) = mnist.load_data()
    X_train = X_train / 4.
    X_test = X_test / 4.

    # ========================== Model ===================================
    seed(init_params['Random_Seed'])
    Mdl = WTA(Net_setup=Net_init)
    if init_params['Run_train']:
        it_counter = 0
        Clean_TempFolder(Flush=True)
        Mdl.Init_State()
        print("================= # TRAINING MODEL # ==================")
        X_pre = Mdl.preProcess(X_data=X_train[:init_params['Train_dt']], preInp=
    init_params['Gabor_filter'])

        for ep in range(init_params['Epoch']):
            for idx in tqdm(range(len(X_pre)), desc='Loading ' + str(ep + 1)):
                it_counter += 1
                Mdl.Norm_SynW(Norm_w=True)

                Mdl.RunModel(X_single=X_pre[idx], preInp=init_params['Gabor_filter'
    ], norm=init_params['Norm'], phase='Stimulus')
                Mdl.RunModel(phase='Resting')

                if it_counter <= 5000:
                    if it_counter % 1000 == 0:
                        temp_ep = 'Temp_It_' + str(it_counter)
                        Mdl.net.store(temp_ep, 'Temp/' + temp_ep + '.b2')
                else:
                    if it_counter % 5000 == 0:
                        temp_ep = 'Temp_It_' + str(it_counter)
                        Mdl.net.store(temp_ep, 'Temp/' + temp_ep + '.b2')
        Mdl.net.store(init_params['Filename'],'Trained_Models/' + init_params['
    Filename'] + '.b2')
        np.save('Temp/Homeo/V_thr', Mdl.get_HomeoThr())
    else:
```

```
        if init_params['Load_Temp'] == True: Mdl_addr = 'Temp/'
        else: Mdl_addr = 'Trained_Models/'
        Mdl.net.restore(init_params['Filename'],Mdl_addr + init_params['Filename'] +
 '.b2')
```

A.2 Recording Phase

```
def get_Spikes(X_data:np.ndarray, init_params:dict, Net_params:dict, presen_stg:
str='Train', Run_trial:bool=False):

seed(init_params['Random_Seed'])

if presen_stg == 'Train':
    dir_data = 'Activity/Train/'
    state_title = 'MAPPING'
elif presen_stg == 'Test':
    dir_data = 'Activity/Test/'
    state_title = 'VALIDATING'

if init_params['Load_Temp'] == True: addr_load = 'Temp/'
else: addr_load = 'Trained_Models/'

if Run_trial:
    print("================== # "+ state_title +" MODEL # ==================")
    Mdl = WTA(Net_setup=Net_params)
    Mdl.Init_State()
    X_pre = Mdl.preProcess(X_data=X_data, preInp=init_params['Gabor_filter'])
    Train_Sp, Input_Sp = [], []
    for idx in tqdm(range(len(X_pre)), desc='Validating'):
        Mdl.net.restore(init_params['Filename'], filename=addr_load +
init_params['Filename'] + '.b2')
        # Rate Monitor for Counting Spikes
        mon = SpikeMonitor(Mdl.net['Exc'], name='CountSp')
        Mdl.net.add(mon)
        if presen_stg == 'Test':
            monInp = SpikeMonitor(Mdl.net['Input'], name='Count_InpSp')
            Mdl.net.add(monInp)

```

```
            Mdl.Norm_SynW(Norm_w=True)
            Mdl.RunModel(X_single=X_pre[idx], preInp=init_params['Gabor_filter'],
    norm=init_params['Norm'], phase='Stimulus')

            Train_Sp.append(np.array(mon.count, dtype=np.int8))
            if presen_stg == 'Test': Input_Sp.append(np.array(monInp.count, dtype=np
    .int8))

            Mdl.RunModel(phase='Resting')
            Mdl.net.remove(Mdl.net['CountSp'])
            if presen_stg == 'Test': Mdl.net.remove(Mdl.net['Count_InpSp'])
        np.save(dir_data + init_params['Filename'] + '_' + str(len(X_data)),
    Train_Sp)
        if presen_stg == 'Test': np.save('Activity/Test/Input/Poisson_Count_' + str(
    len(X_data)), Input_Sp)
    else:
        Train_Sp = np.load(dir_data + init_params['Filename'] + '_' + str(len(X_data
    )) + '.npy')
        print('Loaded Excitatory Spike Data Shape: ' + str(np.array(Train_Sp).shape)
    )

        if presen_stg == 'Test':
            Input_Sp = np.load('Activity/Test/Input/Poisson_Count_' + str(len(X_data
    )) + '.npy')
            print('Loaded Input Count Data Shape: ' + str(np.array(Input_Sp).shape))
    if presen_stg == 'Test':
        return Train_Sp, Input_Sp
    else:
        return Train_Sp
```

A.3 Repository

The network connection and its settings can be checked out at the following link: https://github.com/PentaCarlos/SNN_WTA.

Appendix B

Model Parameters

The "Leaky-integrate-and-fire" model is defined by the equation (2.1). The parameters for the LIF model for both excitatory and inhibitory neurons are shown in table B.1.

-	Excitatory	Inhibitory
V_{rest}	$-60mV$	$-60mV$
V_{reset}	$-65mV$	$-45mV$
V_{thresh}	$-52mV$	$-40mV$
V_{rev}	$0mV$	$-100mV$
τ_{mem}	$100ms$	$10ms$
τ_{ge}	$1ms$	$1ms$
τ_{gi}	$2ms$	$2ms$
τ_{theta}	$0.6e7ms$	-
V_{homeo}	$0.05mV$	-

Table B.1: Parameters for the LIF neuronal model. From V_{rest} to τ_{gi} model the neuronal behavior for firing an action potential, while the last two parameters (τ_{theta} and V_{homeo}) described the intrinsic plasticity rule for a homogeneous constrain on the firing rate for the excitatory neurons only.

The learning parameters for pairSTDP (all-to-all), pairSTDP (nearest-neighbor), and Triplet-STDP can be appreciated in the table B.2.

-	pairSTDP (all-to-all)	pairSTDP (NN)	TripletSTDP
A_{pre}	0.1	-	-
A_{post}	-0.105	-	-
τ_{pre}	$20ms$	$20ms$	$20ms$
τ_{post}	$20ms$	$20ms$	$20ms$
τ_{post2}	-	-	$40ms$
pre_{rate}	0.001	0.0001	0.01
$post_{rate}$	0.1	0.01	0.1
G_{max}	1.0	1.0	1.0

Table B.2: Parameters for the STDP-based learning rules. The hyperparameters (learning rates) were adjusted to match the specific best behavior of each learning rule.