Optimization of Neuronal Morphologies for Pattern Recognition

Giseli de Sousa

A thesis submitted in partial fulfilment of the requirements of the University of Hertfordshire for the degree of Doctor of Philosophy

This research was carried out in the department of Computer Science, University of Hertfordshire

December 2011

Abstract

This thesis addresses the problem of how the dendritic structure and other morphological properties of the neuron can determine its pattern recognition performance.

The techniques used in this work for generating dendritic trees with different morphologies included the following three methods. Firstly, dendritic trees were produced by exhaustively generating every possible morphology. Where this was not possible due to the size of morphological space, I sampled systematically from the possible morphologies. Lastly, dendritic trees were evolved using an evolutionary algorithm, which varied existing morphologies using selection, mutation and crossover. From these trees, I constructed full compartmental conductance-based models of neurons. I then assessed the performance of the resulting neuronal models by quantifying their ability to discriminate between learned and novel input patterns. The morphologies generated were tested in the presence and absence of active conductances.

The results have shown that the morphology does have a considerable effect on pattern recognition performance. In fact, neurons with a small mean depth of their dendritic tree are the best pattern recognizers. Moreover, the performance of neurons is anti-correlated with mean depth. Interestingly, the symmetry of the neuronal morphology does not correlate with performance.

This research has also revealed that the evolutionary algorithm could find effective morphologies for both passive models and models with active conductances. In the active model, there was a considerable change in the performance of the original population of neurons, which largely resulted from changes in the morphological parameters such as dendritic compartmental length and tapering. However, no single parameter setting guaranteed good neuronal performance; in three separate runs of the evolutionary algorithm, different sets of well performing parameters were found. In fact, the evolved neurons performed at least five times better than the original hand-tuned neurons. In summary, the combination of morphological parameters plays a key role in determining the performance of neurons in the pattern recognition task and the right combination produces very well performing neurons.

Acknowledgements

I would like to thank my principal supervisor Volker Steuber who believed I could accomplish this project. My supervisors Reinoud Maex, Rod Adams and Neil Davey for the availability and patience during our usual Wednesday meetings. My officemates Faisal Rezwan, Karen Safaryan, Nicolas Oros and David Gray for the productive discussions during our tea breaks. Also, my officemate Johannes Luthman for our long discussions during commuting time. A special thanks go to my friends Thiago Matos, who helped a lot, specially in the final phase of this journey; and Burak Erdeniz, for our crazy conversations. I cannot forget my friends from AADE school, specially Georgij and Marija Shmelin, Behrad Vahedi and George Haritos, as we spent nice moments together on many occasions. And also a special thanks to my friend Claudia Pisac, who was always there to listen to me in every moment I needed.

I would like to thank my Brazilian friends, Marcelo Carlomagno and Vania Zanelli, Tulio Salvaro and Giani Hoffman, and Fausto Vetter, who supported me during this journey. Many thanks to my Italian friend, Eleonora Rossi, who followed this process since the beginning, when we first met at the English language course in Cambridge. I am also thankful to my former colleagues from Labtrans, in special to Fernanda Miranda, Helio Rodak and Manolo Caramez, who supported me when I decided to get a PhD. Moreover, I cannot forget my current officemates from L3C, in special my former supervisor Mauro Roisenberg, who introduced me to this area and allowed me to write up at his lab. I am very thankful for it.

I am also very grateful to my parents, who always supported me to finish this journey, even though they missed me due to our long physical distance. My sister Rafaela, my brother-in-law Henrique and my nephew Henriquinho, thanks for the good moments we spent together during my visits to Brazil. The same for my in-laws who always sent me positive thoughts. And last but not least, my special thanks to my husband Jean, who pushed me to start this journey and supported me during these four years, never let me give up about anything. Thanks my love, for all this.

I know this will be the most read page of my thesis, and probably the only one for many people. But if you have time, take a look at the rest of it as I spent a lot of time and effort in preparing this nice piece of work. So, please enjoy!

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
CF	Climbing fibre
CV2	coefficient of variation
EPSP	excitatory postsynaptic potential
GABA	gamma aminobutyric acid
Glu	glutamate
IPSP	inhibitory postsynaptic potential
ISI	interspike-interval
LTD	Long-term depression
LTP	Long-term potentiation
PC	Purkinje cell
PF	Parallel fibre
PSC	postsynaptic current
PSP	postsynaptic potential
RC	resistor-capacitor
s/n	signal-to-noise

Chapter 1

Introduction

1.1 Motivation and Goals

The brain contains many different types of neurons with a large number of different morphologies. One of the main questions in neuroscience is the role these characteristic morphologies play in determining neuronal function. Previously, it has been shown that the morphology of a neuron can affect its firing pattern [42, 76, 75]. Specifically, some morphologies tend to favour bursting patterns, while others produce high firing frequencies, with these functional changes coming about by only changing the dendritic topology of the neuron. These studies have shown that the neuronal morphology is an important factor in modulating firing behaviour.

Previous work on associative memory in cerebellar Purkinje cells, a type of neuron that has been implicated in motor control and motor learning, has suggested that the generation of burst-pause sequences is important for information storage in the cerebellum [66, 33]. It was found that Purkinje cells use a novel neural code, where information about learned patterns is represented by the length of silent periods, and not by the number or the exact timing of individual spikes, as has been classically assumed. These previous results have important implications for the coding of information in the brain, but they are specific to one particular neuron with a very specialised morphology. Given that neuronal morphologies affect the ability of neurons to generate burst-pause sequences, they should also affect their ability to act as associative memory devices.

The aim of this research is to characterise the implications of neuronal morphology for associative memory and pattern recognition. It is assumed that associative memory is based on changing synaptic weights, initially in this thesis focusing on Long-Term Depression (LTD), observed at synapses between parallel fibres and Purkinje cells in the cerebellum, and later on Long-Term Potentiation (LTP), a common form of synaptic plasticity that occurs for example at synapses between CA3 and CA1 pyramidal cells in the hippocampus. Based on this assumption, I investi-

gated the importance of dendritic structure and other morphological properties of the neuron for its pattern recognition performance. To do this, I took three different approaches: two involved generating dendritic trees systematically and one evolved trees by using an evolutionary algorithm (EA). The first two algorithms were designed to produce trees with a certain number of terminal points, one exhaustively, by covering the whole range of morphologies; and the second one selectively, to produce trees by sampling from the range chosen. This second approach was the required method when the number of terminal points was large, so that the exhaustive method could not be applied. The third approach, evolving trees using an EA, was inspired by biological evolution theory [48, 5], where large sets of dendritic morphologies could be evolved by producing variations of existing morphologies, initially generated at random, using biologically inspired mechanisms such as mutation and crossover. Using these three approaches, I could generate a large number of dendritic trees to analyse which morphological parameters affected the pattern recognition performance.

The work presented here has tried to answer the following questions:

- How does the morphology of a neuron affect to the storage and recall of memories?
- Which morphological parameters such as compartmental length and tapering, determine the performance of the neuron in the pattern recognition task?
- Can we evolve effective neuronal morphologies to perform pattern recognition in the presence and absence of active conductances?
- Which measurable characteristics of neuronal morphologies, such as asymmetry index or mean depth, can predict the pattern recognition performance of a neuron?

1.2 Contribution to Knowledge

The following contributions to the field of computational neuroscience were achieved during my PhD:

- In both active and passive neuronal models, I found an almost linear correlation between the mean depth of the dendritic trees and the pattern recognition performance; the best performing neurons were the ones with the smallest mean depth. A similar result was found for the variance of depth, mean path length and mean electronic path length, which correlated with the neuronal performance in some experiments. Interestingly, the asymmetry index did not correlate with the performance for the full range of tree morphologies. In fact, any

morphology with an asymmetry index below 0.5 performed as well as the most symmetric one.

- The values of neuronal parameters play a major role in determining the performance of neurons in the pattern recognition task. In particular, the values of the morphological parameters, dendritic compartmental length and tapering, affected the ability of the model neurons to be good pattern recognizers. However, no single parameter setting guaranteed good neuronal performance; in three separate runs of the evolutionary algorithm, different sets of well performing parameters were found.

- In my search for optimal neuronal morphologies for pattern recognition, I investigated different optimisation procedures that could alter different neuronal features such as dendritic topologies and morphological parameters. Following a survey of existing algorithms, I developed my own algorithms to generate dendritic morphologies either in a systematic way or by evolving them using an Evolutionary Algorithm (EA). My EA was designed to meet six requirements that were not met entirely by existing algorithms. As a result, I could generate neuronal morphologies that allowed me to understand some of the neuronal features that are important for pattern recognition.

- The final EA could evolve effective morphologies in the presence of active conductances, with a pattern recognition performance that was five times better than that of hand-tuned neurons.

Moreover in a preliminary investigation, I studied the effects of synaptic plasticity in a Purkinje cell model. In these studies, I found that the best pattern recognition performance in the model resulted from LTD that saturated at a lower bound value of zero, which corresponds to silencing the PF synapses completely. On the other hand, the ability of the model to discriminate between learned and novel input patterns was unaffected by the presence of inhibitory plasticity for a wide range of parameter values.

1.3 Thesis Outline

This thesis is organized as followed:

Chapter 2 gives a review of neural systems, focusing on the structures involved in the process of pattern recognition. It explains how biological neurons are represented by computational models, where each dendritic segment can be described by mathematical equations using cable or compartmental models. Two examples of compartmental models are presented, a

cerebellar Purkinje cell model and a hippocampal pyramidal cell model, which served as a basis for my studies of pattern recognition in neuronal models with different morphologies.

Chapter 3 describes the concept of learning and memory based on different types of synaptic plasticity. Two types of synaptic plasticity are explained, long-term depression and long-term potentiation, which then form the basis of the learning rules used by the models studied in this work. These learning rules are implemented in two types of computational models, artificial neural networks (ANNs) and compartmental models, which are used to analyse the effect of synaptic plasticity on the pattern recognition task.

Chapter 4 presents the results of a study of synaptic plasticity in a compartmental Purkinje cell (PC) model. The effect of biologically realistic synaptic plasticity that saturates at lower bound values is analysed, for both synapses from excitatory parallel fibres to the PC and for synapses between inhibitory interneurons and the PC. The main result of this study is that the pattern recognition performance of the PC model is very sensitive to the lower bound value at which LTD at PF synapses saturates, and that the performance is unaffected by the presence of inhibitory synaptic plasticity. Moreover, I describe an unsuccessful attempt to use a reduced version of the PC model for my studies of pattern recognition.

Chapter 5 introduces a review of previous work on dendritic morphologies. First, I described the generic neuronal models used in my exploration of the best morphologies for pattern recognition. These multi-compartmental models were used as they are relatively simple models and are not based on any real neuronal morphology [76]. Then I present the morphological metrics used to characterise dendritic trees with different topologies. At the end of the chapter, an analysis of existing tree morphology generation algorithms is carried out. The algorithms used by me should meet six requirements to be able to generate all the desired morphologies necessary to fully investigate the pattern recognition performance. As none of the existing algorithms met all of the requirements, a new tree generation algorithm was implemented using the best features of the existing algorithms, which is described in the next chapter.

Chapter 6 describes the algorithms used to generate neuronal morphologies for the pattern recognition task, focusing on the dendritic tree structure. Three algorithms are described, two to generate dendritic trees systematically and one to evolve trees by using an EA. The systematic tree generation algorithms were implemented to generate trees exhaustively, covering the whole range of dendritic trees with a certain number of terminal points, and selectively, by generating samples of trees from a larger range of morphologies, where the number of

terminal points was very large. The third algorithm was implemented to evolve dendritic tree structures and morphological parameters, meeting the six requirements specified in the previous chapter. The algorithm evolved new individuals by applying genetic operations like mutation and crossover, and its fitness function was based on neuronal performance by calculating a signal-to-noise ratio over the neuronal response from stored and novel patterns. The aim of each EA run was to improve neuronal performance by evolving neurons with an increasing fitness. The results of these three algorithms are presented in the next two chapters.

Chapter 7 presents the results of the evaluation of the effect of neuronal morphologies on pattern recognition using passive neuronal models. The three algorithms described in the last chapter were used to generate the dendritic trees, and their performance was measured by calculating a signal-to-noise ratio over their EPSP responses to the stored and novel pattern presentations. The performance of these trees was tested in four different experiments: comparing the two most distinct morphologies, comparing the evolved trees generated by the EA, and comparing the trees generated systematically by exhaustive and selective searches. The results show that morphology has a major impact on pattern recognition performance. Moreover, I found that the mean depth correlated well with performance. Another relevant point found was that the EA could evolve morphologies that performed well for both a specific or a variable set of patterns.

Chapter 8 presents the results of the study of pattern recognition in active neuronal models. The same trees generated in Chapter 6 were tested, but the neurons now included a set of active conductances taken from the simplified models described in Chapter 5. The EA was used to evolve not only dendritic trees as in the previous chapter, but also parameters related to morphology and input patterns. The performance of the active models was measured by calculating a signal-to-noise ratio over the number of spikes generated in response to stored and novel patterns. A similar set of experiments as in the previous chapter tested the performance of two distinct morphologies, trees generated selectively and finally, trees evolved using the EA. The results for the evolved trees were analysed in three ways: by comparing the neuronal morphology and its performance, by comparing the neuronal parameters evolved and the resulting neuronal performance, and by a sensitivity analysis over the parameters evolved. From these results, I can conclude that the EA could effectively generate active morphologies where the morphological parameters, dendritic compartmental length and tapering, appear to be most important parameters for the pattern recognition task. Moreover,

the mean depth of the trees was shown to be as good a predictor of performance in active models as well as in passive models.

Chapter 9 gives a summary of the main contributions of in this work. It also discusses some research not dealt with in this work, but which could represent suitable extensions to my work, such as an optimisation of ionic conductances and input patterns, which could be studied in the future. At the end, I present a list of publications that I have presented at conferences during these four years of research.

Chapter 2

Neural Systems

2.1 Introduction

Neurons are cells responsible for learning and storing information in the brain. In my PhD work I tried to understand the influence of neuronal morphologies – or in other words, the sizes and shapes neurons can have – on their ability to store and recall input patterns. Thus, two relevant points are described in this chapter. First, the morphological properties of neurons, which are related to their shape, structure and branching pattern. Second, the way neurons transmit information, focusing on the type of neuronal signals used in this work: synaptic potentials and action potentials. Both points are detailed in Section 2.2.

As this study uses neuronal models to simulate the activity of neurons during pattern recognition, an overview about how biological data are translated into computational models is given. In Section 2.3, I present the details about how mathematical equations are used to represent the morphology of a neuron, which can be modelled by cable models or compartmental models. As this work uses only compartmental models, the details about this model are presented in Section 2.3.2. At the end of this chapter, two examples of compartmental models are presented: a CA1 pyramidal cell and a cerebellar Purkinje cell. Both models were used to understand how neurons can perform the pattern recognition tasks, which is the subject of the next chapter.

2.2 Neuronal Structure and Function

2.2.1 Neuronal Morphology

Neurons consist of four main morphological parts: soma, dendrites, axons and axon terminals. The soma or cell body contains the nucleus where the genetic information is stored. Dendrites are the elements where the neurons receive most of the inputs. From the dendrites, the signals are passed

on through the soma and axon to the axon terminals, where they are transmitted by synapses to the next cell (Figure 2.1). Most neurons have many dendritic branches which form their dendritic trees, but only one axon, although this axon usually also gives rise to many axon collaterals and terminals.

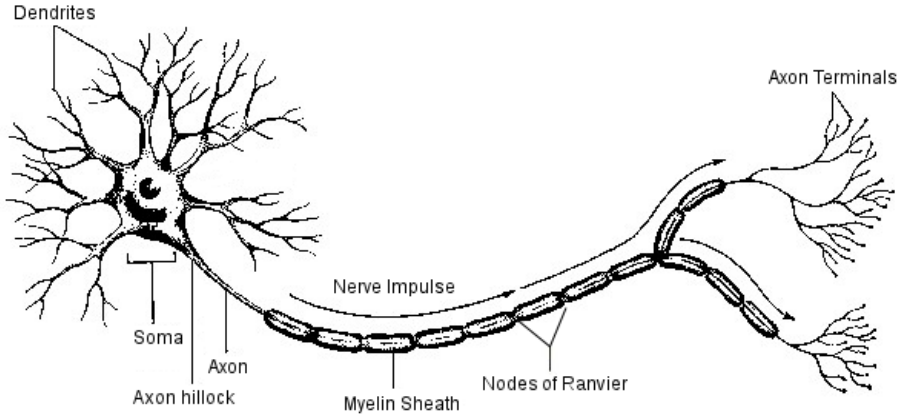


Figure 2.1 – Typical structure of a motor neuron. Modified from inside.salve.edu/~walsh/neuron.jpg.

Figure 2.1 shows a classical neuronal morphology (of a motor neuron), where the main parts described previously can be easily identified. However, there is a lot of variability: the human brain contains a total of 10^{11} neurons and thousands of different types of neurons with different morphologies [34]. Figure 2.2 shows examples of different neuronal morphologies, highlighting that the main differences are found in their dendritic trees. Thus, this present study of neuronal morphologies focuses mainly on the properties and structures of the dendritic trees; more details are given in the next section.

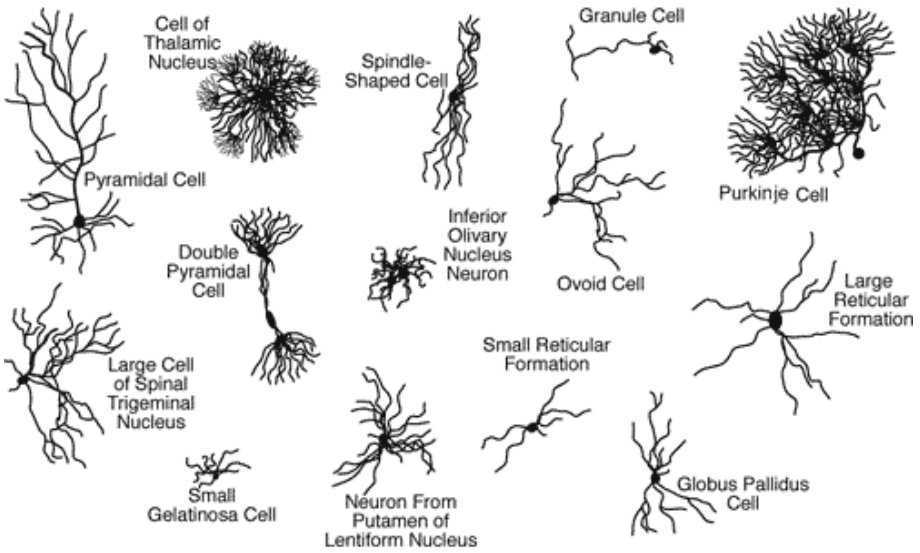


Figure 2.2 – Examples of neuronal morphologies. Based on the drawings by Ramón y Cajal. From Stufflebeam [68].

2.2.1.1 Dendritic Trees

One interesting question related to the function role of neuronal morphologies is why different neurons exhibit distinct dendritic trees, like the ones shown in Figure 2.2. It is known that dendrites play an important role in synaptic integration, integrating synaptic input often through dendritic spines [19, 41, 11, 83]. A second role of dendritic trees is information processing, determining the propagation of synaptic potentials, the backpropagation of action potentials and the induction of synaptic plasticity, which is responsible for associative memory (explained later in Chapter 3). Due to the importance of dendritic trees for transmitting information, this research focuses mainly on dendritic trees.

The studies of Ramón y Cajal allow us to understand some of the dendritic properties and branching structures. Morphologically speaking, most dendrites are very complex trees with a large number of bifurcations and ramifications. According to Segev and London, the number of dendritic trees per neuron can vary between 1 and 16, and the total number of terminal points can vary even more: from 10 to 400 dendritic tips per neuron [58]. To give an example of these extremes, cerebellar Purkinje cells have on average only one main tree with approximately 400 terminal points, whereas α-motoneurons from the cat spinal cord can have 8 to 12 dendritic trees with an average of 30 terminal points for each tree [7, 58]. In terms of length and diameter, dendrites can also have a large variation. Dendrites are thin tubes which vary their diameter as they get far from the soma: usually dendrites close to the soma have 1 to 6 μm in diameter, and distal dendrites vary from 0.3 to 1 μm [58]. The dendritic length can also vary from very short trees with 100 μm, such as the spiny stellate cell in the mammalian cortex, to longer ones, such as the spinal α-motoneurons with a total dendritic length between 1 and 2 mm [7]. The total dendritic length on average lies between 1 and 10 mm [58].

Some researchers proposed that these dendritic dimensions and branching structures are optimised to minimize the cost of transporting information from synapses to soma [11, 83]; other studies relate the dendritic topology, or the connectivity pattern of the dendritic segments, to the neuronal firing pattern [42, 16, 36, 76]. To understand the relationship between neuronal morphology and function, the next section presents an overview of the mechanisms underlying neuronal signalling.

2.2.2 Neuronal Function

A feature of neurons that is central to integration and transmission of their signals is their ability to generate and modulate membrane potentials. The membrane potential, also called membrane

voltage, is caused by the difference in the electrical potentials across the cell membrane. This potential difference is controlled by ion channels, which are membrane proteins that work as gates allowing ions to cross the membrane. These gates are selective for ions such as sodium (Na^+) and potassium (K^+) by using the size and charge of the ions and their interaction with water [34]. Hence, the difference in concentration of these ions between the inner and outer part of the cell generates the membrane potential.

Neuronal voltage signals can be divided into two types: synaptic potentials and action potentials (Figure 2.3). Both types of potentials are used by neurons to transmit information to other neurons, but the main difference between them is given by the distance the signal can cover. Synaptic potentials are used by neurons to transmit signals over the short distance from the synapses to the soma and initial segment of the axon, while action potentials can travel longer distances as they are larger and, most importantly, self-regenerating. Both types of potentials are considered in this work: synaptic potentials in the study of pattern recognition in passive models (Chapter 7) and action potentials in the active models (Chapter 8); therefore more details about both potentials are given in the next sections.

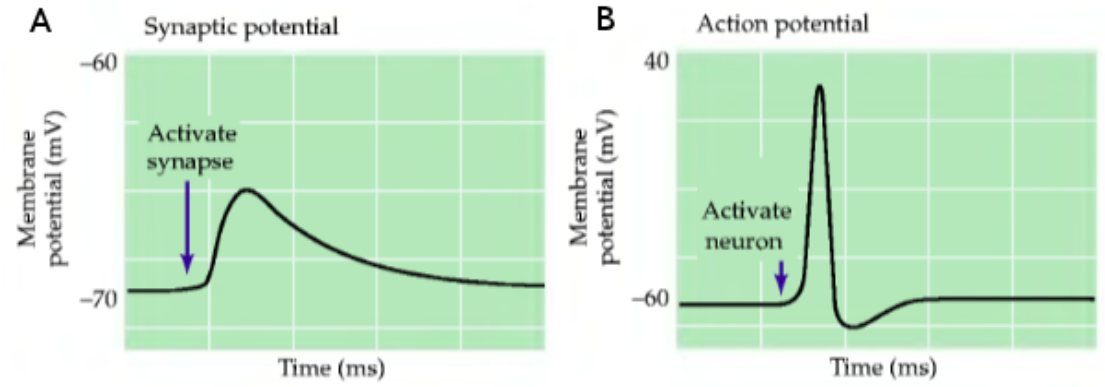


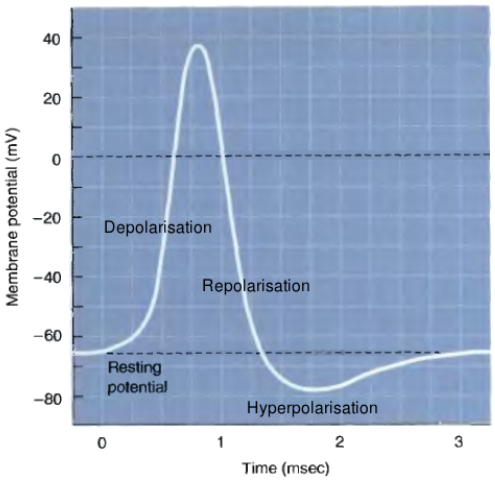
Figure 2.3 – Neuronal electrical signals: synaptic potential (A) and action potential (B). Note the different scales of the y-axes. Modified from Purves el al. [54].

2.2.2.1 Action Potentials

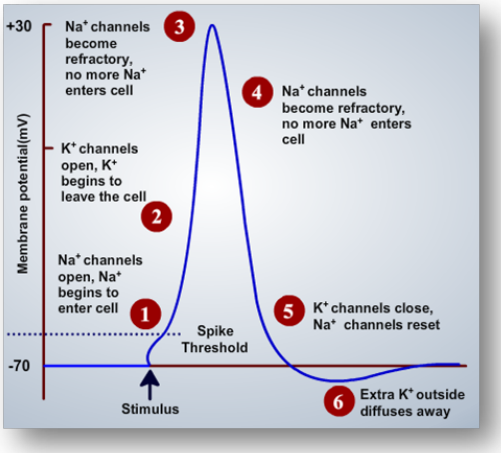
Action potentials are used by neurons to transmit information; their frequency and temporal pattern are assumed to represent information under different conditions and in different brain systems. Action potentials are characterised by a rapid increase in the membrane potential called depolarization, followed by slightly a less sharp decrease called repolarisation. In the final stages of the action potential, the membrane potential reaches a level lower than the resting potential, which is called hyperpolarisation, before reaching the resting potential phase once again (phases show in Figure 2.4a). Action potentials are initiated only if the membrane potential reaches a

certain degree of depolarisation, which is called threshold, otherwise the signal is classified as a subthreshold postsynaptic response (explained in the next section). Action potentials are usually generated in the segment of the axon that borders the soma, called axon hillock, and pass through the axon before arriving at the axon terminals, which then transmit chemical signals to the cells downstream via synaptic contacts (see neuronal parts in Figure 2.1) .

The rapid change in membrane potential during the upstroke of an action potential is caused by the inner part of cell becoming more positively charged than the outer part [6]. This difference is determined by the different concentration of $Na^+$and K^+ ions. The flux of these ions across the membrane is regulated by special types of channels called voltage-gated ion channels. The relative permeability of these channels in each phase of the action potential is shown in Figure 2.4b. In the first phase, the neuron is in the resting potential where the membrane is more permeable to K^+ than Na^+, which makes the membrane potential negative as it approaches the equilibrium potential for K^+. Then, after the action potential is initiated, for example by an arrival of a synaptic potential, the membrane becomes more permeable to Na^+. As a consequence, the membrane potential becomes more positive and moves towards the equilibrium potential of Na^+. This membrane depolarization results in the opening of additional (delayed rectifier) K^+ channels, and the Na^+ permeability is transient, both of which lead to a repolarisation and hyperpolarisation of the neuronal membrane back towards the K^+ equilibrium potential.



(a) Phases of the action potential. Modified from Bear et al. [6].



(b) Relative membrane ionic permeability during an action potential. From http://amrita.vlab.co.in/?sub=3&brch=212&sim=742&cnt=1.

Figure 2.4 – Action potential.

2.2.2.2 Synaptic Transmission

Synapses are the functional connections between neurons, which allow information to be transmitted from one neuron to the next. Each neuron forms on average 1000 synaptic connections with other neurons, but this number is highly variable and can go up to 200,000 like in the Purkinje cells [34]. Synapses can be divided into two types of synapses: electrical and chemical. As electrical synapses are less common and not used in this work, I will restrict the explanation about synapses to the chemical ones only.

In chemical synapses, the action potential which arrives at the presynaptic terminal activates Ca^{2+} channels, which causes Ca^{2+} influx into the terminal. This Ca^{2+} influx leads to the release of neurotransmitters, chemical agents present in synaptic vesicles, which then bind to receptors on the postsynaptic cell (see Figure 2.5). The most common types of these receptors are neurotransmitter-gated ion channels; their activation by the neurotransmitters results in a change of the postsynaptic conductance [6, 54]. This change in conductance generates an electrical current, called postsynaptic current (PSC), which consequently changes the membrane potential in the postsynaptic cell, generating a postsynaptic potential (PSP).

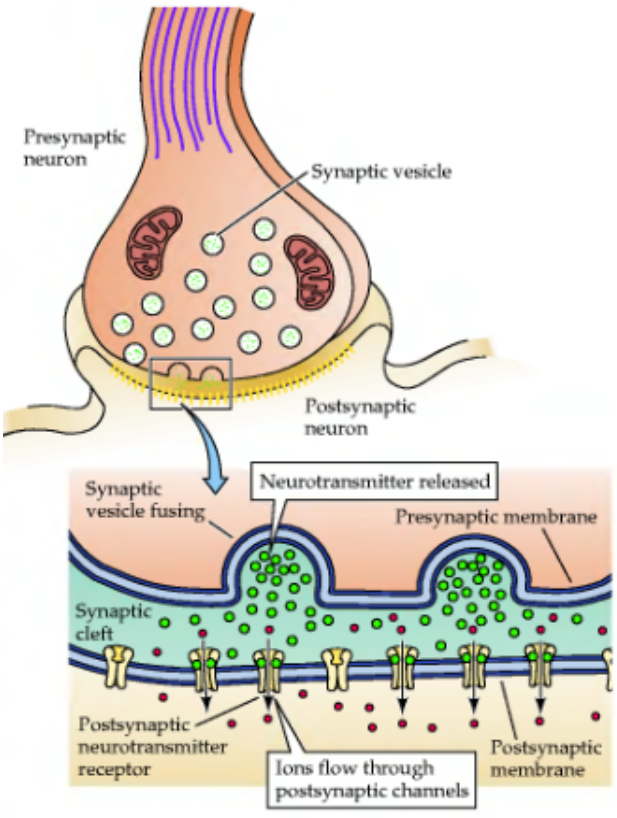


Figure 2.5 – Chemical synapse. From Purves et al. [54]

PSPs are ultimately responsible for triggering action potentials in the postsynaptic cells. How-

ever, depending on the type of PSP, the probability of action potential firing can change: the PSP is considered excitatory (EPSP) if it increases the probability of action potential firing, and the PSP is inhibitory (IPSP) if it decreases this probability. The generation of an excitatory or inhibitory PSP depends on the type of synapse, or more precisely, on the type of ion channel that is activated in the postsynaptic membrane. For example, glutamate (Glu) is a type of neurotransmitter released by excitatory synapses, which makes the membrane more permeable to Na^+, and consequently it generates EPSPs in the postsynaptic cell. On the other hand, if an inhibitory neurotransmitter, such as gamma aminobutyric acid (GABA), is released in the synapse, it activates receptors to ions such as chloride (Cl^-) which consequently can generate IPSPs.

2.3 Neuronal Models

As described in Section 2.2.2, neurons transmit information using the voltage across their membrane. Based on this fact, neurons are modelled by analysing their electrical properties. The initial step to understand the electrical activity of neurons is to characterise their passive properties. Passive properties are neuronal properties that do not depend on voltage or ligand gated ion channels which affect the membrane resistance; a model is named passive if its membrane resistance does not change [64]. A list of passive properties is presented in Table 2.1, where the typical range of values found in dendritic trees is presented in the fourth column.

Table 2.1 – Passive properties of dendrites. The fourth column shows the typical range of values found for dendritic trees [58].

Property	**Symbol**	**Unit**	**Value Range**
Membrane resistance	R_m	$k\Omega cm^2$	1-100
Axial resistivity	R_a	Ωcm	70–300
Membrane capacitance	C_m	$\mu F/cm^2$	0.5–2
Membrane time constant	τ_m	ms	1–100

To characterise a neuron based on its passive properties, two types of models can be used: cable models and compartmental models (Figure 2.6). Cable models are based on the idea that the dendrites can be represented by an electric cable, if the membrane is passive and uniform. This type of model calculates the voltage across the membrane as a continuous function of the time and distance along the cable. On the other hand, compartmental models represent each small segment of the dendrite by an iso-electric compartment, which is a homogeneous cylinder that can be represented by an equivalent electrical circuit. Compartmental models use differential equations to describe the behaviour of each compartment as well as the interactions between neighbouring

compartments [7]. Both types of models are described in the next sections.

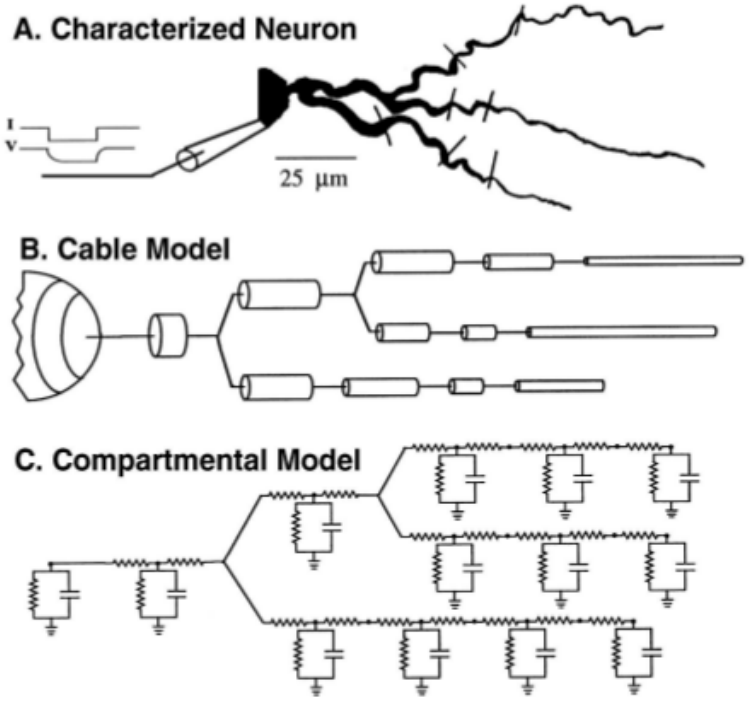


Figure 2.6 – Dendritic models. A. First, the neuronal morphology is reconstructed and the response of the neuron to current injections is used to determine some of its passive properties. Then, the neuronal model can be represented by either a cable model (B) or a compartmental model (C). The cable model represents each dendritic segment as a cable, where the voltage can be measured at any point of the dendritic tree by using the cable equation (Section 2.3.1). In the compartmental model, each segment is represented by a RC circuit and the membrane voltage of each compartment can be computed by using the neuronal passive and input properties. From Bower and Beeman [7].

2.3.1 Cable Model

In 1959, Rall started to apply the cable theory, which had been previously used to model electric current flow through transatlantic cables, to computational models of neurons, where he represented neurons by cylinders of finite length [55]. Cable theory uses mathematical models to describe the flow of electric current in a dendritic tree that receives synaptic inputs at various sites and times [7].

To analyse the spatio-temporal evolution of the membrane potential in a cable model, a specialised equation called the Cable Equation is applied. The cable equation is derived from the combination of the principle of the conservation of electrical charge with Ohm's law [9]. In the cable equation, the membrane potential V_m is calculated as a function of the distance x along the cable, the time t and the injected current I_e at the point x:

$$C_m \frac{\partial V_m}{\partial t} = \frac{E_m - V_m}{R_m} + \frac{d}{4R_a} \frac{\partial^2 V_m}{\partial x^2} + \frac{I_e}{\pi d} \tag{2.1}$$

where the passive properties are given in Table 2.1, V_m is the membrane potential, E_m is the leakage reversal potential, $\partial V/\partial t$ is the partial derivative of the membrane potential, d is the diameter and $d/4R_a \partial^2 V/\partial x^2$ is the density of current flowing along the cable length into the point x.

From the cable equation we can derive the three main passive properties when using cable models [7]:

$$C_m = \pi d l C_M \tag{2.2}$$

$$R_m = \frac{R_M}{\pi d l} \tag{2.3}$$

$$R_a = \frac{4 l R_A}{\pi d^2} \tag{2.4}$$

where l is cable length, d is the diameter, and C_M, R_M and R_A are the specific capacitance, resistance and axial resistance respectively. More details can be found in [55, 7, 9, 64].

2.3.2 Compartmental Model

2.3.2.1 RC Circuit

Compartmental models discretise the Cable Equation by representing each small piece, or segment, of the neuron as an electrical circuit [7]. The generic circuit used in compartmental models is a resistor-capacitor circuit (RC circuit), which is composed of a capacitor and a resistor in parallel, a battery in series with the resistor, and a current source (see equivalent circuit in Figure 2.7).

The circuit shown in Figure 2.7 can represent a membrane segment in passive models. In this circuit, the change of membrane potential V_m at the time t can be calculated by combining Kirchhoff's current and voltage laws [64], which gives the following equation:

$$C_m \frac{dV_m}{dt} + I_{ion} = I_{pulse} \tag{2.5}$$

where the passive properties are given in Table 2.1, I_{pulse} is the externally applied current and I_{ion} is the ionic current flowing across the membrane, which can be expressed as:

$$I_{ion} = I_{leak} = g_{leak}(V_m - E_{leak}) \tag{2.6}$$

where g_{leak} is the resting conductance and E_{leak} is the resting potential.

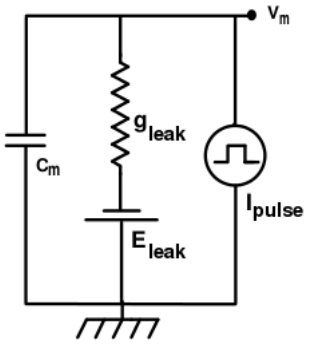


Figure 2.7 – Equivalent RC circuit for the electrical model of a passive membrane. The elements shown are: C_m is the membrane capacitance, g_{leak} is the resting conductance represented by a resistor, E_{leak} is the resting potential represented by a battery, I_{pulse} is the externally applied current, and V_m is the voltage between the cell interior and the cell exterior. Modified from Bower and Beeman [7].

2.3.2.2 Hodgkin-Huxley Model

This model is based on the observations made on a squid axon by Hodgkin and Huxley beginning of the 1950s [27, 24, 23, 25, 26]. Their idea was to model segments of the membrane based on their electrical properties, using an equivalent circuit as shown in Figure 2.8.

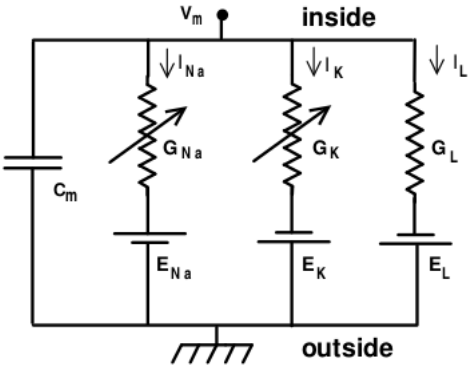


Figure 2.8 – Electrical equivalent circuit proposed by Hodgkin and Huxley. From Bower and Beeman [7].

The change of membrane potential V_m in the Hodgkin-Huxley circuit can be calculated by the Equation 2.5, where the ionic current I_{ion} is composed of three different currents: sodium (I_{Na}), potassium (I_K), and a leakage current (I_L). Each of these ionic currents is determined by a conductance value g and a reversal potential E as shown in the following equations:

$$I_{Na} = g_{Na} m^3 h \left(V_m - E_{Na}\right) \tag{2.7}$$

$$I_K = g_K n^4 \left(V_m - E_K\right) \tag{2.8}$$

$$I_L = g_L \left(V_m - E_L\right) \tag{2.9}$$

where m and n are the activation variables and h the inactivation variable, which change according to the following equation:

$$\frac{dx}{dt} = \alpha_x \left(V_m\right)\left(1 - x\right) - \beta_x \left(V_m\right) x \tag{2.10}$$

where x is the type of the gate (m, n or h) and α, β are voltage-dependent rate constants. Hence, the total ionic current I_{ion} can be expressed as:

$$I_{ion} = g_{Na} m^3 h \left(V_m - E_{Na}\right) + g_K n^4 \left(V_m - E_K\right) + g_L \left(V_m - E_L\right) \tag{2.11}$$

2.3.2.3 Modelling Synapses

As shown in Section 2.2.2.2, presynaptic terminals release neurotransmitters which bind to postsynaptic receptors; for ionotropic receptors this leads directly to the opening of ion channels. The opening of these neurotransmitter receptor ion channels results in a conductance change that produces a postsynaptic current (PSC), as previously explained in Section 2.2.2.2. This current can be expressed as:

$$I_s(t) = g_s(t) \left(V_m - E_s\right) \tag{2.12}$$

where g_s is the synaptic conductance at the time t, V_m is the voltage across the membrane, and E_s is the reversal potential of the synaptic channels [7].

The changes in the postsynaptic conductance can be represented by simple functions of time t when the postsynaptic signal arrives. Three types of functions that are commonly used to represent the conductance changes are single exponential decay, alpha function, and dual exponential. Each of these waveforms is shown in Figure 2.9.

In this work I use a synaptic model with a dual exponential function, as it is replicates the conductance changes of typical synapses well [64]. The dual exponential function is governed by two time constants, a rise time constant τ_1 and a decay time constant τ_2 , and can be expressed as:

$$g_s(t) = \overline{g}_s \frac{\tau_1 \tau_2}{\tau_2 - \tau_1} \left(e^{-\frac{t}{\tau_2}} - e^{-\frac{t}{\tau_1}}\right) \tag{2.13}$$

where $\overline{g}_s$ is the synaptic peak conductance at the time of the most recent event.

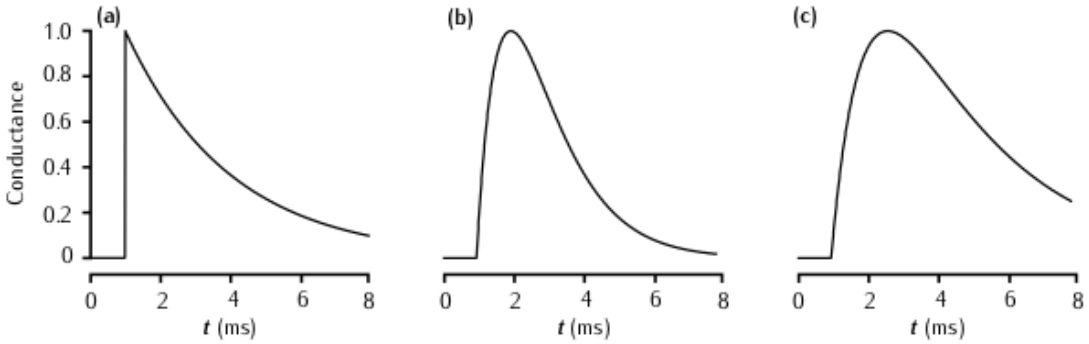


Figure 2.9 – Synaptic conductances changes. Three types of synaptic conductance waveforms are presented here: single exponential decay (a), alpha function (b), dual exponential (c). The respective time constant of each waveform is given as (a) $\tau = 3ms$, (b) $\tau = 1ms$, and (c) $\tau_1 = 3ms$ and $\tau_2 = 1ms$. From Sterratt et al. [64].

2.3.3 Examples of Multi-Compartmental Models

Here I present two examples of multi-compartmental models which were used in previous studies of pattern recognition: a hippocampal CA1 pyramidal neuron model [18] and a cerebellar Purkinje cell model [66]. The first model was used as basis for my studies of pattern recognition in passive and active neurons; the results are presented in Chapters 7 and 8. The second model was used to analyse the effects of different forms of synaptic plasticity on pattern recognition in the cerebellum (results given in Chapter 4). Both models are presented in the next sections.

2.3.3.1 Pyramidal cell

This multi-compartmental model was proposed by Graham [18] and it was based on the reconstruction of a rat hippocampal CA1 pyramidal cell made by Major et al. [43]. The model has 890 compartments; the dendritic tree was divided into three areas (see Figure 2.10). The default model was passive, which means no voltage gated ion channels were present in the dendrites and soma, and the response was given by EPSPs. However, some experiments included four types of voltage-gated ion channels (Na^+, Ca^{2+}, KA and h-type), which were used to amplify the EPSP responses.

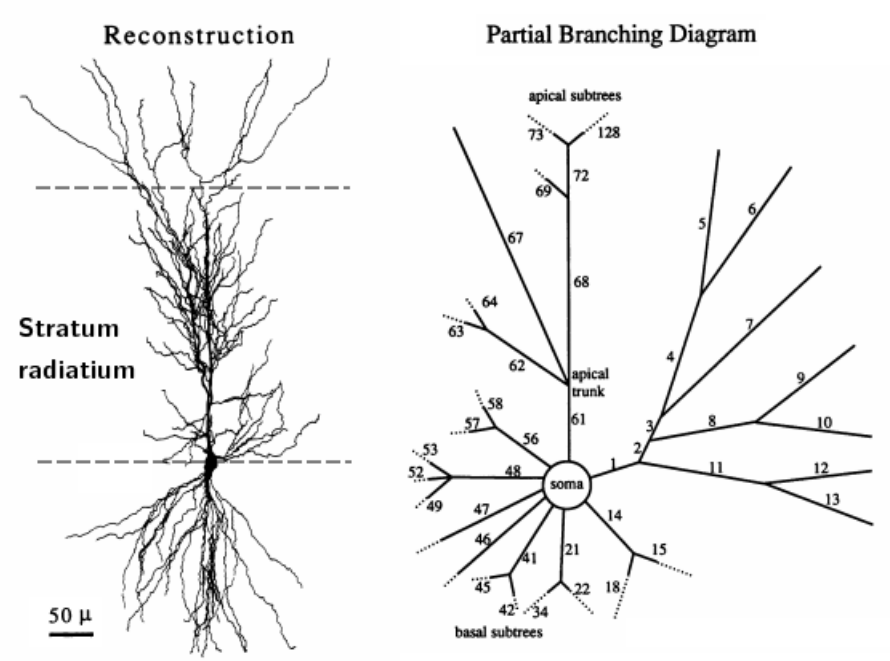
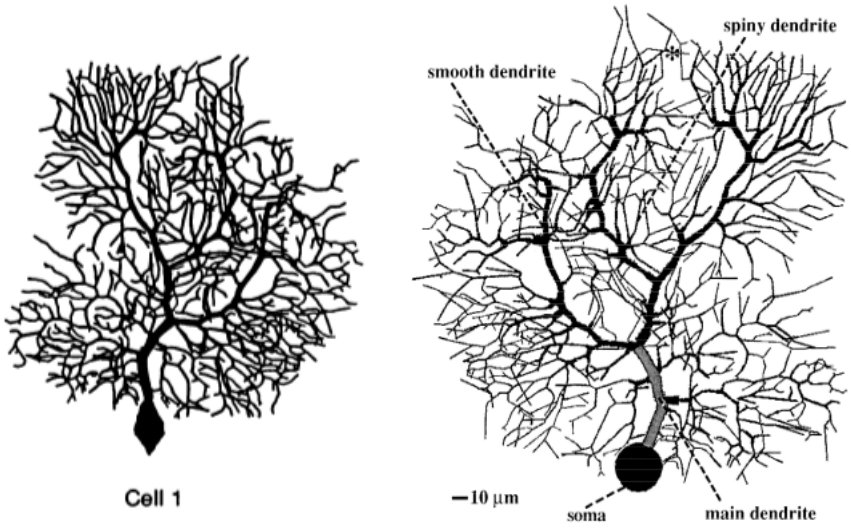


Figure 2.10 – CA1 pyramidal cell compartmental model. On the left, the reconstruction of a rat hippocampal CA1 pyramidal cell is shown. On the right, the branching diagram of the same cell is given, showing part of the dendritic tree branching from the soma. Modified from Major et al. [43].

More details about this neuronal model are presented in Section 3.3.2.2, which details my work on pattern recognition using this model.

2.3.3.2 Purkinje cell

This multi-compartmental model of a cerebellar Purkinje cell was proposed by De Schutter and Bower [13, 14], based on a reconstruction and a passive model of a guinea-pig Purkinje cell by Rapp et al. [56]. It is composed of 4550 compartments and 147,400 dendritic spines. Ten different types of voltage-dependent channels were modelled using Hodgkin-Huxley-like equations. The soma compartment had a fast and persistent Na^+ conductance, a delayed rectifier, a transient A-type K^+ conductance, a non-inactivating M-type K^+ conductance, an anomalous rectifier and a low-threshold T-type Ca^{2+} conductance. The dendritic compartments contained a Purkinje-cell specific high-threshold P-type and a low-threshold T-type Ca^{2+} conductance, two different types of Ca^{2+}-activated K^+ (KCa) conductances and an M-type K^+ conductance. Figure 2.11 shows the morphology of the reconstructed Purkinje cell (a) and the compartmental model (b), where the soma and three types of dendrites are identified (main, smooth and spiny). Each of these four regions contains a different set of ion channels. Details about the study of pattern recognition in this model are presented in Section 3.3.2.1.



(a) Reconstruction of a cerebellar Purkinje cell. From Rapp et al. [56].

(b) Multi-compartmental Purkinje cell model. From De Schutter and Bower [13].

Figure 2.11 – Cerebellar Purkinje cell compartmental model.

2.4 Conclusion

In this chapter, I presented the biological foundations of this work. The neuron was the core element presented here, which was described in terms of morphology and function. The description of neuronal morphology concentrated on the dendritic tree, which is the focus of this work as it is the structure with the most diverse morphology and the place where neurons receive information from other cells. Moreover, this discussion presented how neurons encode information by integrating and transmitting signals through their membrane. This information is then used by neurons to perform the pattern recognition task which is explained in details in the next chapter.

Another topic described in this chapter was how neurons are computationally modelled, where electrical circuits represent each segment of the neuronal membrane, and mathematical equations are used to describe the behaviour of these circuits. I also showed how ion channels and synapses can be represented in computational models, as they play a key role in neuronal function. At the end, I presented two relevant computational models which I used to understand the effects of synaptic plasticity (presented later in Chapter 4) and which I used as basis of my work on passive and active models, presented in the later chapters (Chapters 7 and 8).

Chapter 3

Review of Synaptic Plasticity and Pattern Recognition

3.1 Introduction

My first goal in this research was to understand the mechanisms that underlie learning and memory. As I based my research on pattern recognition in two neuronal models, a cerebellar Purkinje cell and a hippocampal pyramidal cell, my initial studies were focused on the theories which are fundamental for the learning process in these two cells: the Marr-Albus-Ito theory of cerebellar learning and the theory of Hebbian learning. As memory is known to be encoded using long-lasting modifications of synaptic strength, both of these learning theories use long-term synaptic plasticity as their learning mechanism. For the cerebellar Purkinje cells, the learning process is assumed to be based on a process called long-term depression (LTD); whereas hippocampal pyramidal cells are assumed to use long-term potentiation (LTP) to learn their input patterns. These two types of synaptic plasticity are presented in more detail in Section 3.2. After understanding how these synaptic mechanisms occur, I then present the pattern recognition models which are used in this research. Pattern recognition is studied in two types of computational models: artificial neural networks (Section 3.3.1) and compartmental models (Section 3.3.2). For each type of model, I present two examples of how the pattern recognition task can be implemented, using different learning rules reflecting the types of synaptic plasticity previously mentioned. For example, in the compartmental models, I explain pattern recognition in a cerebellar Purkinje cell model, which uses an LTD learning rule (explained in Section 3.3.1.1) and in a CA1 pyramidal cell model, which uses an LTP learning rule (explained in Section 3.3.1.2). The results obtained for each of these models are presented in the next chapter.

3.2 Synaptic Plasticity

Synaptic plasticity is a change of synaptic strength that happens after repeated synaptic stimulation or by pairing specific pre and postsynaptic activations [60]. The change in plasticity at a synapse can remain for milliseconds to a few seconds, called short-term synaptic plasticity; or for minutes to hours or even days, which is then called long-term plasticity. This research focuses on long-term synaptic plasticity, which is the form of plasticity that is assumed to be the basis of learning and memory. There are two main forms of long-term plasticity: long-term depression and long-term potentiation. These two types of plasticity are assumed to be responsible for the learning process in cerebellar Purkinje cells and hippocampal pyramidal cells, respectively, which are the cells used in this study of pattern recognition (presented later in Section 3.3.2). Hence, these two types of synaptic plasticity are detailed in the next sections.

3.2.1 Long-term Depression in the Cerebellum

Although many different types of cerebellar plasticity have been characterised, the type of cerebellar plasticity that has received by far the most attention is long-term depression (LTD) at the synapses between parallel fibres (PFs) and Purkinje cells (PCs). PF LTD, which is often called cerebellar LTD, has been implicated in motor learning, which is responsible for the rapidity, smoothness and precision of movements [74]. PF LTD is an associative process in which the strength of a PF synapse onto a PC is depressed when the PF is activated together with climbing fibre (CF) input to the PC [32, 31] (see Figure 3.1A, B). This synaptic weakening occurs due to the loss of AMPA receptors in the postsynaptic membrane, which reduces the PC response to the PF activation, as shown in Figure 3.1C.

3.2.2 Long-term Potentiation in the Hippocampus

Long-term potentiation (LTP) is a type of synaptic plasticity which can occur when both presynaptic and postsynaptic cells are activated at same time. The result of this process is a strengthened synapse between the pre and postsynaptic cells, which is assumed to be the basis of information storage by the postsynaptic cell. LTP occurs in many synapses in different brain areas such as hippocampus, cortex, amygdala, and cerebellum (although the mechanism of LTP induction in the cerebellum is different) [54].

In the hippocampus, three synaptic sites have been identified where LTP has been observed (Figure 3.2A): in the dentate gyrus, between the perforant path and granule cells; in area CA3, between the mossy fibres from dentate granule cells and CA3 pyramidal cells; and in area CA1,

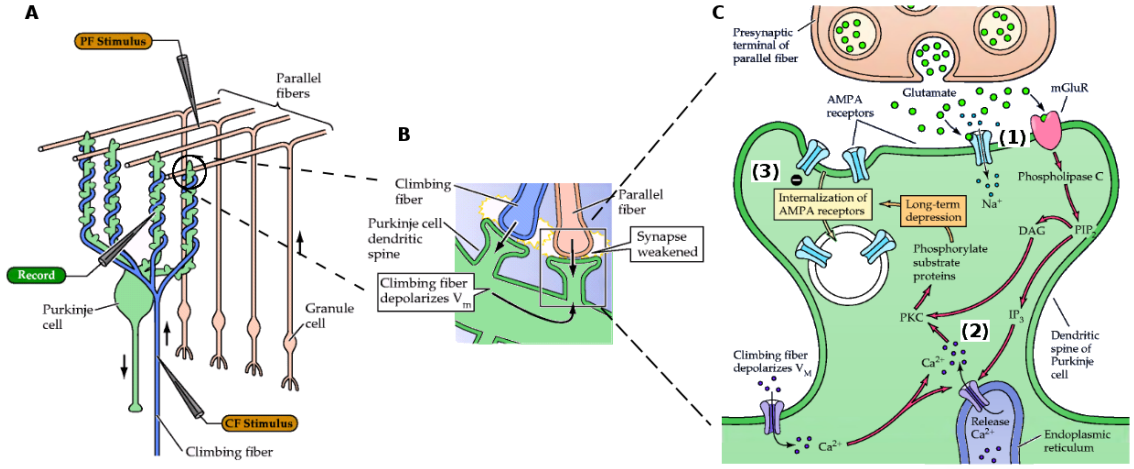


Figure 3.1 – Cerebellar (PF) LTD. A. LTD is found in the cerebellum at the synapses between parallel fibres (PFs) and Purkinje cells (PCs). B. LTD weakens the synapses between PFs and PCs after coincident PF and climbing fibre (CF) activation. C. The mechanism which controls LTD: 1. glutamate neurotransmitter released by the PFs binds to AMPA receptors and metabotropic glutamate receptors (mGluRs) in the PC, which results in the production of the second messenger DAG in the PC. 2. The activation of the CF results in depolarization of the PC and influx of Ca^{2+} into the PC. Ca^{2+} and DAG activate the protein kinase PKC. 3. The phosphorylation of AMPA receptors by PKC results in their internalisation, which decreases subsequent PC responses to PF activation. Modified from Purves et al. [54].

between Schaffer collaterals from CA3 pyramidal cells and CA1 pyramidal cells. This research is based on a study which focuses on the Schaffer collateral pathway, where pattern recognition in CA1 pyramidal cells has been studied previously [18].

LTP at the synaptic connection between CA3 and CA1 pyramidal cells is an associative process, where the synapses are strengthened when both cells are stimulated at the same time, which can occur for example when the Schaffer collateral axons of the CA3 pyramidal cells receive a brief high frequency stimulus train [54] (see Figure 3.2B and C). This kind of Hebbian synaptic plasticity at the CA3-CA1 synapses is assumed to implement a heteroassociative memory, which enables patterns of activity in CA3 to be associated with patterns of activity in CA1.

3.3 Pattern Recognition

In this study, pattern recognition means the process where a neuron, represented by a computational model, recognises particular sets of patterns. The patterns here are represented by a number of random binary inputs, which are learned by two different models: an artificial neural network (ANN) and a neuronal compartmental model. These two models were used in order to compare the pattern recognition performance between the two models, as was done previously in studies of pattern recognition in a hippocampal CA1 pyramidal [18] and a cerebellar Purkinje cell [65, 66]. The steps used to implement pattern recognition in each model are detailed in the next sections.

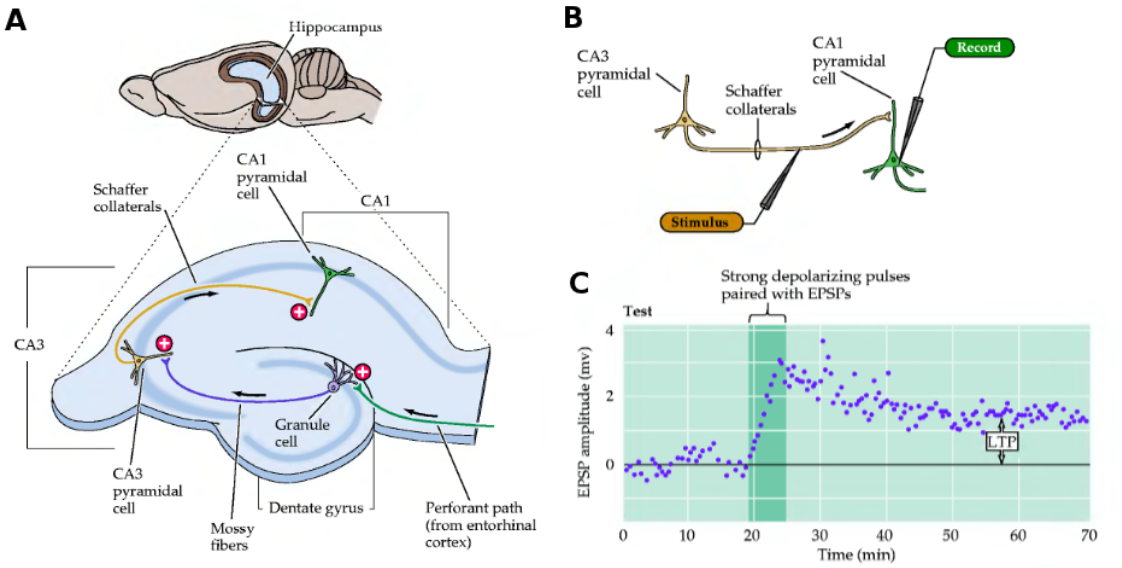


Figure 3.2 – Hippocampal LTP. A. There are three synaptic pathways where LTP is observed in the hippocampus: 1. perforant path, from entorhinal cortex to the granule cells in the dentate gyrus; 2. mossy fibers, from granule cells in the dentate gyrus to CA3 pyramidal cells; 3. Schaffer collaterals, from CA3 pyramidal cells to CA1 pyramidal cells. B. When CA3 and CA1 pyramidal cells are activated simultaneously by a strong signal (for example a strong electrical stimulus to the Schaffer collateral axons of the CA3 cells), LTP occurs at the synapses between CA3 and CA1 pyramidal cells. C. LTP increases the response of CA1 cells (their EPSP amplitude) to Schaffer collateral stimulation. Modified from Purves [54].

The results of the experiments described here are to be found in Chapter 4.

3.3.1 Pattern Recognition in Artificial Neural Networks

The ANN used in this work is a modified version of an associative net with feed-forward connections between its inputs and output [84]. A number of patterns are learned by the ANN model in the learning stage, these are called stored patterns. The new synaptic weights are then transferred to the synapses in the compartmental model, explained in Section 3.3.2. In the recall stage the same stored patterns are presented to both the ANN and the compartmental model together with an equal number of new, unseen patterns, called novel patterns. The synaptic weights of the ANN are modified depending on the learning rule that is used. In my study of the effect of synaptic plasticity in cerebellar PCs, presented in Chapter 4, I used an LTD learning rule based on cerebellar LTD explained in Section 3.2.1. For my studies of the effect of dendritic morphologies on pattern recognition (Chapter 6), which were based on a previous study on pattern recognition in pyramidal cells, I used an LTP learning rule that was based on hippocampal LTP, explained in Section 3.2.2. Both learning rules are presented in the next sections.

3.3.1.1 LTD Learning Rule

This learning rule was implemented based on previous studies of pattern recognition in cerebellar Purkinje cells [65, 66]. As explained in Section 3.2.1, Purkinje cells can discriminate activity patterns presented by their afferent parallel fibres based on the depression of their synaptic strengths when a coincident activation of parallel fibres and climbing fibre occurs. It is important to note that the PC model recognizes patterns by decreasing its output, in contrast to what is done by traditional ANN learning rules which increase the neuronal output for learned patterns. Thus, the LTD learning in the ANN was implemented as a simplified rule where the synaptic strength was decreased when patterns were stored. In the original LTD learning rule, proposed by Steuber and De Schutter [65], the weights of activated synapses were decreased by half each time a pattern was learned, which represented the AMPA receptor conductance being depressed by 50% during the learning process. However, consecutive applications of this learning rule could result in very small synaptic weights, while experiments with LTD induction in cerebellar slices show that the mean AMPA receptor conductances are hardly depressed to less than 50% of the pre-induction baseline [82, 49]. Therefore, I decided to investigate the effect of different degrees of synaptic plasticity on learning and recall in the ANN and PC model. In this study, I used a set of LTD saturation or lower bound values, which varied from 0 to 80% of the pre-learning baseline. This meant that the synaptic weight was decreased by the LTD factor chosen only when the first active input was stored in a particular synapse, instead of for each active input as used in the original learning rule, so that the weight could never fall below this saturation or lower bound value. In Figure 3.3, an example of the new LTD learning rule is presented, where an LTD factor of 0.5 was used, representing synaptic saturation at a lower bound of 50%. The results of this study are presented in the next chapter (Section 4.2).

Using the LTD learning rule, pattern recognition in the ANN was implemented in two phases: learning and recall. In the learning phase, the weight of all the active synapses are adjusted; the network learns each pattern by decreasing the synaptic weight by a specific LTD factor (in Figure 3.3, the LTP factor chosen was 0.5). In the recall phase, the network response is calculated by summing all the synaptic weights associated with active inputs (or in other words, by calculating the inner or dot product of input and weight vector). As shown for the recall phase in Figure 3.3, the stored patterns result in lower responses than novel patterns (middle and right graphs respectively).

The discrimination between the stored and novel patterns was evaluated by calculating a signal-to-noise (s/n) ratio [12]:

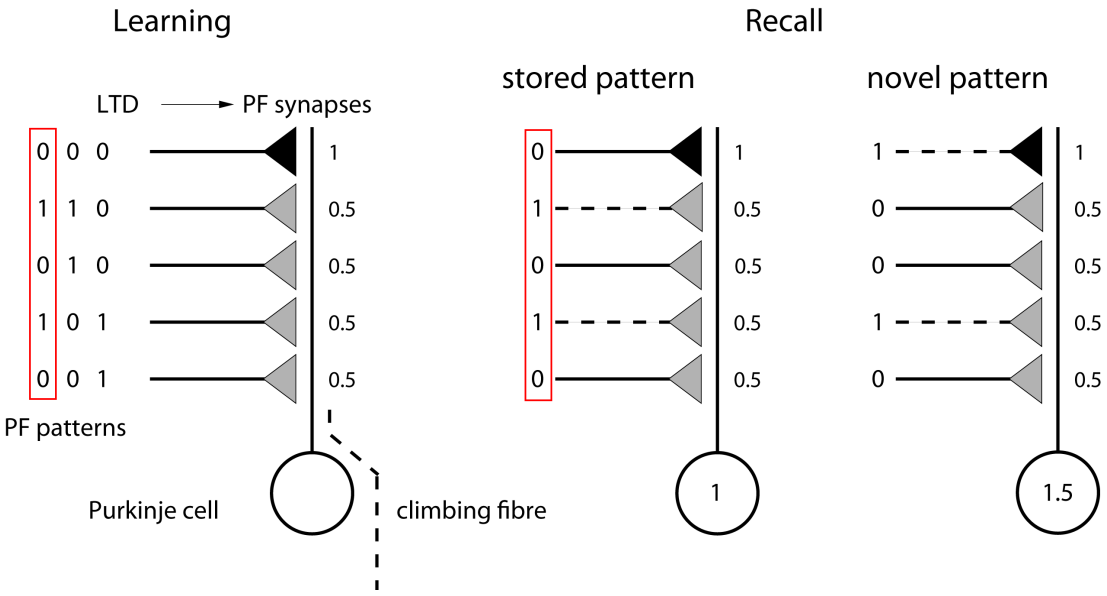


Figure 3.3 – LTD learning rule. Learning phase: three patterns are stored by decreasing the synaptic weight by a factor of 0.5 at the first active input and only at the first active input. Recall phase: the network response is calculated by the inner product of the input and weight vector. The response to a stored pattern (graph shown in the middle) is compared to the response to a novel pattern (right graph). The resulting responses show a lower output value for the stored pattern (1) when compared to the novel pattern (1.5). Note that a low output is found for stored patterns whereas a high output is found for novel patterns, in contrast to the traditional LTP based method used by most ANNs.

$$s/n = \frac{(\mu_s - \mu_n)^2}{0.5\,(\sigma_s^2 + \sigma_n^2)} \tag{3.1}$$

where μ_s and μ_n represent the mean values and σ_s^2 and σ_n^2 the variances of the responses to stored and novel patterns, respectively.

3.3.1.2 LTP Learning Rule

In contrast to the PC model, the pyramidal cell produces high responses for learned patterns, while lower response are found for novel patterns. As previously explained in Section 3.2.2, CA1 pyramidal cells store information through a type of synaptic plasticity called long-term potentiation (LTP). This type of plasticity increases the synaptic strength between the pre and postsynaptic neuron when they are both active at the same time. Given this associative process, learning in the pyramidal cells appears to be based on the Hebbian learning rule. This rule, formulated by Donald Hebb in 1949, postulates that an increase in synaptic strength is caused by coincident pre and postsynaptic neuronal activity [17, 63].

Based on Hebb's theory, the LTP learning rule used in this work was implemented by incrementing the synaptic weight, which were originally set to zero, by adding a value of one every time when both pre and postsynaptic cells were activated. Differently from the learning rule implemented

by Graham in his work on pattern recognition in CA1 pyramidal cells [18], which uses a clipped learning with an upper bound value of one, the learning rule implemented here does not have an upper bound value. Using this Hebbian learning rule, the pattern recognition was implemented in two phases, learning and recall, as for the LTD learning rule (described in Section 3.3.1.1). In the learning phase, the synaptic weights of all active synapses are adjusted, increasing the synaptic weight by a value of one for each active input (see Figure 3.4 - Learning). In the recall phase, the network output is again given by the sum of all synaptic weights associated with active inputs. This results in a higher response for the stored patterns than for novel ones (see bottom values in Figure 3.4 - Recall).

The discrimination between stored and novel patterns is evaluated as for the LTD learning rule, that is a s/n ratio is used (see Equation 3.1).

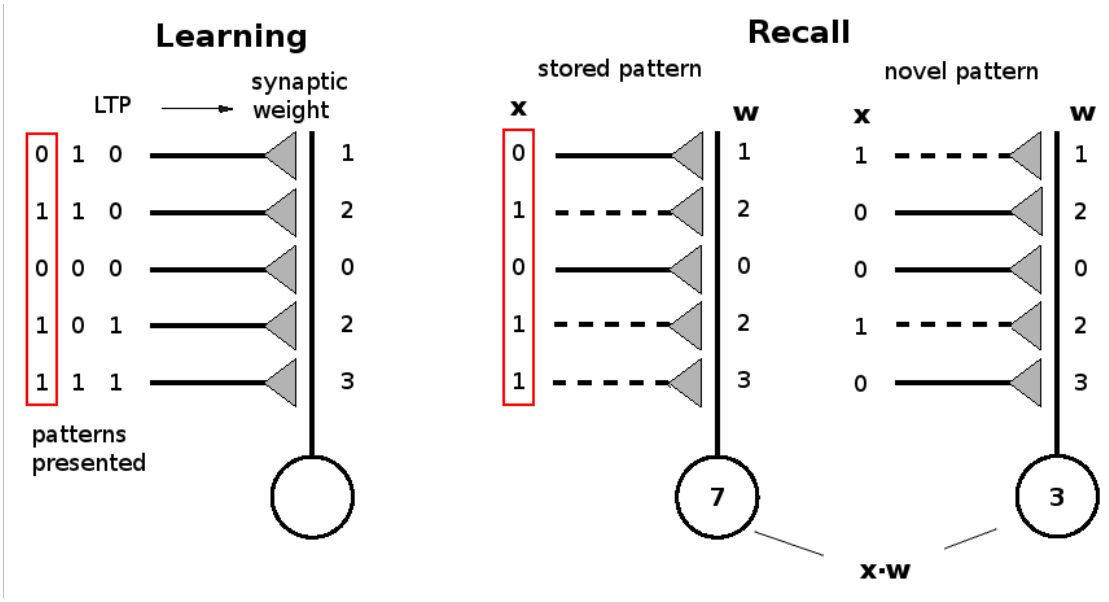


Figure 3.4 – LTP learning rule. Left side: during the learning phase, three patterns are stored by increasing the synaptic weight by a value of one for each active input. Right side: during the recall phase, the response to a stored pattern (indicated by a red box) is compared to the response to a novel pattern. Responses are calculated as the dot product of input vector **x** and weight vector **w**. The resulting responses show a higher output value for stored pattern (7) when compared to novel pattern (3).

3.3.2 Pattern Recognition in Compartmental Models

3.3.2.1 Cerebellar Purkinje Cell

The classical cerebellar learning theory proposed by Marr [45], Albus [2] and Ito [30] suggests that motor learning is based on the association of particular patterns of PF inputs and PC outputs. This theory suggests that a PC can learn to discriminate between different activity patterns presented

by its thousands of afferent PFs based on cerebellar LTD. As explained in Section 3.2.1, cerebellar LTD occurs at the synapses between PFs and PCs, where a teaching signal is given by a CF input. The learning process is accomplished by decreasing the strength of the PF synapses activated when the CF input is given. As a result of this PF LTD, the PC firing rate has been assumed to be reduced in response to presentation of a learned PF pattern, and thus the PC is expected to exert less inhibition on the deep cerebellar nuclei (see circuitry in Figure 3.5). As a consequence of this process, the cerebellar output would be increased, so implementing the motor learning.

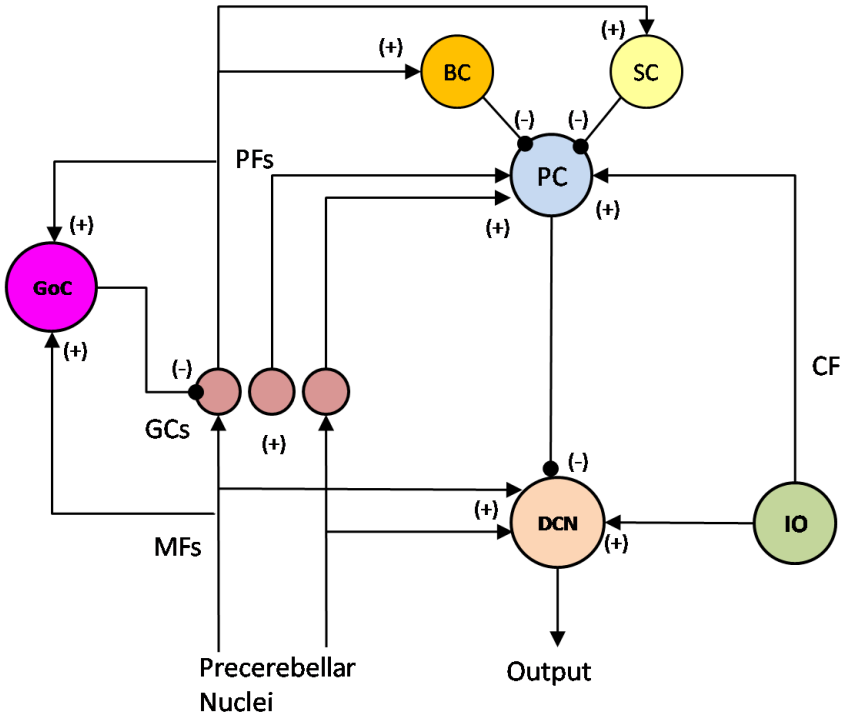


Figure 3.5 – Schematic diagram of the cerebellar circuitry. Purkinje cells (PCs) receive excitatory inputs (+) from ~150,000 parallel fibres (PFs) and a single climbing fibre (CF), and inhibitory inputs (-) from basket cells (BCs) and stellate cells (SCs), and in turn inhibit the deep cerebellar nuclei (DCN). Also shown are: mossy fibres (MFs), granule cells (GCs), Golgi cells (GoCs) and the inferior olive (IO).

Inspired by this classical cerebellar learning theory, previous studies evaluated the pattern recognition performance of PCs using somatic excitatory postsynaptic potential (EPSP) amplitudes [65] and spike responses [66, 80, 81]. Steuber and collaborators [65, 66] used a multi-compartmental PC model (presented in Section 2.3.3.2) to study pattern recognition under more biologically realistic conditions. This work used two versions of the PC model proposed by De Schutter and Bower [13, 14]. An extended version of the PC model contained 147,400 dendritic spines, which were activated randomly by a sequence of PF inputs at an average frequency of 0.28 Hz. The background excitation was balanced by tonic inhibition, which made the model fire simple spikes at an average frequency of 48 Hz. Due to the large number of dendritic spines, which made the simulations

computationally expensive, a simplified version of the PC model was also used where the number of spines was decreased to 1% of the original number. To compensate for this reduction, the rate of PF excitation was increased to an average frequency of 28 Hz and each spine was set to receive 100 synapses. As this simplified model gave identical results as the full model, it was used in most of the simulations done by Steuber and De Schutter [65, 66] as well as in this follow-on study of the effect of different features of synaptic plasticity; the results are presented in Chapter 4.

The pattern recognition process in the PCs was simulated in two steps. First, 200 input patterns were generated, each of them with 1000 active PF inputs, and half of these patterns were learned by the ANN, described in Section 3.3.1.1. In the second step, the vector of synaptic weights was transferred from the ANN onto AMPA receptor conductances in the multi-compartmental PC model. This represents the mechanism that the PC model learns the input patterns by depressing the corresponding AMPA receptor conductances based on LTD induction. To test the recall of learned patterns, the PC model was then presented with the corresponding stored and novel patterns of synchronous AMPA receptor activation at the PF synapses, as in the recall phase for the ANN (shown in Figure 3.3). To evaluate the discrimination between stored and novel patterns, s/n ratios (Equation 3.1) were calculated for different features of the PC spike response, and these results were compared with the s/n ratios for the corresponding ANN using the LTD learning rule, described in Section 3.3.1.1.

In the study of pattern recognition in PCs, Steuber and collaborators [66] found that PCs can use a novel neural code based on the duration of silent periods or pauses after the presentation of a PF input pattern to the PC. They found that the PC model produced a burst sequence of spikes followed by a pause, with shorter pauses in response to learned patterns when compared to novel patterns, as can be seen in Figure 3.6A. The length of the pause duration is shorter for learned patterns due to a negative feedback effect based on KCa channels, as the cell receives less Ca^{2+} influx into the dendritic tree, which consequently results in a shorter afterhyperpolarization (AHP). This form of neural coding diverges from the classical view, which assumes that the number or timing of individual spikes is used to distinguish between novel and learned patterns. In the study by Steuber et al., the pause was compared with two other spike features: the number of spikes in a fixed time-window (25 ms) after pattern presentation and the latency of the first spike in the response. Examples of response distributions for the latency, number of spikes and pause duration are given in Figure 3.6B, where the respective s/n ratios are 0.33, 0.21 and 15.6. These results suggest that the pause was the best criterion for pattern recognition by cerebellar PCs. In a follow-up study, I used the same model to analyse the effect of synaptic plasticity saturation on pattern recognition by PCs; these results are presented in the next chapter.

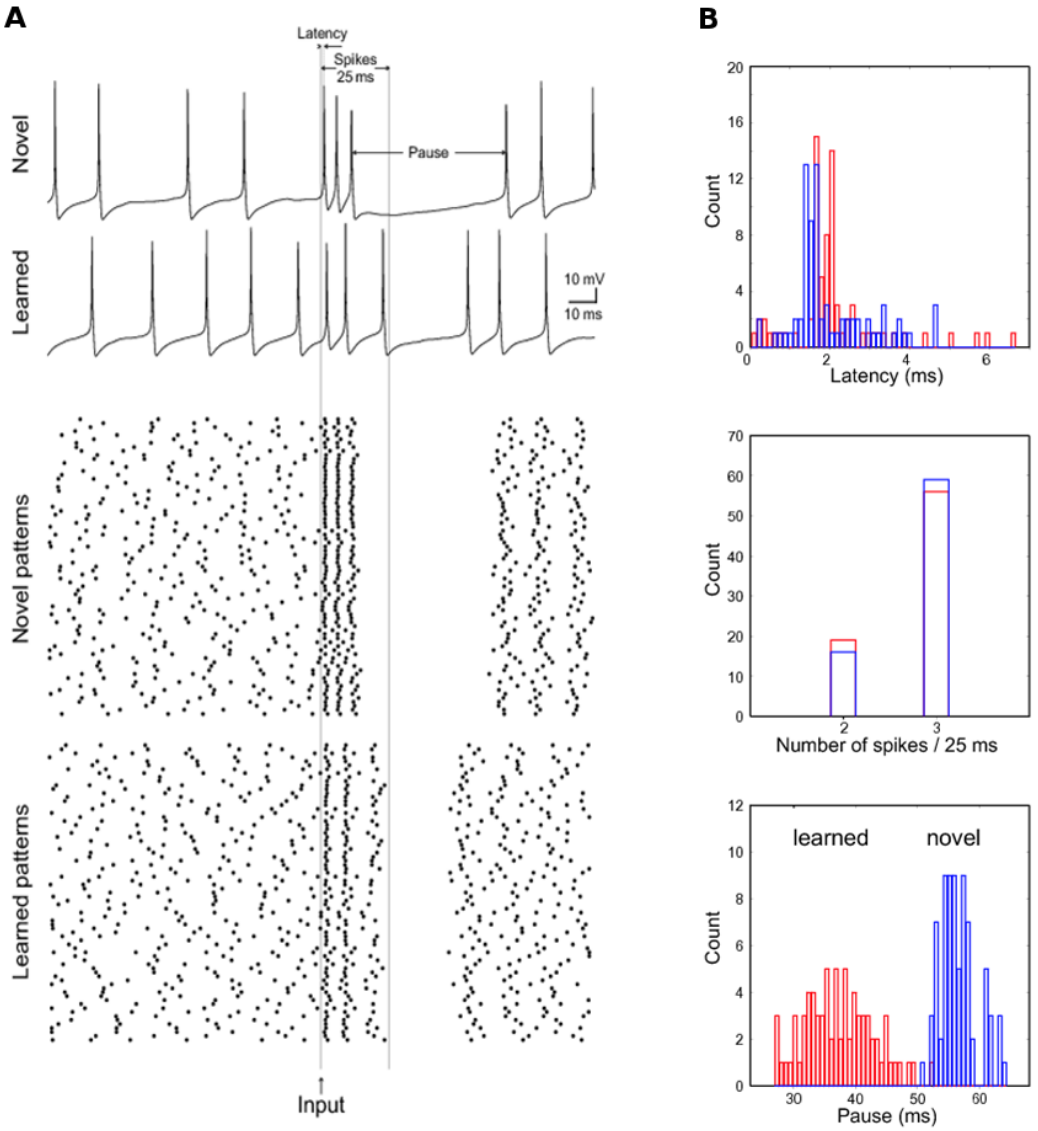


Figure 3.6 – Responses of the Purkinje cell model to novel and learned patterns of PF input. (A) Upper: The pauses evoked by the novel pattern are longer than those for the learned pattern. Lower: Raster plot of spike responses to presentations of 75 learned and 75 novel patterns. (B) Response distribution for three different spike features. Upper: Latency of first spike after pattern presentation. Middle: Number of spikes in the first 25ms. Lower: Length of pause. Modified from Steuber et al. [66].

3.3.2.2 Hippocampal Pyramidal Cell

This model of pattern recognition in a hippocampal CA1 pyramidal cell was proposed by Graham [18]. Using the multi-compartmental pyramidal cell model described in Section 2.3.3.1 and an ANN based on an associative net which uses an LTP learning rule (explained in Section 3.3.1.2), Graham compared the pattern recognition capability of a CA1 pyramidal cell in the presence and absence of noise.

The pattern recognition simulation in the CA1 pyramidal cell model was implemented in two

steps, as in the Purkinje cell model explained in Section 3.3.2.1. In the first step, 60 binary input patterns were generated, with initially 200 active inputs out of 8000 inputs, and each pattern was learned by an ANN. As in the Purkinje cell, the pyramidal cell stores patterns by using a clipped Hebbian learning, as explained in 3.3.1.2. In the second step, the vector of binary synaptic weights was transferred from the ANN onto the AMPA receptor conductances of the excitatory synapses in the pyramidal cell model, where each active input was represented by a single synaptic activation. This represents the mechanism by which the pyramidal cell learns the patterns by strengthening the excitatory conductances. The responses to the input patterns were determined by calculating the inner products of input and weight vector (here called the dendritic sum) in the ANN (Figure 3.7A) and by measuring the somatic EPSP amplitudes in the neuronal model (Figure 3.7B). Again, a s/n ratio was used to measure the capability of both models to discriminate stored and novel patterns, shown in Figure 3.7.

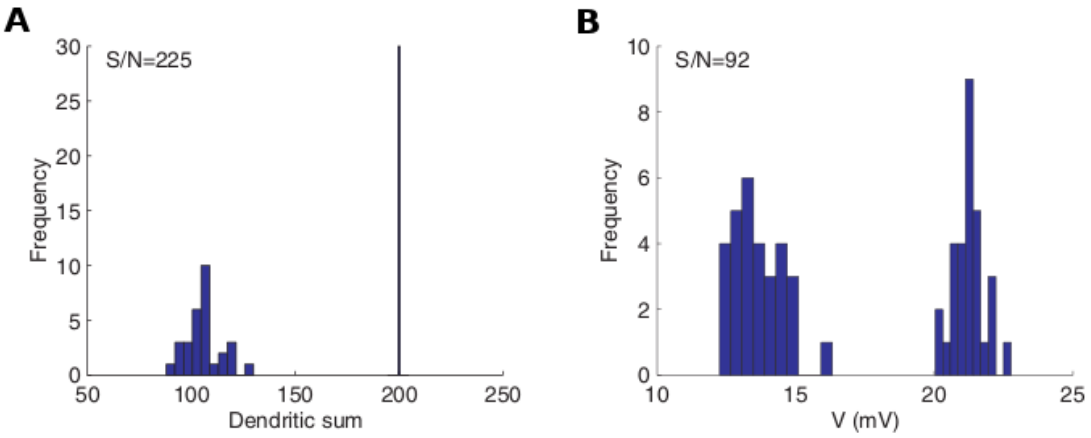


Figure 3.7 – Response distributions for novel and stored patterns (left and right distributions respectively). A. The ANN response is calculated by the sum of synaptic weights associated with active inputs (as shown in Figure 3.4). B. The CA1 pyramidal cell response is given by the somatic EPSP amplitudes. For both models, the pattern recognition performance is measured by the s/n ratio given on the top of each graph. Modified from Graham [18].

The results found by Graham showed that pyramidal cells could operate as an associative memory and that their pattern recognition performance was robust even in the presence of many sources of noise. However, the relevant part of his work was the pattern recognition model which I used as a basis of my model for the exploration of dendritic morphologies for pattern recognition (presented in Chapter 6).

3.4 Conclusion

To start my search of optimal morphologies for pattern recognition, I first had to study the fundamental concepts behind learning in neuronal systems. It is widely believed that learning and

memory in the brain are based mainly on long-term changes in the strength of synaptic connections. Based on this assumption, I started my studies of pattern recognition by studying the types of long-term synaptic plasticity used in this work: long-term depression and long-term potentiation. Based on these two types of synaptic plasticity, I then described the learning rules used in the pattern recognition models studied here.

I first presented an LTD learning rule, which was used in studies of pattern recognition in cerebellar Purkinje cells. For this learning rule, the ANN and compartmental PC model learned input patterns by depressing the synaptic weights. In the studies by Steuber and collaborators, presented in Section 3.3.1.1, the synaptic weights were decreased by half each time a pattern was learned. However, as previous experiments have shown [49, 82], this could result in unrealistically small values. So in my work a new learning rule was tested where the synaptic weights were decreased to a fixed lower bound or saturation value. The results of these experiments are presented in the next chapter (Chapter 4).

The second learning rule, based on LTP, was used in previous pattern recognition simulations of a CA1 pyramidal model. In the LTP learning rule, the patterns presented to the ANN and compartmental pyramidal cell model were learned by strengthening the synaptic weights by adding a constant value. For the pattern recognition model implemented in my exploration of dendritic morphologies, presented in Chapter 6, this LTP value was equal to one, which means that for each active input of each pattern the synaptic weight was incremented by one. The results of all of these studies are presented later in Chapters 7 and 8.

It is important to notice that the PC model used a gap (pause) between the spikes after pattern presentation to encode the pattern recognition. This makes the background activity an import feature of the model, as without a continuous level of activity, the pause could not be measured.

Chapter 4

Effect of Saturating Synaptic Plasticity and Inhibitory Plasticity

4.1 Introduction

The original study of pattern recognition in the cerebellar Purkinje cell (PC), explained in the previous chapter (Section 3.3.2.1), used a simplified learning rule. This LTD learning rule was implemented by decrementing the synaptic weight by 50% for each active synapse each time a pattern was learned. As this oversimplification could result in unrealistically small values of the weights [49, 82], I decided that one of the first aims of my research should be to investigate the effect of LTD saturation in the PC model. For this analysis, all synapses that received an active input during pattern storage were decreased to a constant value. This saturation or lower bound value of the synaptic weight was then kept constant and was unaffected by further pattern presentations. So, if a synapse changed at all it was immediately set to the saturation value, which therefore acted as a lower bound on all the synaptic weights, effectively implementing one-shot Hebbian learning. This learning rule was supported by experimental evidence in cerebellar Purkinje cells of rats which shows that once the saturation value was achieved, synaptic weights would not change any longer [49]. For example, the initial weight of all synapses was one and a saturation value of 0.5 or 0.6, etc. was tested. The results of this new LTD learning rule are presented in Section 4.2.1.

Another issue which was not covered in the previous research was the effect of LTD at the synapses between inhibitory interneurons and PCs. As LTD of inhibitory synapses can be induced when the PC receives coincident CF input [49], it could potentially counteract the effect of the depression of the excitatory PF synapses. Hence, I studied the effect of LTD of the inhibitory synapses on pattern recognition in the PCs models; the results are presented in Section 4.2.2.

The last point discussed in this chapter concerns the complexity of the PC model simulations.

Firstly, the pattern recognition simulations in the PC model could take a long time, varying from days to weeks, depending on the number of parameters being analysed. Secondly, the original PC model was a very complex model, including 4550 compartments, which were divided into four different types with distinct morphological and passive properties as well as different sets of ion channels. Thus, the large number of variables involved in the model description made it difficult to find the relationship between the neuronal morphology and performance, which is the main focus of this research. Therefore, to achieve quicker results and have a better understanding of the morphological properties related to the pattern recognition performance, a reduced PC compartmental model was used. This reduced model and the results obtained for it are presented in Section 4.3.

4.2 Effect of Synaptic Plasticity Constraints on Pattern Recognition in the Cerebellar Purkinje Cell

4.2.1 LTD Saturation at Excitatory Synapses

In my research into the effect of synaptic plasticity at excitatory synapses, I initially investigated two essential parameters involved in cerebellar pattern recognition: LTD saturation and the number of active PFs. Firstly, to study the effect of LTD plasticity at excitatory synapses, I varied the LTD saturation value over a range of values from zero to 0.8, where zero represented that the synapses were silenced completely and 0.8 represented that the synaptic weight was decreased from 1 to 0.8. For this experiment, I used the same numbers of active PFs (1000), which represents 0.7% of the total number of PF inputs (147,400), and PF patterns (100 novel and 100 stored) as in previous work [66]. Using an ANN driven by the new LTD learning rule (explained in Section 3.3.1.1) and the PC model where the duration of pause was considered the relevant response feature (explained in Section 3.3.2.1), the performance of each model was calculated by averaging their s/n ratio, where 10 different sets of patterns were presented for each models. Figure 4.1 show the results for the ANN (A) and PC model (B) where the LTD saturation value is given on the x-axis and their pattern recognition performance is presented on the y-axis.

From these results, we can see that the ANN was insensitive to the amount of LTD induced (Figure 4.1A). In contrast, the pattern recognition capacity based on the duration of pauses in the PC model improved when the LTD saturation value decreased, finding an optimal performance when the synaptic weights of active PFs were set to zero (Figure 4.1B). The relative performance of the ANN and the PC model to the amount of LTD induced are compared in Figure 4.1C. While the ANN was unaffected by varying the amount of LTD, increasing the LTD saturation value to

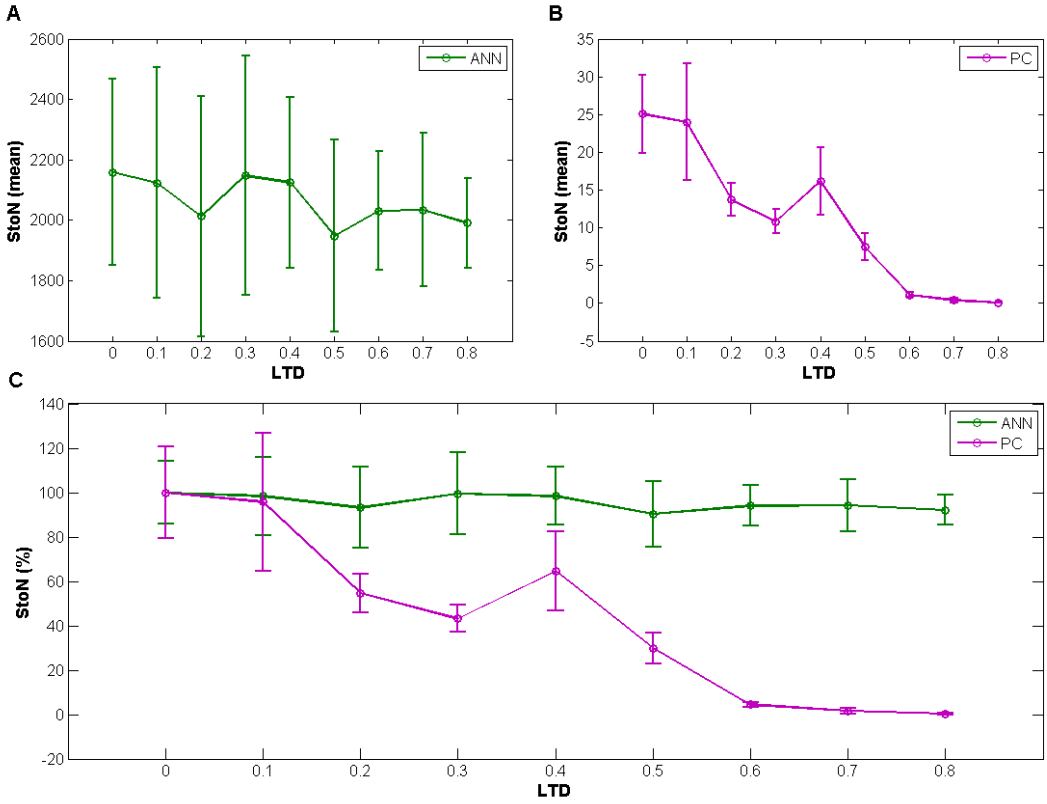


Figure 4.1 – Pattern recognition performance of the ANN and PC model for a range of LTD saturation or lower bound values. The performance was evaluated by calculating the mean s/n ratio for the ANN (A) and the PC model using the pause duration (B), averaging over 10 different sets of patterns. The relative decreases in s/n ratio are compared in (C), showing that the PC model is more sensitive to the LTD saturation value than the ANN. Error bars indicate standard deviation (SD).

0.8 in the PC model reduced the s/n ratio down to 0.4 ± 0.4% (n = 10) of the maximal value obtained by switching off the synapses completely. For LTD saturation values below 0.5, the PC model performed as well as or better than the previous model with a non-saturating learning rule, for which the average s/n ratio was around 15 [66].

The reason for the difference in sensitivity of the ANN and the PC model to varying amounts of LTD became apparent when the mean responses of the two models to stored and novel patterns were plotted against the LTD saturation value (Figure 4.2). In the PC model, increasing LTD saturation values reduced the difference in pause duration between stored and novel patterns, with standard deviations that were affected to a much lesser extent (Figure 4.2B). This led to the drastic reduction in s/n ratio for weak LTD shown in Figure 4.1. In the ANN, the difference between the mean responses to stored and novel patterns was affected as well by the LTD saturation value, but the standard deviation of the responses to novel patterns decreased with increasing LTD saturation values (Figure 4.2A). Based on Equation 3.1 (given in Section 3.3 on page 26), the constant s/n ratio of the ANN in the presence of varying amounts of LTD can be explained by a linear relationship

between the squared difference of the mean responses to stored and novel patterns $(\mu_s - \mu_n)^2$ and the variance of the responses to novel patterns σ_n^2. In other words, as the separation between the novel and learned patterns decreases so does the variance of the response to the novel patterns. In fact, it can be shown analytically that the s/n ratio of the ANN can be approximated as

$$S/N = 2\frac{A(S-D)}{D} \tag{4.1}$$

where S is the total number of synapses, A is the number of active synapses per pattern and D is the number of depressed synapses after learning, which can be approximated as

$$D = S(1-(1-\frac{A}{S})^L) \tag{4.2}$$

where L is the number of learned patterns (R. Maex, personal communication, 3 Feb 2009). For analytical demonstration, see Appendix A. So, for the configuration of the ANN used here, $S = 147,400$, $A = 1000$, $L = 100$, D will be approximately 72,779, and the s/n ratio will be 2050 which correspond to the number shown in Figure 4.1A. It is important to note that the s/n ratio is independent of the actual value of the synaptic depression. By contrast, in the PC model the s/n ratio is dependent on the actual value of LTD saturation as the variance of the response to novel patterns does not change very much while the mean difference between the responses to learned and novel patterns decreases.

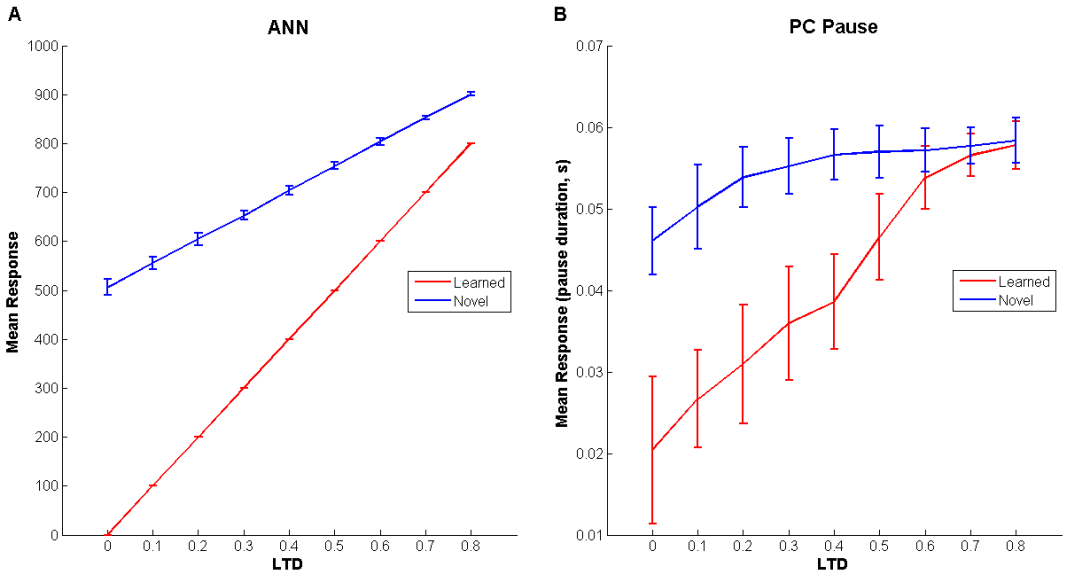


Figure 4.2 – Relationship between the LTD saturation value and the mean responses to learned and novel patterns in the ANN and the PC model using pause duration as response feature. Although the difference between the mean responses to stored and novel patterns decreases with increasing LTD saturation values in both cases, in the ANN the variance of responses to novel patterns also decreases. This results in s/n ratios in the ANN that are independent of the LTD saturation value. Same simulation parameters as in Figure 4.1. Error bars indicate SD.

In a second experiment, I measured the effect of variation in the number of active PFs. The range of active PFs was varied from 500 to 5000, which correspond to 0.3 to 3.4% of all PF inputs respectively (147,400), where the same synaptic conductance was used regardless of the number of active PFs. The LTD saturation values were varied over the same range as in previous experiments (from zero to 0.8). Comparing once more the effects on pattern recognition by the ANN and the PC model based on pause duration, we can see from Figure 4.3A that the ANN showed a worse performance for a higher number of activated PFs, although the LTD factor did not affect its performance. However, in the PC model, lower LTD factors produced better s/n values for pattern recognition, especially in an optimal range of active PF numbers between 1000 and 2000 (Figure 4.3B). Interestingly, a worse performance was found in the PC model when compared to the ANN for the LTD saturation value of zero with a low number of active PFs (500). This can be explained by the fact the synaptic conductances were not scaled when varying the number of active PFs. A further experiment has shown that in the case of 500 active PFs, the best performance was achieved (s/n~38) when the synaptic conductance was doubled from the original conductance value (0.7 nS), which was set initially for 1000 active PFs (results not shown here).

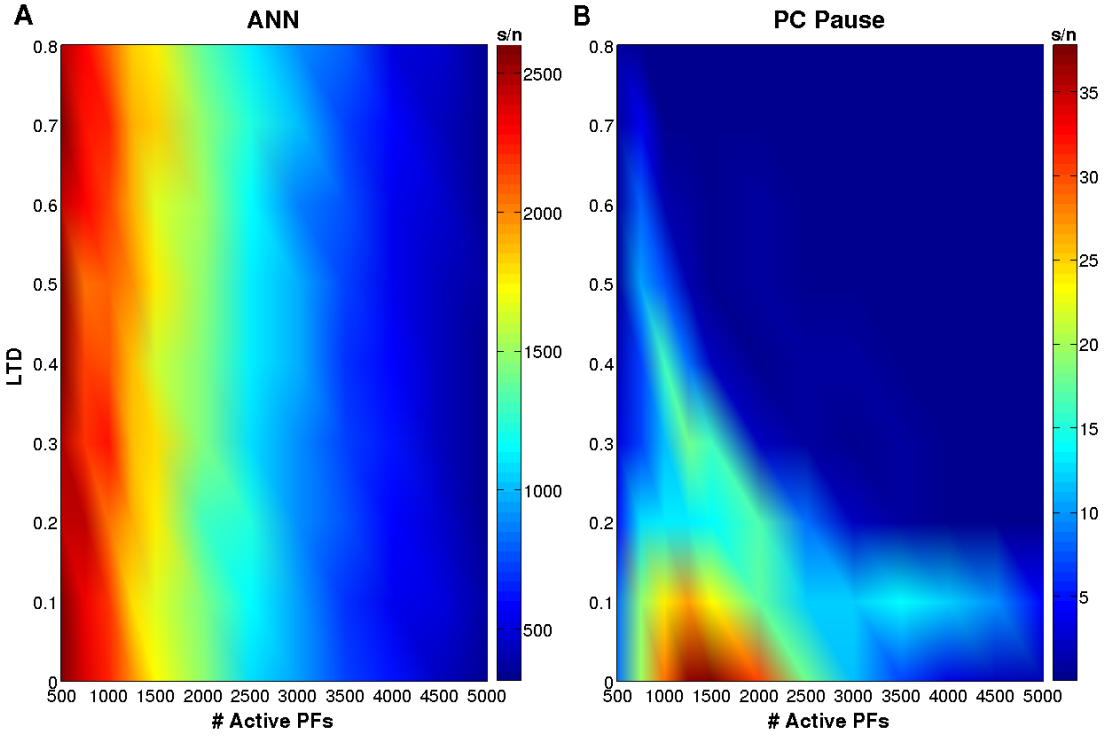


Figure 4.3 – Pattern recognition performance of the ANN (A) and PC model (B) using pause duration as response feature. The numbers of active PFs simulated were: 500, 750, 1000, 1250, 1500, 2000, 2500, 3000, 3500, 4000, 4500 and 5000. The colour represents the resulting average s/n ratio (10 trials) for each combination of a number of active PFs for each pattern (indicated on the x-axis) and an LTD saturation value (y-axis).

4.2.2 LTD at Inhibitory Synapses

LTD of inhibitory synapses can also be induced when the PC receives coincident CF input [49] and this could potentially counteract the effect of the depression of the excitatory PF synapses. Thus, to investigate the effect of saturating LTD at inhibitory synapses, the original PC model described in Section 3.3.2.1 was provided with an inhibitory input by activating a variable number of inhibitory synapses onto the soma and main dendrite. To simplify the implementation, a single inhibitory signal was introduced at the soma. The inhibitory input followed the synchronous activation of excitatory PFs synapses with a delay of 1.4 ms [49]. The ratio of inhibitory current to excitatory current was measured as the ratio of the mean inhibitory postsynaptic current (IPSC) peak to the mean excitatory postsynaptic current (EPSC) peak when the model was voltage clamped to -40 mV. I initially used an inhibition/excitation ratio of one, which is in the range of experimentally observed data from cerebellar slices [49]. Then, I simulated LTD at the inhibitory synapses and evaluated the pattern recognition performance of the PC for different numbers of patterns. The results are presented in Figure 4.4 which shows on the x-axis varying numbers of patterns tested. For example, in the first set there are 40 patterns in total, consisting of 20 stored and 20 novel patterns.

In this experiment, I did not train the inhibitory synapses explicitly. Instead the effect of inhibitory LTD was examined by depressing the IPSP amplitude to a range between 25% and 75 % of the pre-depression baseline. The simulations were configured with four different setups (see Figure 4.4): no depression (100%), and values of 75%, 50% and 25% depression. Moreover, this degree of depression was applied both separately and together for stored and novel patterns. For example, the dark blue bars in Figure 4.4 represent depression down to 0% of the original value, in other words, no inhibitory signal at all. The light blue bars represent the full value of the inhibitory signal, 100% of the original value [49].

From the results presented in Figure 4.4, it is possible to see that the pattern recognition performance of the PC model was unaffected by the presence of inhibitory LTD, even in the extreme case where the inhibitory plasticity was restricted to stored PFs patterns.

4.3 Reduced Purkinje Cell Model

To speed up the pattern recognition simulations, a reduced version of the original PC model was used. This reduced model, available from the GENESIS simulator library [7], was composed of 460 compartments (142 spines). This represented approximately 10% of the original model size, which had 4550 compartments (1,474 spines), as shown in Figure 4.5. To compensate the model

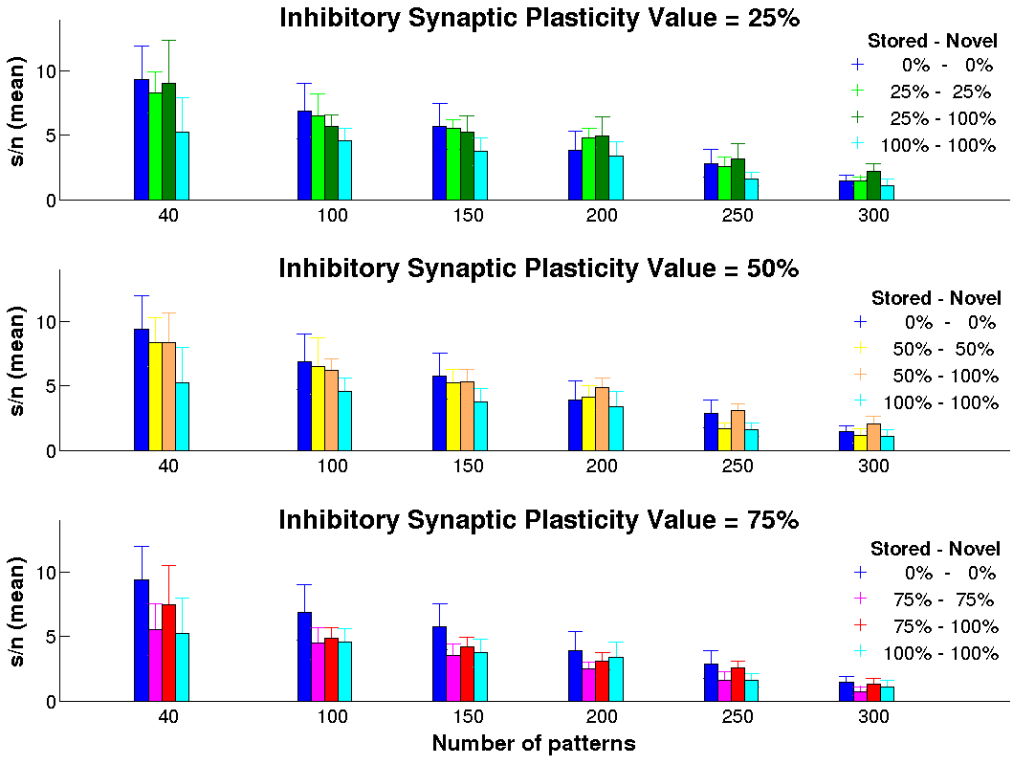


Figure 4.4 – The effect of LTD at inhibitory synapses on pattern recognition. Three different inhibitory synaptic plasticity rules were applied for varying numbers of patterns, from 40 to 300 where 40 patterns means 20 stored and 20 novel patterns (x-axis). The first bar of each graph shows the s/n ratio when no inhibition is applied for either stored and novel pattern, resulting in the best pattern recognition performance. The other bars represent cases with inhibition present, with from left to right: plasticity for both stored and novel patterns, plasticity for stored patterns only and no plasticity for either type of patterns, using the original inhibitory conductances. Error bars indicate SD.

reduction, all the parameters that influenced the pattern recognition performance, such as the number of active inputs per pattern and the size of patterns, were proportionally reduced.

A range of simulations were performed using this reduced model, varying for example, the LTD saturation value, the number of patterns presented to the model, among other model parameters. The results achieved for the variation of LTD saturation are presented in Table 4.1, where both the original LTD learning rule (with no saturation value) and new learning rule (where the LTD value saturated at lower bound value after the first active input in a pattern had been learned) were used. The results obtained for the reduced model were then compared to the original one, calculating the neuronal performance for the corresponding ANN and the three spike features studied previously, length of pause, latency of first spike and number of spikes after the pattern been presented. For the duration of pause as response feature, which was the best criterion for pattern recognition in the PC model, in all cases but one (LTD saturation value of 0.5), the reduced PC model exhibited a worse performance when compared to the original full sized model. Because one of the differences

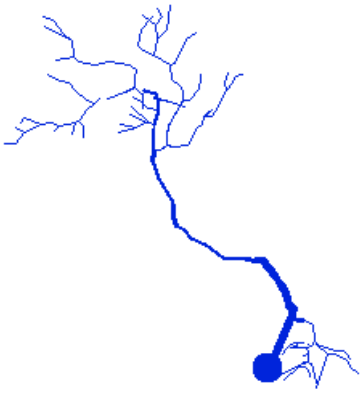


Figure 4.5 – Reduced Purkinje cell model. This model had 176 dendritic compartments, which represents 10% of the original model size (to compare models, see Figure 2.11b on page 20).

between the full and reduced PC models was the absence versus presence of spontaneous activity, I first checked whether this difference could account for the decline in s/n ratios. This experiment is described in the next section.

Table 4.1 – Comparing the performance of the ANN, the original full sized PC model and the reduced version for a range of LTD saturation values. For each PC model, the pattern recognition performance was calculated for the three spike response features studied previously in the original PC model (explained in Section 3.3.2.1): pause, latency and number of spikes. The pattern recognition performance was calculated by averaging the s/n ratio over the response for stored and novel patterns, using 10 different sets of patterns. The parameters used in each model are (original/reduced): number of PFs = 147400/14200; number of active PFs per pattern = 1000/100; number of stored patterns = 100/10. The LTD value marked as * corresponds to the original learning rule where the LTD did not saturate (explained in Section 3.3.2.1).

LTD	ANN		Pause		Latency		Spikes Number	
	Original	Reduced	Original	Reduced	Original	Reduced	Original	Reduced
0.5*	2354.26	3105.47	14.87	12.27	0.28	0.51	0.21	0.23
0.5	2097.17	3138.31	7.48	12.89	0.11	0.56	0.02	0.39
0.0	2208.84	4146.41	28.48	16.14	3.48	3.33	14.23	2.69
0.1	2196.37	3405.28	23.99	10.89	1.27	1.23	7.56	0.93

4.3.1 Effect of Spontaneous Activity in the Reduced Purkinje cell model

Another set of experiments done in the reduced Purkinje cell model was to analyse the effect of spontaneous activity on pattern recognition performance. The reduced PC model was originally characterised by spontaneous activity, as opposed to the original model which was silent if no background activity was present. In the previous experiment (Table 4.1), I compared the performance of the reduced PC model with the original model. However, the difference between both models in terms of spontaneous activity did not allow a fair comparison between them. Therefore,

to compare these models, a new version of the reduced PC model was developed which did not exhibit spontaneous activity. To achieve this, the reduced model was hand tuned by changing its leak reversal potential until the spontaneous activity ceased. The results for this new reduced model (without spontaneous activity) are shown in Table 4.2, where it is possible to see that the pause feature exhibited very low s/n values when compared to the original reduced model (with spontaneous activity). From this experiment, it appeared that the reduced PC model was over-sensitive to small changes in its parameters. Hence I sought a more robust neuron model to study the effect of dendritic parameters on the pattern recognition performance.

Table 4.2 – Comparison between the reduced PC model with and without spontaneous activity, using different LTD saturation values. The reduced PC model without spontaneous activity show very low s/n ratios for pause duration as a response feature, which means that the model is unable to use pause duration to distinguish between novel and learned patterns. The PC models with and without spontaneous activity give similar low values for other features of the spike response (latency and number of spikes after presenting a pattern), the only exception being the number of spikes in the not spontaneously active model with a LTD saturation value of 0.0, which also resulted in good pattern separation (s/n = 12.01). These simulations used the same parameters as the reduced PC model in Table 4.1.

LTD	Pause		Latency		Spikes Number	
	With	Without	With	Without	With	Without
0.5*	12.27	1.86	0.51	1.27	0.23	2.15
0.5	12.89	2.97	0.56	0.82	0.39	3.51
0.0	16.14	2.43	3.33	2.74	2.69	12.02
0.1	10.89	1.12	1.23	2.46	0.93	0.00

4.4 Conclusion

Previous computer simulations and experiments in cerebellar slices and awake behaving mice suggested that the cerebellum can use a novel neural code that is based on the duration of silent periods in neuronal activity [66]. These simulations used a complex multi-compartmental model of a cerebellar Purkinje cell that had been tuned to replicate a wide range of behaviours in vitro and in vivo [13, 14], but they applied a simplified LTD learning rule, which involved dividing the synaptic weights of active PF inputs by two every time a PF pattern was learned. This could result in very small synaptic weights and does not fit experimental data on LTD induction in cerebellar slices, where the mean AMPA receptor conductances saturate and are hardly ever depressed to less than 50% of their pre-depression baseline values [49, 82]. Moreover, the previous simulations did not include the plasticity at synapses between inhibitory interneurons and PCs that has recently been characterised [49]. Given these points, I decided to study the effect of inhibitory synaptic plasticity and saturating LTD in the complex PC model. I found that the ability of the PC model

to discriminate between learned and novel PF input patterns was unaffected by the presence of inhibitory plasticity for a wide range of parameter values. However, the pattern recognition performance of the PC model was very sensitive to the value at which LTD saturated. In contrast to a corresponding ANN, which was unaffected by the level of LTD induced, the performance of the PC model was improved by lower LTD saturation values. The best performance resulted from LTD saturation values of zero, which corresponds to silencing the PF synapses completely. Interestingly, large numbers of silent PF synapses have been observed by monitoring microscopically identified PF–PC connections in cerebellar slices [29]. My simulation results indicated that the discrepancy between the existence of these silent synapses and the apparent saturation of LTD in induction experiments needs to be resolved to understand the connection between LTD and cerebellar learning.

In an attempt to speed up the pattern recognition simulations in the PC model, I decided to use a reduced version of the original model which had approximately 10% of the size of the original model. The results obtained for this reduced model did not reproduce the original PC model performance, which could be due to the reduced model being spontaneously active, while the full model was not. An attempt was made to remove the spontaneous activity in the reduced PC model to allow for a fair comparison with the original model. However, once again the results found were unsatisfactory, as the reduced model showed a very poor performance when its spontaneous activity was removed. Hence, after many failed attempts trying to tune the reduced PC model to behave as the original full-sized model, I concluded that this model could not be used in further experiments. As a result, I started to look for different neuronal models on which to carry out my research into pattern recognition. As the Purkinje cell model and other biologically realistic models had very complex morphologies, I decided to look for models with simpler morphologies, which are discussed in the next chapter.

Chapter 5

Review of Dendritic Morphology and Neural Physiology

5.1 Introduction

In my previous experiments, a reduced Purkinje cell model was used to investigate the effects of synaptic plasticity on pattern recognition. However, as shown in the previous chapter, the results obtained from this reduced version did not reproduce the results from the full-sized model; hence the reduced Purkinje cell could not be used for further experiments. As the reduced Purkinje cell was not a reliable model and the full-sized model had a very complex morphology to analyse the effect of its firing pattern on pattern recognition performance, I decided to modify the project direction using simpler generic morphologies. The morphologies chosen were based on previous work from van Ooyen and collaborators entitled "*The effect of dendritic topology on firing patterns in model neurons*"[76]. In this work, simple dendritic morphologies, built with the same electrophysiological properties but with different topological arrangements were shown to produce different firing patterns. This model was not based on any real neuronal morphology, but it was based on an abstraction of dendritic morphology. The abstract model assumed all morphologies were binary trees with a simplified structure, including all dendritic segments being the same length. The authors also restricted the model to produce morphologies with the same number of terminal segments, just varying their dendritic topology, that is the way the dendritic segments are connected to each other. Because of the simplicity of the morphologies produced by this model, they were chosen to be used as initial morphologies in my search for optimal morphologies for pattern recognition. This model is presented in more detail in Section 5.2.

After I defined the type of morphologies used as the starting point of this research, my next step was to find systematic ways to generate different examples of valid tree shapes. To do this, a

critical analysis of the existing tree morphology generation schemes was performed. Each of these morphology generating algorithms was evaluated with respect to some predetermined requirements, which are described in Section 5.3.1. These requisites were formed keeping in mind the goal of generating morphologies to perform pattern recognition. A full description of this analysis is given in Section 5.3. Finally, a summary of the features chosen from the algorithms tested is presented in the conclusion of this chapter.

5.2 Simple Dendritic Morphologies

In this section, I present the study by van Ooyen et al. [76] about how different dendritic morphologies can produce different firing patterns using a simple neuronal model. In this work, the authors described a model composed of a set of multi-compartmental neuronal morphologies with active conductances. This model, named from this point on as van Ooyen's model, is presented in detail in the next section as it served as basis for my evolutionary algorithm discussed later in Section 6.4. Moreover, five different morphometrics are presented as they were used to compare dendritic morphologies. The first two metrics described were presented with van Ooyen's model and widely used in the literature (Section 5.2.2): the asymmetry index and the mean path length. The third one was based on the mean electrotonic path length defined in van Ooyen's most recent publication [75]. The last two metrics used were the mean depth, which was defined as a simplified version of the mean path length; and the variance of depth, which was calculated using the variance of each normalized dendritic path. These five metrics are discussed in the Section 5.2.2.

5.2.1 Van Ooyen's Model

The idea behind the research by van Ooyen and collaborators was to investigate how neuronal morphologies with the same membrane properties can generate different firing patterns when varying only the dendritic topology. All neuronal models they studied have the same membrane properties (see Table 5.1) as well as the same number of terminal points (namely 8). The axo-somatic compartment was a cylinder with a length and diameter equal to 20 μm, and all dendritic segments were cylinders with the same length (varied in each experiment) and a diameter equal to 5 μm. The only difference found between the studied morphologies was the dendritic topology.

The ion channel conductances used were based on those of the two-compartmental model by Mainen and Sejnowski [42]. This model was written in the Neuron programming language [22] and can be freely downloaded from ModelDB (http://senselab.med.yale.edu/modeldb/ShowModel.asp?model=2488) The only difference is found in the somatic conductance densities which are 10 times smaller than

Table 5.1 – Passive properties used in van Ooyen's model [76]. The membrane properties are based on Mainen and Sejnowski's two-compartmental model [42].

Membrane Properties		**Reversal Potentials**	
Cm	0.75 $\mu F/cm^2$	E_{Na}	60 mV
Rm	30 $k\Omega cm^2$	E_K	-90 mV
Ra	150 Ωcm	E_{Ca}	140 mV
		E_{Leak}	-70 mV

in the original model to compensate for the larger axo-somatic compartment used in van Ooyen's model (see conductances in Table 5.2).

Table 5.2 – Ion channel conductances from van Ooyen's model. These conductances are expressed in $pS/\mu m^2$.

Conductance	**Symbol**	**Soma**	**Dendrite**
Fast Na^+ conductance	$\overline{g}_{Na}$	3000	15
Fast non-inactivating K^+ conductance	$\overline{g}_{Kv}$	150	-
Slow voltage-dependent non-inactivating K^+ conductance	$\overline{g}_{Km}$	-	0.1
Slow Ca^{2+}-activated K^+ conductance	$\overline{g}_{KCa}$	-	3
High voltage-activated Ca^{2+} conductance	$\overline{g}_{Ca}$	-	0.3
Leak conductance	$\overline{g}_{Leak}$	-	0.33

To investigate the firing pattern generated by each morphology, van Ooyen et al. used a set of binary dendritic trees each with 8 terminal points. The structure of these trees followed the definition given by van Pelt and Verwer [79], named ambilateral tree types. The ambilateral tree types are defined by Toroczkai [73] as *"a set of trees whose elements cannot be obtained from each other via an arbitrary number of reflections with respect to vertical axes passing through any of the nodes on the tree"*. One way to ensure that a set of binary trees is ambilateral is to make the trees right heavy. In other words, at any vertex the right subtree has at least as many leaves as the left subtree. These tree types constitute a subset of two-dimensional topological tree types, which describe the whole range of binary trees [79]. For example, for trees with five terminal segments, the total number of possible different 2D-tree types is 14, which can be divided into three ambilateral types, the right-heavy representatives of which are shown in the right-side column of Figure 5.1.

The total number of ambilateral-tree types with n terminal segments is given by the recurrent expression:

$$N^n = \frac{1}{2}\left(\sum_{r=1}^{n-1} N^r N^{n-r} + (1-\varepsilon(n))\, N^{n/2}\right) \tag{5.1}$$

where $N^1 = 1$, $\varepsilon(n) = 0$ for even n and $\varepsilon(n) = 1$ for odd n. Using this equation, it is possible to calculate the total number of trees with 8 terminal segments, which gives the set of 23 trees used in van Ooyen's model (see Figure 5.2).

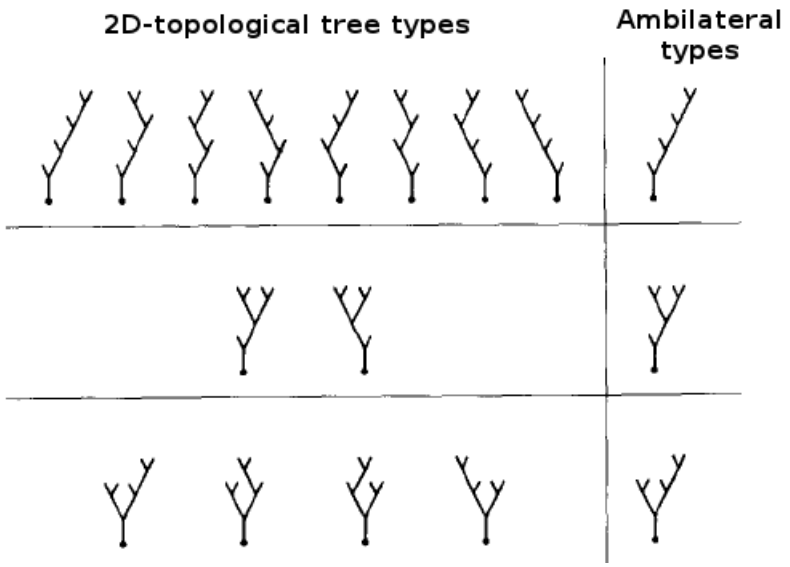


Figure 5.1 – 2D and ambilateral-tree types with 5 terminal segments. The left side shows the 2D-topological tree types, covering the whole range of possible trees with 5 terminal points (14 trees). On the right side, all the possible ambilateral-tree types are shown. Modified from van Pelt and Verwer [79].

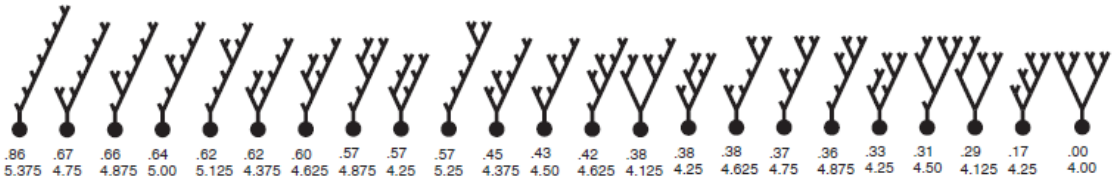


Figure 5.2 – Morphological trees of van Ooyen's model. All possible ambilateral-tree types with 8 terminal points are shown. These trees are electrophysiologically unique, as each of them represents a set of the 2D-tree types (as shown in Figure 5.1). The numbers presented below the trees are asymmetric index (top) and mean path length (bottom) with the segment length equal to 1 μm. Both are discussed in the next sections. From van Ooyen et al. [76].

The neurons in the van Ooyen model were stimulated by injecting a 100 pA current into the axo-somatic compartment and the firing pattern was also recorded at the soma. Figure 5.3 shows examples of output patterns, where a fully symmetric and fully asymmetric morphology are compared, using two different dendritic lengths. These results show that for trees with the same number of terminal segments and total dendritic length, the firing pattern can vary from single spikes to bursts, by only varying the tree topological arrangement (see graphs in first column of Figure 5.3). They also show that for the same topological tree, it is possible to obtain different outputs when increasing the total dendritic size (compare firing patterns for 1150μm and 2100 μm total dendritic tree length).

The results obtained by van Ooyen and his collaborators demonstrate that tree topology and dendritic length are important neuronal morphology features to be studied, as regards to firing pattern analysis.

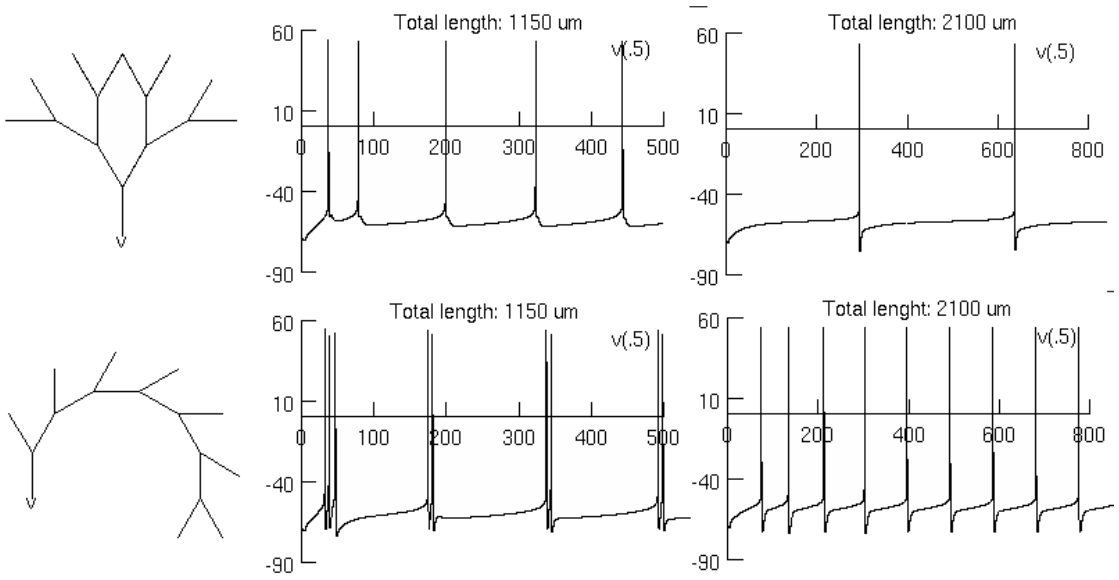


Figure 5.3 – Example of firing patterns I obtained for fully symmetric (upper traces) and fully asymmetric (lower traces) trees using van Ooyen's morphology model. The output patterns from both neurons were generated with Neuron simulator and compared for trees with larger dendritic length (right traces) and smaller ones (left traces). These graphs reproduce the results found by van Ooyen et al., 2002 (see original results in Figure 2 of [76]).

5.2.2 Morphological Tree Metrics

To study how the dendritic morphology affects the firing pattern, I needed to analyse different dendritic topologies. In the next sections, five different metrics are presented which enabled me to distinguish dendritic trees with different topologies.

5.2.2.1 Tree Asymmetry Index

This metric was defined in 1992 by van Pelt et al. [78]. The asymmetry index is simply the mean of the partition asymmetries of all vertices in the tree. So, the index of asymmetry A_t for a given tree α^n, with n terminal segments and $n-1$ bifurcation points, is defined as:

$$A_t(\alpha^n) = \frac{1}{n-1} \sum_{j=1}^{n-1} A_p(r_j, s_j) \tag{5.2}$$

The partition asymmetry A_p at a given vertex j is defined as :

$$A_p(r_j, s_j) = \frac{|r_j - s_j|}{r_j + s_j - 2} \tag{5.3}$$

where r_j and s_j are the number of terminal segments in the two subtrees of the vertex j and $A_p(1,1)$ is equal to zero. Given this equation, we find that the asymmetry index is zero for the most symmetric tree and close to one for the most asymmetric tree (see top index in Figure 5.2).

5.2.2.2 Mean Path Length

Van Ooyen and collaborators use the sum of all dendritic path lengths from the soma to the terminal points to calculate the mean path length of a tree [76]. For the purpose of this work, I assumed that the mean path length is calculated over all dendritic segments instead only for terminal ones. This is required as the mean path length needs to consider the location of all synapses, which are uniformly distributed over all dendritic segments (more details are presented later in Section 7.2). So, for a given tree α^n with n terminal segments, the mean path length P_t is here defined as:

$$P_t(\alpha^n) = \frac{1}{2n-1} \sum_{i=1}^{2n-1} P_i \tag{5.4}$$

where P_i is the total length of the dendritic path from the ith segment to the soma. See examples of mean path length calculated for trees with 8 terminal points in the bottom indices of Figure 5.2.

5.2.2.3 Mean Electrotonic Path Length

Mean electrotonic path length is defined by van Elburg and van Ooyen [75] as the dendritic path length from each terminal segment to the soma. However, as defined in the mean path length, I calculate the mean electrotonic path length using the dendritic path from each dendritic segment to the soma, as I am interested in the location of all synapses instead only terminal segments.

So, to calculate the electrotonic path length, each dendritic segment i which has its length ℓ_i is normalized by an electrotonic length constant λ_i, which is defined as:

$$\lambda_j = \sqrt{\frac{d_i r_m}{4 r_a}} \tag{5.5}$$

where d_i is the diameter of the dendritic segment i, r_m is the membrane resistance and r_a the axial resistance of the cell. So, the normalised electrotonic length Λ_i is given as:

$$\Lambda_i = \frac{\ell_i}{\lambda_i} \tag{5.6}$$

To calculate the mean electrotonic path length (MEP) for the dendritic tree with n segments, the following equation is used:

$$MEP(\alpha^n) = \frac{1}{2n-1} \sum_{i=1}^{2n-1} \Lambda_i \tag{5.7}$$

where Λ_i is defined as the electrotonic lengths in the path from the dendritic segment i to the soma.

5.2.2.4 Mean Depth

This metric is calculated similarly to the mean path length, where the mean is computed over all dendritic paths. However, instead of using the actual dendritic length, the compartmental length is normalised and assumed to be equal to 1. This means that the mean will be calculated using the number of steps required to reach each dendritic synapse instead of its actual path length, as calculated in Equation 5.4.

Following this concept, the lower index previously shown in Figure 5.2 for van Ooyen's model in fact shows the mean depth of each tree, which is equivalent to the normalised mean path length presented in his model.

5.2.2.5 Variance of Depth

Variance of depth is defined as the variance of the dendritic path of each dendritic segment using the normalised compartmental length (1). This metric was introduced as it correlates with the calculation of the neuronal performance, which involves the variance of the stored and novel patterns (as previously explained in Section 3.3.1.1). Using a large set of trees with 22 terminal points (later discussed in Section 6.3.2), it is possible to see that this metric differs from the mean depth, as shown in Figure 5.4.

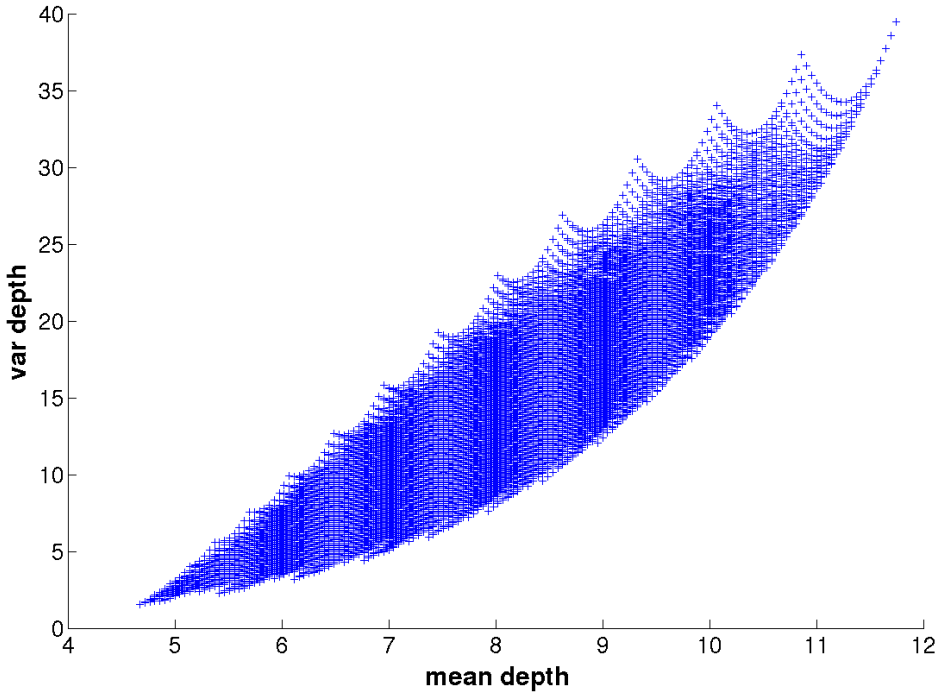


Figure 5.4 – Plotting mean depth against variance of depth. This graph was plotted using $1,721,998$ trees with 22 terminal points each. As variance of depth is not completely correlated with mean depth metric, this fourth metric was introduced to evaluate pattern recognition performance in the different neuronal morphologies studied.

5.3 Morphological Tree Generation Algorithms

5.3.1 Algorithm Requirements

Currently in the literature it is possible to find many algorithms to generate morphological tree structures (for references see [4, 67, 71, 70]). Most of these algorithms are designed to make the resultant trees look biologically realistic. However, since the visual appearance of the neuronal trees does not make any difference to the performance of neurons to act as associative memories and pattern recognisers, this feature is not relevant to this research. As a result, virtually none of the current tree generation tools available in the field were appropriate for my work.

Here in this section, I present several tree generation algorithms found in the literature that I have tested. Most of these existing algorithms have two stages: first, an algorithm to generate morphological trees and then, an evolutionary algorithm to optimise the fitness of the population. In order to generate and search for suitable morphologies for my research, the algorithm must meet a number of requirements as described below:

I. Maximise the number of morphologies generated: this is the main requirement which the algorithm must obey. The algorithm selected should be able to maximise the number of trees generated for any given number of terminal points. Ideally, it should be able to generate all possible tree morphologies (but see requirement II below), so that my analysis can cover the whole search space. As the main goal of this research is to find the implications of neuronal morphologies for pattern recognition, I am interested in testing the largest possible number of achievable morphologies.

II. Generate electrophysiologically unique trees: the selected algorithm should ideally only generate tree structures that are electrophysiologically unique. This requirement means that the resultant neurons should not produce an identical firing pattern due to the similarity of their equivalent electric circuits. Because the software packages for simulating the electrical properties of neurons (like Genesis and Neuron) treat trees in essence as one-dimensional (branching) cable structures, only the branching pattern is important and the rotation angles of branches can be ignored. Still, different branching sequences can generate tree structures that are electrically identical mirror images. So, the algorithm should avoid such duplications and produce a set of ambilateral trees rather than 2D morphological ones (Section 5.2.1).

III. Automatically generate morphologies: another prerequisite for the algorithm is that it should be able to generate automatically all the trees containing a certain number of branch points, or terminal points, for illustration and initial testing purposes. Tables listing the numbers of

different trees with a given number of branching points are available from the literature [78], and the algorithm should be able to produce them all automatically and without duplication.

IV. Interact with evolutionary algorithms: an essential requirement for my purposes is that the algorithm must interact with an evolutionary algorithm to search the possible tree morphologies, which also implies that the algorithm must allow a suitable tree representation for genetic operations. This would allow the selected genetic algorithm to create new morphologies, using genetic operators like crossover and mutation over the genotypes. The new trees resulting from these operations should also respect the other requirements: they should have the same given number of terminal points (requirement VI) and ideally not be electrophysiologically equivalent to previous ones, within the same generation (requirement III).

V. Total control over the morphologies generated: this is another essential requirement, which means that stochastic methods would not be suitable for the actual tree generation algorithm. Any algorithm with a stochastic element would not produce a unique tree from a given genotype (possibly also violating requirement II) and would make allocating a fitness value to the genotype problematic. Indeed, multiple runs would be required to average the fitness over a sufficient number of sample trees. In summary, what the chosen algorithm should have is a one-to-one mapping between the phenotype and genotype. Hence, a deterministic algorithm should be used.

VI. Generate binary trees with a fixed number of terminal points m: This representation keeps the size of the resulting trees constant and consequently allows a meaningful comparison between them. Such trees always have m-1 branch points and $2m$-1 branch segments, due to the fact that these trees have a soma. Note that there is no requirement in this research for the selected algorithm to generate morphologies that look like real neurons, as most of the algorithms that generate morphological trees described at the next section aim to do.

The following section presents the description of each algorithm found in the literature which I tested, including details of their implementation. This is followed in each case by a section named *Problems* which explains the reasons why the algorithm presented is not suitable for the proposed research.

5.3.2 Description of the Algorithms

5.3.2.1 EvOL-Neuron

EvOL-Neuron is a tool for generating virtual neurons written by Torben-Nielsen [69, 70]. This tool uses L-System rules to generate morphologies. An L-System is a mathematical formalism used originally to model the growth processes of plants [38, 53]. This system is composed of rules and axioms. Axioms define the initial state that is the branch start point. Rules define how branches grow from the axioms iteratively. An example of axioms and rules is shown in Figure 5.5.

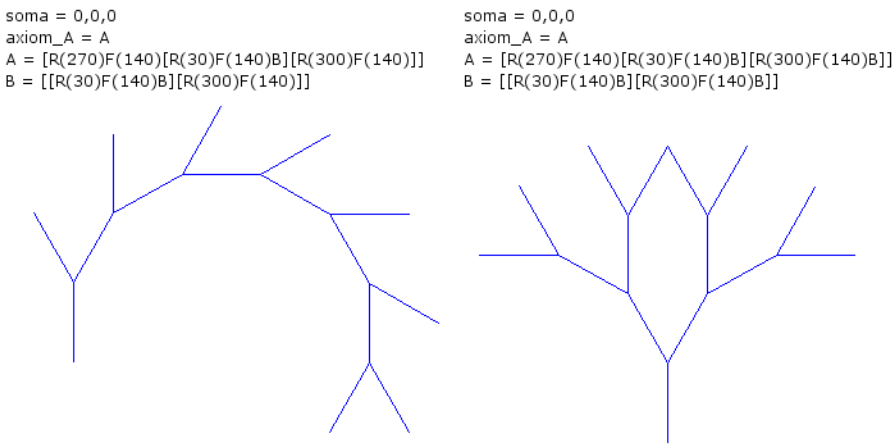


Figure 5.5 – Example of morphologies I generated with EvOL-Neuron. These neurons were generated following the L-system rules given above them, based on the morphologies presented by van Ooyen [76]. Note that the rules to generate both morphologies are quite analogous, just varying in their last part: for the symmetric tree (on the right), the rule B is called at the end of each rule to make the tree grow its left branches as well. To generate the asymmetric tree (on the left), 7 cycles were needed; for the symmetric tree, just 3 cycles were needed.

EvOL-Neuron uses a set of rules based on a simple alphabet. Three different symbols are used: *F(x)* to move forward x times, *R(x)* to rotate by x degrees and *E(x)* to elevate the angle by x degrees. In addition, the symbols [ ] are used to determine when a new branch starts and ends, respectively. Furthermore, the number of branches is determined by the number of times (cycles) the rule is called. To generate simple tree structures like those shown in Figure 5.5, EvOL-Neuron needs just one axiom (*axiom_ A*) and two rules (*A* and *B*). However, to generate complex morphologies, which look like real neurons, a stochastic component is needed [70].

Problems

One focus of EvOL-Neuron is to generate artificial neurons with morphologies close to real ones. To achieve this, the algorithm uses a stochastic component (a random seed), which makes it inappropriate for my purposes, because I wish to have full control over the generation process (see requirement V). Another problem found is that this algorithm does not allow me to keep control

of valid trees. My goal is to restrict the generation process to result in trees that have a certain given number of terminal branches, since I want to compare trees of identical sizes (see requirement VI). As shown above, two neurons with the same number of branch points can require different numbers of cycles in EvOL-Neuron. Moreover, the algorithm generates many unwanted neuronal structures, such as duplicated or mirror-symmetric morphologies, which violates requirement II.

5.3.2.2 L-Neuron

This model, proposed by Ascoli and Krichmar [4], uses the same L-System syntax as the previous model. However, the production rules (see Section 5.3.2.1) are replaced by neuroanatomical rules, which are based on Hillman's approach [21]. These rules allow the model to describe neurons in terms of "fundamental" parameters, which are expressed by dendritic features like diameter, length, tapering, and branch ratio and angle. Examples of the morphologies produced by L-Neuron are shown in Figure 5.6.

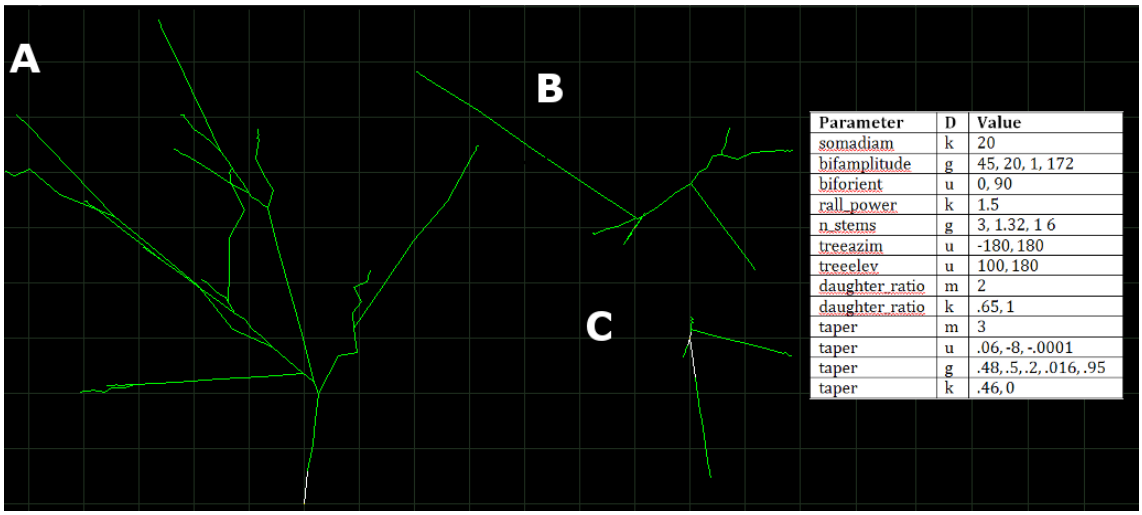


Figure 5.6 – Example of morphologies I obtained with L-Neuron. The set of neurons were obtained using Hillman's algorithm [21]. The parameter values used to generate these neurons are given in the table. For each parameter, the distribution used is given (column D at table), where k denotes constants, g Gaussian distribution, u uniform distribution and m mean. The only parameter modified for each morphology is the random number generator seed, where neuron A uses a seed of 2, B a seed of 50 and C a seed of 100.

L-Neuron uses a set of parameters to generate a morphological class instead of a single neuron - for instance, to generate morphological variability among the various neurons in a network model. To do that, the model reads a list of neuroanatomical parameters and samples them stochastically, applying statistical distributions (Gaussian or uniform). The result is a list of multiple non-identical neurons, as can be seen in Figure 5.6.

Problems

The use of a random seed to generate a morphological class results in a loss of control over the morphologies generated, which violates requirement V. The need to use a random number as seed does not allow the model to generate neurons like those shown in Figure 5.5, which are the types of morphologies aimed for in this work. Another considerable limitation of the model is the number of neuroanatomical rules necessary to describe a morphological class. This constraint goes against requirement III, since the algorithm needs to specify a limiting list of values to generate the morphologies. Additionally, the algorithm infringes requirement II since it is impossible to limit the rules to produce only electrophysiologically unique morphologies.

5.3.2.3 EvOL-Neuron II

In the first version of the algorithm (see Section 5.3.2.1), EvOL-Neuron used an L-System formalism to generate the neuronal morphologies. In that version, some morphogenetic parameters were used to validate the morphologies produced. However, in this second version [72], EvOL-Neuron uses statistical information about morphological features to describe neuronal morphologies. These features are expressed as the morphogenetic parameters (see Table 5.3) and presented in the form of density distributions. This algorithm is based on Burke's algorithm [8], which uses a similar set of parameters as employed by L-Neuron (see Section 5.3.2.2).

Table 5.3 – Parameters used by EvOL-Neuron to generate morphologies. Each parameter is described following a distribution given in the last column. The symbols used are described by Ascoli et al. [3]: μ (mean) is the location parameter and σ (standard deviation) is the scale parameter of a Gaussian distribution; α is the shape parameter and β is the scale parameter of a γ-distribution; U is a uniform distribution and G is a Gaussian distribution for the initial values given. From Torben-Nielsen and Stiefel [72].

Parameter	Initial value	Drawn from
Stem elevation	$\mu = \mathrm{U}(-80, 80)$, $\sigma = 10$	Gaussian
Stem rotation	$\mu = \mathrm{U}(0, 360)$, $\sigma = 10$	Gaussian
Stem diameter	Uniform (0.2, 10)	Constant
Segment length	Uniform(5, 30)	Gaussian
Branch rotation	$\mu = \mathrm{G}(0, 8)$, $\sigma = 10$	Guassian
Branch elevation	$\mu = \mathrm{G}(0, 8)$, $\sigma = 10$	Gaussian
Taper rate	-0.125	Constant
Bifuraction probability	$\alpha = \mathrm{U}(0, 4) + 1$, $\beta = \mathrm{U}(90, 170)$	Scaled γ (path to soma)
Termination probability	$\alpha = \mathrm{U}(0, 4) + 1$, $\beta = \mathrm{U}(5, 5\mathrm{O})$	Cumulative γ (path to soma)
I_{KA} stem density	U(0, 0.08), range [0, 0.2]	Constant
I_{KA} tapering rate	U(0.85, 1.15)	Constant
I_{CaT} stem density	U(0, 0.0001), range [0, 0.02]	Constant
I_{caT} tapering rate	U(0.85, 1.15)	Constant

This model uses an evolutionary algorithm to optimise neuronal models to perform a certain computational goal, such as coincidence detection. To generate more realistic neuronal models, the system generates multi-compartmental models with active dendrites, which incorporate ion channels (for instance KA and CaT). The fitness function to evaluate the system output is based on the desired computational function. As an example, the system was tested for input order detection, as assessed from the amplitude of the compound postsynaptic potential (EPSP) evoked by the input pattern [72]. The EvOL-Neuron genetic algorithm is explained in Figure 5.7.

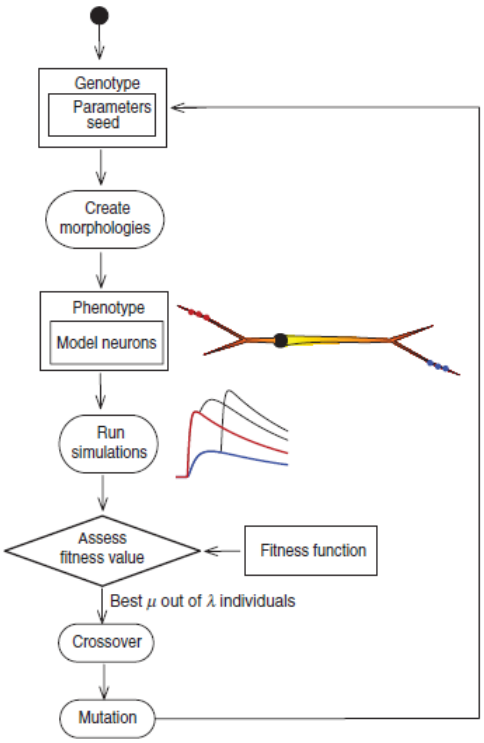


Figure 5.7 – Genetic algorithm steps used by EvOL-Neuron II to optimise neuronal morphologies. First, the initial population is generated following a set of parameters. Then these parameters are used to generate the morphologies. After running simulations, the EPSPs are evaluated by the fitness function and the best individuals are selected. Finally, crossover and mutation are applied to the selected neurons to restart the GA process. From Torben-Nielsen and Stiefel [72].

Problems

As one of the main objectives of this research is to keep control over all morphologies generated, this model does not fit with requirement V because it uses stochastic methods to generate the morphologies. Moreover, the distributions employed in this model (Gaussian and Gamma-distribution) do not allow it to generate the whole range of morphologies aimed at here, disobeying requirement I. The algorithm described here seeks to produce morphologies that were of a specific type with synapses in a certain region, whereas I want to generate all possible morphologies. Again, limit-

ing the rules to just produce unique electrophysiological morphologies is problematic, which goes against requirement II.

5.3.2.4 Stiefel's algorithm

Similar to EvOL-Neuron II, this algorithm described by Stiefel and Sejnowski [67] uses a list of morphometric parameters and a genetic algorithm to evolve morphologies. One of the main reasons to consider this algorithm was the absence of a stochastic component, in contrast to the previous models. Moreover, many parameters are allowed to vary with distance from the soma, which enhances the number of possible morphologies. Stiefel's algorithm uses 14 morphometric parameters (see Table 5.4) to encode the genome, obeying requirement IV. As the description lacks a stochastic component, trees can always be reproduced from the reduced set of 14 parameters.

Table 5.4 – Parameters used by Stiefel model. From Stiefel and Sejnowski [67].

Index	*Parameter*	*Encodes*	*Initial Values for GA*
0	m_0	Terminal degree	32 ± 16
1	L_0	Peak length to branch point	100 ± 100
2	L_μ	Position of the peak length to branch point	100 ± 10
3	L_σ	Half width of the Gaussian determining the length to branch point.	200 ± 100
4	a_0	Peak asymmetry for the splitting of m.	0.5 ± 0.2
5	a_μ	Position of the peak asymmetry for the splitting of m.	100 ± 10
6	a_σ	Half width of the Gaussian determining the asymmetry for the splitting of m.	100 ± 10
7	Θ_0	Peak branching angle.	$\pi/10\pm\pi/10^5$
8	Θ_μ	Position of the peak branching angle.	10 ± 10
9	Θ_σ	Half width of the Gaussian determining the branching angle.	100 ± 10
10	d_0	Initial dendritic diameter.	10 ± 2
11	Δd	Slope of the change of the dendritic diameter.	0.04 ± 0.004
12	α_0	Initial orientation.	$\pi/2\pm4\pi$
13	β_0	Initial orientation.	$0\pm4\pi$

The model is also able to generate neurons in which multiple dendritic trees originate from the soma, requiring a separate chromosome for each dendrite. The steps from genome to phenotype are described in Figure 5.8.

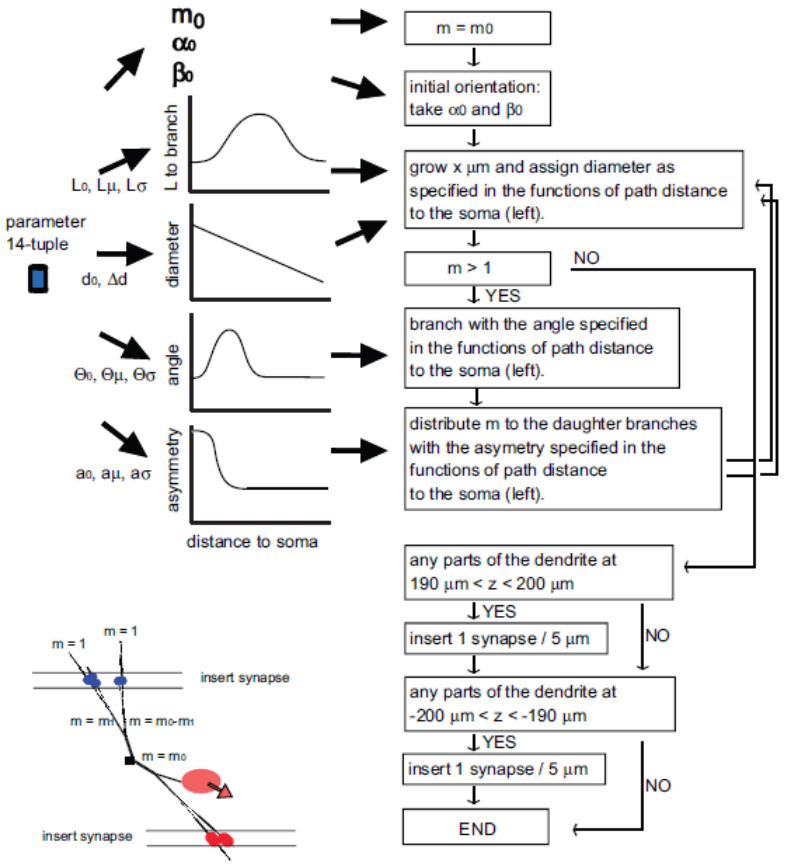


Figure 5.8 – Steps from genome to phenotype encoding in the Stiefel model. From Stiefel and Sejnowski [67].

Problems

To generate a range of morphologies, the algorithm encodes features of the dendrites using a single Gaussian function whose parameters (distance of centre from soma and standard deviation) are part of the genome and can be varied by the genetic algorithm. However, this kind of function does not allow the algorithm to generate all the possible morphologies with a specific terminal degree (requirement I). Indeed, the constraints that the distance function has a single maximum and that the same Gaussian distance function specifies a given parameter in all of the sub-branches simultaneously, unduly restricts the space of tree morphologies. One example of a neuron that cannot be generated is the degree-9 tree (a tree with 9 terminal points) given in Figure 5.9. This is because the Gaussian asymmetry index (see asymmetry graph in Figure 5.8) is used to distribute the branches to the left and right sides of the tree. To adapt the algorithm for my purposes, it would need to use a more complicated and flexible set of functions than a single Gaussian. In fact, to generate this tree an asymmetry function with two maxima is required since the growth of the tree requires two places where the asymmetry needs to be high. Larger, more complicated, trees

of the type needed in this research would require a function with many maxima.

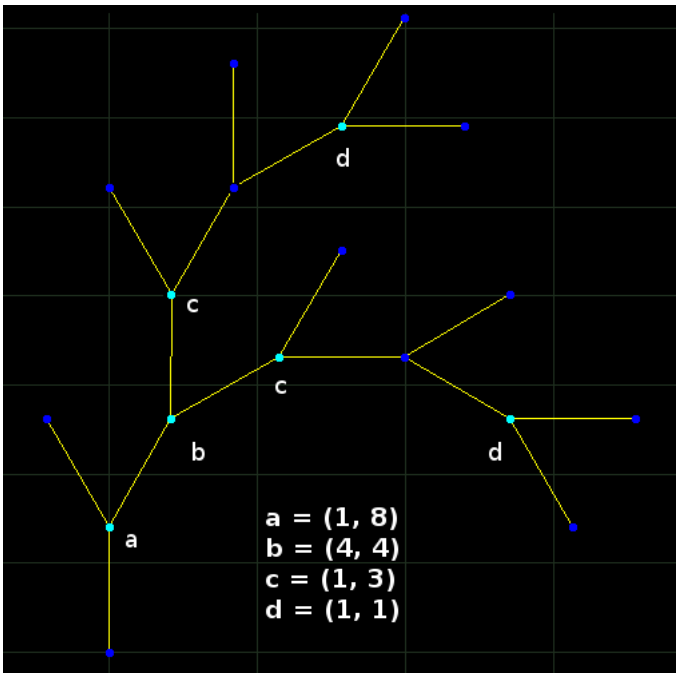


Figure 5.9 – Example of a tree that can not be generated by Stiefel's algorithm. The highlighted points (**a-d**) are the examples of branch splitting points. As can be seen using the partition representation (the numbers between brackets), the points **a** and **c** are asymmetric points and the points **b** and **d** are symmetric points. Since this tree has more than one asymmetric point (**a**, **c**), the function to generate this tree needs two maxima, which is not possible using Stiefel's algorithm (a Gaussian function just has one maximum).

5.4 Conclusion

In this chapter, I have described all the algorithms in the literature which initially looked appropriate for optimising neuronal morphologies for pattern recognition. However, having tested them all, none of them was suitable for my proposed task. The main problems found were:

- the use of a stochastic component as seed to generate the morphologies;
- the need of an extensive parameter list to generate morphologies covering the whole search space;
- the use of inappropriate functions like Gaussian distributions, restricting the range of possible morphologies.

As none of the models analysed in this chapter were able to achieve all the requirements specified, a new tree generation algorithm was implemented using the best features of the existing algorithms. This tool, presented in the next chapter in Section 6.4, was designed to meet all six requirements described for the pattern recognition task and consequently generates all the morphologies desired.

Chapter 6

Exploration of Dendritic Morphologies for Pattern Recognition

6.1 Introduction

In this chapter, three ways of generating tree morphologies are introduced. Firstly, for trees of small order (having few terminal nodes), it is possible to generate every topologically different binary tree using an exhaustive search algorithm. As this is not possible for higher tree orders, I implemented an algorithm which generates samples of trees with a particular number of terminal points. In order to explore unique morphologies, my algorithm allows me to force the trees to be particularly symmetric or asymmetric. Both algorithms to generate high and low tree orders are described in Section 6.3. Finally I describe my evolutionary algorithm (Section 6.4) for finding morphologies that give rise to neurons that are functionally desirable.

All the three algorithms discussed in this chapter use the same dendritic tree representation based on the partition notation defined by van Pelt and Verwer [77]. The details are given in the following section (Section 6.2).

6.2 Representation and Construction of Dendritic Trees

To represent and generate dendritic trees, the partition notation from van Pelt and Verwer [77] was used. A partition at a bifurcation point in a binary tree is defined by a pair of numbers which denotes the degree of each subtree. Each partition represents a bifurcation point, where the nodes of its subtree are split into those in its left branch and those in its right branch. The topology of the whole tree can therefore be characterised by the set of partitions at its bifurcation points. So for example, the most asymmetric tree with 5 terminal points can be described by the partitions

5(1 4(1 3(1 2(1 1)))) (see Figure 5.1, top right side).

In general, a binary tree T_n with n terminal points can be described using the following rule:

$$T_n = n(T_a\ T_b) \tag{6.1}$$

where $a + b = n$; a, $b > 0$ and $T_1 = 1$.

For example, for the tree described above with 5 terminal points we have:

$$T_5 = 5(T_1\ T_4) = 5(1\ 4(T_1\ T_3)) = 5(1\ 4(1\ 3(T_1\ T_2))) = 5(1\ 4(1\ 3(1\ 2(T_1\ T_1)))) = 5(1\ 4(1\ 3(1\ 2(1\ 1))))$$

This representation was used for binary trees not only because it is commonly employed in this field, but also because it facilitates the construction of trees with particular values of asymmetry. It is also helpful when applying genetic operators like mutation and crossover used in the evolutionary algorithm described later in Section 6.4.

6.3 Systematic Generation of Dendritic Morphologies

To understand the relationship between neuronal morphology and pattern recognition, the initial task required a systematic algorithm to generate the dendritic trees. The algorithm to implement this was based on the work from van Ooyen and collaborators presented in Section 5.2.1, where simple morphologies were compared based on their asymmetry index. In this section, I present the details about how the dendritic trees were generated, starting by comparing the most distinct dendritic morphologies: the fully symmetric and the fully asymmetric trees. In the following sections, I describe the two algorithms which I have implemented to cover the whole range of dendritic trees (Section 6.3.2) and to generate samples of trees (Section 6.3.3) given a certain number of terminal points. The pattern recognition results obtained from the trees generated here are presented later in Chapters 7 and 8.

6.3.1 Fully Symmetric and Fully Asymmetric Trees

To understand the effects that dendritic morphologies can have on pattern recognition performance, simple morphologies based on van Ooyen's model (presented in Section 5.2.1) were chosen to start this research. As was shown by van Ooyen and his collaborators [76], the most symmetric and asymmetric morphologies exhibited completely different firing patterns for neurons with the same membrane properties and ionic conductances (see Figure 5.3 on page 47). So given this fact, I decided to initially compare the results using these two distinct morphologies (Figure 6.1).

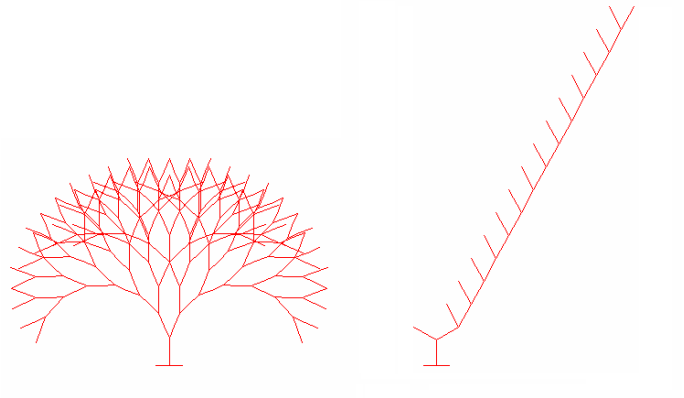


Figure 6.1 – Fully symmetric tree with 128 terminal points (left side) and fully asymmetric tree with 16 terminal points (right side). These morphologies were generated using the rule presented in Section 6.2 and plotted by the Neuron simulator tool [22].

These morphological trees were generated using the tree generation rule presented in Section 6.2. The results obtained in pattern recognition tasks comparing both morphologies are presented in Section 7.4 when running passive models and in Section 8.4 for active models.

6.3.2 Trees Exhaustively Generated

After having compared two extreme morphologies, the next step was to compare a large set of neuronal morphologies. The initial idea was to cover the whole search space by generating all possible binary trees for a given number of terminal points. To do this, I implemented a program where the main steps are explained using the pseudocode below:

Listing 6.1 – Exhaustively tree generation algorithm

```
1 For a given number of terminal points N, create every possible partition (a
    b) such that: a <= b and a + b = N
2 For each new branch a and b, call the partition method recursively (step 1)
3 Repeat steps 1–2 until N = 1 (when a leaf has been reached)
```

As trees are recursive structures, the best method to generate these morphologies uses a recursive algorithm as described above. Due to this fact, the algorithm was implemented in Common Lisp, a functional programming language which has a natural representation of trees. See the full Lisp code in Appendix B.1.

This algorithm was tested for different tree orders. As previously shown by Harding [20], the number of possible trees S_n for a given number of terminal points n is calculated by the following equation:

$$S_n = \frac{\alpha^{3/2}\beta}{2\pi^{1/2}} \cdot \frac{\alpha^{-n}}{n^{3/2}} \tag{6.2}$$

where Harding demonstrates that $\frac{\alpha^{3/2}\beta}{2\pi^{1/2}} = 0.3187766$ and $\alpha^{-1} = 2.4832535$. This equation gives an approximated number of 8.042×10^{46} trees for 128 terminal points, which is the main tree order investigated in this study. However, this large number of trees makes the simulations of this tree order computationally infeasible. As a consequence, to compare a whole set of morphologies, the tree size studied had to be reduced. So, the chosen tree order was 22 terminal points, which using the Lisp code explained above generates a total number of $1,721,998$ trees. This number is close to the one calculated by using Equation 6.2, which gives a total number of $1,514,661$ trees. The reason for this difference is that my Lisp code generated some trees of the same ambilateral type, which means they are not electrophysiologically unique trees. However, the removal of these extra trees was computationally quite expensive, so I opted to keep these trees and carry on the pattern recognition task using them.

Samples of the trees generated are presented in Figure 6.2. The performance of these trees when executing the pattern recognition task is presented later in Section 7.7.

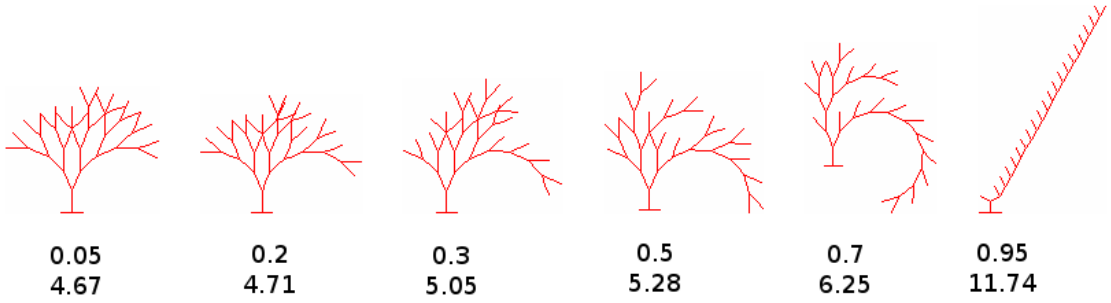


Figure 6.2 – Samples of tree morphologies with 22 terminal points generated by the exhaustive tree generation algorithm. The trees are ordered according to their degree of symmetry, where the values shown indicate asymmetry index (top) and mean depth (bottom) of the respective tree.

6.3.3 Trees Selectively Generated

As it was not possible to simulate the whole range of morphologies for the desired neuronal morphology order (128 terminal points), a third method was used, comparing randomly generated morphologies. To achieve this, I implemented an algorithm to produce random samples of morphological trees with a given number of terminal points. The following pseudocode explains the main steps of the algorithm (see the full Lisp code in Appendix B.2):

Listing 6.2 – Selective tree generation algorithm

1 For a given number of terminal points N, create a partition by randomly splitting N in two branches (a b), for a and b such that: $a <= b$ and $a + b = N$

2 Use parameter $asym$ to define whether to generate trees more symmetric ($asym = 0$) or more asymmetric ($asym = 1$)

3 Use parameter $bias$ to control the asymmetry of the tree, where $bias$ varies from 0.01 to 0.5. To generate random partitions use $bias$ equal to 0.5; for most symmetric or asymmetric partitions use $bias$ equal to 0.01. For example for $m = 100$ $bias = 0.1$: $asym = 0$ => (46 54) to (49 51); $asym = 1$ => (5 95) to (10 90)

4 For each new branch a and b, call the partition method recursively (step 1)

5 Repeat steps 1–4 until $N = 1$ (when a leaf has been reached)

6 Repeat steps 1–5 until number of desired trees is achieved

This algorithm differs from the one to generate trees exhaustively mainly at the splitting function. Instead of generating the whole possible range of partitions for each pair of branches a and b, this algorithm depends on a bias value which controls the partition generation. For example, if bias is equal to 0.1, the left branch a will have up to 10% of the total number of terminal points for that branch, when generating the most asymmetric tree. In the example shown in the Program Listing 6.2, where the number of terminal segments is 100, the possible partitions generated in this case will be within the range (5 95) to (10 90). In the case of the most symmetric tree, the bias value means this value will be subtracted from half of the total number of terminal points for that branch. For the same example given previously, the possible partitions generated are between (46 54) and (49 51). In summary, low bias is more likely to generate trees with extreme values, which means more symmetric or asymmetric trees. On the other hand, if the bias is equal to 0.5 it generates completely random trees.

A sample of trees generated with 128 terminal points using this algorithm is presented in Figure 6.3 and the results from these trees are presented later in Section 7.7.

6.4 Evolving Dendritic Morphologies

After analysing the four different algorithms from the literature with no success, as presented in Section 5.3, I decided to develop my own algorithm. This algorithm named *Evol-Patrec* (for Evolutionary Pattern recognition) had as its main purpose the evolution of optimal neuronal morphologies for performing pattern recognition. Using a tree generation procedure based on van Pelt

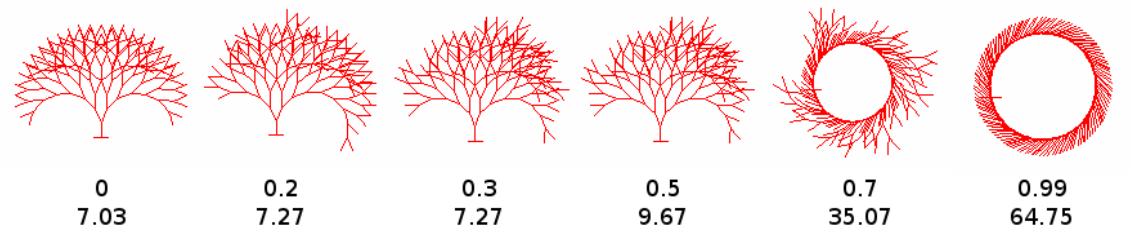


Figure 6.3 – Sample of tree morphologies with 128 terminal points generated by the selective tree generation algorithm. As previously shown for trees with 22 terminal points, the trees are ordered according to their degree of symmetry. Upper values indicate the tree asymmetry index and lower values its mean depth. The last two trees are plotted with the original visualisation, where the angles of each branch are not specified.

and Verwer's work [77] and an evolutionary algorithm developed in C++ programming language, this tool was designed to meet all six requirements described in Section 5.3.1. As in Stiefel's algorithm (see Section 5.3.2.4), Evol-Patrec did not use a stochastic component to generate trees, allowing full control over the morphologies generated as specified in requirement V (Section 5.3.1). The optimisation process follows a sequence of four steps to generate all the different morphologies, as shown in Figure 6.4.

The evolutionary algorithm (EA) uses the partition notation, defined by van Pelt and Verwer's work [77] and previously described in Section 6.2, as its genotype. To initialise a partition tree, the EA needs as input the number of terminal segments m, which identifies the degree of the full tree. In Figure 6.4, the genotype specifies a tree of degree 8 ($m = 8$). The initial population was composed of 50 individual tree morphologies, each of them represented by a set of partitions randomly initialised. The random initial population was produced using Program Listing 6.2, with no symmetry bias (bias = 0.5, asym = 0). For the active models (used in Chapter 8), the genotype also included a second chromosome: a list of parameters used to define the morphological and pattern recognition features, such as compartmental length and synaptic strength. More detail about this chromosome is presented in Chapter 8.

The next step of the optimisation procedure is the translation of the genotype to a model neuron, which forms the phenotype. The model neuron is produced by converting the tree morphology to a NEURON *hoc* format [22]. This format describes the morphology of a multi-compartmental model in the NEURON simulator language.

After the phenotype or compartmental model has been generated, the fitness value is assessed to check if the suitable morphology has been generated. The fitness is computed by calculating the signal-to-noise ratio of the neuronal output for pattern recognition. This ratio is measured as previously described in Section 3.3.1.1 on page 25. Then the actual fitness of each individual is evaluated by calculating the average s/n for a given number of pattern sets (trials). The fitness

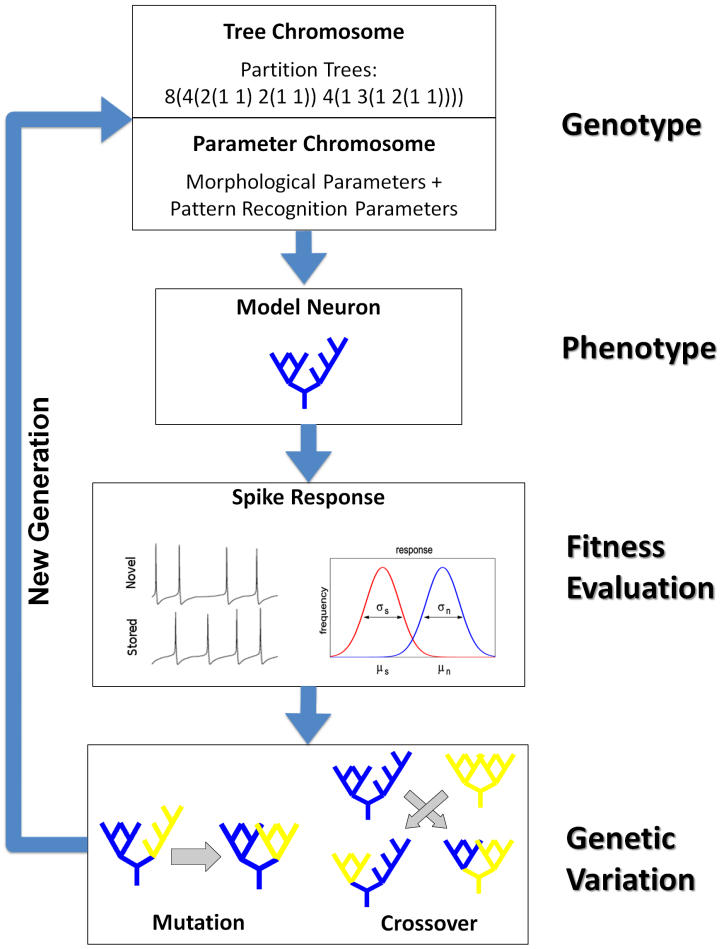


Figure 6.4 – Evolutionary algorithm used to optimise neuronal morphologies. The diagram shows the sequential steps followed to generate all the morphologies. First, the genotypes are randomly initiated specifying the binary tree structure. Then, the genotype is expressed as a model neuron phenotype and then converted to a multi-compartmental model. After that, the fitness values are assessed by evaluating the pattern recognition performance (the separation of the distributions of responses to stored and novel patterns). Finally, genetic variation is introduced using a process where the genes are modified by crossover and mutation operators. This final step generates the morphologies which will be part of the population in the next generation. Source code available at http://code.google.com/p/evol-patrec.

function calculation is discussed in more detail in Section 7.3 for passive models and Section 8.3 for active models.

As the final stage of the evolutionary process, the genes are submitted to genetic variation. This stage is composed of four main steps:

1. Select the parents.
2. Crossover the parents to generate new individuals (offspring).
3. Mutate offspring to introduce genetic variability in population.
4. Replace old individuals in the population by the new ones.

These steps are explained in more detail in the following sections.

The files needed on each EA step are automatically generated by the algorithm written in C++ language, where the main steps are explained by the pseudocode below:

Listing 6.3 – Evolutionary algorithm written in C++ code

```
1 create new population by applying mutation and crossover in the genotypes (
    tree and parameter chromosomes)
2 for each individual in the population, generate the model description in the
    NEURON simulator language and the set of patterns used in the pattern
    recognition task
3 send files generated in step 2 to a workstation to run the simulation in
    NEURON
4 when the simulations finish, NEURON simulator saves the output files with
    the neuronal response (EPSP or spike traces) into output files
5 program reads files generated in step 4 to calculate signal-to-noise ratio
    for each individual
6 repeat steps 1-5 until termination codition achieved (number of generations
    or found desired fitness)
```

6.4.1 Genetic Operators

6.4.1.1 Selection

To ensure that the fittest genes are passed to the next generation without any genetic modification, a selection process called elitism is used. Elitism ensures that the best individuals are not lost when genetic operators like mutation and crossover are applied over the population [48]. For all the tasks performed by the EA, the individuals are ranked by their fitness and the best 10% of the individuals are copied to the next generation.

The remaining 90% of the new population are selected, to apply crossover and mutation, using a rank order roulette-wheel based selection. This method combines two classical selection algorithms: roulette-wheel and rank selection. In the roulette-wheel selection, each individual is assigned to a slice of a roulette wheel where the slice size corresponds to the fitness value of that individual [48]. To select the parents, the wheel is spun N times, where N is the number of the individuals in the population. In the rank selection, the individuals are ranked by their fitness where the least-fit individual receives a rank 1 and the fittest a rank N; the slice size of the roulette wheel is now the rank of the individual. Combining rank order with roulette-wheel selection allows an easy algorithm to be implemented. At the same time, it prevents the EA from converging too quickly,

as the best individuals which occupied a large part of the roulette-wheel are not selected every time the wheel is spun [62]. The rank order roulette-wheel based selection was implemented by the following pseudocode :

Listing 6.4 – Rank order roulette-wheel based selection algorithm

```
1 rank the population in ascending order, using the fitness value of each
    individual
2 select a random number I between 1 and population size N
3 select a random number J between I and N
4 return the individual of index J
```

The code above is used twice to selects individuals with high fitness to be a pair of parents, which thereafter undergoes modifications by crossover and mutation operations to produce a pair of offspring.

6.4.1.2 Crossover

Crossover is a genetic operator where two individuals are combined to produce offspring. To combine the neuronal morphologies within the EA population, I used a common form of crossover called *subtree crossover* [52]. This operation was implemented by selecting a random subtree in each parent tree and swapping them. The resulting offspring are produced by copying the original root branches from each parent and the selected random subtrees from each other parent. To ensure that this operator only produces trees of the same order (number of terminal points), only the subtrees that have the same order between the parent genes can be selected for crossover. An example is shown in Figure 6.5, where the subtree chosen has the same order ($m = 6$) in both parents, guaranteeing that the offspring have the same number of terminal points. For the tasks performed, every individual within the population is eligible to be submitted to a crossover operation as the best-fit individuals are already kept by the elitism operation.

As I mentioned earlier, a second chromosome is used in the active models which contains a sequence of parameters (described in Section 8.6.1). To crossover two of these chromosomes, a simple point-crossover is applied that means the parameters will be simply swapped between the parents. More detail about this crossover operation is given later in Section 8.6.1.

6.4.1.3 Mutation

Mutation introduces genetic variability by randomly modifying the individual genetic information. In genetic algorithms, it prevents the algorithm from falling into local minima [62]. Mutation

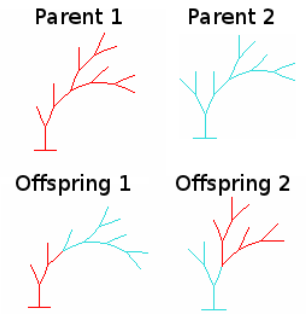


Figure 6.5 – Example of crossover operation in trees with 8 terminal points. The branches selected in both parents have the same order (6 terminal points) to ensure that the total number of terminal points is maintained as described by requirement VI (Section 5.3.1). The resulting offspring show that they originated by swapping the branches from both parent trees (see the distinct colours from each parent branch).

was implemented by randomly selecting a point in the tree and then replacing the subtree at this point in the tree with a randomly generated new subtree. For this work, like in the crossover operation, mutation also needs to ensure that the resulting dendritic trees have the same number of terminal points. To do this, the subtrees generated after mutation need to keep the same number of terminal points as the randomly selected ones on the parent genes. This rule also assures that the trees generated maintain the electrophysiological uniqueness, obeying requirement II described in Section 5.3.1. An example of mutation is shown in Figure 6.6 where a new tree with 8 terminal points was produced by mutating its parent tree.

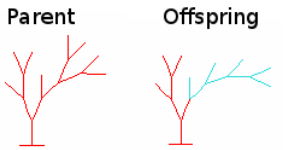


Figure 6.6 – Example of mutation in a tree with 8 terminal points. In the parent tree, a branch with 5 terminal points in its right side was selected. After mutation, the new tree branch was generated (blue branch), keeping the same number of terminal points, only varying its topology.

In the active model where there is a second chromosome, mutation simply changes one or more parameters with low probability. The details of this operation are given later in section Section 8.6.1.

6.4.2 Testing the Evolutionary Algorithm

To test the performance of the evolutionary algorithm described here, before using it for the real task to be investigated in this thesis, three simple tasks were selected to generate unique morphological trees: finding the most symmetric tree, the most asymmetric tree and trees with a certain mean depth. These tasks used the metrics described in Section 5.2.2, so it could be tested

if the EA was able to generate the desired trees varying only their topologies and using the metrics as fitness functions.

All three EA tasks were performed using the same parameters:

- a population composed by 100 individuals;
- a fixed rate for elitism (10%), tree mutation (20%) and tree crossover (100%);
- a fixed number of terminal points (16).

The first EA task was to generate the most symmetric morphology. The fitness function used was the asymmetry index of each individual (described in Section 5.2.2.1). For the degree of trees chosen, which have 16 terminal points, the desired fitness was an asymmetry index equal to zero. As shown in Figure 6.7, the most symmetric tree was obtained after only two generations. The figure shows all the individuals which were involved in the evolutionary process to obtain the final tree (indiv 16).

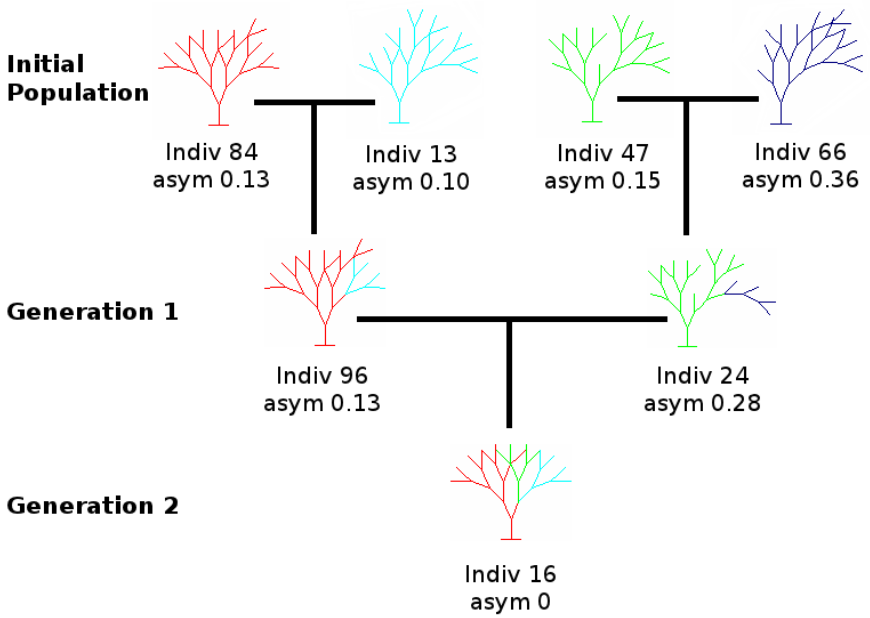


Figure 6.7 – Genealogical tree showing the evolution of the most symmetric morphology. All the ancestors of the final individual evolved (indiv 16) are shown.

The second task performed was to evolve the most asymmetric tree. This task was similar to the previous one, as the fitness function was implemented using the tree asymmetry index. For trees with 16 terminal points, the most asymmetric trees has an asymmetry index equal to 0.93, which was defined as the desired fitness of this task. However, differently from the previous task, which quickly evolved the desired tree after only two generations, this task required many more generations (see Figure 6.8). This happened mainly because the EA had a low mutation

rate (20%). Moreover, to speed up the evolutionary process it was necessary to implement a new genetic operator where the tree branches could be moved around the tree, like a transposition mechanism [61].

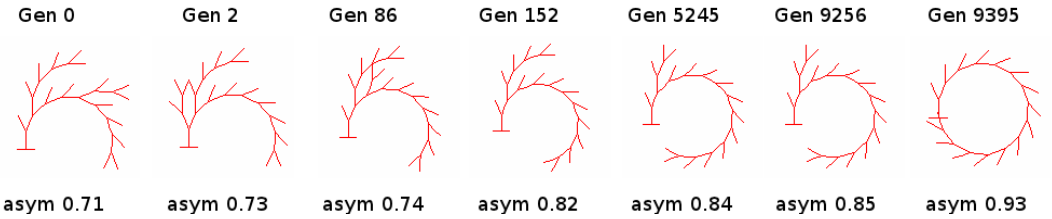


Figure 6.8 – Evolution of the most asymmetric morphology for trees with 16 terminal points. The figure shows each generation where the fittest individual was different from the fittest one in the previous generation.

The last task used to test the EA performance was to generate all possible trees with a certain mean depth (metric described in Section 5.2.2.4). The mean depth chosen was 7, which corresponds to the trees with asymmetry index equal to 0.68 and mean path length equal to 35 μm. The first morphology with the chosen mean depth was evolved at generation 50. After 100 generations, 86% of the total population was composed by this specific morphology (Figure 6.9).

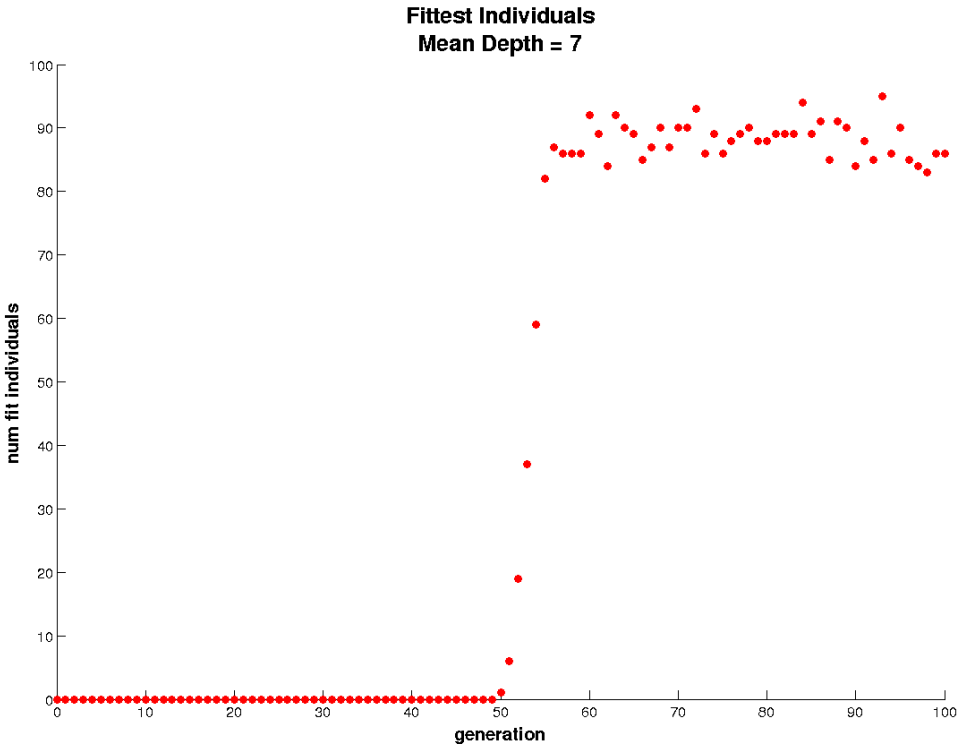


Figure 6.9 – Evolving trees with mean depth equal to 7. This graph shows the number of individuals evolved with the required topology over the generations. After 100 generations, the EA was halted and a total of 86 individuals with this mean depth were found. However, the highest percentage of fit individuals was obtained at generation 93, where 95% of the population was composed of the desired morphology.

6.5 Conclusion

In this chapter I have described all the algorithms which I developed to generate dendritic morphologies whether in a systematic way (Section 6.3) or by evolving them using a genetic algorithm (Section 6.4). All these algorithms used the same dendritic tree representation, based on the binary tree structures of van Pelt and Verwer [77], providing an easy definition for trees, showing a clear tree structure visualisation, and facilitating the genetic modification applied by the EA operators.

The tasks performed using the evolutionary algorithm described in this chapter could test all the genetic operators described in Section 6.4.1. Moreover, I could also test the EA performance, showing that some morphologies were evolved more quickly than others. As a result, the EA seems to be ready to perform the final task which it was designed for: the evolution of morphologies for pattern recognition. The results of this task as well as the results from the trees generated systematically are presented in the next two chapters: Chapter 7 describes the results for morphologies with no active conductances, called here passive models; and Chapter 8 presents the results obtained from active models, which means the morphologies with active conductances.

Chapter 7

Effect of Dendritic Morphology in Passive Neurons

7.1 Introduction

In the previous chapter, I presented the algorithms I use to generate and modify the neuronal morphologies to start my study about pattern recognition. The next step was to understand how different dendritic morphologies could affect the neuronal firing pattern, and consequently, the capability of neurons for storing information. In order to study the implications of dendritic morphology for pattern recognition, the initial experiments used compartmental models with passive parameters.

Following the methodology described in the previous chapter, four different experiments were designed to compare the dendritic morphologies generated. The first experiment was to study the fully symmetric and the fully asymmetric tree which allowed me to compare the results of these two distinct morphologies when performing the pattern recognition task. The idea of this experiment was trying to find if there was, in fact, any difference between the performance of the two most extreme morphologies. If little differences were found, the following experiments were unlikely to be successful. However, as the most symmetric and asymmetric trees showed a clear difference in pattern recognition performance, a second experiment was designed to evolve neuronal morphologies using the EA previously described in Section 6.4. The results found in this experiment were generally disappointing. So, in order to find out why the EA did not work, I decided to explore the morphological space more systematically. To do this, a third experiment was designed to evaluate all possible ambilateral trees that could be generated with a small number of terminal points. Finally, to cover the morphology range of trees with the desired degree, the last experiment was designed to compare the results from samples of trees with 128 terminal points. The results of

these four experiments are presented in Sections 7.4 to 7.7, where I investigated how the generated morphologies performed when presented with either a fixed or different sets of patterns. Furthermore, I was able to evaluate which of the three morphometrics studied, asymmetry index, mean depth and variance of depth, could characterise better the morphologies studied when performing the pattern recognition task.

7.2 Model Neurons

To evaluate how the neurons described in the previous chapter performed the pattern recognition task, a set of common parameters was defined to be used in both types of neuronal morphologies: the ones generated systematically (presented in Section 6.3) and the evolved morphologies (Section 6.4). All parameters were based initially on previous works [76, 18], however most of these parameters were hand-tuned later on, trying to optimise the pattern recognition performance. The parameters used are related to the morphological and pattern recognition features and are detailed in Table 7.1.

Table 7.1 – Morphological and pattern recognition parameters used to generate the morphologies in passive models.

Morphological Parameters		
terminal points (m)		128
dendritic compartments ($2m-1$)		255
soma	length	20 μm
	diameter	20 μm
dendritic compartment	length	10 μm
	diameter	2.5 μm
	tapering	false

Pattern Recognition Parameters	
pattern size	255 (1 bit per synapse)
patterns presented	20 (10 stored + 10 novel)
active synapses	25 (10% of total synapses)
synaptic strength	1 nS
number of spikes	1 per stimulus
spike interval	10 ms
noise	false

The membrane properties used are the same as those defined in van Ooyen's model (Table 5.1), except for the reversal potential E_{leak} which was defined as -65 mV. This value was based on the resting potential defined in a previous study about pattern recognition in CA1 pyramidal cells [18]. The spike interval value defined in Table 7.1 gives the mean interval between the spikes in the spike train, which have a spike distribution governed by a negative exponential distribution. Thus, this value affects the timing of the first spike and therefore needs to be specified.

The neuronal degree of 128 terminal points was chosen to allow a reasonable pattern size (with 255 bits) to be presented, as each dendritic segment had one synapse per compartment, located in the middle of each compartment. This synaptic distribution was chosen as it is the

simplest way of positioning the synapses, whilst allowing a clear comparison between the topology of the morphological trees. The lengths and diameters for soma and dendritic compartments were also based on the values used in van Ooyen's model (Section 5.2.1). However, the dendritic compartmental length, combined with the pattern recognition parameters such as synaptic strength and number of spikes, were tuned to acquire a clear distinction between the neuronal outputs when presenting stored and novel patterns.

The input presented to the model was similar to that given in the Purkinje cell model, where a number of random binary input patterns were generated. To present the patterns to the model, each bit of the input pattern was mapped to a specific synapse. To do this, each synapse was numbered by the location of its dendritic compartment in the tree. The compartments were indexed from the left side of the tree to the right. An example is given in Figure 7.1 where the same pattern was mapped to the dendritic trees from both the most symmetric and the most asymmetric morphologies.

Pattern	Synapse
0	1
1	2
1	3
0	4
1	5
0	6
1	7
0	8
0	9
1	10
1	11
1	12
0	13
1	14
0	15

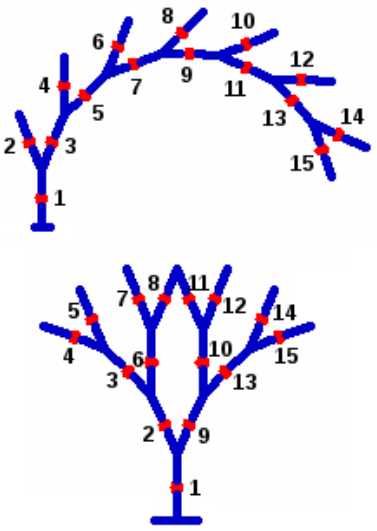


Figure 7.1 – Mapping pattern to trees. The diagram shows how the same input pattern is mapped to each synapse in the most symmetric and the most asymmetric morphologies.

Twenty input patterns were presented, where half of these patterns had previously been stored by an ANN. This ANN, previously described in Section 3.3.1.2, was trained using a LTP learning rule where the synaptic weight was increased by a value of 1 each time an active input pattern was presented. The synaptic input was driven by an excitatory synapse with an alpha function like conductance [9] with a rise time (*tau1*) and a decay time (*tau2*) constant, where $tau2 > tau1$ (see values in Table 7.2 left side). The peak conductance was calculated from the synaptic weight learned by the ANN, where the weight w was translated to a peak conductance of $w\ nS$. The passive model did not receive any background input.

Table 7.2 – Synaptic properties and simulation parameters used in the passive models. The type of synapse produces as alpha function like conductance, where the parameters are described in the text. The simulation parameters are defined as: t_{sim} is the simulation run time, dt is the integration time step, t_{stim} is the time the stimulus is presented to the neurons and V_{init} is the initial membrane potential.

Synaptic Properties	
Synaptic type	*Exp2Syn*
Rise time (tau1)	0.2 ms
Decay time (tau2)	2 ms
Reversal potential	0 mV

Simulation Parameters	
t_{sim}	200 ms
dt	0.025 ms
t_{stim}	50 ms
V_{init}	-65 mV

All the simulations were performed using the Neuron simulator [22]. The parameters used in each simulation are summarised in the Simulation Parameters table (Table 7.2), where the values were based on previous pattern recognition simulations [18, 66].

7.3 Performance Evaluation

The pattern recognition performance was measured in a similar manner to that used in the Purkinje cell model (Section 3.3.2.1), using the s/n ratio between the response distributions to the stored and novel input patterns. However, in this study using passive models, the s/n ratio was calculated over the EPSP amplitude of the neuronal responses as shown in Figure 7.2.

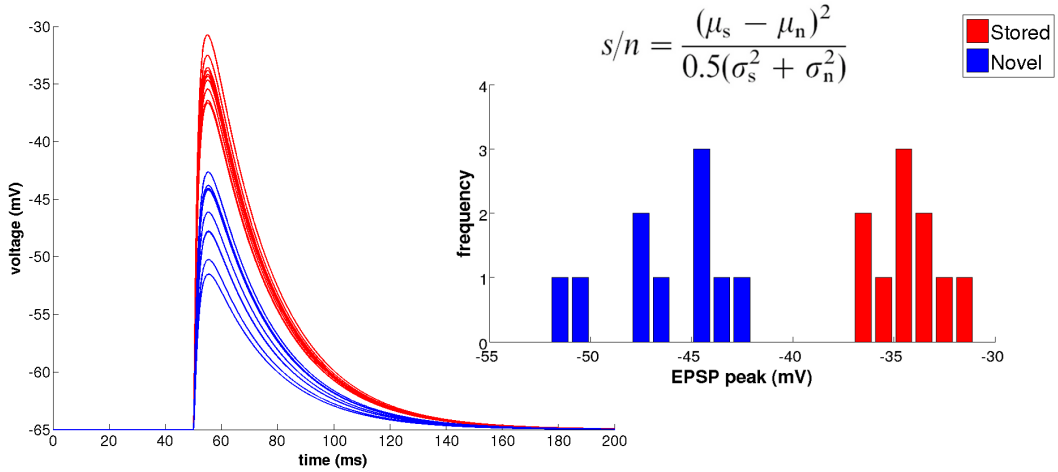


Figure 7.2 – Typical EPSP response for neuronal morphologies in passive models. The voltage traces show the responses to 10 stored patterns (red traces) and 10 novel patterns (blue traces). The inset on the right side of the figure shows how the s/n ratio is calculated using the EPSP peak responses for both stored $\boldsymbol{s}$ and novel $\boldsymbol{n}$ patterns, where the actual s/n value found here is equal to 23.76.

7.4 Comparing Fully Symmetric and Asymmetric Morphologies

The first experiment performed on the passive models was to compare the pattern recognition performance for the fully symmetric and fully asymmetric morphologies. These distinct morphologies were obtained as previously described in Section 6.3.1, where trees with 128 terminal points were generated. For this experiment, the s/n ratio was calculated over the EPSP responses when presenting a single set of patterns (10 stored + 10 novel patterns) for both morphologies, as shown in Figure 7.3.

From the results shown in Figure 7.3, it is possible to see a clear distinction between the traces resulting from stored and novel patterns for the fully symmetric morphology (right hand side of each of the lower traces of the figure). As a consequence, the s/n ratio for the most symmetric morphology is more than twice as large as the one calculated from the most asymmetric morphology when presenting the same pattern set. The differences between the EPSP amplitudes can be explained by the different locations of the synapses over the whole dendritic tree. It is clear from the traces shown in the left side of Figure 7.3 that, the further the active synapses are located from the soma, the lower are the resulting EPSPs amplitudes. Consequently, the EPSP amplitudes in the asymmetric morphology have a larger variance compared to the symmetric one, as shown in the bottom graphs of Figure 7.3, which results in a lower s/n ratio (see equation in Figure 7.2). So, from these results, we can conclude that the most asymmetric morphology performs less well than the most symmetric one.

7.5 Evolutionary Algorithm

Since there are clear differences in s/n performance between different morphologies, it makes sense to attempt to evolve an optimal morphology using an evolutionary algorithm. So the second step of this study of dendritic morphologies in passive models was to evolve morphologies using the evolutionary algorithm detailed in Section 6.4. In the present section, I describe details of the EA parameters used to evolve the passive morphologies. I also describe implementation issues that were found when running the experiments and how they were solved. Finally I present the results found from the evolved morphologies.

7.5.1 Chromosome Details

As already mentioned in the previous chapter, the EA genotype in passive models was composed of a tree chromosome which defined the morphological parameter used to generate the neuronal morphologies studied here. These parameters as well as the pattern recognition parameters were

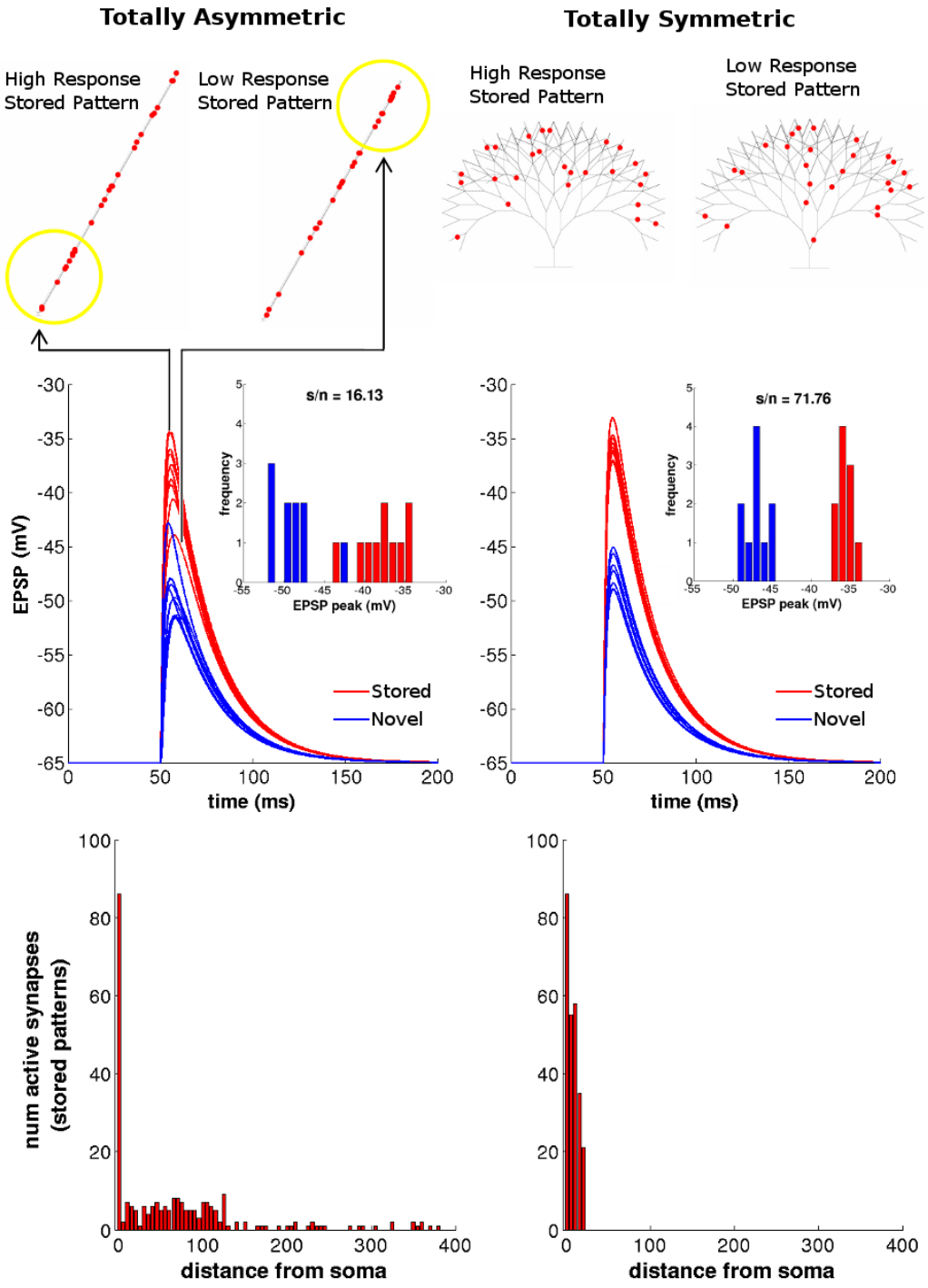


Figure 7.3 – Example of traces comparing the most symmetric and the most asymmetric morphologies. The s/n ratios, shown on the top of each histogram, are calculated using the EPSP peaks resulting from presentation of both sets of stored and novel patterns, red and blue traces respectively. On the top of each of the EPSP traces the neuronal morphologies used in this experiment are shown, where the red dots represent the location of each active synapse for the lowest and highest response obtained from stored patterns (right and left morphologies respectively). The yellow circles shown in the asymmetric morphologies indicate the location of some of the active synapses. It can be seen that where there is a high response, there are more active synapses close to the soma; whereas where there is a low response, there are more active synapses at the far end of the dendritic tree. The synaptic distributions of active synapses for the stored patterns (bottom graphs) also show that the performance is higher in the most symmetric morphology (right graph) as it has a higher number of active synapses closer to the soma and consequently, a lower variance of the synaptic distribution, when compared to the most asymmetric morphology (left graph).

detailed in Table 7.1, and other simulation properties were also defined in that section (Section 7.2).

For all experiments using the EA in the passive models, the population was composed of 100 individuals, initialised with random morphologies (as explained in Section 6.4). The fitness of each individual was evaluated by calculating the average s/n ratio for the given number of pattern sets.

The genetic operators rates were defined to cover all set of simulations here presented. As mentioned in Section 6.4.1.1, 10% of the total population was selected to be passed to the next generation by elitism. The morphological trees were modified by mutation and crossover operations, using a rate of 20% and 100% of the population respectively.

7.5.2 Implementation Issues

A significant implementation issue that needed to be addressed was assessing the fitness of any individual in the population. To run five different sets of patterns (5 trials), composed by 20 patterns each, each individual could take up to 2 minutes of computational time to be assessed, using a single 64 bit core. This means that each individual was taking approximately 12 seconds per trial in passive models, and 22 seconds per trial in active models (which is explained later in Chapter 8). It was therefore not feasible to run the EA in a normal SISD (single instruction, single data stream) fashion. So, the evaluation of the population fitness needed to be parallelised, using a MIMD (multi instruction, multiple data) technique. In order to do this, a parallel version of the code was implemented using the MPI (message passing interface) specification, which combined C++ libraries and Neuron simulator code [22]. To run the parallel code, a cluster composed of 48 nodes with 8 cores each was used, where each individual in the population was run on a separate core. Using this parallelised version of the EA, each individual started to take around 0.23 seconds to run one set of patterns in the passive models, which was 10 times faster than the sequential version of the same code. Additionally, a higher improvement was found when running the EA for active models, where the parallelised code speed up was 14 times compared to the previous version.

7.5.3 Results

7.5.3.1 Fixed Training Set

The initial experiment was designed to test the capability of the EA to evolve morphologies for a given set of patterns. The test was performed using a population of 100 individuals, where each individual was presented with the same set of 20 patterns in each generation. With this test it was

possible to check which kind of morphology was evolved as well as to measure the performance of the evolutionary algorithm in evolving the neuronal morphologies for the pattern recognition task.

Figure 7.4 shows that it was possible to evolve morphologies using only 150 generations, which improved the average pattern recognition performance by 15% when compared with the initial population performance. However, no correlation was found between fitness value and asymmetry index (see graphs in Figure 7.4). The reason for the increase in fitness (left graph) and the fluctuation in asymmetry index (right graph) could be that I had a fixed set of patterns and the EA was therefore trying to find morphologies that performed well for that specific set of 20 patterns.

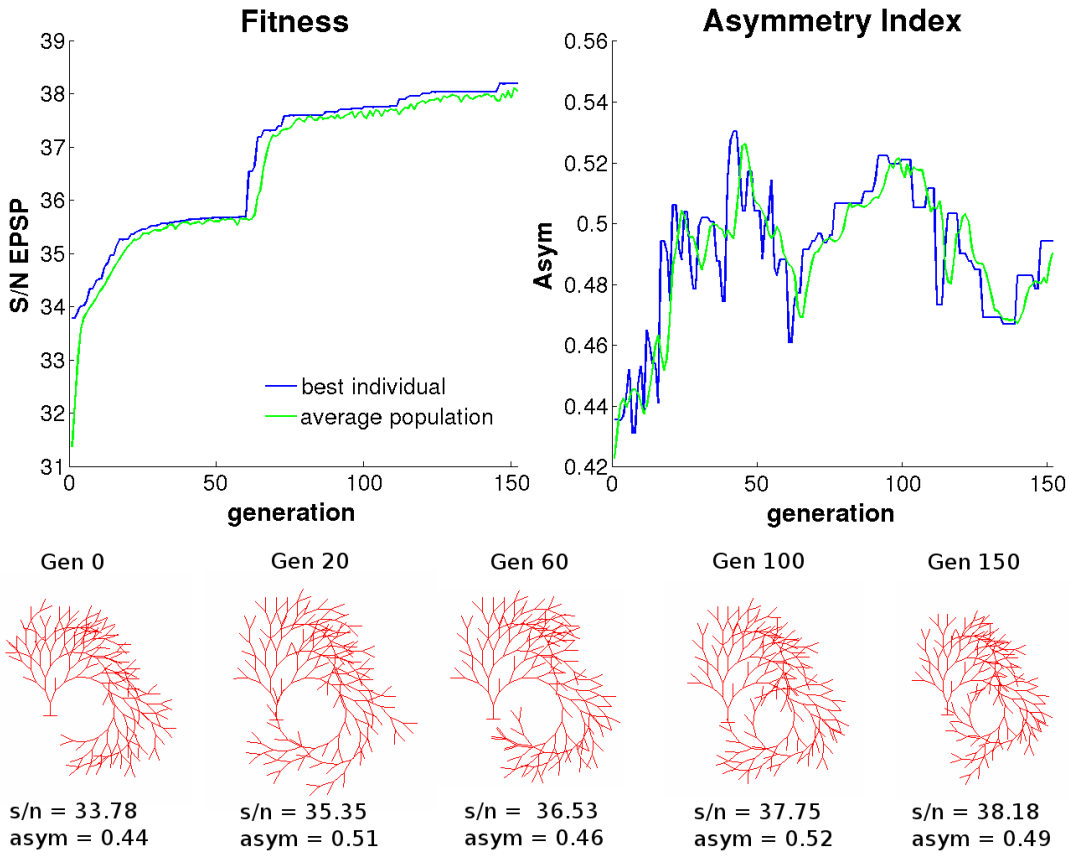


Figure 7.4 – Pattern recognition performance using a fixed set of patterns. The EA was run for 150 generations. The fitness of the best individual (blue line) and the average over the population (green line) are plotted for each generation. The top graphs show that the performance of the best individual and the overall population were slightly increasing over the generations (left graph), even though their asymmetry index did not correspond to this improvement (right graph). The morphologies shown on the bottom of the figure highlight the evolution of the best individual, where a different individual was obtained in each generation.

7.5.3.2 Varying Training Set

In order to force the EA to find morphologies that performed well regardless of the particular pattern set, I ran a second experiment in which a different training set was presented for each individual every time the fitness was assessed. For this task, I used the same population size from previous experiments (100 individuals), where for each individual I presented ten different sets of patterns (10 trials). As the asymmetry index did not seem to represent a good metric to predict the pattern recognition performance, a second metric, mean depth, was tested. This metric, explained in Section 5.2.2.4, was chosen for the simplicity of its calculation and because it includes the location of each synapse in its computation. The results are presented in Figure 7.5.

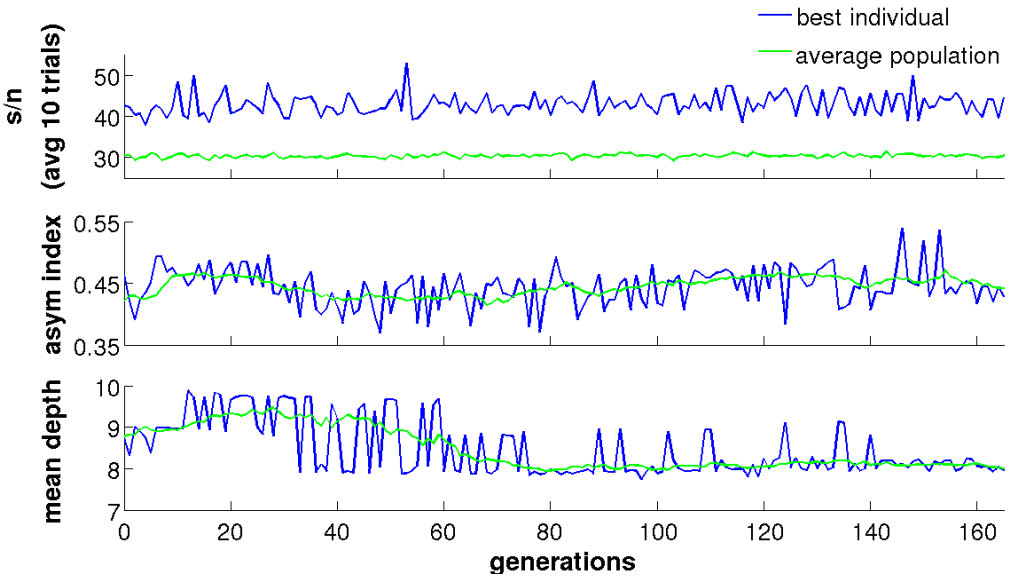


Figure 7.5 – EA results from passive models when presenting different sets of patterns. Each individual in the population was presented with 10 different pattern sets and the individual fitness was calculated using the averaged s/n ratio over the patterns sets presented (top graph). After 160 generations, no improvement was found in the neuronal performance compared to the initial population.

The results plotted in Figure 7.5 showed that the average s/n (top graph) did not improve over time and the morphometrics show no clear directional change (middle and bottom graphs). Even though the average population shows a decrease in mean depth (green line in the bottom graph), this does not correlate with the population performance (green line in the top graph). The possible reason for this is discussed later in Section 7.8.

7.5.3.3 Control Simulation

As both results from the EA that were shown in the previous sections did not correspond to the results obtained from the distinct morphologies (Section 7.4), based on which the most symmetric

morphologies were expected to be evolved as the best pattern recognizer, some experimental controls were performed to try to identify the model flaws. One task which was designed was to try to identify if the EA results could be related to the asymmetry index of the initial population. For this control experiment, the initial population was deliberately constructed to contain all morphologies with an asymmetric index equal to 0.99. This means that the EA population was initialised with the most asymmetric morphologies, which were supposed to have the lowest performances in the pattern recognition task, as shown in Section 7.4. The results from this control experiment are presented in the figure below (Figure 7.6).

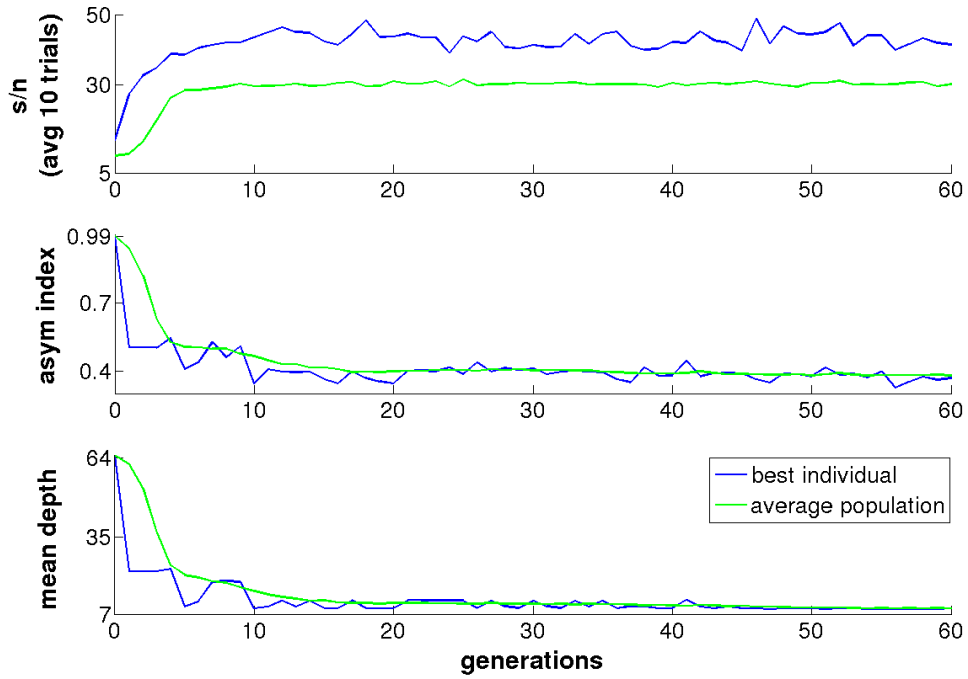


Figure 7.6 – Control simulation using an initial population with asymmetric individuals. The average asymmetry index of the initial population was equal to 0.99. The results show that in the first generation the best individual asymmetry was already 0.5 and after 9 generations the average asymmetry index has decreased to 0.47 (middle graph). In the same generation, the average s/n ratio (top graph) has reached a plateau ($\approx$ 30). The asymmetry index has reached its plateau ($\approx$ 0.4) in generation 17. However, the mean depth (bottom graph) did not reach a plateau as the average value kept decreasing until the last generations, which reached a mean depth around 9. These results indicate that individuals that are in the middle of the range of the asymmetry index and have low values of mean depth are already close to the best morphologies evolved for pattern recognition and can no longer be improved.

This control experiment showed that the morphologies with an asymmetric index value in the middle of the range (close to 0.5) performed as well as the most symmetric ones (asymmetry index equal to 0). This middle range asymmetry index value was already found as the best pattern recognizer in the initial population of the second EA experiment (generation 0 in Figure 7.5). This can be explained by the fact that the algorithm that generates random trees tends to

generate random partitions, which means partitions that are neither completely symmetric (such as the partition (64 64) for 128 terminal points) or asymmetric (like the partition (1 127)), as it uses a stochastic component to create the dendritic tree partitions (explained in Program Listing 6.2). Hence, the initial population did not really start with a very wide range of morphologies and the mutation operator was unlikely to generate new morphologies that fully sampled the morphology space. However, by forcing the EA to start with asymmetric trees I could see that it was able to evolve trees that were more symmetric. To test the prediction that the most symmetric morphologies did not perform better than the morphologies with an asymmetry index in the middle of the range, I conducted the experiments described in the next two sections.

7.6 Comparing Exhaustively Generated Morphologies

In order to determine what is happening when the trees are evolved it is necessary to evaluate a larger sample of the tree morphology space, or even to evaluate the whole tree space. In this section we consider the whole tree space. As mentioned in Section 6.3.2, the morphology degree chosen for this study with 128 terminal points was too computationally expensive to evaluate the pattern recognition performance when covering all possible trees. So, the solution was to choose the largest tree size for which it was feasible to compute the performance of all generated trees. This tree degree was 22 terminal points which generated a total number of trees equal to $1,514,661$ using the Lisp code given in Program Listing 6.1 (the trees with 24 terminal points, the next possible number of terminal points for binary trees, generated a total of $8,197,377$ trees, which was too expensive to compute in terms of computational time). Setting the terminal points to 22 made it possible to computed the pattern recognition performance of all trees within three weeks using five different Apple Macintosh servers with 8 cores each. To present this large amount of data, I opted to plot each morphometric studied here, asymmetry index, mean depth and variance of depth (previously described in Section 5.2.2), using three different bin widths. For example, for the asymmetry index, for which the bin width chosen was equal to 0.01, the first bin contained all the trees from asymmetry index 0 to less than 0.01, the second bin ranged from 0.01 to less than 0.02 and so on. The width of each morphometric bin was chosen to allow a comparison between the metrics since each metric varied in magnitude. These results are presented in Figure 7.7, where the graphs on the left (a,c,e) show error bars indicating the standard deviation calculated over all the trees within the respective bin, and the graphs on the right (b,d,f) plot only the mean values obtained for each metric studied, using the same average bins. Since the graphs on the right do not include error bars, the y-axis could be expanded which highlights the overall trends more clearly.

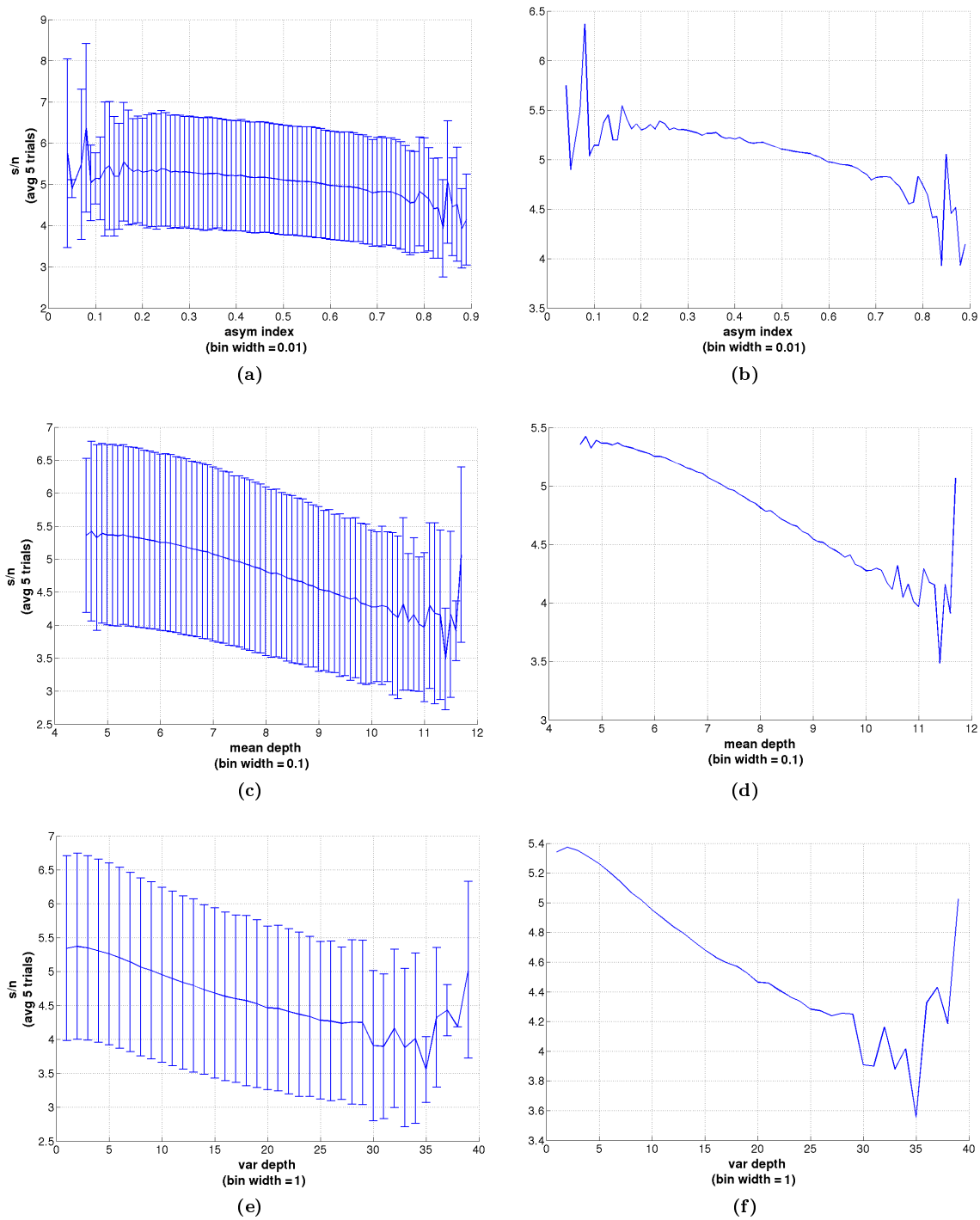


Figure 7.7 – Trees with 22 terminal points comparing three different morphometrics: asymmetry index (a,b), mean depth (c,d) and variance of depth (e,f). Each metric uses a different bin width (highlighted by the x-axis label) to result in a similar resolution of data. The pattern recognition performance was calculated by averaging over the s/n ratio in response to five different sets of patterns presented to each neuronal morphology. Error bars (graphs a,c,e) represent the standard deviation calculated over the average s/n for the trees within the bin. The plots on the right (b,d,f) highlight the trend found for each metric, showing that the three metrics proved to be good predictors of the pattern recognition performance.

The results demonstrate that for trees with 22 terminal points, the trees with lower values of the three metrics, which represent the most symmetric morphologies, are the ones with better a pattern recognition performance. The trends presented on the right side of Figure 7.7 show a decrease of performance when the morphologies become more asymmetric. The fluctuations found in the last bins of each metric as well as the initial bins for the asymmetry index metric result from the low number of trees that are contained in these bins. These results correspond to the ones found in Section 7.4, where the fully symmetric morphologies are better pattern recognizers when compared with the fully asymmetric ones.

7.7 Comparing Selectively Generated Morphologies

The results presented in the previous section raised the question if they could be generalised to larger trees. This section therefore tests the pattern recognition performance over the tree size (128 terminal points) selected for my research. For this size of trees we could not generate and evaluate all the trees so a sampling procedure was needed. To do this, a set of trees was generated using the algorithm described in Section 6.3.3. This sample of trees was designed to cover the morphological space more fully than the ones generated by the EA described earlier in Section 6.4. A total of 155,000 trees were generated and tested using the same set of parameters defined in Section 7.2. The top graph of Figure 7.8 shows the performance the whole set of trees generated (blue data points). The six distinct clusters of data that can be seen in the figure are due to an artefact from the algorithm used to produce the trees.

For testing the performance of this large set of neurons, different numbers of pattern sets (trials) were presented and the final results compared. In Figure 7.8 there are 155,000 blue points representing all of the selectively generated trees. As they were so many each tree was assessed using only 5 trials. In order to verify the overall trend visible in the data, 100 trees were selected across the full range and run with 100 trials each. These results were compared with the mean s/n ratio calculated over all trees in bins with a width of 0.01. The main difference between these two results is the variability of the data, where the large number of trials resulted in a higher standard deviation (see error bars in the bottom graph of Figure 7.8).

These results show that trees which are more symmetric tend to perform better when compared to the most asymmetric ones, confirming the previous results obtained with the trees exhaustively generated. However, the trees which are in the middle range of the asymmetry index (0.2-0.4) also performed as well as the most symmetric ones. This result is different from what was shown for trees with 22 terminal points (Figure 7.7), and it explains the performance of the EA. Once the

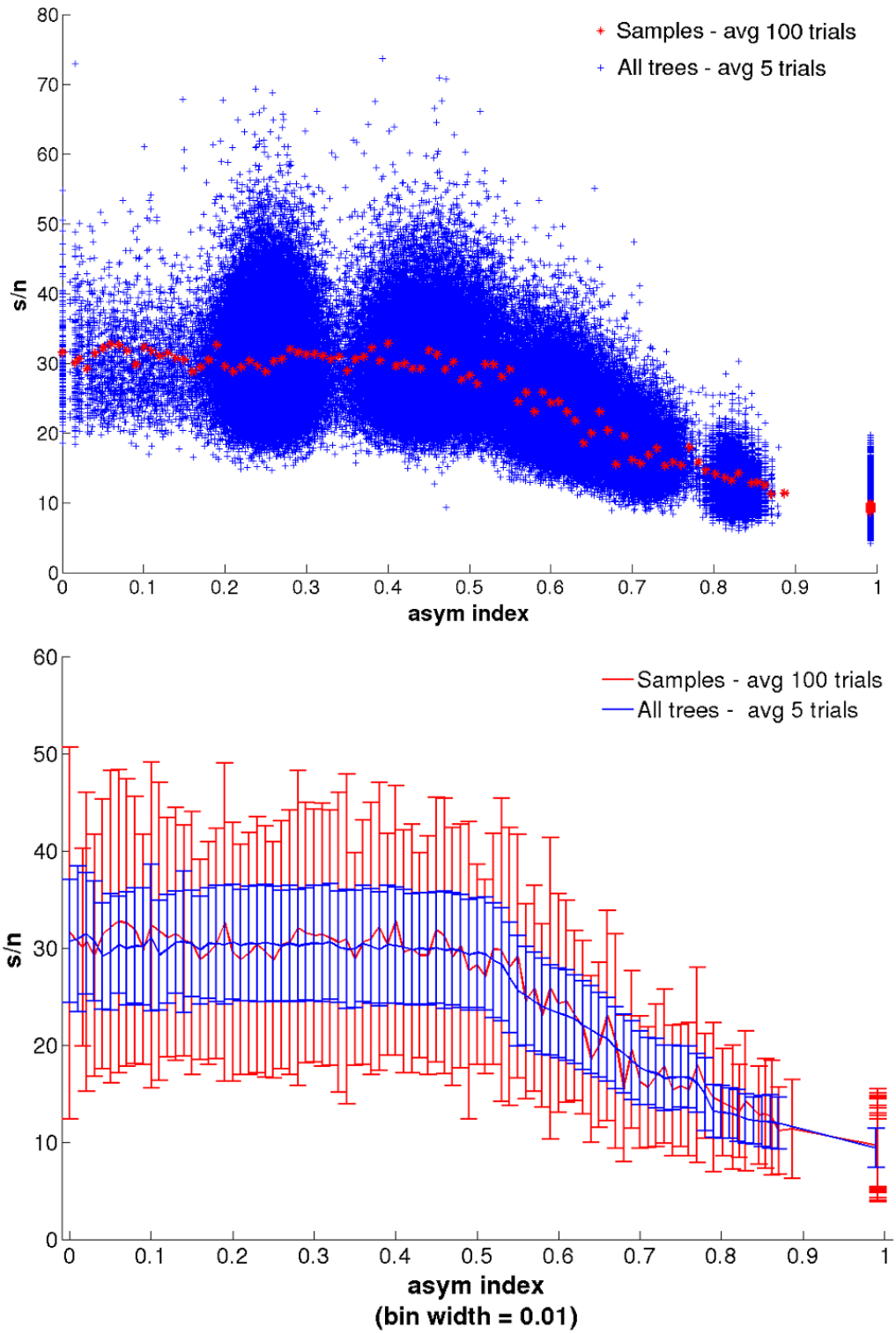


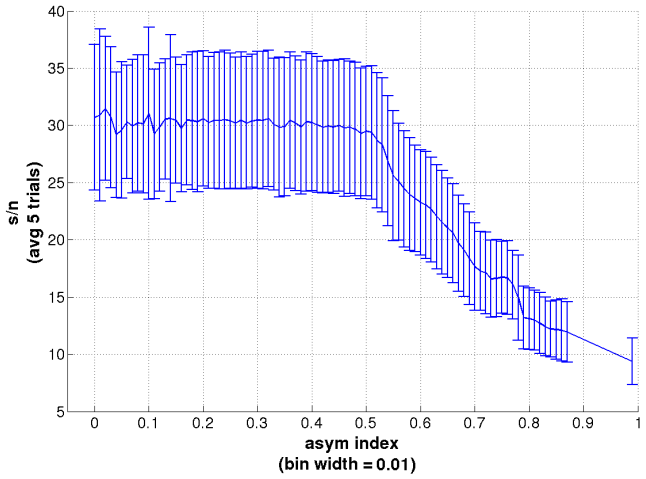
Figure 7.8 – Trees selectively generated with 128 terminal points. The scatter plot shown in blue is composed of approximately 155,000 trees (top graph); the red points correspond to 100 samples of morphologies, covering the whole range of trees with different asymmetry indexes (bin width = 0.01). Two methods were applied to measure the pattern recognition performance on these trees: 1. presenting, for each tree out of 155,000, five different sets of patterns and averaging the resultant s/n ratios (blue data points); 2. for each sample tree, from the 100 trees selected, presenting 100 sets of patterns (red data points). The error bars represent the standard deviation. The blue error bars were calculated for the mean of 5 trials over all the trees within the respective bin. The red error bars were calculated over 100 trials for each sample tree.

EA had produced trees (or had them in the initial generations) that had an asymmetry index of about 0.4-0.5 it did not evolve any further because these trees are as good at pattern recognition as the more symmetric ones. These graphs also show that the 100 samples could well represent the whole range of selectively generated morphologies, which was composed by 155,000 trees, showing that the main results could be obtained using a fairly small sample of morphologies only.

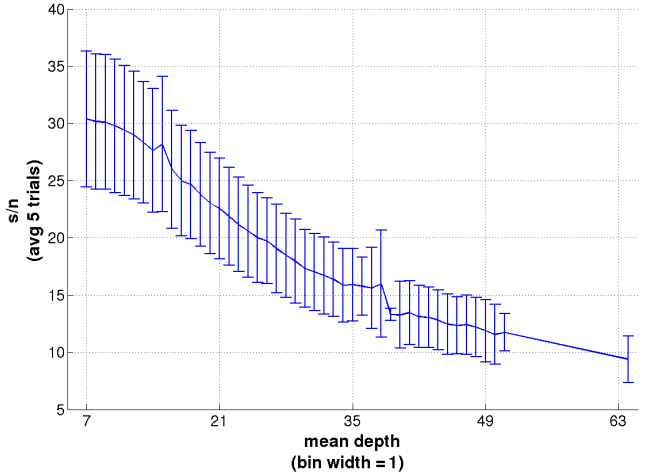
The next step was to compare the same three metrics used to evaluate the results from the exhaustively generated trees (Figure 7.9). The results obtained from the 155,000 samples of trees show that the mean depth (b) and variance of depth (c) are good predictors for the pattern recognition performance for the large dendritic trees (128 terminal points); these results are similar to the ones obtained with smaller dendritic trees. However, the asymmetry index could not correlate with pattern recognition performance in the large trees, since morphologies with a different asymmetry index, varying from zero to about 0.5, performed equally well (see graph c in Figure 7.9). This results from the fact that all of the trees with asymmetry index up to 0.4 correspond to a range of trees with the same low mean depth (close to 7), as shown in Figure 7.10.

7.8 Conclusion

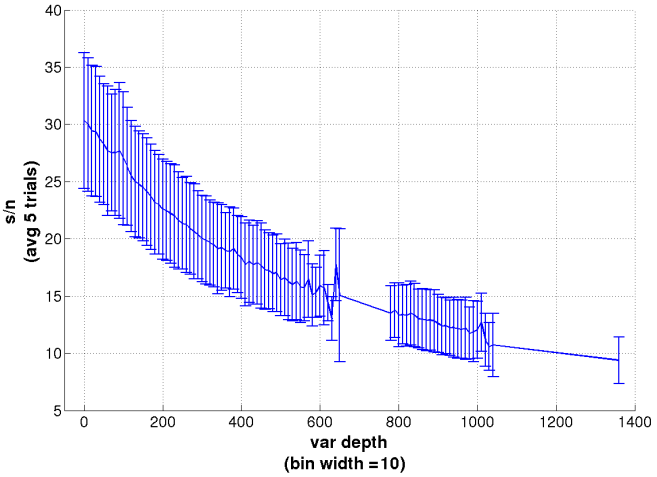
Different techniques were used to compare the performance of the generated morphologies for pattern recognition and their results were presented in this chapter. The initial results, when comparing two distinct morphologies (Section 7.4), showed that the most symmetric morphology proved to be the best pattern recognizer. The next experiment used an EA to evolve morphologies for the pattern recognition task. The EA was able to evolve morphologies that performed well on a particular pattern set. However, when the pattern set was varied for each individual, the mean fitness of the population did not improve over time (Figure 7.5). The reason for this is that the initial population in fact contained morphologies that already produced almost optimal performance. This is evident from the average of the initial population, where the asymmetry index was about 0.45 and the mean depth was about 9, and from Figure 7.9 it can be seen that this puts these trees into an area of high performance. Another relevant point about the EA results was that the performance measure of the best individuals evolved ($s/n \approx 50$) was higher than the best value found for the systematically generated morphologies ($s/n \approx 30$). This can be explained by the fact that the number of trials used to determine the fitness in the EA experiments was 10, which is 10 times lower than the number used in the experiments with selectively generated morphologies. In fact, the fitness function of the EA is stochastic rather than deterministic, and the individuals with the highest fitness values will always have been presented with pattern sets



(a)



(b)



(c)

Figure 7.9 – Comparing selectively generated trees using three morphometrics: asymmetry index (a), mean depth (b) and variance of depth (c). A set of 155,000 trees with 128 terminal points each was tested, and three different metrics were plotted against the pattern recognition performance. From these results, it is possible to see that the best metrics to predict the pattern recognition performance are the mean depth (shown in graph b) and the variance of depth (graph c).

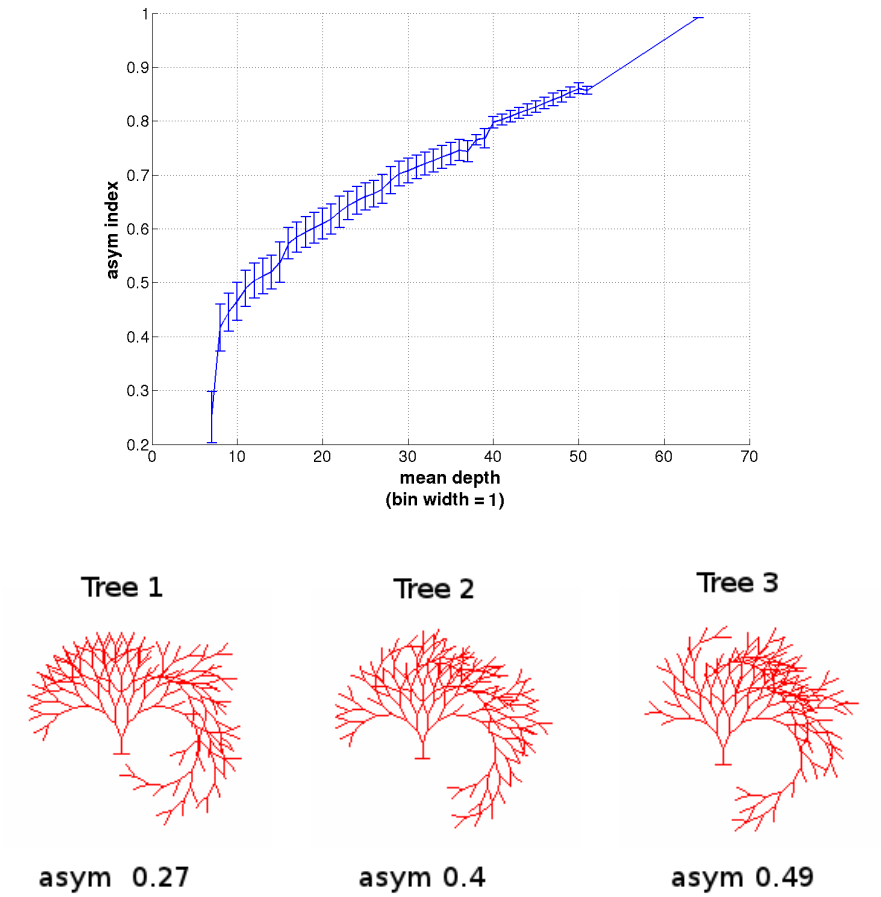


Figure 7.10 – Mean depth against asymmetry index for selectively generated trees with 128 terminal points. The plot on the top of the figure shows that all trees with asymmetry index up to 0.4 correspond to the same set of trees with the lowest mean depth, which explains the poor correlation between asymmetry index and neuronal performance shown in Figure 7.9. The diagrams shown on the bottom are examples of trees with the same mean depth (8.15) and different asymmetry indexes (presented below each tree), which correspond to the trees found in the beginning of the graph.

that are particularly easy to discriminate. Indeed, when the fitness of these best morphologies was re-evaluated by increasing the number of trials to 100, the average s/n decreased from 50 close to 30 (results not shown).

Similar results were found for exhaustively generated trees (Section 7.6) and selectively generated trees (Section 7.7) where the fully symmetric morphologies showed a better performance when compared to the fully asymmetric ones. However, the results from the selectively generated trees with 128 terminal points showed that the morphologies with an asymmetry index up to 0.5 performed as well as the most symmetric ones. These middle value range morphologies were also found to be the best pattern recognizers in the evolved morphologies produced by the EA (Section 7.5.3), as explained earlier. These results indicated that the asymmetry index could not correlate with the neuronal performance for the pattern recognition task across its whole range.

Moreover, the results showed that the best metric to predict neuronal performance was the mean depth (Figures 7.7 and 7.8). This can be explained by the way this metric is calculated, which uses the distance of each synapse from the soma. As shown in Figure 7.3, the main difference in performance between the most distinct morphologies is highlighted by the synaptic distribution of the active synapses, where the best performing tree shows active synapses closer to soma. Then, minimising the variation in synaptic distances should also minimise variation in the EPSP amplitudes and so, improve pattern recognition performance. Thus, from this point on I decided to use the mean depth as the main metric for comparison with the neuronal performance, as the variance of depth showed similar results to the mean depth metric (showed in Figure 7.9). I also decided that for the next step of this research, which was the investigation of the neuronal morphologies in the presence of active conductances (results presented in the next chapter), I would use the other metrics described in Section 5.2.2, trying to obtain a better correlation between the metric and the pattern recognition performance of the neurons studied.

Chapter 8

Effect of Dendritic Morphology and Parameters in Active Neurons

8.1 Introduction

The results found in the previous chapter gave an understanding about how dendritic morphology can affect pattern recognition performance. Using the EA I firstly found that I could evolve morphologies for a specific set of patterns. However, when the pattern sets were randomly generated for each individual the EA did not show an improved performance over the initial population. Having systematically examined a large sample of tree morphologies I found that the reason for this was that the original population already contained the well performing individuals. From the results obtained for systematically generated morphologies (Section 7.4, Section 7.6 and Section 7.7), I found that more symmetric trees performed better than more asymmetric ones. I also determined that the best metric to predict neuronal performance was the mean depth.

According to Cook and Johnston, "realistic active dendrite models had improved recall performance as compared with the passive model" [10]. Moreover, real neurons contain active conductances, so the study of pattern recognition in the presence of active conductances is more biologically relevant. Based on these considerations and on the results obtained from other research that has studied the effects of neuronal morphologies in the presence of active conductances ([42, 76, 75]), I decided to extend my research to use morphologies with an active soma and dendrites. The active conductances used were based on the values from van Ooyen's model, as presented in Section 5.2.1.

In the passive model, some important features of the pattern recognition task were not explored such as the optimization of parameters related to the morphology and input patterns and the effect of background input during the pattern presentation, which all could affect the pattern recognition

performance. Furthermore, some morphological metrics should be tested due to the variation of parameters. Metrics such as mean path length and mean electrotonic path are now relevant to this work as they can distinguish morphologies not only by their topology, but also by using some morphological parameters such as dendritic compartmental length and tapering. Thus, all these features are also included in the model proposed here, so trying to optimize the performance of the resulting morphologies. Then, the next three logical steps to be studied were: 1. Test the other morphometrics described in Section 5.2.2 on the new active model; 2. Evaluate the morphologies generated systematically in the presence of active conductances; 3. Modify the EA to optimise morphological parameters as well as parameters related to the input patterns to evolve active morphologies. After describing the new model, the results of these experiments are presented. At the end I summarise the metrics and parameters found that characterise the best morphologies for pattern recognition in the presence of active conductances.

8.2 Model Neurons

A set of common parameters was defined to be used in the initial experiments with active models, including the comparison of the distinct morphologies (Section 8.4) and the experiments with selectively generated ones. These parameter values, presented in Tables 8.1 and 8.2, were are all based on reasonably realistic ranges of values which were initially defined for the passive models (presented in Section 7.2). From Table 8.1 it is possible to see that most of the parameters are identical to the ones presented in Table 7.1 for passive models. The main differences are found in the dendritic compartmental length which was reduced, and some of the pattern recognition parameters related to the input stimulus, such as synaptic strength and number of spikes, which were increased. The differences are highlighted in colour blue in Table 8.1 when compared to values from Table 7.1 found on page 73. These changes in parameter values from passive to active models were made to improve the distinction of the neuronal output when comparing stored and novel patterns. To obtain these values, I spent a considerable amount of time hand-crafting the parameters to give a good performance for the most distinct morphologies, which also worked well for the morphologies generated systematically as they are morphologies found between these extremes morphologies.

The synaptic properties are the same as those used in the passive model, presented in Table 7.2. However, differently from the previous models, active models receive background input. This background input was necessary to include some noise in the spike response in order to study the effect of noise on the distinction between stored and novel patterns. The full list of background

Table 8.1 – Morphological and pattern recognition parameters used to generate the morphologies in active models.

Morphological Parameters		
terminal points (m)		128
dendritic compartments ($2m-1$)		255
soma	length	20 μm
	diameter	20 μm
dendritic compartment	length	5 μm
	diameter	2.5 μm
	tapering	false

Pattern Recognition Parameters	
pattern size	255 (1 bit per synapse)
patterns presented	20 (10 stored + 10 novel)
active synapses	25 (10% of total synapses)
synaptic strength	1.5 nS
number of spikes	5 per stimulus
spike interval	3 ms
noise	true

input parameters is presented on the left side of Table 8.2. The right side of this table lists the parameters used to control the simulation, where the simulation time (t_{sim}) and the time when the stimulus was presented (t_{stim}) were adjusted to support the time window of the output detection used in these simulations (more details about the neuronal response evaluation are presented in the next section).

Table 8.2 – Background input and simulation parameters used in active models. The parameters related to the simulation are: simulation time (t_{sim}), integration time step (dt), time the stimulus is presented (t_{stim}) and initial membrane potential (V_{init}).

Background Input Parameters	
synaptic strength	0.5 nS
number of spikes	1 per stimulus
spike interval	1000 ms
noise	true

Simulation Parameters	
t_{sim}	250 ms
dt	0.025 ms
t_{stim}	100 ms
V_{init}	-70 mV

The last change in the model when compared to the passive models is related to the initial membrane potential (V_{init}), which had previously been defined as - 65 mV, based on a previous work with passive models [18]. For simulations using active models, the initial membrane potential was based on the value used in van Ooyen's model, presented in Section 5.2.1. The membrane properties and ion channel conductances also use the same values as defined in van Ooyen's model (see Tables 5.1 and 5.2 for more details).

8.3 Performance Evaluation

The spike response used as the criterion to discriminate between patterns was given by the number of spikes after pattern presentation. The number of spikes was counted from the time when the stimulus is presented (given in Table 8.2) and within a time window equal to 100 ms for both

stored and novel patterns. This size for the time window was defined to accommodate the large number of spikes generated when presenting stored and novel patterns, as shown in the traces from Figure 8.1.

The pattern recognition performance was measured using the signal-to-noise ratio (equation given in the top of Figure 8.1), which was calculated using the number of spikes within the time window described above.

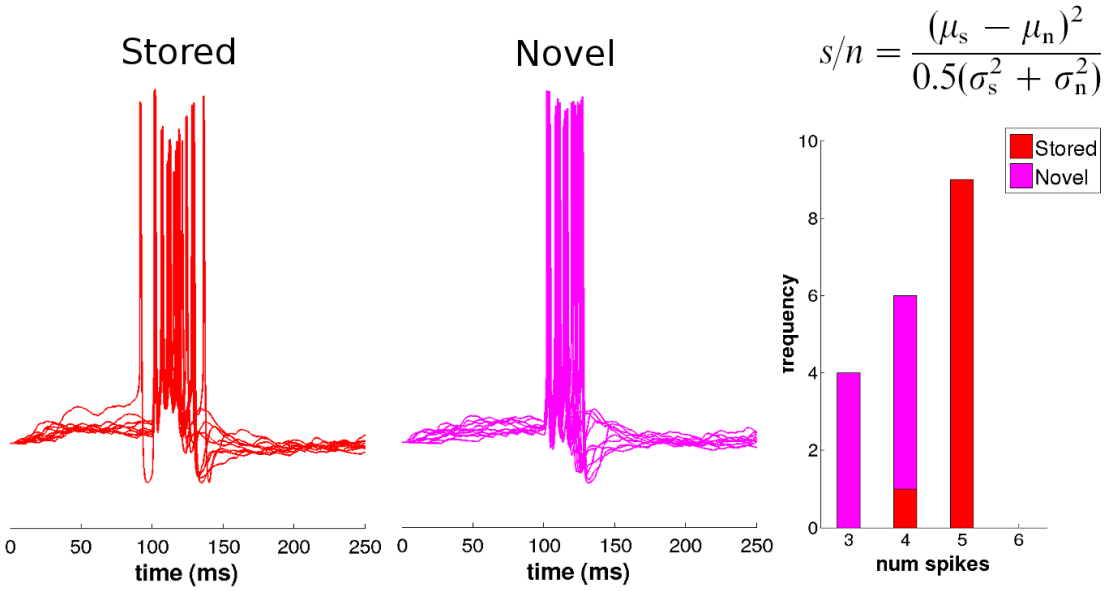


Figure 8.1 – Pattern recognition performance in active models was determined by counting the number of spikes. The spikes represent the responses to 10 stored patterns (left spikes) and 10 novel patterns (right spikes). The histogram shows the frequency of each number of spikes produced for stored (red bars) and novel (magenta bars) patterns, which was used for the s/n calculation (the actual s/n value is presented in Figure 8.2, bottom right graph).

8.4 Comparing Fully Symmetric and Asymmetric Morphologies

As in the passive models, the first experiment was to compare the fully symmetric and fully asymmetric morphologies to determine if any difference could be found in the neuronal response between these two distinct morphologies. This experiment used trees with 128 terminal points as previously described in Section 6.3.1. The results show that the distinction between the traces are not as clear as for the EPSP traces shown in the passive models (compare traces from Figure 7.3 for passive model with traces in Figure 8.2 for active model). However, from the histogram shown in Figure 8.2, it is possible to find a clear distinction between the number of spikes for stored and novel patterns, which is more pronounced in the symmetric trees. As a consequence, the symmetric morphology has a s/n ratio seven times larger than the asymmetric one, as shown on the top of each histogram in Figure 8.2.

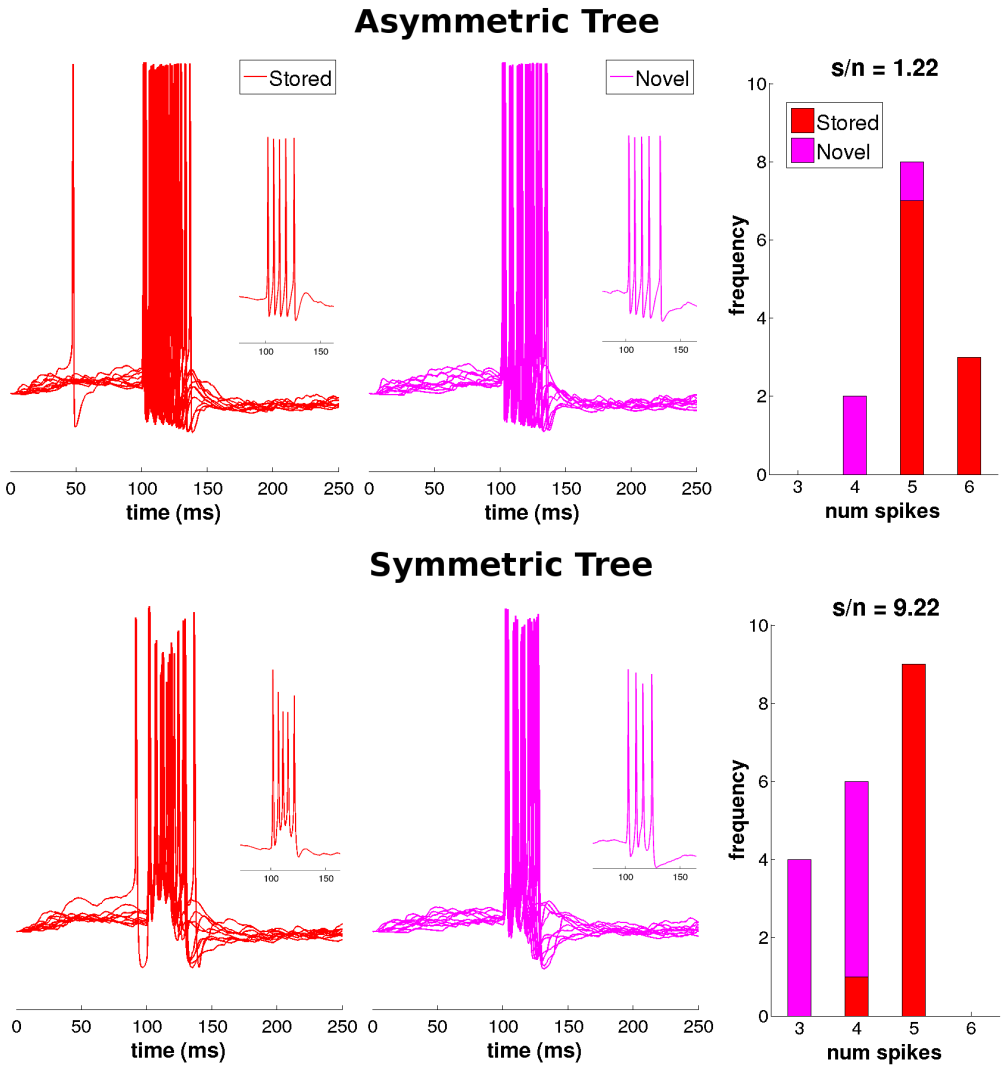


Figure 8.2 – Traces from fully symmetric and fully asymmetric trees in active models. The histograms on the right show the frequency of the number of spikes when presented with stored and novel patterns (red and magenta bars respectively). The performance of each tree is measured by the s/n ratio; the s/n values shown on top of each histogram confirm that the symmetric morphologies (bottom graph) perform seven times better than the asymmetric ones (top).

8.5 Comparing Selectively Generated Morphologies

The same set of 155,000 trees generated previously for passive models was used to test the active ones (see Section 6.3.3 for details about how the trees were generated). These trees with 128 terminal points were tested using the parameters given earlier in Section 8.2. The pattern recognition performance was calculated by averaging a large number of pattern sets (100 trials), with different sets being presented for each trial for each model. Five random trees were selected from each bin and their performance was averaged to calculate the final s/n ratio for that bin. The results were plotted against the same metrics used for passive models: asymmetry index, mean depth and variance of depth. From the graphs shown in Figure 8.3 it can be seen that the active morphologies

performed similarly to the passive ones. Unsurprisingly, the asymmetry index was again the worst metric for distinguishing between different well performing neuronal morphologies (graph (a) in Figure 8.3). The other two metrics, mean depth and variance of depth performed as well as in the passive models, showing a roughly linear anti-correlation with the neuronal performance across their whole range (graphs (b) and (c) in the same figure).

8.5.1 Comparing Active and Passive Models

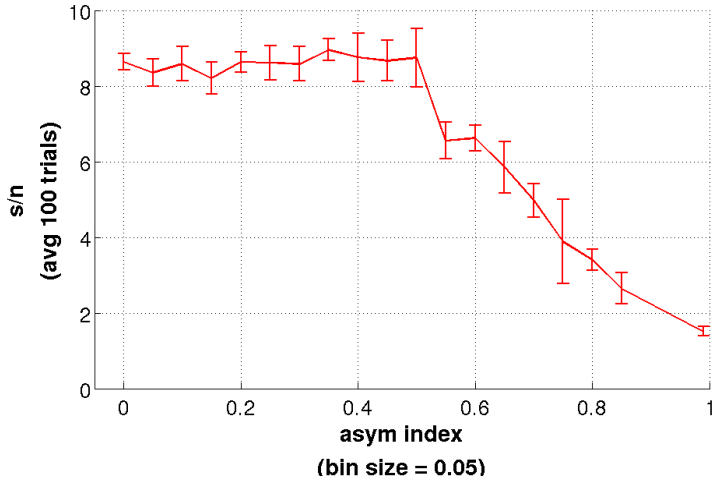
To compare the results from selectively generated trees between active and passive models, a new experiment was designed where both models used the same set of trees, as given in the previous section, and the same set of parameters, as presented in Section 8.2. The neuronal performance was calculated as presented previously in Section 8.5, where the s/n ratio of five randomly selected trees were averaged. The results were plotted against two metrics only, asymmetry index and mean depth, as the variance of depth has already shown similar results to the mean depth metric. The results shown in Figure 8.4 suggested a strong correlation between pattern recognition performance and mean depth across its whole range, where a less accentuated performance difference was found in passive models when compared with active ones (graph B). However, as shown in graph A, the asymmetry index did not correlate with the performance over its full range for either the active or the passive models.

8.6 Evolutionary Algorithm

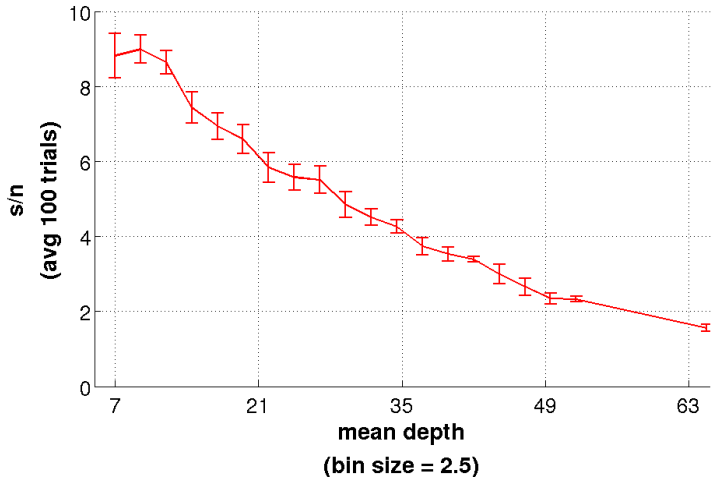
8.6.1 Chromosome Details

As mentioned in Section 6.4, a second chromosome was defined in the active models. This chromosome, named the parameter chromosome, contained a sequence of parameters related to the morphology and pattern input, as shown in Table 8.3. The initial values for each parameter were based on the values presented in Table 8.1. Then, for each parameter, a range of possible values was defined based on realistic values found in related research ([18, 76, 67, 75]), and the combination of them were analysed to ensure that the neuronal morphology and the neuronal output for the pattern recognition task were biological realistic (discussed in Section 8.6.2).

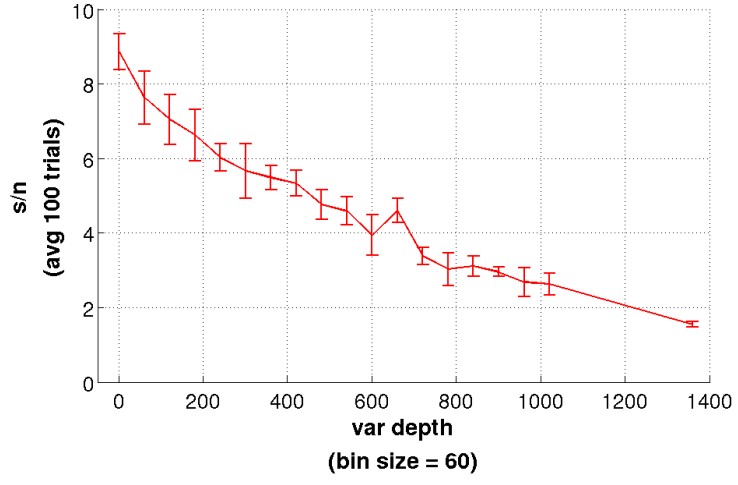
The fixed features of the model such as number of terminal points, number of dendritic compartments, somatic size, and the neuronal training set for the EA experiments were the same as for the systematically generated morphologies (presented in Table 8.1 in Section 8.2). The simulation parameters are also the values given in Section 8.2 (Table 8.2).



(a)



(b)



(c)

Figure 8.3 – Pattern recognition performance in active models for selectively generated morphologies. The results were obtained by generating a population of 155,000 random trees, and then averaging over 5 randomly selected ones in each bin, using three metrics: asymmetry index (a), mean depth (b) and variance of depth (c). The simulations use the set of parameters given in Tables 8.1 and 8.2. The error bars represent the standard deviation calculated over the average neuronal performance of the five selected trees of each bin.

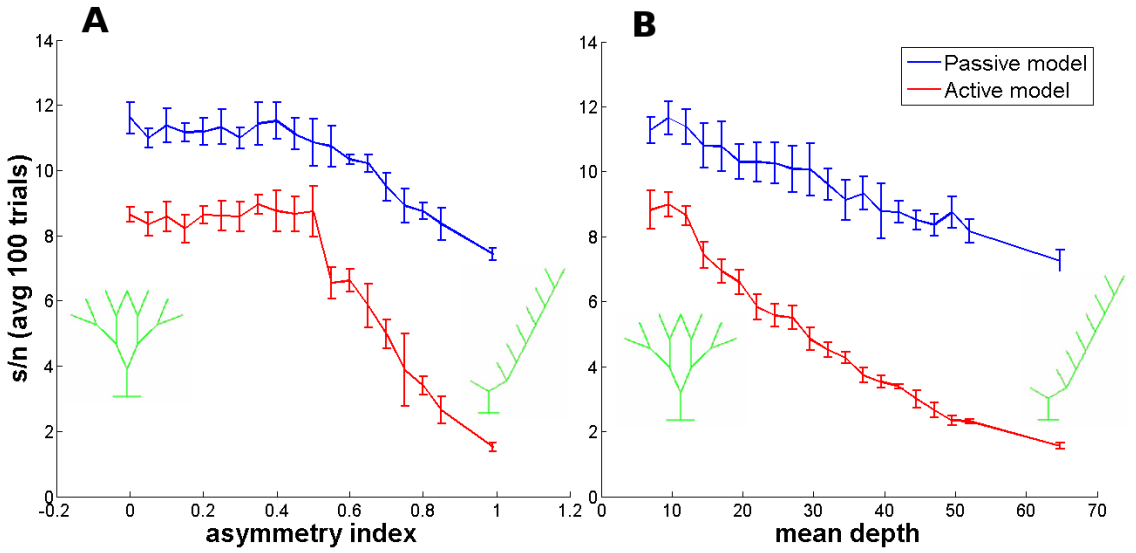


Figure 8.4 – Pattern recognition performance in active and passive models for selectively generated morphologies. Results were obtained by generating a population of 155,000 random trees, and then averaging over 5 randomly selected ones in each bin, using two metrics: asymmetry index (A) and mean depth (B). Error bars represent the standard deviation calculated over the five trees selected in each bin.

Table 8.3 – Morphological and pattern recognition parameters used by the EA to evolve active morphologies. The range of values were defined based on previous work (see text). The other fixed parameters, such as somatic length and diameter, use the values given in Table 8.1.

Morphological Parameters		
parameter	**range**	**type**
dendritic compartmental length	1-10 μm	continuous
tapering	1-0.8	continuous

Pattern Recognition Parameters		
parameter	**range**	**type**
synaptic strength	0.5-2 nS	continuous
number of spikes	1-10	discrete
spike interval	1-10 ms	continuous
noise	true, false	categorical
bg. synaptic strength	0-1 nS	continuous
bg. spike interval	500, 1000 ms	categorical
active synapses	5, 10, 15, 20%	categorical

One of the parameters given in Table 8.3 is tapering, which is given by the ratio between the diameter of the child branch and its parent ($tapering = \frac{diam_{child}}{diam_{parent}}$). From this equation, we can see that $tapering = 1$ means no tapering as the diameter of the child branch is equal to its parent's diameter, and $tapering = 0.8$ means that the diameter of the child branch will be 20% smaller than its parent branch. Tapering is applied to all dendritic branches, from the soma to the terminal points, until the minimum diameter is reached, defined as 0.1 μm.

As also shown in Table 8.3, each parameter had as a numerical type, which could be continuous, discrete or categorical. This type was determined based on three factors: 1. Some parameters,

such as the number of spikes, can only adopt discrete values; 2. To restrict the search space for some parameters, such as noise and active synapses, the parameter type was set to categorical; 3. To facilitate the mutation of parameters, as later described in Section 8.6.1.2.

8.6.1.1 Crossover

Whenever two individuals are selected as parents, crossover was always applied to their tree chromosomes. The method used for the tree chromosome was as described in Section 6.4.1.2, where two random branches were selected and swapped between parents. For the parameter chromosome described in Section 8.6.1, crossover was applied following a one-point crossover technique: a point was randomly selected in the parameter chromosome and all the parameters from that point on were swapped between the parent individuals (see Figure 8.5).

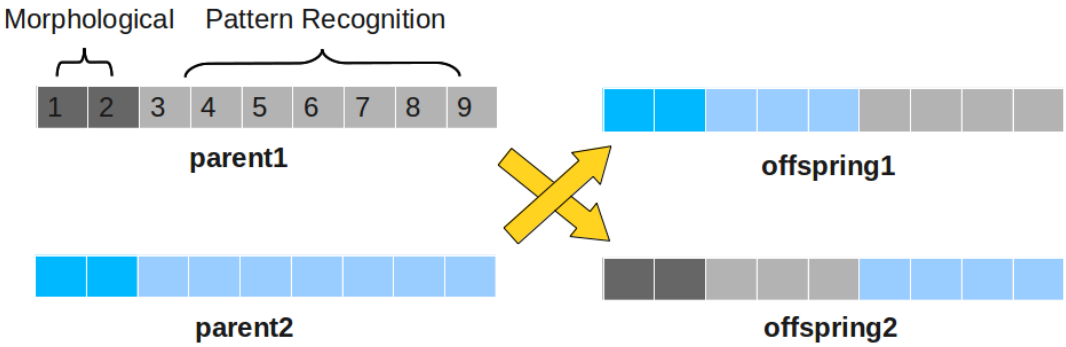


Figure 8.5 – Crossover of the parameter chromosome. The parameters are numbered following the list of parameters given in Table 8.3, where dendritic length is parameter 1 and number of active synapses parameter 9. In this example of a crossover operation, the parameter 6 (noise) was randomly selected to be the crossover point, where the resulting offspring received all the genes from parameter 1 to 5 from one parent, and from parameter 6 to 9 from the other parent (see colour scheme for parents and offspring).

As the parameter chromosome was divided into two genomic blocks, morphological and pattern recognition, the parameters of each parent could be selected during the crossover operation depending on the genomic block they belong to. To be able to investigate the evolution of morphological and pattern recognition parameters both together and separately, three different strategies were defined, depending on which genomic block the parameters could be selected from:

- Strategy 1. The whole genomic block: the crossover point of parent individuals could be selected from any genomic block, which means any morphological and pattern recognition parameter could be selected for the crossover operation.
- Strategy 2. Only morphological parameters: crossover was applied only over parameters within the morphological genomic block, which means dendritic length and tapering.

- Strategy 3. Only pattern recognition parameters: crossover was only applied over parameters within the pattern recognition genomic block, such as synapse strength and number of spikes.

8.6.1.2 Mutation

The mutation of the tree chromosome followed the rules given in Section 6.4.1.3, where a random branch was selected and replaced by a new branch with the same order. In active models, the binary trees were mutated with a probability of 20% (the same as in the passive models).

For the mutation of the parameter chromosome, the genes were selected based on the type of each parameter, as given in Section 8.6.1. For categorical parameters, the new parameter value was randomly selected from the number of possible categories for that parameter. For example, for number of active synapses, for which the parameter range has four categories (5, 10, 15, 20%), a random value x between 1 and 4 was selected and the mutation algorithm returned the correspondent value for the category selected (for example, $x = 1$ gives 5%). To initialise the continuous and discrete parameters, the parameter was selected following a uniform distribution over the range values. However, to mutate these parameters in the following generations, the algorithm below was used:

Listing 8.1 – Selection algorithm to mutate parameters.

1 Get a random value x from the standard normal distribution (mean=0; var=1)
2 Use a default standard deviation sd for all given ranges where $sd = max/4$ and max is the highest value within the range
3 Calculate a new parameter value based on the current one, such that: $new = current + (x * sd)$
4 If new value is out of range, change to min–max values: if $new < min : new = min$ if $new > max : new = max$
5 If range is discrete, round the value: $new = round(new)$

The probability of mutating the individual parameters depended on the strategy being used for the crossover operation as described earlier. So, the corresponding mutation strategies and their respective probabilities were given as:

- Strategy 1. The whole genomic block: each individual had a 2% chance of a mutation taking place.
- Strategy 2. Only morphological parameters: each individual had a 3% chance.
- Strategy 3. Only pattern recognition parameters: each individual had a 10% chance.

In fact, my initial experiments showed that Strategy 1 was more effective than Strategies 2 and 3, so the results presented from here on relate to Strategy 1 only.

8.6.2 Preliminary Investigation

An initial run was made using the EA and an investigation of the results was undertaken that consisted of analysing the firing patterns produced by the morphologies evolved by the EA. This was necessary as a large combination of parameters could be used by the EA, depending on the strategy the EA was running (as described in the parameter chromosome variations - Sections 8.6.1 and 8.6.1.2). From the initial tests, it was found that some parameter combinations resulted in irregular firing patterns which could not be used in the pattern recognition task as the EA uses the number of spikes after the pattern presentation as the criteria to discriminate between patterns. Examples of these irregular patterns are shown in Figure 8.6 (middle and lower traces) in comparison with normal ones (top traces). Each of these anomalies is described in the next sections together with the procedures implemented to eliminate them.

Plateaus

The combination of some parameters such as a small dendritic length with a strong synaptic input or background synaptic input resulted in a kind of response pattern called a *plateau*. Plateau responses can be described as depolarized potentials that start with a transient burst of action potentials and then stay at an approximately constant level of depolarization for a period of time [35]. This kind of neuronal response can be found occasionally in CA3 hippocampal pyramidal cells and subthalamic neurons [36, 35].

In this work, plateaus were characterized by noisy depolarised periods between bursts or individual spikes (see middle traces in Figure 8.6), which usually measured around -20 mV and took up to 40 ms of duration. To detect plateaus, a "band-pass filter" was implemented with a passband between -30 to -10 mV, and the membrane voltage was tested between 110 to 150 ms after the start of the simulation (patterns were presented 100 ms after the simulation start). If at least a 5 ms consecutive part of the trace during this period was within the voltage range of the pass band, the trace was classified as a plateau.

Plateaus were considered irregular responses, and they could not be used to compute the pattern recognition performance (which counts the number of spikes). So, to deal with this problem, a new rule was implemented in the EA where the average s/n ratio was set to zero if a plateau was detected, regardless of the number of plateaus found within that trial. This means that the

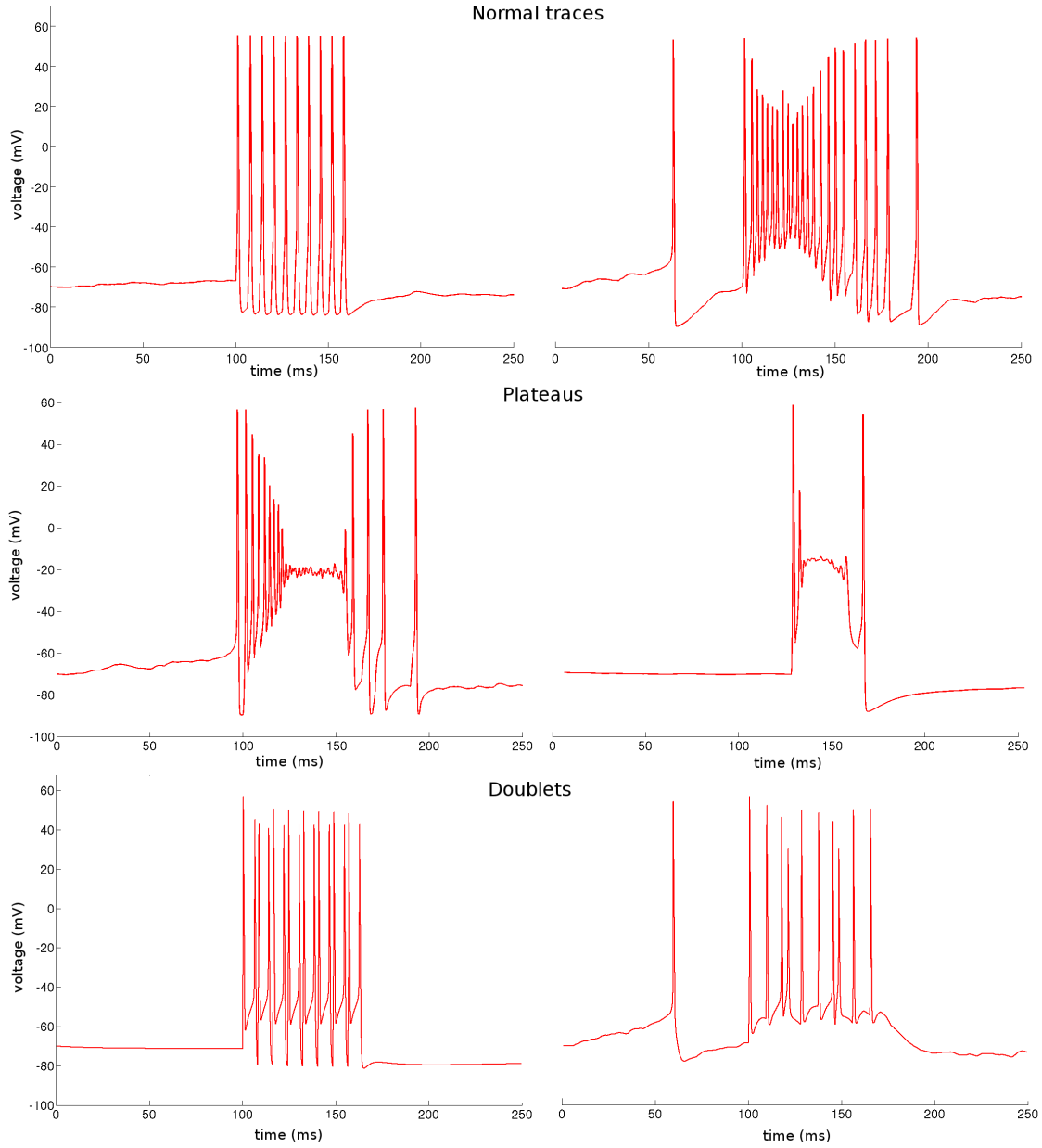


Figure 8.6 – A comparison of normal traces with irregular ones: plateaus and doublets. For each kind of trace, two samples of firing pattern are given.

individual was penalised with a low fitness score for each generation when the individual generated a plateau.

Lack of Variability

Another problem found during the evolution of active models were situations where the s/n value was undefined. This problem was always caused by the lack of variation in response to stored and novel patterns which could be a consequence of either a small number of active synapses or a weak background input. As I considered this lack of variability not biologically plausible, I decided to

eliminate it by setting the s/n value to zero for that trial, whenever this occurred. As a result, the fitness of the individual was affected by receiving a lower average s/n value that decreased its rank in the current population and reduced its chances of being selected in the next generation.

Doublets

The last problem found was related to a phenomenon which was caused by what I referred to as "super neurons". Super neurons were individuals with a much better performance than normal ones; their s/n value could go up to 100 times the expected value (something around 12 for the performance of the most symmetric morphology, as shown in Section 8.4). This huge difference in performance was a consequence of another irregularity found in the firing patterns, named doublets. According to Simpson (1969), doublets are "two discharges of a single (motor) neuron with uniquely short interspike intervals (ISIs) ranging between 2.5 and 20 ms" [qtd. in 37]. Doublets are a type of firing pattern also found in cerebellar Purkinje cells, which has been associated with dendritic Ca^{2+} spike firing [39, 40, 15], and more recently with Ca^{2+}-dependent K^{+} currents [44]. These two types of conductances were also present in the active models studied here, which could be the reason for the doublets being generated by these models. In this work, doublets were found as repetitive doublet patterns, where the spike train was mainly composed by a sequence of pairs of spikes (bottom left trace in Figure 8.6), or as single doublets within a normal firing pattern (bottom right trace).

The problem found here was that the EA evolved individuals which inconsistently generated doublets, sometimes in response to the presentation of stored patterns only and in other trials for both stored and novel patterns. This resulted in an extremely variable performance of these neurons, which could not be correlated with their morphology or the set of parameters used by them; therefore I wished to eliminate them from the population. To solve this problem, several modifications in the EA were tried, such as increasing the search spaces of some parameters and changing the fitness function used by the EA. This last adjustment seemed to be effective initially as the original fitness function was calculated by only averaging the s/n ratio for a given number of pattern sets, which could result in a large variability of the actual fitness from the fittest individual. So to reduce this variability, seven different fitness functions were tested, most of which included the standard deviation in the fitness calculation (such as fitness = mean s/n ratio divided by standard deviation of s/n ratio). However, although some improvements were noticed, that is the number of doublets found was reduced and the pattern recognition performance was increased in some cases, none of the fitness functions tested removed completely the doublets from the neuronal output. Thus, another attempt to eliminate the super neurons from the population was made, aiming to

detect any neuronal output with a doublet and penalising the individuals where this pattern was found. To do this, a method to detect doublets was implemented as described next.

As shown before, doublets are characterized by short ISIs that are followed by longer ISIs. So to detect them, a standard method to measure the variability of ISIs in spikes trains was used. This method, named coefficient of variation (CV2), was defined by Holt et al. in 1996 [28]. The CV2 method calculates the difference of two adjacent ISIs divided by their mean, as given in the following equation:

$$CV_2 = \frac{2\,|\triangle t_{i+1} - \triangle t_i|}{\triangle t_{i+1} + \triangle t_i} \tag{8.1}$$

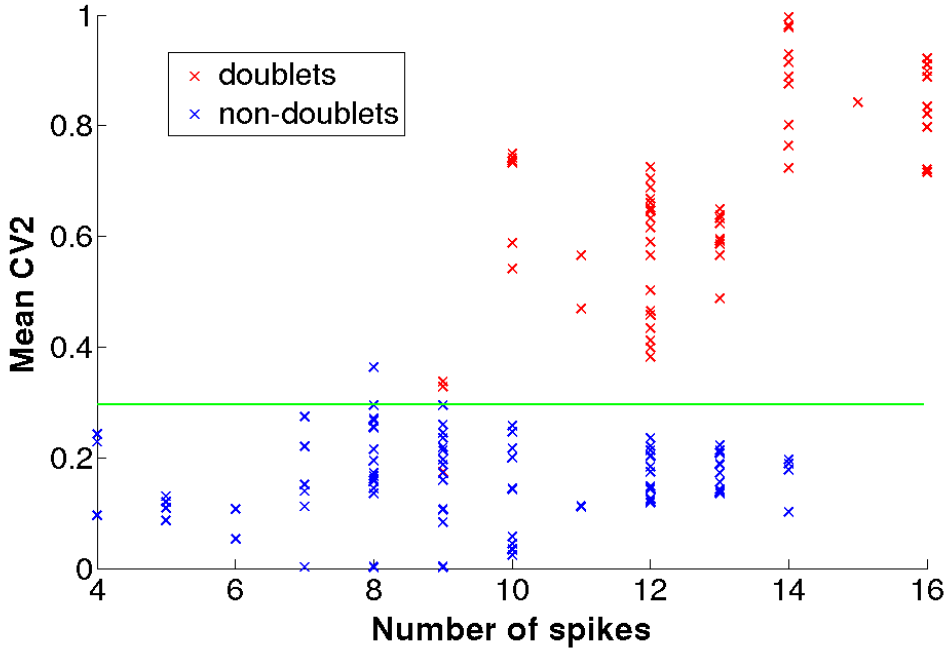
where $\triangle t_i$ gives the ISI for $1 \leq i \leq N$ and it is calculated as

$$\triangle t_i = t_i - t_{i-1}$$

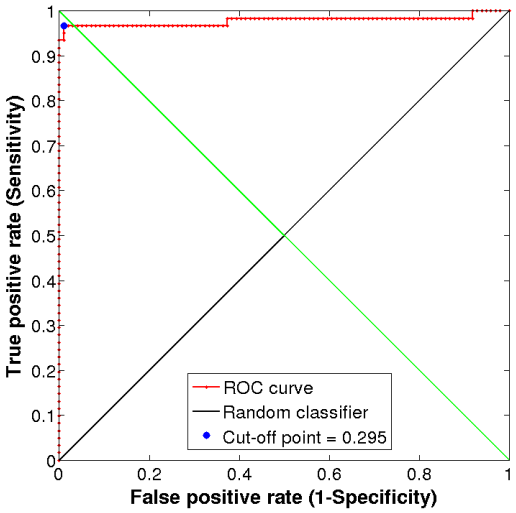
and t_i is the time at which the spike i occurs in the spike train, with $0 \leq i \leq N$. The CV2 is can adopt values between 0 and 2, with a mean of 1 for Poisson spike input.

From Equation 8.1, it is clear that the CV2 is low for a regular spike train and high for a large variability of ISIs. Based on this, I decided to calculate the mean CV2 over a set of neuronal spike trains obtained when presenting different sets of stored and novel patterns. For this spike train set, I identified manually which of the traces were normal (no doublets found) and which ones had doublets. Using 160 traces from which 38% contained doublets, I plotted the number of spikes per trace against their mean CV2. Figure 8.7a shows a clear distinction between traces with doublets (red data points) and normal traces (blue data points), where a possible cut-off point around 0.3 was defined (green line). However, some data points presented an overlap between traces with and without doublets, which required a further study to define where exactly the cut-off point for the doublet detection should be defined.

To define this cut-off point, I used a method named receiver operating characteristic (ROC) analysis, which was developed in statistical decision theory and has largely been applied in medical diagnostics and classifier systems [47, 50, 46]. This method plots a curve of the sensitivity or true positive rate against the specificity or true negative rate. For the set of spike trains plotted in Figure 8.7a, the ROC curve suggested a cut-off point of 0.295, as shown in Figure 8.7b. For this set of spike trains, this cut-off point resulted in a high rate of true positives (96.7%) and a low rate of false negatives (3.3%), as shown in Table 8.4b. These rates indicated that the proportion of traces with doublets which could be misclassified as normal was low. As my major concern was excluding doublets, the CV2 value of 0.295 was chosen as my cut-off point to detect doublets.



(a) Mean CV2 against spike number, showing which spike trains contained doublets .



(b) ROC curve.

Figure 8.7 – ROC curve for the detection of doublets.

Table 8.4 – ROC curve analysis.

(a) Contingency table. The following fractions were calculated from the ROC curve: TP - true positive; FP - false positive; FN - false negative; TN - true negative.

		Actual Value		
		Positive	**Negative**	**Total**
Prediction	**Positive**	TP = 59	FP = 1	60
	Negative	FN = 2	TN = 98	100
	Total	61	99	

(b) Test performance. This table was derived from the contingency table (Table 8.4a).

Positive Rate		**Negative Rate**	
Sensitivity (True Positive Rate)	96.7%	Specificity (True Negative Rate)	99%
False Positive Rate	1%	False Negative Rate	3.3%
Positive Likelihood Ratio	95.8	Negative Likelihood Ratio	0.03
Positive Predictive Value	98.3%	Negative Predictive Value	98%

Using the CV2 value chosen, a rule to eliminate doublets - and consequently the super neurons - was implemented: every time a doublet was detected in the neuronal response, the average fitness of that individual was set to zero. As a consequence of this rule, all individuals which produced doublets as output were graded low in the next EA generation.

8.6.3 Results

After I had solved the problems found in the preliminary investigation (presented in the previous section), I then used the EA to evolve effective morphologies with active conductances. To do this, I ran three different EA simulations using the same set of parameters, as given in Table 8.3. The population of each EA simulation was composed of 50 individuals, with dendritic trees with 128 terminal points. Each individual was run for 50 trials, which means that the individual's fitness was calculated by averaging over 50 different sets of patterns, each set composed of 10 stored and 10 novel ones. For crossover and mutation operations, Strategy 1 was chosen, where these operations were applied over the whole genomic block (morphological + pattern recognition parameters). The simulation setup is described in more detail Sections 6.4.1.2 and 6.4.1.3.

The evolution of the pattern recognition performance in these three simulations is shown in Figure 8.8, where the best individual performance (blue line) and the average performance of the population (green line) are plotted. Each simulation shown in the figure was performed for a different number of generations (x-axis), which was large enough to evaluate the results obtained from each EA simulation.

In all three simulations, populations of neurons were found with an average s/n ratio of more

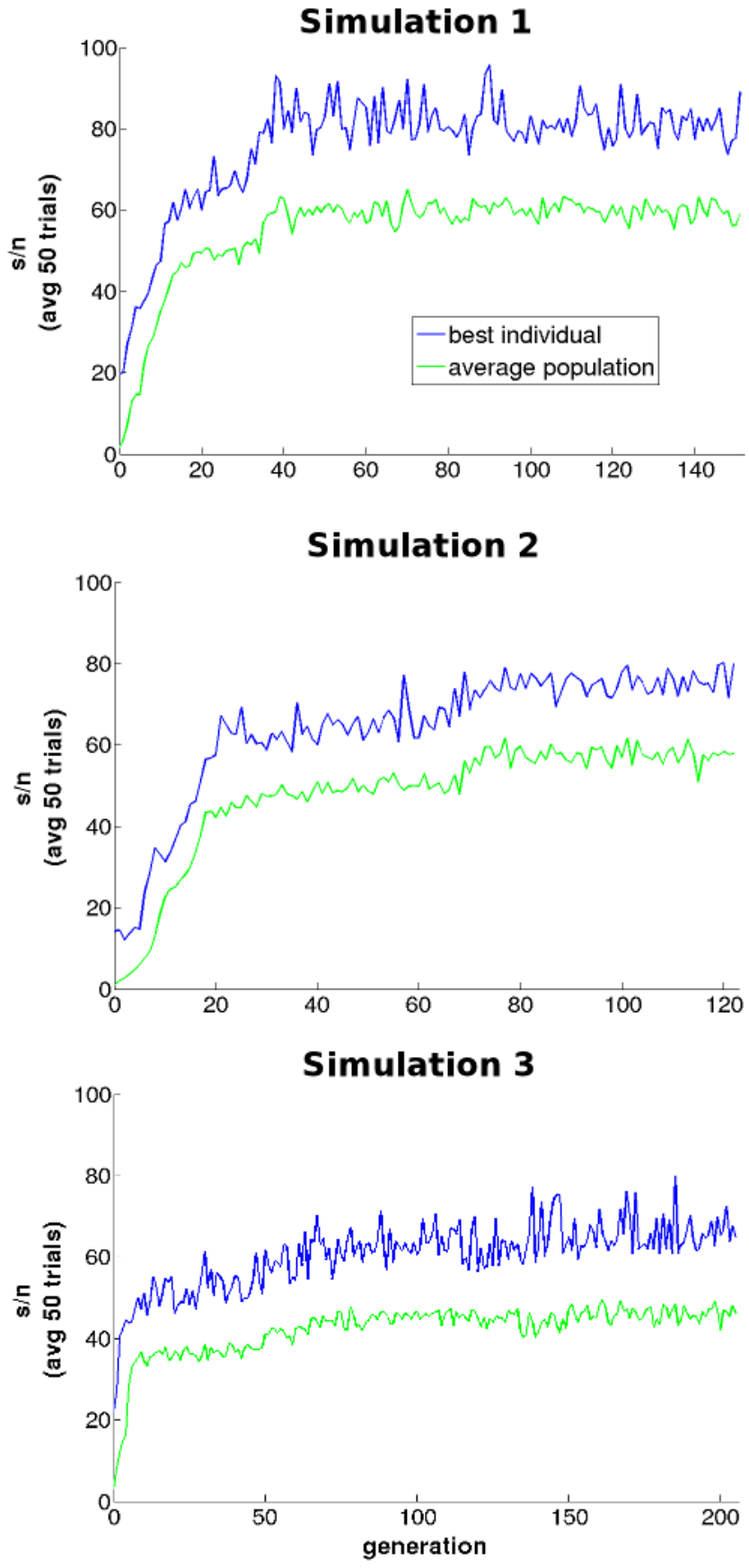


Figure 8.8 – Comparison of the performance of three different EA simulations.

than 40; this should be compared with the hand-crafted symmetric neuron shown in Figure 8.2, whose s/n ratio was about 9. Thus, by evolving neurons the performance increased by a factor of 5. In all three cases we have a rapid improvement in the mean fitness of the population between the first and twentieth generation. In the first two simulations the maximum fitness was reached approximately by generations 40 and 80, respectively, and in the last simulation the performance kept increasing gradually until the last generation.

In the next two sections I describe an investigation of how the neurons produced such a good performance.

8.6.3.1 Morphology and Performance

In this section I present the results of an investigation of how the morphologies of the individuals in the population changed over time. There are two parts of the genome that determine the morphology of the phenotype: firstly the tree chromosome and secondly the morphological part of the parameter chromosome. To analyse the results I used four morphometrics: the two metrics used previously for the passive models, mean depth and asymmetry index; and two new metrics which I did not test before, mean path length and mean electrotonic path length, both previously explained in Section 5.2.2. The decision to use the old metrics again was made to allow a comparison between these metrics, which were the best and worst metrics found for passive models respectively, and the new metrics chosen.

To present the results of each simulation, I opted to plot two versions of the data: the original data, which are presented on the left side of each of the following graphs (Figures 8.9 to 8.14); and a smoothed version, which in some cases allowed a better visualisation of the data trend. The smoothed data were generated by applying a simple moving average filter [1], with a lag of 5 data points.

The results from Simulation 1 exhibited no meaningful change in the tree topology, as shown by the absence of a consistent change in the mean depth after generation 25 in Figure 8.9. However, there was a step decrease in the mean electrotonic path length which corresponded to an increase in performance, as indicated by the red dashed line in Figure 8.10. In fact, as the mean path length did not change over time, whereas the mean electrotonic path length did, this suggested that this change was mainly due to a reduction in the dendritic diameter, as its length seemed to not be affected, which otherwise would have caused a decrease in both metrics. A further discussion of the parameters will be presented in Section 8.6.3.2.

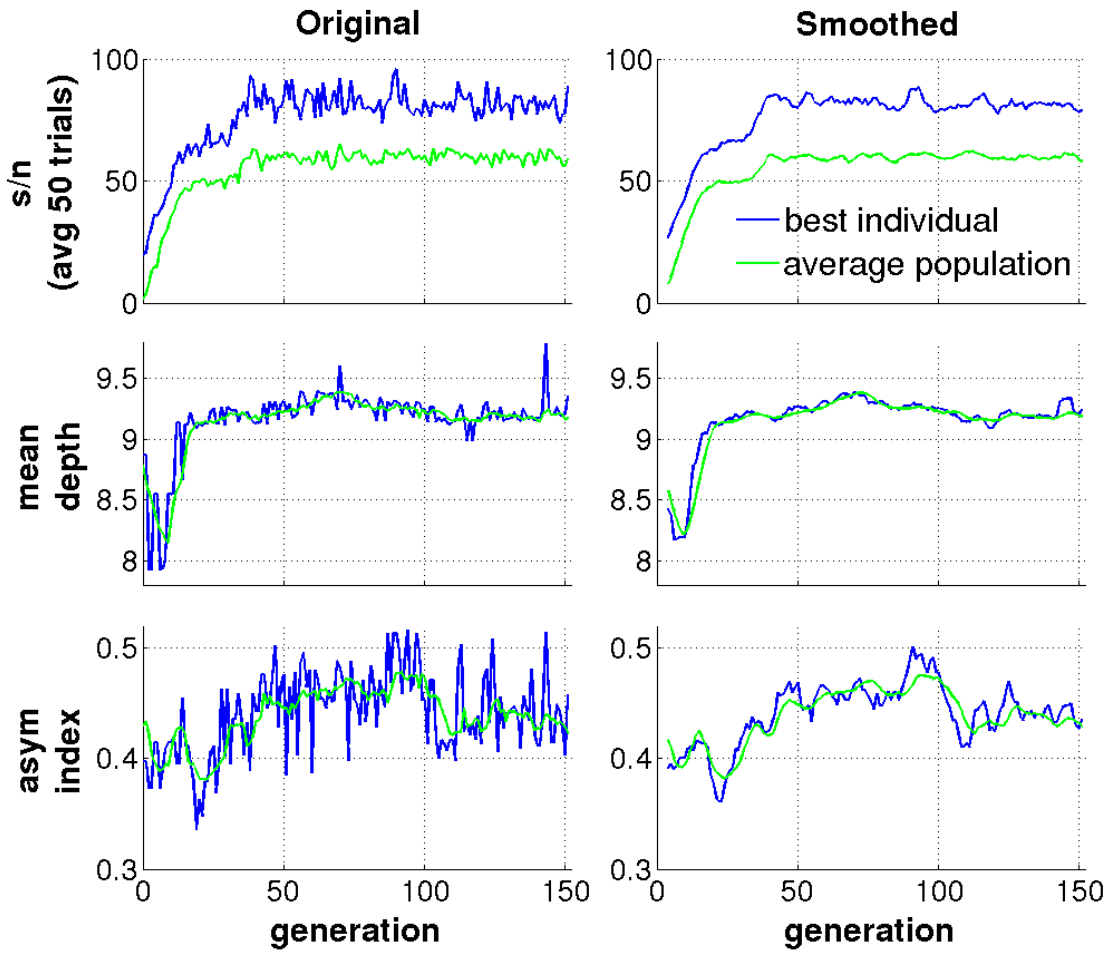


Figure 8.9 – Simulation 1 - Comparing mean depth and asymmetry index with neuronal performance.

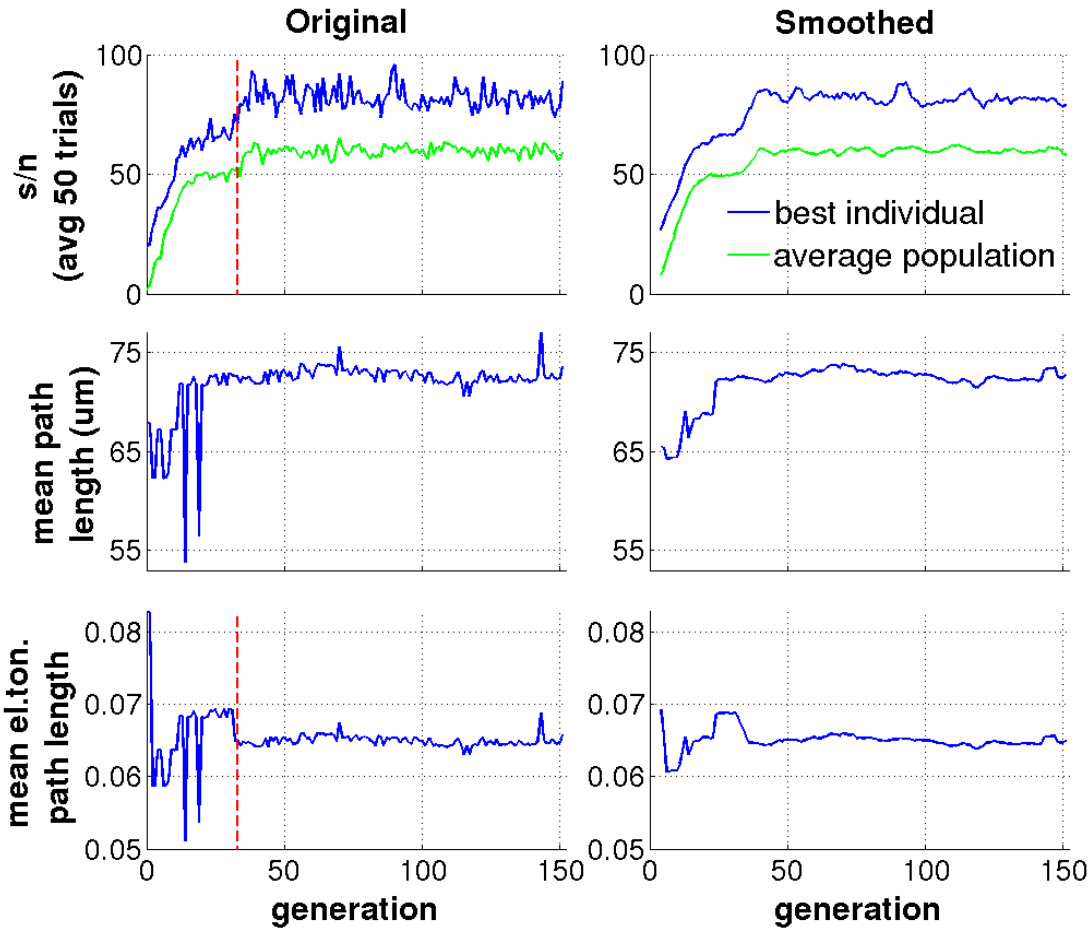


Figure 8.10 – Simulation 1 - Comparing mean path length and mean electrotonic path length with neuronal performance.

In Simulation 2, once again the tree topology was not responsible for the increase of the neuronal performance, as shown in Figure 8.11. On the other hand, the metrics which are dependent on the morphological parameters, mean path length and mean electrotonic path, showed a step decrease which corresponded to an increase in the neuronal performance (Figure 8.12). This significant change found in both metrics (generation 67 – as marked in Figure 8.12) suggested that it was a consequence of a decrease in the dendritic compartmental length (discussed in more detail in Section 8.6.3.2).

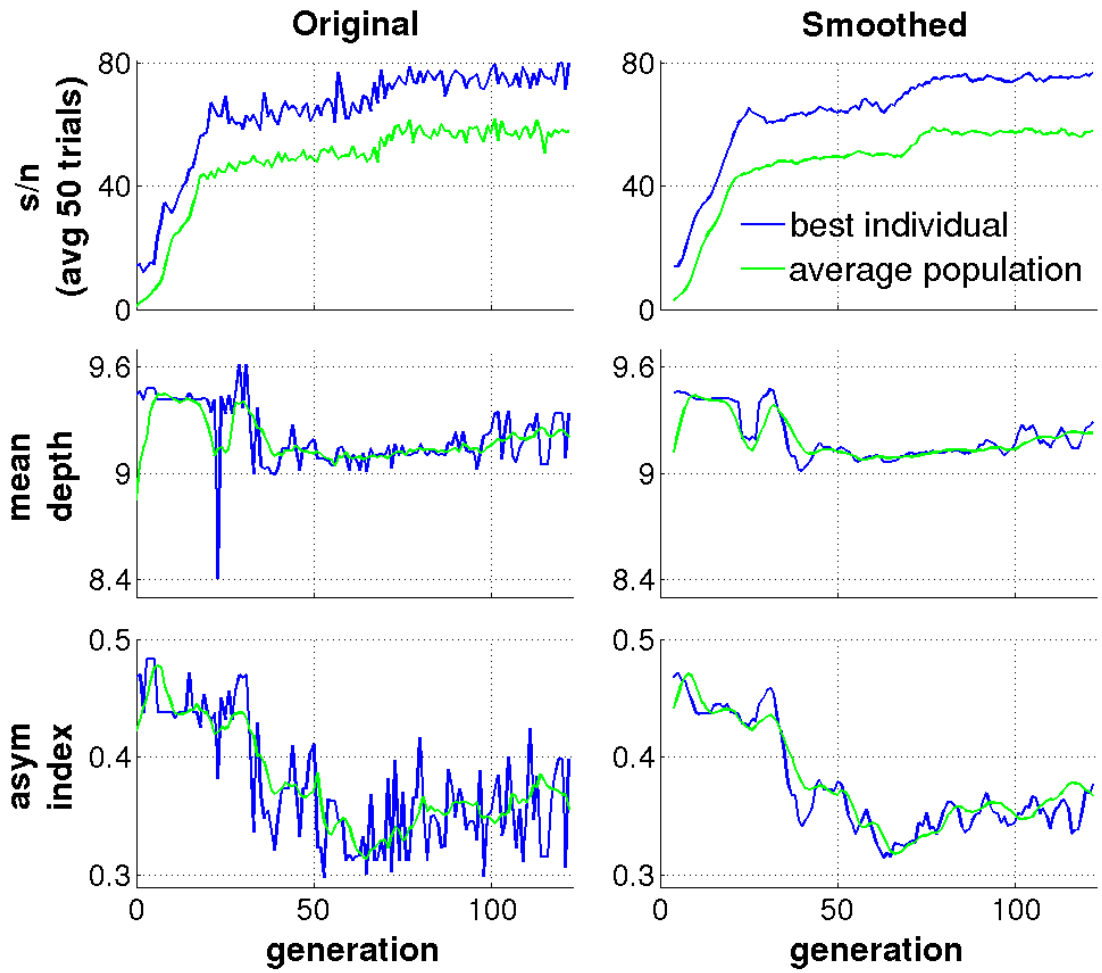


Figure 8.11 – Simulation 2 - Comparing mean depth and asymmetry index with neuronal performance.

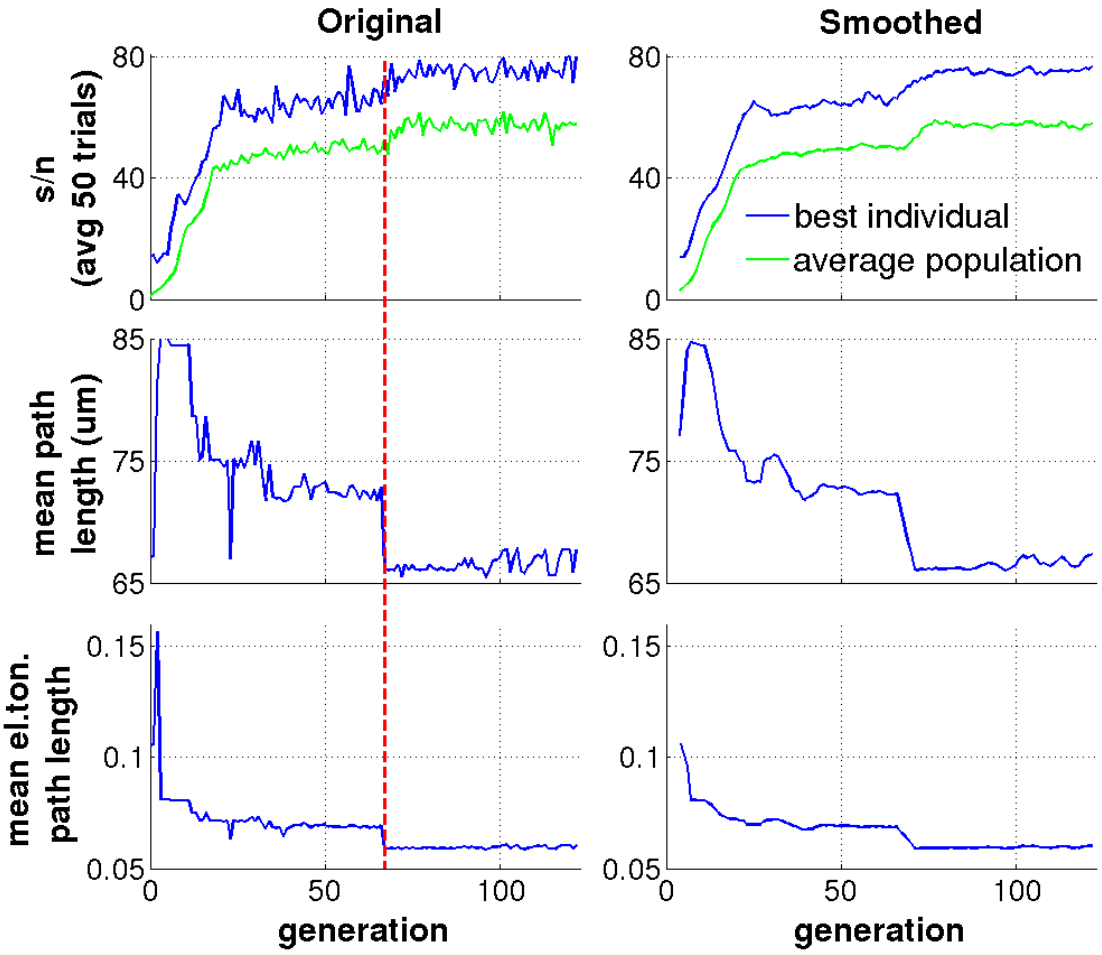


Figure 8.12 – Simulation 2 - Comparing mean path length and mean electrotonic path length with neuronal performance.

Simulation 3 was the only simulation which showed a persistent change in the dendritic tree topology as the mean depth decreased as the performance increased (Figure 8.13). As expected, a similar trend was found in both mean path length and mean electrotonic path length (Figure 8.14).

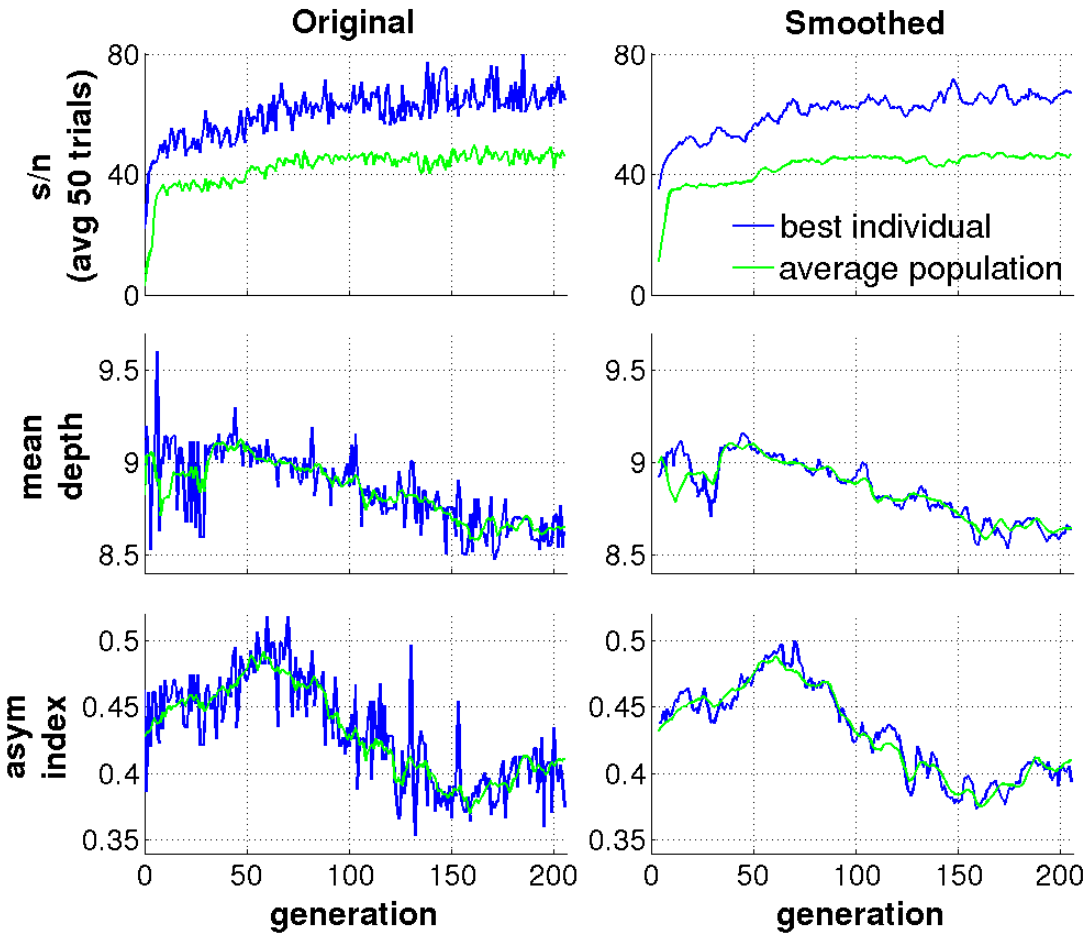


Figure 8.13 – Simulation 3 - Comparing mean depth and asymmetry index with neuronal performance.

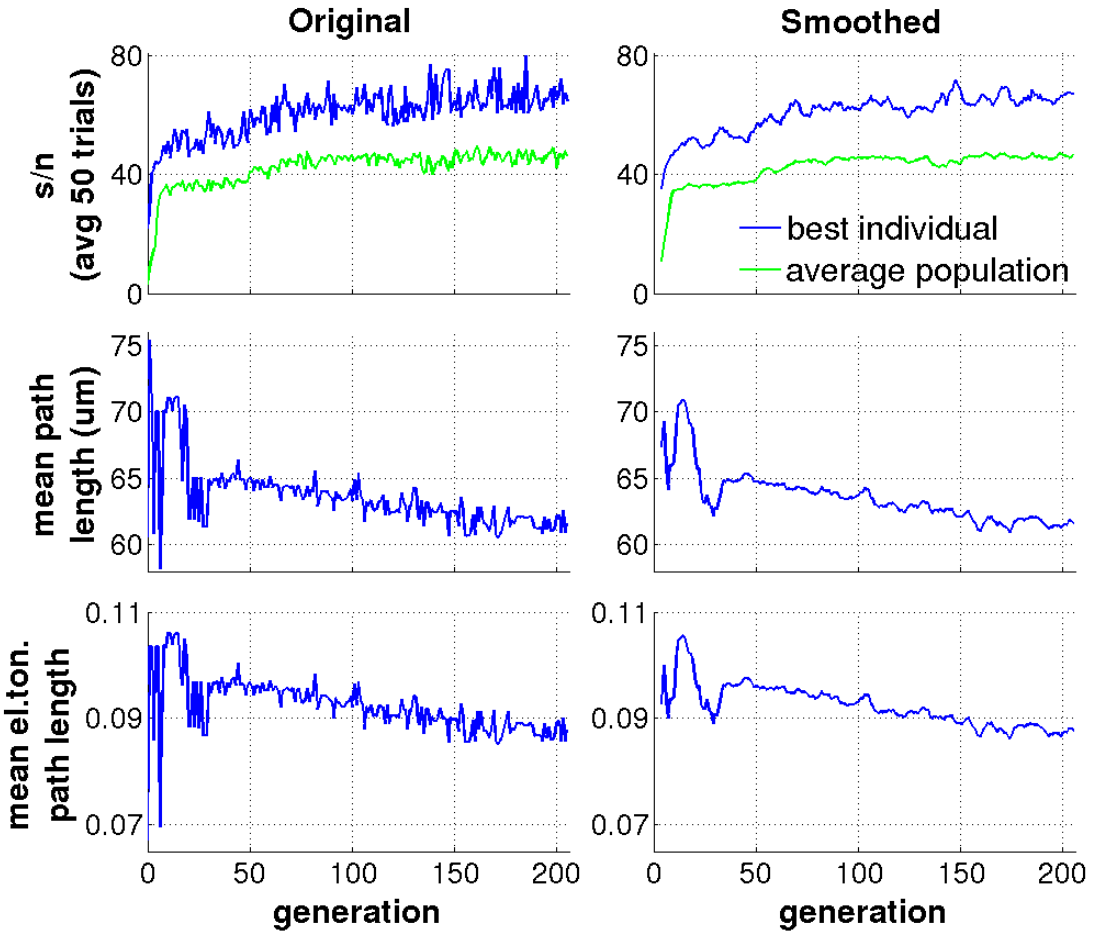


Figure 8.14 – Simulation 3 - Comparing mean path length and mean electrotonic path length with neuronal performance.

To summarise the results of these three simulations, we found that only one simulation showed an improvement in performance related to a change in the dendritic topology (Simulation 3). In the other two simulations, the only neuronal performance increase related to morphology was due to changes of the morphological parameters, such as dendritic compartmental length and diameter (tapering). These changes in the morphological parameters are explained in detail in the next section.

In summary, any change in the actual tree morphology played only a minor role in the evolution of successful neurons. In fact, as can be seem in Figure 8.8, very rapid evolution happens in all three simulations in the first 20 generations, which suggests significant changes in the parameter chromosome. This is discussed in the next section.

8.6.3.2 Neuronal Parameters and Performance

To examine how the parameter chromosome changed over the generations, I calculated the mean value of the parameters at the beginning of each simulation, at generation 20 and at the final generation. Table 8.5 shows the average s/n ratio for the population (column 3) and the mean values of the nine parameters which compose the parameter chromosome (columns 4-12), where the colour code represents each simulation. From this table, it is possible to see that the fitness increases very rapidly from the initial population to generation 20, which is a consequence of the parameter tuning that happens in the initial generations until the values obtained are close to optimal. From generation 20 until the final one, we find a much smaller improvement in the fitness, which results from a smaller variations in the parameters and small changes in the dendritic tree topology (as explained in the previous section).

From Table 8.5 we can also identify values of five of the parameters that are required to obtain the best performing neuron for the pattern recognition task: compartmental length should be close to 7 or 8 μm, no tapering (tapering = 1), a high value for the number of input spikes (close to 10), no noise (noise = 0), a low value for background synaptic strength (close to 0). The other four parameters, synaptic strength, interval, background interval and sparseness, do not seem to correlate with the average s/n value and so appear to be able to take on several different values while still maintaining a high performing neuron, although there appears to be an inverse correlation between sparseness (lower values indicate sparser input patterns) and synaptic strength.

Table 8.5 – A comparison of parameters for different generations in the three simulations. The colours indicate the three simulations previously described. The values given from the third column on represent the average s/n of the population and the nine parameters which compose the parameter chromosome as described in Table 8.3. The parameters presented are (units): dendritic compartmental length (μm), tapering, synaptic strength (nS), number of spikes, interval (ms), noise, background synaptic strength (nS), background interval (ms), sparseness (fraction of active synapses).

Sim	Gen	Avg S/N	Length	Tapering	SynStren	NumSpikes	Interval	Noise	BgSynStren	BgInterval	Sparseness
1	0	1.56	6.02	0.91	1.25	5.24	5.22	0.58	0.51	740	0.12
1	20	49.48	7.70	0.99	0.71	9.00	7.79	0.00	0.16	1000	0.10
1	151	59.10	7.86	1.00	0.70	8.92	8.56	0.00	0.15	940	0.10
2	0	0.97	5.90	0.90	1.10	5.22	5.60	0.54	0.45	710	0.13
2	20	42.25	8.16	0.99	0.59	9.80	4.90	0.00	0.04	520	0.15
2	122	57.84	7.29	1.00	0.60	10.00	4.98	0.00	0.00	910	0.15
3	0	3.11	5.17	0.90	1.27	6.04	5.45	0.64	0.52	800	0.14
3	20	36.82	7.48	0.90	1.21	9.92	8.70	0.00	0.44	1000	0.05
3	205	46.28	7.11	0.90	1.18	9.96	8.81	0.00	0.19	500	0.05

Having identified the parameter changes related to the initial large scale increase in fitness, I now discuss some of the jumps identified in the fitness that occur in later generations. These jumps, easily identified in two of the three simulations, are interesting because in both cases only two parameters were responsible for these changes.

As shown earlier in Section 8.6.3.1, Simulation 1 showed a decrease in the mean electrotonic

path without there being a similar decrease in the mean path length (see the dashed red line at about generation 32 in Figure 8.10). From the equation used to calculate the mean electrotonic path length (given in Section 5.2.2.3), it can be seen that the only parameters that are involved in this calculation are the dendritic compartmental length and the diameter. So, given the fact that the mean path length, which only depends on the dendritic length, did not change when the neuronal performance increased, the only parameter that could be responsible for this performance improvement was the dendritic diameter. This hypothesis is confirmed by the results shown in Figure 8.15, where the tapering parameter increased (middle graph) in the same generation in which the mean electrotonic path decreased. Surprisingly, the spike interval was another parameter that also seems to be related to the increase of the neuronal performance (bottom graph in Figure 8.15). From further experiments which only varied tapering and spike interval (initial parameter values from generation 31 and final values from generation 32), the results show that none of these two parameters can improve performance when they are varied alone. In fact, when tapering was increased from 0.98 to 1, without changing the spike interval, the actual neuronal performance decreased (results not shown here). The results also show that when both parameters, tapering and spike interval, were increased at the same time, the actual neuronal performance was improved. Thus, we can conclude that the performance improvement found in generation 32 is a consequence of the increase of both parameters, as shown in Figure 8.15.

As noted previously for Simulation 2, there is a similar clear increase of neuronal performance at generation 67 (shown in Figure 8.12). This occurred at the same point as a decrease in the mean path length and the mean electrotonic path length, which suggests that the main parameter responsible for this performance improvement was the dendritic compartmental length. This was confirmed by the results shown in Figure 8.16, where the dendritic length decreased in the same generation in which the s/n ratio increased. Again, tapering was another morphological parameter that exhibited a step change coinciding with a step change in the neuronal performance (bottom graph of Figure 8.16), which could explain the larger relative decrease of the mean electrotonic path length (15%) compared to the decrease of the mean path length (9%). No pattern recognition parameter was found to be responsible for the improvement of the neuronal performance in this simulation.

As mentioned in the previous section, in Simulation 3 there were no clearly defined jumps in performance that matched clear changes in metrics, so no parameter was found that could be directly related to the increase of the neuronal performance. This suggests that this improvement was only a result of the change in the dendritic tree topology (shown in Figure 8.13).

In the three simulations studied here, no further clear relationship was found between the

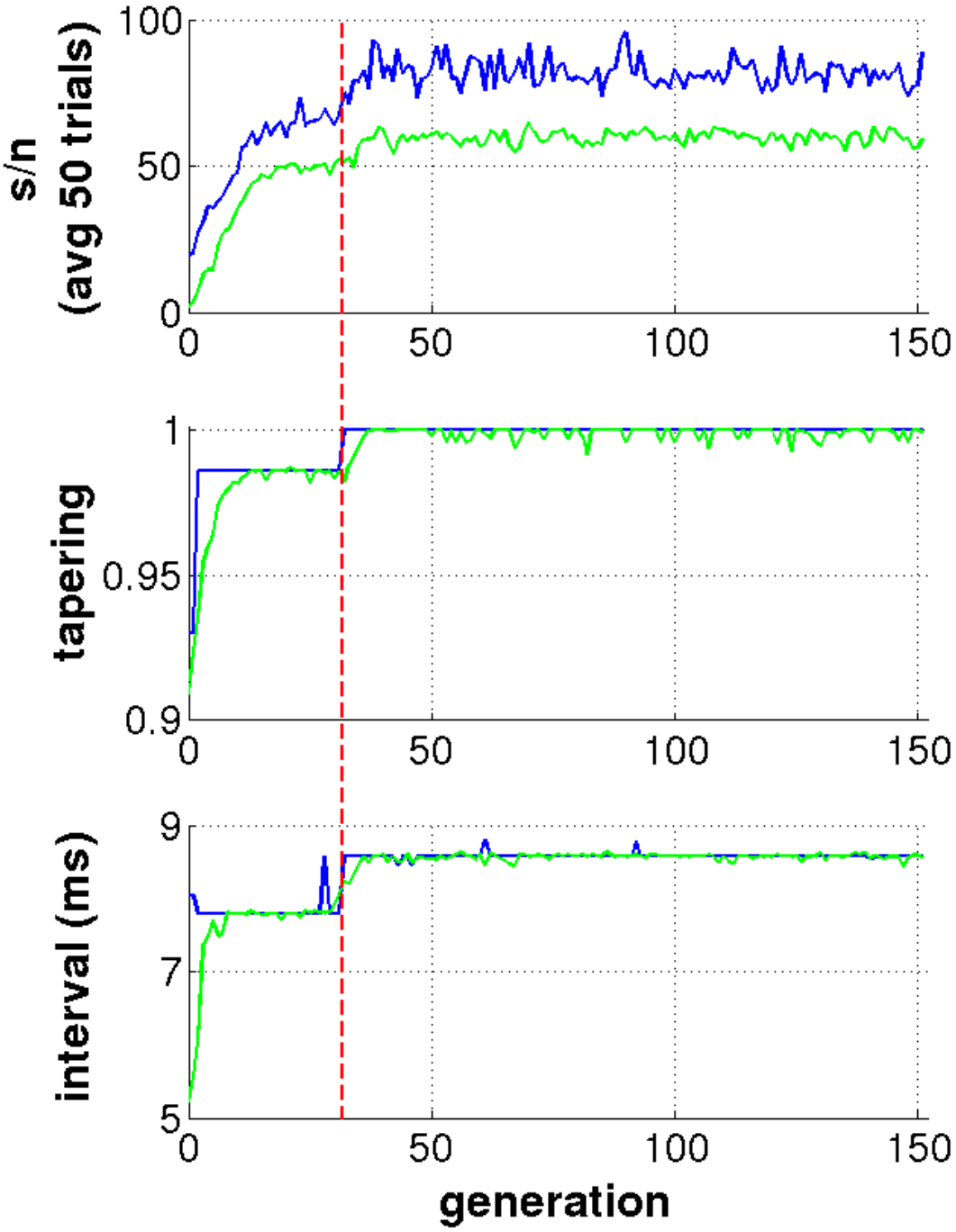


Figure 8.15 – Simulation 1 - Comparing neuronal performance with two parameters: tapering and spike interval.

parameters and the neuronal performance (see Appendix C for a full list of graphs comparing all parameters in each simulation).

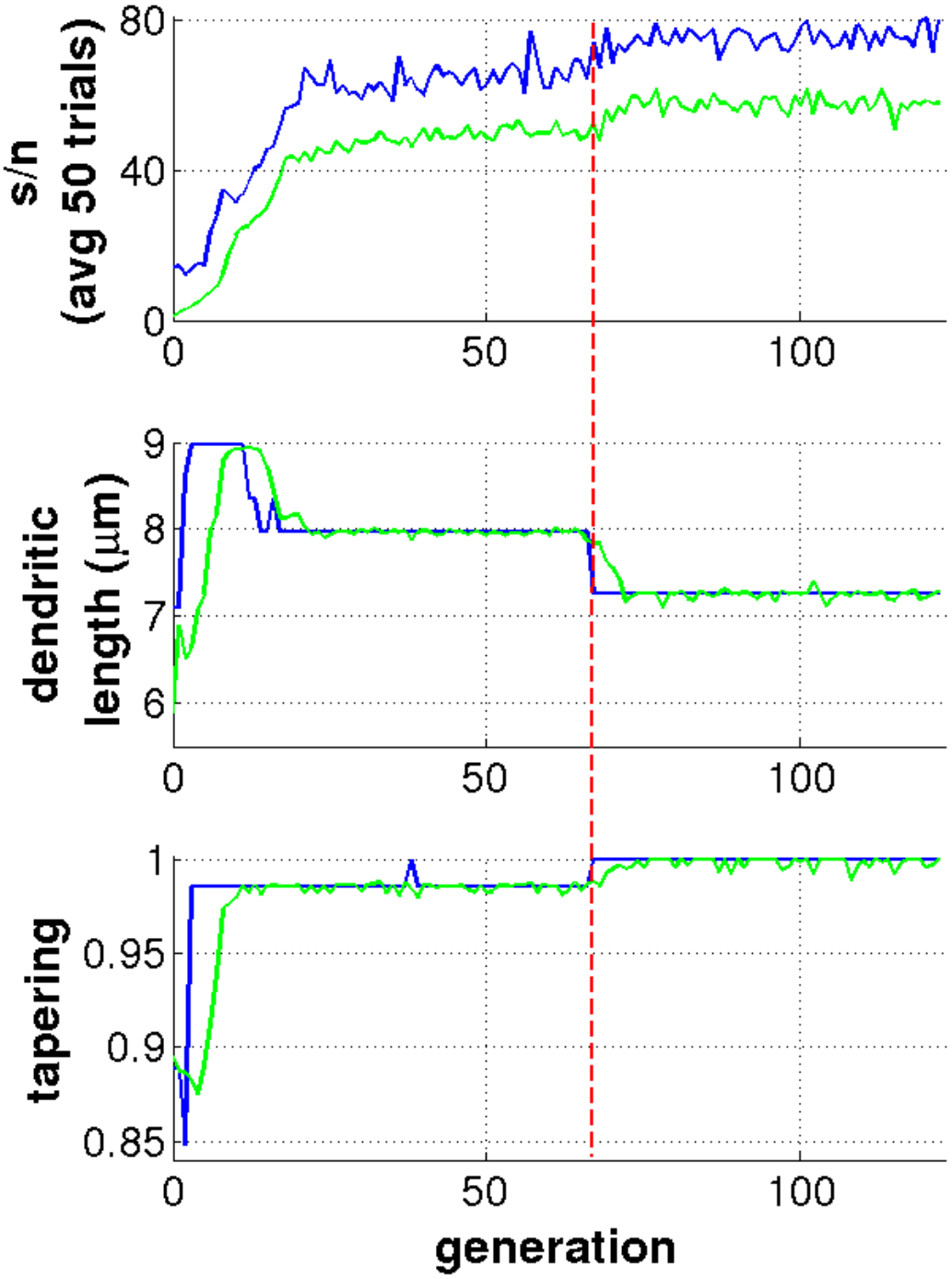


Figure 8.16 – Simulation 2 - Comparing neuronal performance with parameters: dendritic length and tapering.

8.6.3.3 Sensitivity Analysis

The last analysis carried out on the EA results was to identify how sensitive were the parameter values obtained from final evolved individuals. Table 8.6 lists all the parameters values (for morphological and pattern recognition parameters) for the fittest individual in the last generation of each simulation, where each column represents one of the three simulations studied previously. In the last column, a list of the original parameter values used in the experiments with the systematically generated morphologies is presented; these values were hand-tuned to obtain a good neuronal performance. From the comparison between these values and the parameter values obtained from the EA simulations, it is clear that most of the evolved parameters differ from the original values. This comparison shows an increase of the dendritic length, the number of spikes, and the spike interval, a decrease of the synaptic strength for the input pattern and background input and a removal of the input noise in the evolved morphologies. These changes in the models that were evolved resulted in an improvement of the pattern recognition performance by a factor of at least five when compared to the hand-tuned ones.

Table 8.6 – Parameters of the best individual evolved in each simulation. To allow a comparison with the systematically generated morphologies, a column with the hand-tuned original values was included which were used in those experiments (described in Section 7.2).

Parameters	**Sim 1**	**Sim 2**	**Sim 3**	**Original Values**
dendritic length (μm)	7.86	7.25	7.14	5
dendritic tapering	1	1	0.89	1
synaptic strength (nS)	0.71	0.59	1.2	1.5
number of spikes	9	10	10	5
spike interval (ms)	8.57	4.98	8.90	3
noise	0	0	0	1
background synaptic strength (nS)	0.16	0	0.18	0.5
background spike interval (ms)	1000	500	500	1000
sparseness (% active synapses)	10	15	5	10

To analyse the importance of each of the EA parameters more systematically, a one-way sensitivity analysis was chosen where I could determine how the parameter values obtained could influence the fitness function. The results of this kind of analysis are usually presented in a tornado diagram, which is a bar chart where the length of each bar represents the variation of the fitness function for changes of each parameter [57]. This sensitivity analysis is presented in Figure 8.17, where the tornado diagram was plotted along the x-axis to allow a comparison between the results of the three simulations. The analysis was made by varying just one of the parameter values

by 10% up (blue bars) and 10% down (red bars) and while keeping the other parameter values the constant. The new fitness was calculated for each of the parameter variations and by averaging the final fitness over 100 trials. The bars in Figure 8.17 show the difference between the new s/n value resulting from the parameter variation and the original s/n value of the fittest individual. The error bars indicate the standard error of the mean calculated over the 100 trials.

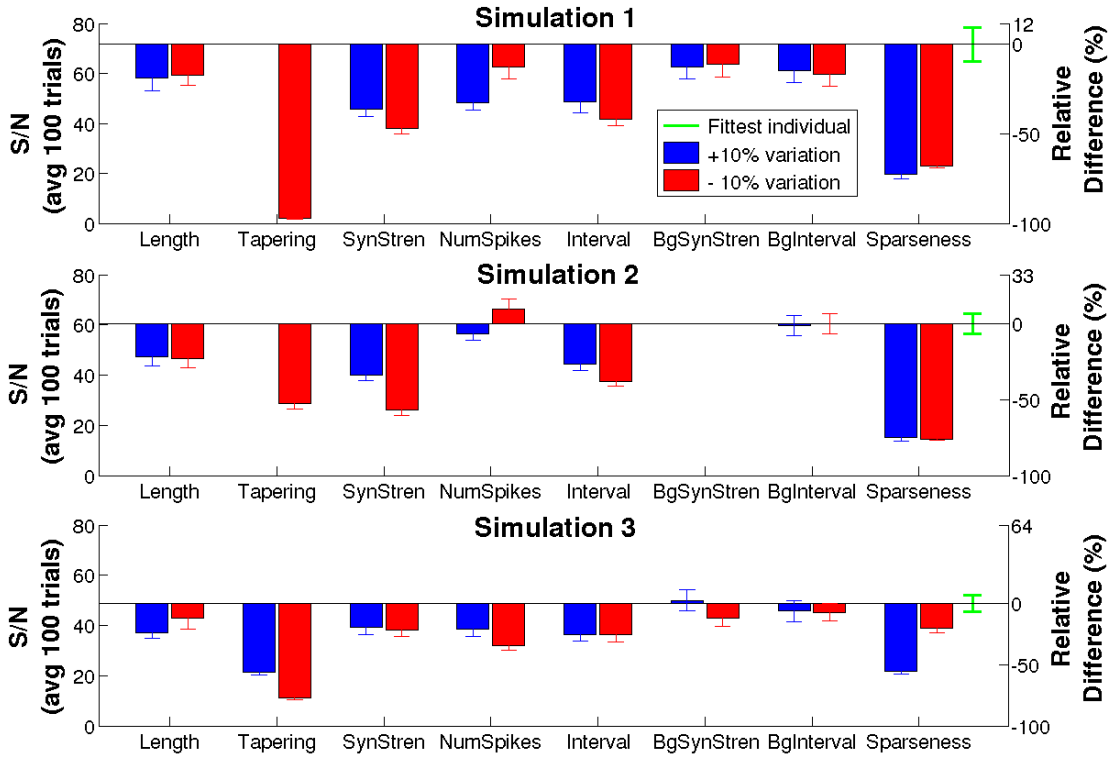


Figure 8.17 – Sensitivity test over the best individual parameters from each simulation. The actual s/n value obtained for the fittest individual evolved is indicated by the horizontal black line, and its standard error of the mean is given by the green error bar on the right side of each graph. The difference between the average s/n value of the fittest individual and the new average obtained from each parameter variation is given by the bars, where the right y-axis shows the relative difference as a percentage. Each parameter was varied by 10% up (blue bars) and 10% down (red bars), and the resulting s/n ratio was averaged over 100 trials, where the error bars indicate the standard error of the mean. The parameter noise is not shown here because it was not possible to vary its value since this is a categorical parameter, accepting only two values (0, 1). Missing bars are due to the parameter concerned already being at the limit of its value so that a decrease or increase in value cannot be made.

It is clear from the sensitivity analysis that the vast majority (42 out of 44) of these 10% changes to the parameters values found by the EA decreased the overall performance, while one change did not have any effect, and only in two cases a very small increase in performance was found. In other words, the EA did well in finding parameters that could not easily be improved. It is noticeable that some parameters appeared to have a larger effect on performance than the others. From Figure 8.17 it is clear that the most sensitive parameters in all three simulations were tapering

and sparseness (fraction of active synapses). From Section 8.6.3.2, it is known that tapering was an important parameter in both Simulations 1 and 2, where the neuronal performance improved when tapering increased. This can explain why the neuronal performance deteriorated when the tapering decreased by 10%, as shown in the first two diagrams in Figure 8.17. The other particularly sensitive parameter, sparseness, can be explained by the fact that it is a parameter that can not be changed independently of the others. As shown in Table 8.6, when the sparseness increased the synaptic strength decreased and vice-versa. Thus, this could explain that when sparseness is changed on its own the neuronal performance degrades, as shown for the three simulations in Figure 8.17.

8.7 Conclusions

In this chapter I have presented results obtained by an extensive investigation of neuronal morphologies in the presence of active conductances. As for the passive models, this study included a comparison of the most distinct morphologies and an analysis of morphologies and parameters that were evolved by using an EA. It also included a comparison of selectively generated morphologies, and highlighted differences between active and passive models. In this chapter both the tree morphology itself and the parameters associated with morphology and pattern recognition where varied and an analysis of the effects on pattern recognition performance carried out. The main conclusions we can draw from the results presented in this chapter are:

1. The EA could evolve effective morphologies in the presence of active conductances, with a resulting pattern recognition performance that was five times better than that of hand-tuned neurons.

2. The specific values of the neuronal parameters play a major role in determining the performance of neurons in the pattern recognition task.

3. Dendritic compartmental length and tapering were shown to be important neuronal parameters when evolving the best pattern recognizers. Moreover, a sensitivity analysis showed that tapering and the number of active synapses were the most sensitive neuronal parameters when measuring the pattern recognition performance of the active neuronal morphologies evolved by the EA.

4. The EA did not evolve the dendritic topology much since the initial population already contained reasonably good morphologies.

5. Mean path length, mean electrotonic path length and mean depth were good predictors of the pattern recognition performance of active models.

6. As in the passive model, the asymmetry index was not a good predictor of good morphologies for the active model as all the morphologies with an asymmetry index in the range between 0 and 0.5 perform as well as the most symmetric one.

Overall we can see that the EA found a population of neurons that perform pattern recognition significantly better than the hand-crafted neurons that I initially used. In fact the EA managed to increase the s/n ratio from 8 to between 40 and 60.

Chapter 9

Conclusion

9.1 Main Contributions

The main contributions of my PhD work to the field of computation neuroscience are:

- In both active and passive neuronal models, I found an almost linear correlation between the mean depth of the dendritic trees and the pattern recognition performance; the best performing neurons were the ones with the smallest mean depth. A similar result was found for the variance of depth, mean path length and mean electronic path length, which correlated with the neuronal performance in some experiments. Interestingly, the asymmetry index did not correlate with the performance for the full range of tree morphologies. In fact, any morphology with an asymmetry index below 0.5 performed as well as the most symmetric one.

- The values of neuronal parameters play a major role in determining the performance of neurons in the pattern recognition task. In particular, the values of the morphological parameters, dendritic compartmental length and tapering, affected the ability of the model neurons to be good pattern recognizers. However, no single parameter setting guaranteed good neuronal performance; in three separate runs of the evolutionary algorithm, different sets of well performing parameters were found.

- In my search for optimal neuronal morphologies for pattern recognition, I investigated different optimisation procedures that could alter different neuronal features such as dendritic topologies and morphological parameters. Following a survey of existing algorithms, I developed my own algorithms to generate dendritic morphologies, either in a systematic way or by evolving them using an Evolutionary Algorithm (EA). My EA was designed to meet six requirements that were not met entirely by existing algorithms. As a result, I could generate

neuronal morphologies that allowed me to understand some of the neuronal features that are important for pattern recognition.

- The final EA could evolve effective morphologies in the presence of active conductances, with a pattern recognition performance that was five times better than that of hand-tuned neurons.

Moreover in a preliminary investigation, I studied the effect of different types and features of synaptic plasticity in a Purkinje cell model. In these studies, I found that the best pattern recognition performance in the model resulted from LTD saturating at a lower bound value of zero, which corresponds to silencing the PF synapses completely. On the other hand, the ability of the model to discriminate between learned and novel input patterns was unaffected by the presence of inhibitory plasticity for a wide range of parameter values.

9.2 Future Research

Some ideas are suggested as extensions to my work:

- Optimisation of ion channel distributions. Ion channels are a key determinant of neuronal function, as they regulate the ion flux into the cell which is responsible for the neuronal transmission of signals. In this work, I used a set of voltage-gated conductances studied previously in related works [42, 76]. However, previous research suggests that including variable types and densities of ion channels could improve the neuronal performance as it can change the firing pattern [13, 16, 72, 75]. Thus, a next step of this research should be the optimisation of voltage-gated conductances by varying their type, density and location.

- Study of input pattern characteristics such as clustered and asynchronous input patterns. Previous work has shown that these features of input patterns play an important role in information processing [81, 51] and consequently they should be investigated in more detail in further studies of pattern recognition by neuronal models.

- Study of different spike response features. To analyse the pattern recognition performance in the neuronal morphologies evolved by the Evolutionary Algorithm, different features of the neuronal response can be considered. Instead of using the amplitude of the EPSP response, as used in the passive models, or the number of spikes, as used in active models, other features of the spike response could be tested. Previous work has used the latency of the first spike or the length of a pause after presentation of a pattern to measure the pattern recognition performance [66]. By adjusting the neuronal model to have intrinsic properties

such as spontaneous activity, these and other spike response features could be used to measure the neuronal performance.

- Investigate the use of information theory to evaluate the neuronal performance. In this work and in previous studies of pattern recognition, the signal-to-noise ratio was the only measurement used to evaluate the neuronal performance of the morphologies studied. A next step suggested is to study information theoretical measurements such as Shannon's entropy [59], trying to identify better measures of neuronal performance.

- Incrementally increase the complexity of the active model. From the passive to the active model, a number of features were included such as optimization of parameters, inclusion of background input and use of new morphological metrics when compared to the passive model, apart of the inclusion of the ionic channels. To acquire a better understanding of the active model and its results, each of these steps should be done individually and progressively, first in the passive model and then in the active one. Thus, the results from both models could be fairly compared.

9.3 List of Publications

During the four years of my PhD, I have attended several conferences; I have published four abstracts, presented as posters, and one full conference paper.

The following papers were presented in conferences in my first two years of research. Their results were presented in Chapter 4.

- Seventeenth Annual Computational Neuroscience Meeting: CNS 2008. Portland, USA. G. De Sousa, R. Adams, N. Davey, V. Steuber. Determinants of pattern recognition by cerebellar Purkinje cells. Published in BMC Neuroscience, 9, Suppl 1 (2008), P67.

- I Congress IBRO/LARC of Neurosciences for Latin America, Caribbean and Iberian Peninsula: NeuroLATAM 2008. Buzios, Brazil. G. De Sousa, R. Adams, N. Davey, V. Steuber. Effect of LTD Saturation on Pattern Recognition by Cerebellar Purkinje Cells.

- International Conference on Adaptive and Natural Computing Algorithms: ICANNGA 2009. Kuopio, Finland. G. De Sousa, R. Adams, N. Davey, R. Maex, V. Steuber. The Effect of Different Forms of Synaptic Plasticity on Pattern Recognition in the Cerebellar Cortex. Published in Lecture Notes in Computer Science, 2009. M. Kolehmainen, P. Toivanen, and B. Beliczynski, Eds., vol. 5495, Springer Berlin / Heidelberg, pp. 413–422.

The following papers were presented in conferences in the last year of my research. They summarize some of the results presented in Chapters 7 and 8.

- Nineteenth Annual Computational Neuroscience Meeting: CNS 2010. San Antonio, USA. G. De Sousa, R. Maex, R. Adams, N. Davey, V. Steuber. Optimization of Neuronal Morphologies for Pattern Recognition. Published in BMC Neuroscience, 11, Suppl 1 (2010), P80.

- Twentieth Annual Computational Neuroscience Meeting: CNS 2011. Stockholm, Sweden. G. De Sousa, R. Maex, R. Adams, N. Davey, V. Steuber. The Effect of Dendritic Morphology on Pattern Recognition in the Presence of Active Conductances. Published in BMC Neuroscience, 12, Suppl1 (2011) , P315.

Bibliography

[1] Achelis, S. B. (2000) *Technical Analysis from A to Z*, pp. 184–192. McGraw-Hill, 2nd edn.

[2] Albus, J. S. (1971) A theory of cerebellar function. *Mathematical Biosciences*, **10**, 25–61.

[3] Ascoli, G. A., Krichmar, J. L., Scorcioni, R., Nasuto, S. J., & Senft, S. L. (2001) Computer generation and quantitative morphometric analysis of virtual neurons. *Anatomy and Embryology*, **204**, 283–301.

[4] Ascoli, G. A. & Krichmar, J. L. (2000) L-neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, **32-33**, 1003–1011.

[5] Back, T., Fogel, D. B., & Michalewicz, Z. (2000) *Evolutionary Computation 1 Basic Algorithms and Operators*, vol. 1. Institute of Pyshics Publishing.

[6] Bear, M. F., Connors, B., & Paradiso, M. (2006) *Neuroscience: Exploring the Brain*. Lippincott Williams and Wilkins, third edn.

[7] Bower, J. M. & Beeman, D. (1995) *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. TELOS.

[8] Burke, R. E., Marks, W. B., & Ulfhake, B. (1992) A parsimonious description of motoneuron dendritic morphology using computer simulation. *J. Neurosci.*, **12**, 2403–2416.

[9] Carnevale, N. & Hines, M. (2006) *The NEURON Book*. Cambridge University Press.

[10] Cook, E. P. & Johnston, D. (1997) Active dendrites reduce location-dependent variability of synaptic input trains. *Journal of Neurophysiology*, **78**, 2116–2128.

[11] Cuntz, H., Borst, A., & Segev, I. (2007) Optimization principles of dendritic structure. *Theoretical Biology and Medical Modelling*, **4**, 21.

[12] Dayan, P. & Willshaw, D. J. (1991) Optimising synaptic learning rules in linear associative memories. *Biol Cybern*, **65**, 253–265.

[13] De Schutter, E. & Bower, J. M. (1994) An active membrane model of the cerebellar purkinje cell. i. simulation of current clamps in slice. *Journal of Neurophysiology*, **71**, 375–400.

[14] De Schutter, E. & Bower, J. M. (1994) An active membrane model of the cerebellar purkinje cell: Ii. simulation of synaptic responses. *Journal of Neurophysiology*, **71**, 401–419.

[15] Etzion, Y. & Grossman, Y. (1998) Potassium currents modulation of calcium spike firing in dendrites of cerebellar purkinje cells. *Experimental Brain Research*, **122**, 283–294, 10.1007/s002210050516.

[16] Fohlmeister, J. F. & Miller, R. F. (1997) Mechanisms by which cell geometry controls repetitive firing. *Journal of Neurophysiology*, **78**, 1948–1964.

[17] Fregnac, Y. (2002) Hebbian synaptic plasticity. Arbib, M. A. (ed.), *The Handbook of Brain Theory and Neuronal Networks*, pp. 515–522, MIT Press, 2 edn.

[18] Graham, B. P. (2001) Pattern recognition in a compartmental model of a ca1 pyramidal neuron. *Network: Computation in Neural Systems*, **12**, 473–492.

[19] Gulledge, A. T., Kampa, B. M., & Stuart, G. J. (2005) Synaptic integration in dendritic trees. *Journal of Neurobiology*, **64**, 75–90.

[20] Harding, E. F. (1971) The probabilities of rooted tree-shapes generated by random bifurcation. *Advances in Applied Probability*, **3**, 44–77.

[21] Hillman, D. (1979) Neuronal shape parameters and substructures as a basis of neuronal form. Schmitt, F. (ed.), *The Neurosciences, Fourth Study Program*, pp. 477–498, MIT Press.

[22] Hines, M. L. & Carnevale, N. T. (1997) The neuron simulation environment. *Neural Computation*, **9**, 1179–1209.

[23] Hodgkin, A. L. & Huxley, A. F. (1952) The components of membrane conductance in the giant axon of loligo. *The Journal of Physiology*, **116**, 473–496.

[24] Hodgkin, A. L. & Huxley, A. F. (1952) Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of Physiology*, **116**, 449–472.

[25] Hodgkin, A. L. & Huxley, A. F. (1952) The dual effect of membrane potential on sodium conductance in the giant axon of loligo. *The Journal of Physiology*, **116**, 497–506.

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

[27] Hodgkin, A. L., Huxley, A. F., & Katz, B. (1952) Measurement of current-voltage relations in the membrane of the giant axon of loligo. *The Journal of Physiology*, **116**, 424–448.

[28] Holt, G. R., Softky, W. R., Koch, C., & Douglas, R. J. (1996) Comparison of discharge variability in vitro and in vivo in cat visual cortex neurons. *Journal of Neurophysiology*, **75**, 1806–1814.

[29] Isope, P. & Barbour, B. (2002) Properties of unitary granule cell-purkinje cell synapses in adult rat cerebellar slices. *Journal of Neuroscience*, **22**, 9668–9678.

[30] Ito, M. (1984) *The cerebellum and neural control*. Raven Press.

[31] Ito, M. (2001) Cerebellar long-term depression: Characterization, signal transduction, and functional roles. *Physiol. Rev.*, **81**, 1143–1195.

[32] Ito, M., Sakurai, M., & Tongroach, P. (1982) Climbing fibre induced depression of both mossy fibre responsiveness and glutamate sensitivity of cerebellar purkinje cells. *J Physiol*, **324**, 113–134.

[33] Jaeger, D. (2007) Pauses as neural code in the cerebellum. *Neuron*, **54**, 9–10.

[34] Kandel, E., Schwartz, J., & Jessell, T. (2000) *Principles of Neural Science*. McGraw-Hill Medical, 4 edn.

[35] Kass, J. I. & Mintz, I. M. (2006) Silent plateau potentials, rhythmic bursts, and pacemaker firing: Three patterns of activity that coexist in quadristable subthalamic neurons. *Proceedings of the National Academy of Sciences of the United States of America*, **103**, 183–188.

[36] Krichmar, J. L., Nasuto, S. J., Scorcioni, R., Washington, S. D., & Ascoli, G. A. (2002) Effects of dendritic morphology on ca3 pyramidal cell electrophysiology: a simulation study. *Brain Research*, **941**, 11–28.

[37] Kudina, L. & Andreeva, R. (2010) Repetitive doublet firing of motor units: evidence for plateau potentials in human motoneurones. *Experimental Brain Research*, **204**, 79–90, 10.1007/s00221-010-2298-z.

[38] Lindenmayer, A. (1968) Mathematical models for cellular interaction in development i & ii. *Journal of Theoretical Biology*, **18**, 280–315.

[39] Llinas, R. & Sugimori, M. (1980) Electrophysiological properties of in vitro purkinje cell dendrites in mammalian cerebellar slices. *Journal of Physiology (London)*, **305**, 197–213.

[40] Llinas, R. & Sugimori, M. (1980) Electrophysiological properties of in vitro purkinje cell somata in mammalian cerebellar slices. *Journal of Physiology (London)*, **305**, 171–195.

[41] London, M. & Häusser, M. (2005) Dendritic computation. *Annual Review of Neuroscience*, **28**, 503–532.

[42] Mainen, Z. F. & Sejnowski, T. J. (1996) Influence of dendritic structure on firing pattern in model neocortical neurons. *Nature*, **382**, 363–366.

[43] Major, G., Evans, J., & Jack, J. (1993) Solutions for transients in arbitrarily branching cables: I. voltage recording with a somatic shunt. *Biophysical Journal*, **65**, 423 – 449.

[44] Mandelblat, Y., Etzion, Y., Grossman, Y., & Golomb, D. (2001) Period doubling of calcium spike firing in a model of a purkinje cell dendrite. *Journal of Computational Neuroscience*, **11**, 43–62, 10.1023/A:1011252730249.

[45] Marr, D. (1969) A theory of cerebellar cortex. *Journal of Physiology (London)*, **202**, 437–470.

[46] Marrocco, C. (2006) *Learning Classification Systems Maximizing the Area Under the ROC Curve*. Ph.D. thesis, Universita degli Studi di Cassino.

[47] Metz, C. E., Herman, B. A., & Shen, J.-H. (1998) Maximum likelihood estimation of receiver operating characteristic (roc) curves from continuously-distributed data. *Statistics in Medicine*, **17**, 1033–1053.

[48] Mitchell, M. (1998) *An Introduction to Genetic Algorithms*. MIT Press.

[49] Mittmann, W. & Hausser, M. (2007) Linking synaptic plasticity and spike output at excitatory and inhibitory synapses onto cerebellar purkinje cells. *J. Neurosci.*, **27**, 5559–5570.

[50] Park, S. H., Goo, J. M., & Jo, C.-H. (2004) Receiver operating characteristic (roc) curve: Practical review for radiologists. *Korean J Radiol*, **5**, 11–18.

[51] Pissadaki, E. K., Sidiropoulou, K., Reczko, M., & Poirazi, P. (2010) Encoding of spatio-temporal input characteristics by a ca1 pyramidal neuron model. *PLoS Comput Biol*, **6**, e1001038.

[52] Poli, R., Langdon, W., & Mcphee, N. (2008) *A field guide to genetic programming*. Lulu Enterprises.

[53] Prusinkiewicz, P. & Lindenmayer, A. (1990) *The algorithmic beauty of plants*. Springer-Verlag New York, Inc.

[54] Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., Lamantia, A. S., Mcnamara, J. O., & Williams, M. S. (2004) *Neuroscience*. Sinauer Associates, third edn.

[55] Rall, W. (1959) Branching dendritic trees and motoneuron membrane resistivity. *Experimental Neurology*, **1**, 491–527.

[56] Rapp, M., Segev, I., & Yarom, Y. (1994) Physiology, morphology and detailed passive models of guinea-pig cerebellar purkinje cells. *Journal of Physiology (London)*, **474**, 101–118.

[57] Reilly, T. (2000) Sensitivity analysis for dependent variables. *Decision Sciences*, **31**, 551–572.

[58] Segev, I. & London, M. (2002) Dendritic processing. Arbib, M. A. (ed.), *The Handbook of Brain Theory and Neuronal Networks*, pp. 324–332, MIT Press, 2 edn.

[59] Shannon, C. E. (1948) A mathematical theory of communication. *Bell System Technical Journal*, **27**, 379–423;623–656.

[60] Sheperd, G. M. (2003) *The Synaptic Organization of the Brain*. Oxford University Press, USA, 5 edn.

[61] Simoes, A. & Costa, E. (2000) Using genetic algorithms with asexual transposition. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)*, pp. 323–330, Morgan Kaufmann.

[62] Sivanandam, S. & Deepa, S. N. (2007) *Introduction to Genetic Algorithms*. Springer.

[63] Sjostrom, P. J., Rancz, E. A., Roth, A., & Hausser, M. (2008) Dendritic excitability and synaptic plasticity. *Physiol. Rev.*, **88**, 769–840.

[64] Sterratt, D., Graham, B., Gillies, A., & Willshaw, D. (2011) *Principles of Computational Modelling in Neuroscience*. Cambridge University Press.

[65] Steuber, V. & De Schutter, E. (2001) Long-term depression and recognition of parallel fibre patterns in a multi-compartmental model of a cerebellar purkinje cell. *Neurocomputing*, **38**, 383–388.

[66] Steuber, V., Mittmann, W., Hoebeek, F. E., Silver, R. A., De Zeeuw, C. I., Hausser, M., & De Schutter, E. (2007) Cerebellar ltd and pattern recognition by purkinje cells. *Neuron*, **54**, 121–136.

[67] Stiefel, K. M. & Sejnowski, T. J. (2007) Mapping function onto neuronal morphology. *J Neurophysiol*, **98**, 513–526.

[68] Stufflebeam, R. (2008), Neurons, synapses, action potentials, and neurotransmission. `www.mind.ilstu.edu/curriculum/neurons_intro/neurons_intro.php`, accessed in 27 Nov 2011.

[69] Torben-Nielsen, B., Tuyls, K., & Postma, E. (2007) On the neuronal morphology-function relationship: A synthetic approach. Tuyls, K., Westra, R., Saeys, Y., & Nowï¿œ, A. (eds.), *Knowledge Discovery and Emergent Complexity in Bioinformatics*, vol. 4366, pp. 131–144, Springer Berlin / Heidelberg.

[70] Torben-Nielsen, B., Tuyls, K., & Postma, E. (2008) Evol-neuron: Neuronal morphology generation. *Neurocomput.*, **71**, 963–972.

[71] Torben-Nielsen, B., Tuyls, K., & Postma, E. O. (2006) Shaping realistic neuronal morphologies: An evolutionary computation method. *International Joint Conference on Neural Networks, IJCNN 2006*, Vancouver, BC, Canada, pp. 573–580, IEEE.

[72] Torben-Nielsen, B. & Stiefel, K. M. (2009) Systematic mapping between dendritic function and structure. *Network: Computation in Neural Systems*, **20**, 69–105.

[73] Toroczkai, Z. (2001) Topological classification of binary trees using the horton-strahler index. *Phys. Rev. E*, **65**, 016130.

[74] Tyrrell, T. & Willshaw, D. (1992) Cerebellar cortex: its simulation and the relevance of marr's theory. *Philosophical Transactions of the Royal Society of London, Series B*, **336**, 239–257.

[75] van Elburg, R. A. J. & van Ooyen, A. (2010) Impact of dendritic size and dendritic topology on burst firing in pyramidal cells. *PLoS Comput Biol*, **6**, e1000781.

[76] van Ooyen, A., Duijnhouwer, J., Remme, M., & van Pelt, J. (2002) The effect of dendritic topology on firing patterns in model neurons. *Network: Computation in Neural Systems*, **13**, 311–325.

[77] van Pelt, J. & Verwer, R. (1985) Growth models (including terminal and segmental branching) for topological binary trees. *Bulletin of Mathematical Biology*, **47**, 323–336, 10.1007/BF02459919.

[78] van Pelt, J., Uylings, H. B. M., Verwer, R. W. H., Pentney, R. J., & Woldenberg, M. J. (1992) Tree asymmetry–a sensitive and practical measure for binary topological trees. *Bulletin of Mathematical Biology*, **54**, 759–784.

[79] van Pelt, J. & Verwer, R. W. H. (1983) The exact probabilities of branching patterns under terminal and segmental growth hypotheses. *Bulletin of Mathematical Biology*, **45**, 269–285.

[80] Walter, J. T. & Khodakhah, K. (2006) The linear computational algorithm of cerebellar purkinje cells. *J. Neurosci.*, **26**, 12861–12872.

[81] Walter, J. T. & Khodakhah, K. (2009) The advantages of linear information processing for cerebellar computation. *Proceedings of the National Academy of Sciences*, **106**, 4471–4476.

[82] Wang, S. S. H., Denk, W., & Häusser, M. (2000) Coincidence detection in single dendritic spines mediated by calcium release. *Nat Neurosci*, **3**, 1266–1273.

[83] Wen, Q. & Chklovskii, D. B. (2008) A cost-benefit analysis of neuronal morphology. *J Neurophysiol*, **99**, 2320–2328.

[84] Willshaw, D. J., Buneman, O. P., & Longuet-Higgins, H. C. (1969) Non-holographic associative memory. *Nature*, **222**, 960–962.

Appendix A

Analytical Calculation of the Signal-to-Noise Ratio in the ANN

(R. Maex, personal communication, 3 Feb 2009)

Suppose that S denotes the total number of Synapses, L denotes the number of Learned patterns, A denotes the number of Active synapses per pattern and D denotes the number of Depressed synapses after learning. In this demonstration, I consider that all synapses are binary and they are all depressed by the LTD saturation value of zero.

The calculation of the S/N ratio consists of two subproblems: a) finding the number D of depressed synapses; b) calculating the responses to unlearned patterns. Let's start with the second problem, so D is supposed to be known.

Calculating the responses to unlearned patterns is equivalent to the problem of drawing A balls from a population of S balls of which D are white and $S - D$ balls are red and calculating the distribution of the number of red balls (drawn on each trial). This should be a hypergeometric distribution, which can however, if S is very large, be approximated by a binomial distribution with the probability of drawing a red ball being equal to $(1 - D/S)$. Then, the average output of the ANN for unlearned patterns is $A * (1 - D/S)$. The variance is $A * (1 - D/S) * (D/S)$. We also know that for learned patterns the mean output is zero with variance zero. Hence, the S/N ratio becomes:

$$S/N = \frac{(\mu_s - \mu_n)^2}{0.5\,(\sigma_s^2 + \sigma_n^2)}$$

$$= 2\frac{(A(1 - D/S))^2}{A(1 - (D/S)(D/S)}$$

$$= 2\frac{A(1 - D/S)}{(D/S)}$$

$$= 2\frac{A(S-D)}{D}$$

What is the value of D? If the number of learned patterns L is very small and/or the patterns are very sparse (A small), then the degree of overlap between learned patterns will also be small, so that D becomes approximately equal to $L * A$ (each depressed synapse is unique).

Then the S/N ratio reduces to:

$$S/N = 2\frac{A(S-LA)}{LA}$$

$$= 2\frac{(S-LA)}{L}$$

$$\approx 2\frac{S}{L}$$

where it is assumed that $LA \ll S$. Hence the S/N is proportional to the number of synapses, and inversely proportional to the number of learned patterns.

If LA is large, then we cannot assume that LA is a good approximation of D. I do not know yet how to calculate D in that case! But let's assume all synapses are independently depressed with probability A/S for each pattern, then the probability that a synapse is not being depressed by any of the L patterns equals $(1 - A/S)^L$, and consequently the mean number D of depressed synapses will be $S - S(1 - A/S)^L$.

Appendix B

Systematically Tree Generation Algorithms

B.1 Lisp Code to Generate Trees Exhaustively

```
;;;;;;;;;;;;;;;;;; Main code ;;;;;;;;;;;;;;;;;;
;this function finds all splits in N, a b such that a <= b and a +
     b = N
(defun find-numeric-pairs (N &aux result)
   (dotimes (i (floor (/ N 2)))
      (push (list (+ i 1) (- N (+ i 1)) ) result))
   result)

(defun generate-trees-exhaustively (N)
   (with-open-file (stream "trees.txt" :direction :output :
       if-exists :supersede)
      (Let ((pt (mapcar 'partition (trees N))))
         (dolist (x pt)
            (format stream "~%~a" x)))))

;;;;;;;;;;;;;;;;;; Auxiliar code ;;;;;;;;;;;;;;;;;;

;;;these two mutually recursive functions
(defun trees (N)
   (if (= N 1) '(1)
      (trees-helper (find-numeric-pairs N))))

(defun trees-helper (splits)
```

```
   (if (null splits) nil
      (append (cartesian (trees (caar splits)) (trees (cadar
          splits)))
         (trees-helper (cdr splits)))))

;cartesian product eg (a b c) * ( d e) = (a d) (a e) (b d) (b e) (
    c d) (c e)
(defun distl (m N &aux res)
   (dolist (x N)
      (push (list m x) res))
         (reverse res))

(defun cartesian ( M N )
   (cond
      ((null M ) nil)
      (t (append (distl (car M) N)
         (cartesian (cdr M) N)))))

(defun partition (tree)
   (cond ((null tree) nil)
      ((equal tree '( 1 1)) (list 2 '(1 1)))
      ((equal (car tree) 1)
      (let ((pt (partition (cadr tree))))
         (list (+ 1 (car pt)) 1 pt)))
      (T (let ((ptl (partition (car tree)))
            ptr (partition (cadr tree))))
         (list (+ (car ptl) (car ptr)) ptl ptr)))))
```

B.2 Lisp Code to Generate Tree Samples from the Search Space

```
;;;;;;;;;;;;;;;;;; Main code ;;;;;;;;;;;;;;;;;;
;;Code to produce random samples of the tree space if bias is 0.5
    or
;; very asymmetric trees if bias is 0
(defun find-a-asymmetric-random-numeric-pair (N &optional (bias
    0.5))
```

```
   (let ((i (+ 1 (random (max 1 (floor (* N bias)))))))
      (list i (- N i))))

;;Code to produce very symmetric trees:
;;this function splits N into a,b such that a + b = N
;;and a <= N/2 a >= N/2 - (bias * N/2) - the lower bias the more
    symmetric the trees become
(defun find-a-symmetric-random-numeric-pair (N &optional (bias
    0.5))
   (let ((i (- (floor (/ N 2))  (random (max 1 (floor (* (/ N 2)
       bias)))))))
      (list i (- N i))))

;;how to use: (lots-of-trees num_terminals num_trees bias asym))
;;bias range: 0-0.5, where 0 generates more (a)symmetric trees and
     0.5 generate random trees
;;asym values: 0,1. Use 0 to generate asymmetric trees or 1 to
    generate symmetric trees
;;eg. (lots-of-trees 120 1000 0.1 0)
(defun lots-of-trees (N how-many &optional (bias 0.5) (asym 0))
   (with-open-file (stream "trees.txt" :direction :output :
       if-exists :supersede)
      (dotimes (i how-many)
         (format stream "~%~a" (partition (car (a-tree N bias asym
             )))))))


;;;;;;;;;;;;;;;;;; Auxiliar code ;;;;;;;;;;;;;;;;;;
(defun a-tree (N &optional (bias 0.5) (asym 0))
   (if (= N 1) '(1)
      (if (= asym 0)
         (a-tree-helper (list (
             find-a-asymmetric-random-numeric-pair N bias)) bias)
      (a-tree-helper (list (find-a-symmetric-random-numeric-pair N
          bias)) bias))))

(defun a-tree-helper (splits &optional (bias 0.5))
```

```
   (if (null splits) nil
      (append (cartesian (a-tree (caar splits)) (a-tree (cadar
          splits) bias))
         (a-tree-helper (cdr splits) bias))))

;cartesian product eg (a b c) * ( d e) = (a d) (a e) (b d) (b e) (
    c d) (c e)
(defun distl (m N &aux res)
   (dolist (x N)
      (push (list m x) res))
         (reverse res))

(defun cartesian ( M N )
   (cond
      ((null M ) nil)
      (t  (append (distl (car M) N)
         (cartesian (cdr M) N)))))

(defun partition (tree)
   (cond  ((null tree) nil)
      ((equal tree '( 1 1)) (list 2 '(1 1)))
      ((equal (car tree) 1)
         (let ((pt (partition (cadr tree))))
            (list (+ 1 (car pt)) 1 pt)))
         (T (let ((ptl (partition (car tree)))
                (ptr (partition (cadr tree))))
            (list (+ (car ptl) (car ptr)) ptl ptr)))))
```

Appendix C

Evolution of EA Parameters

C.1 Simulation 1

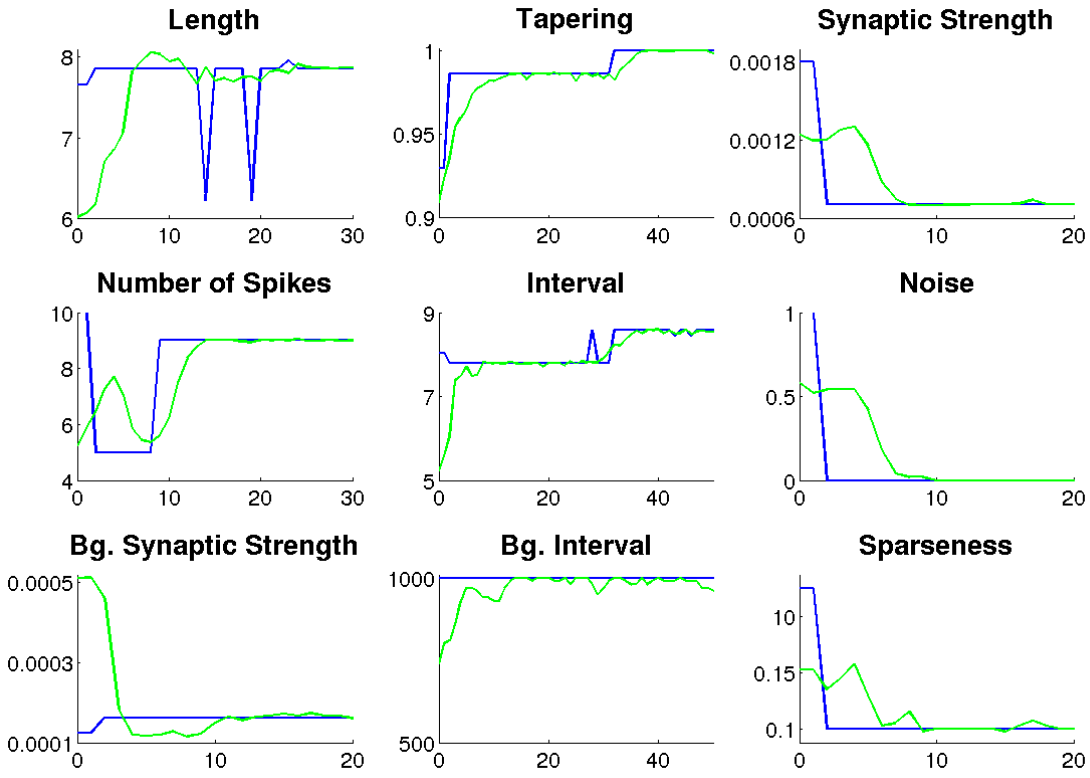


Figure C.1 – Evolution of all the nine parameter optimised by the EA.

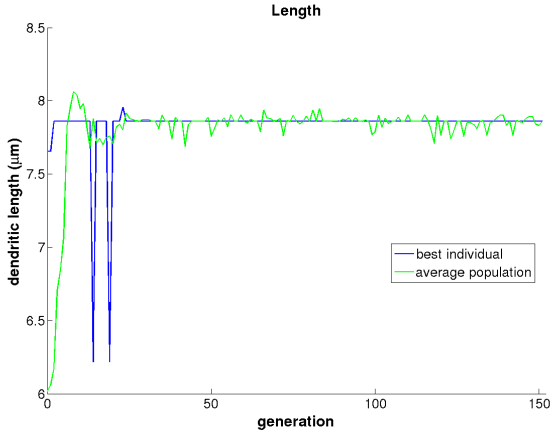


Figure C.2 – Dendritic compartmental length.

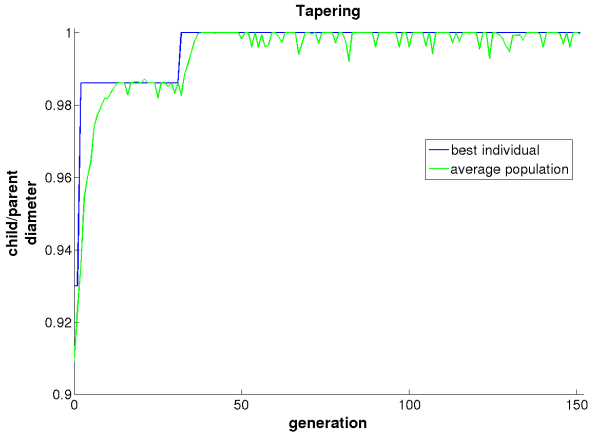


Figure C.3 – Tapering.

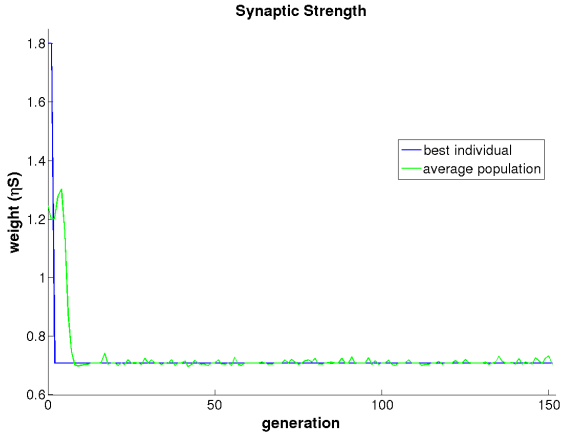


Figure C.4 – Synaptic strength.

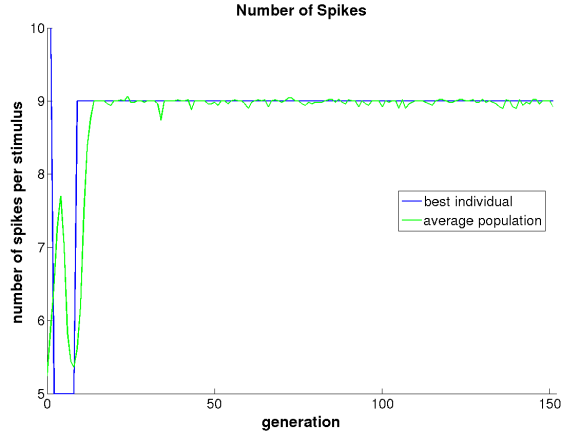


Figure C.5 – Number of spikes.

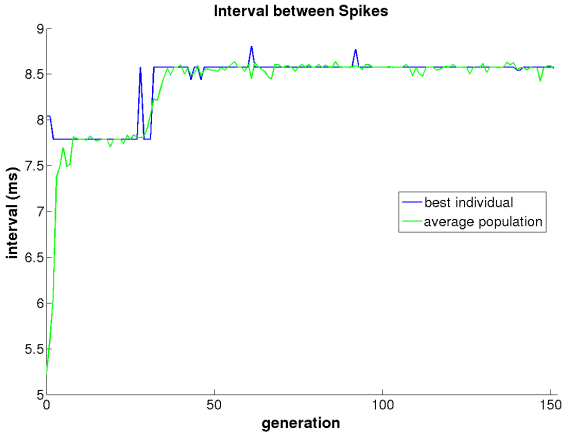


Figure C.6 – Interval between spikes.

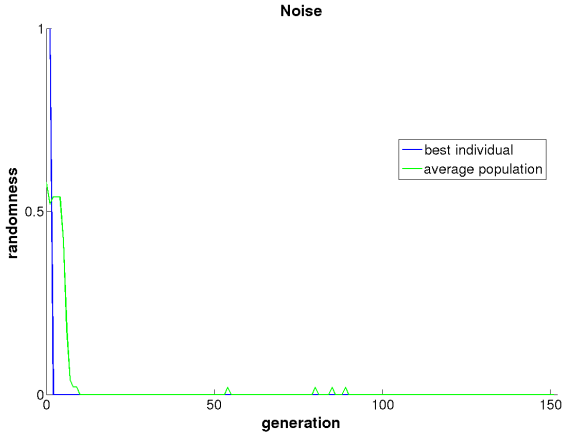


Figure C.7 – Noise.

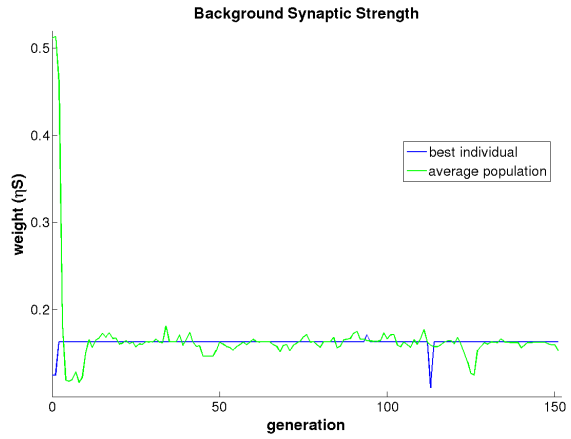


Figure C.8 – Background synaptic strength.

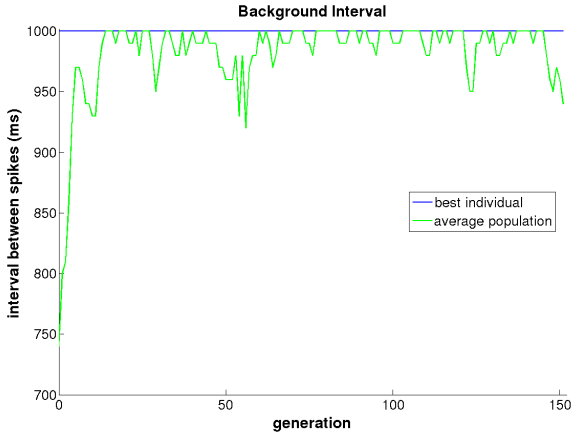


Figure C.9 – Background interval.

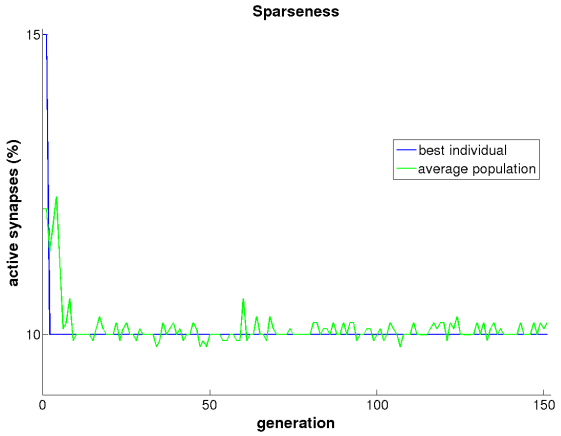


Figure C.10 – Sparseness. It is given by the percentage of active synapses.

C.2 Simulation 2

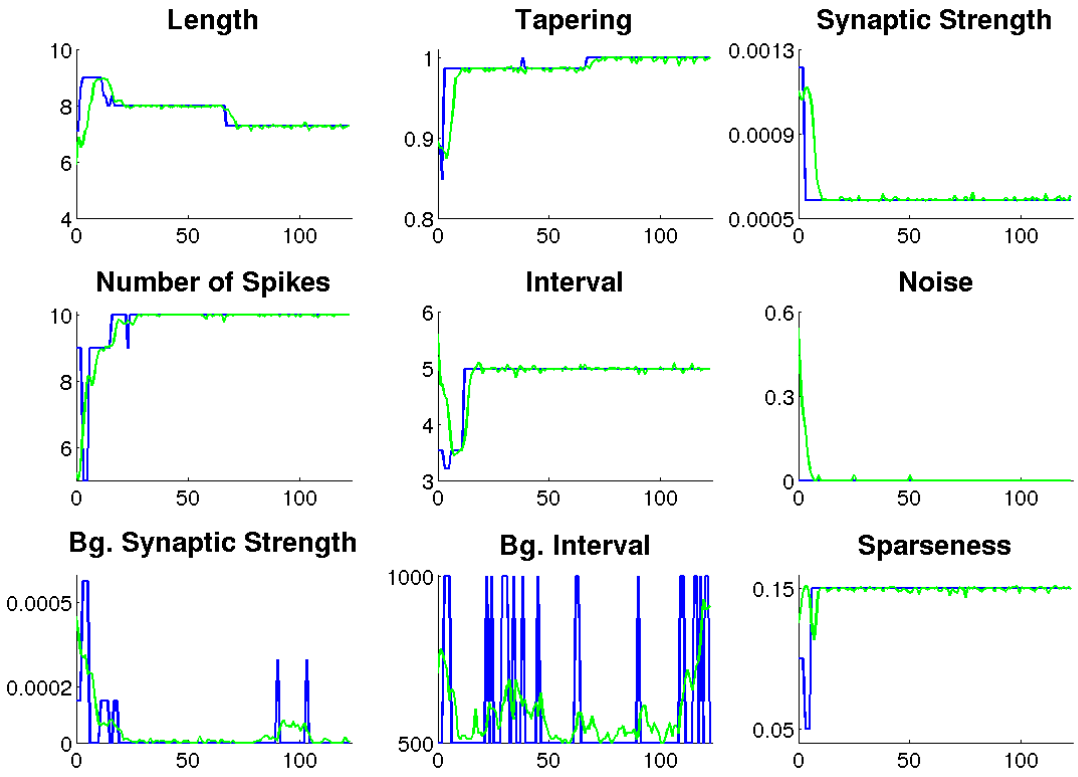


Figure C.11 – Evolution of all the nine parameter optimised by the EA.

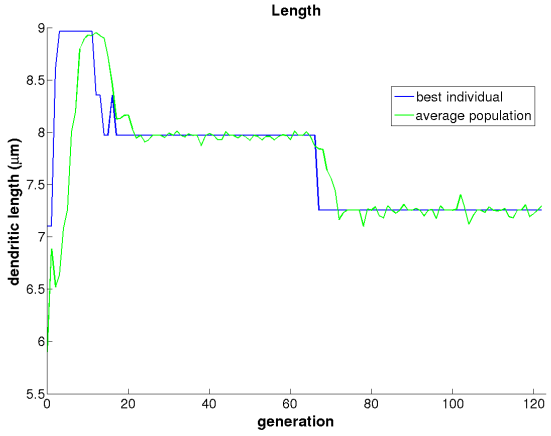


Figure C.12 – Dendritic compartmental length.

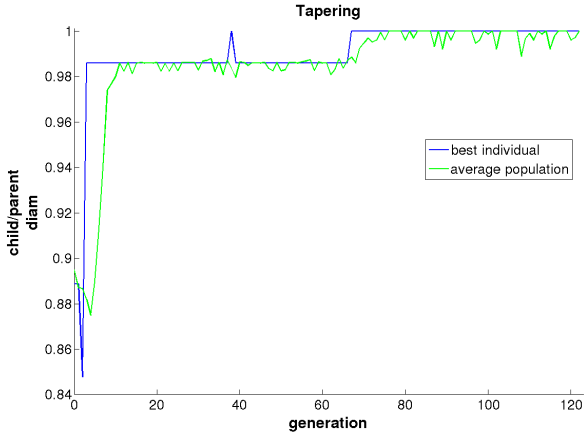


Figure C.13 – Tapering.

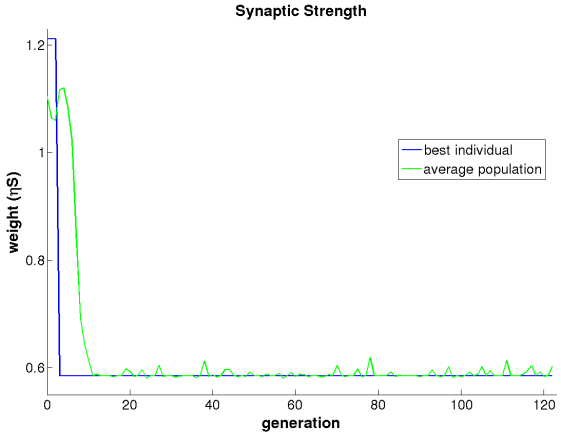


Figure C.14 – Synaptic strength.

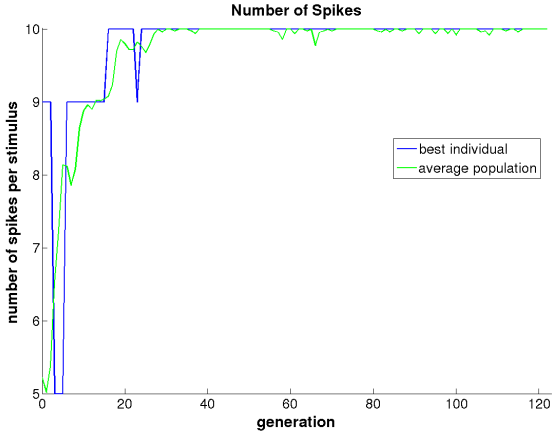


Figure C.15 – Number of spikes.

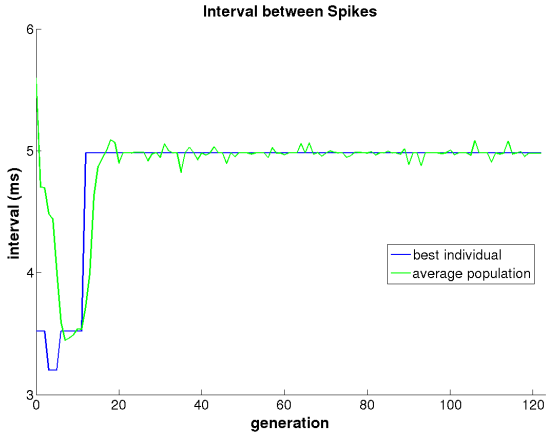


Figure C.16 – Interval between spikes.

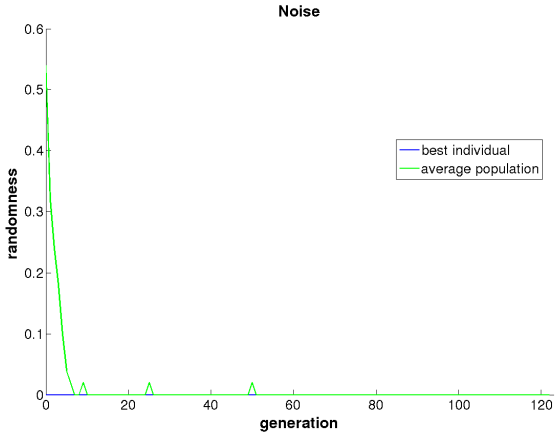


Figure C.17 – Noise.

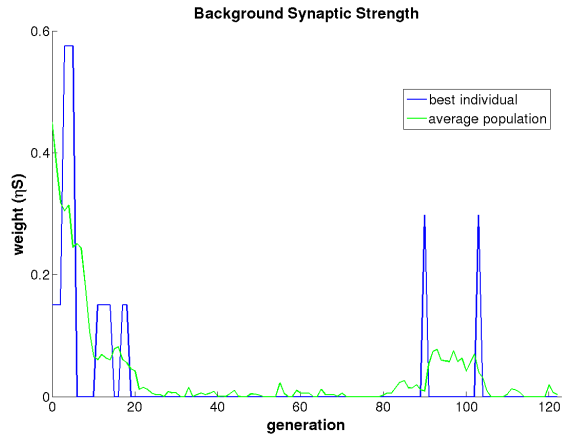


Figure C.18 – Background synaptic strength.

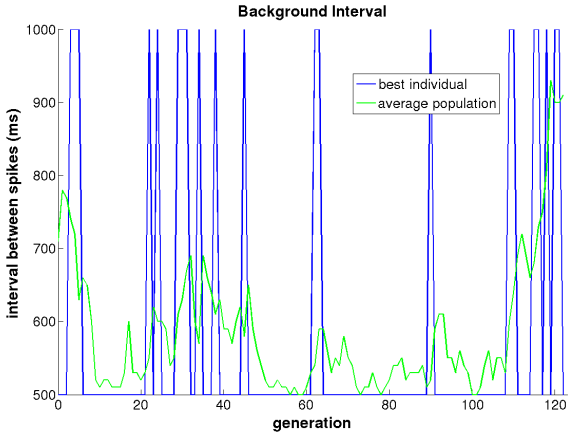


Figure C.19 – Background interval.

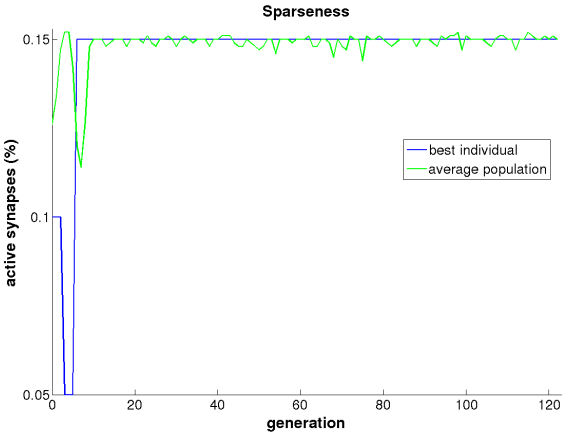


Figure C.20 – Sparseness.

C.3 Simulation 3

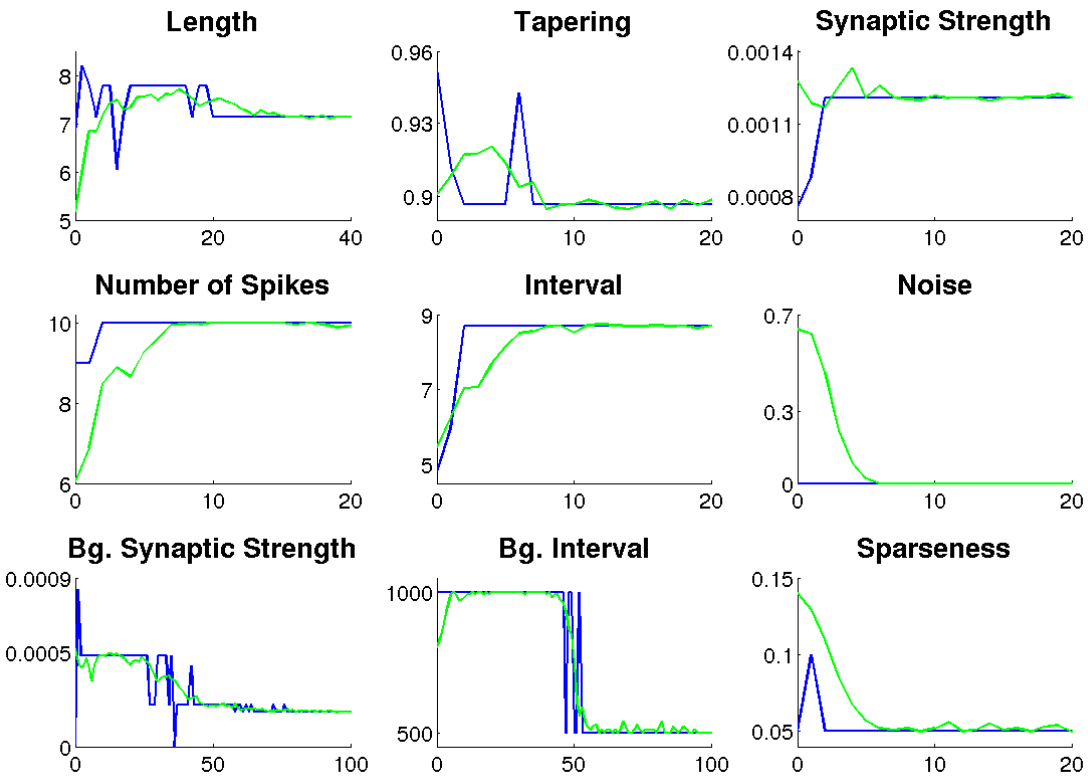


Figure C.21 – Evolution of all the nine parameter optimised by the EA.

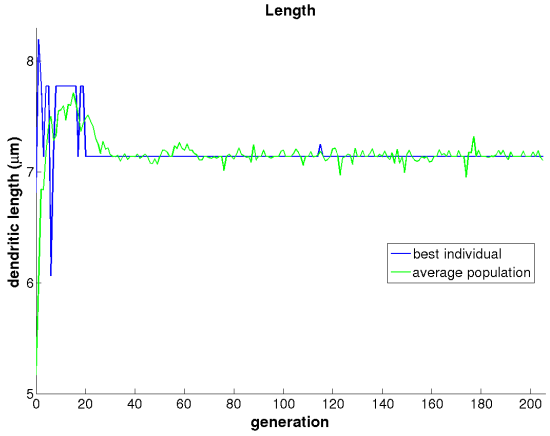


Figure C.22 – Dendritic compartmental length.

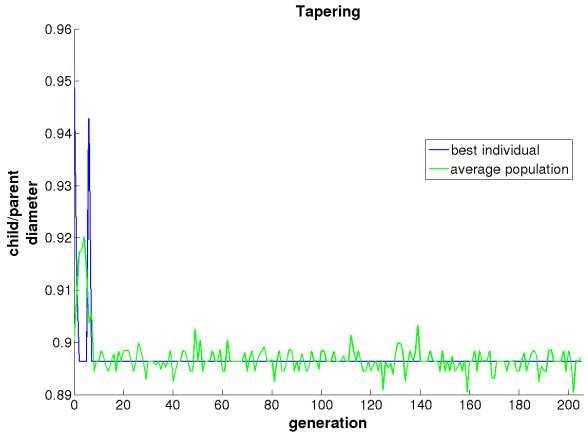


Figure C.23 – Tapering.

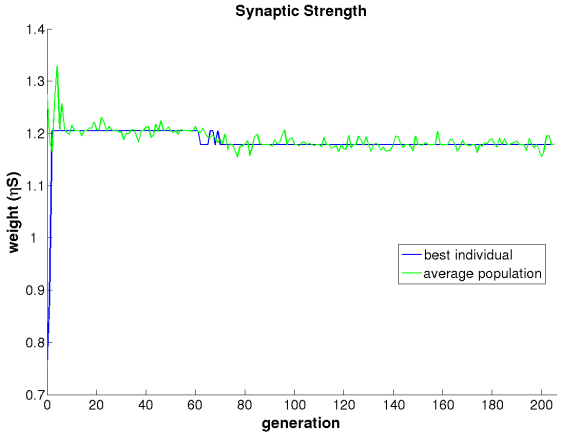


Figure C.24 – Synaptic strength.

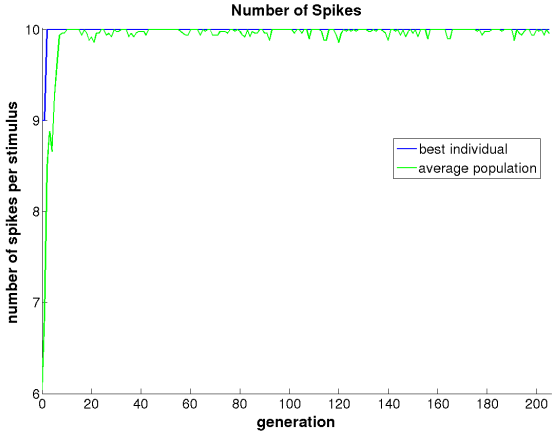


Figure C.25 – Number of spikes.

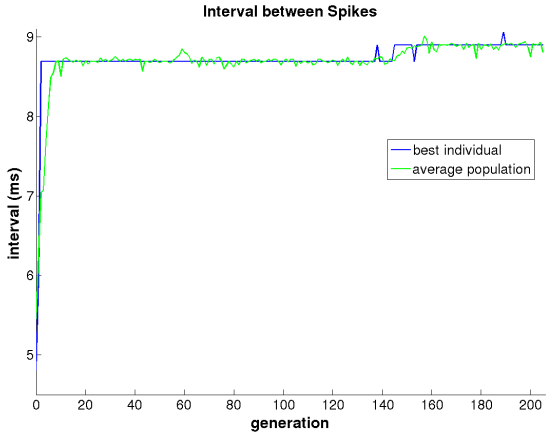


Figure C.26 – Interval between spikes.

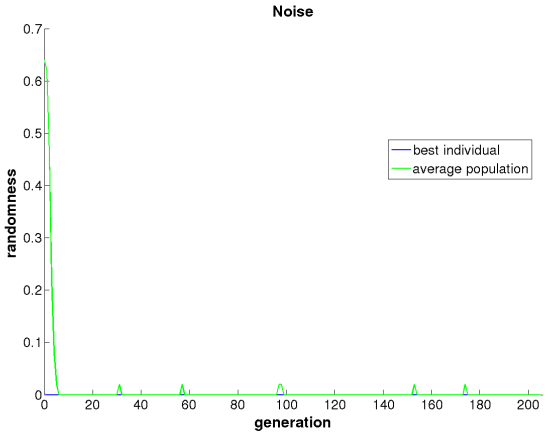


Figure C.27 – Noise.

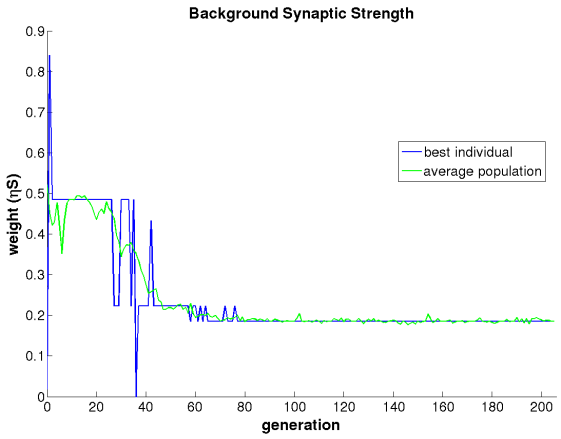


Figure C.28 – Background synaptic strength.

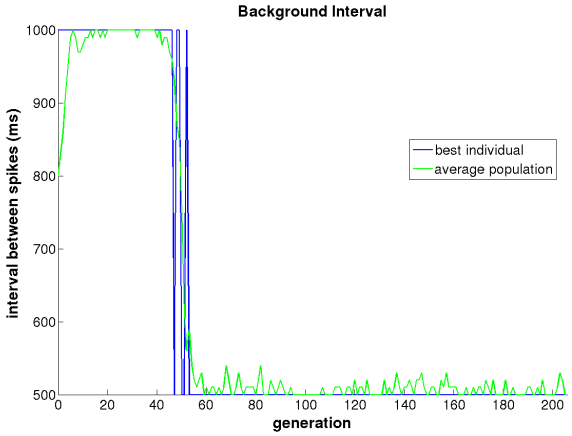


Figure C.29 – Background interval.

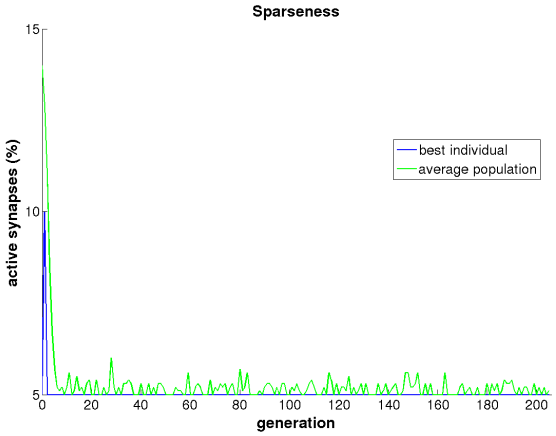


Figure C.30 – Sparseness.

Appendix D

Published Papers

Determinants of Pattern Recognition by Cerebellar Purkinje Cells

Giseli de Sousa, Rod Adams, Neil Davey, Volker Steuber
Science and Technology Research Institute, University of Hertfordshire, Hatfield Herts, AL10 9AB, UK

E-mail: g.sousa@herts.ac.uk

Many theories of cerebellar function assume that long-term depression (LTD) of parallel fiber (PF) synapses enables Purkinje cells (PCs) to learn to recognize PF activity patterns. According to the classic view, a PC can store and learn to distinguish PF activity patterns that have been presented repeatedly together with climbing fibre (CF) input to the cell. The resulting LTD of the PF synapses is often assumed to lead to a decreased rate of PC simple spike firing, a reduction in the inhibition of their target neurons in the deep cerebellar nuclei and thus an increased output from the cerebellum. We have recently shown by combining computer simulations with electrophysiological recordings in slices and in awake behaving mice that the readout of learned patterns in PCs may operate in a fundamentally different way. Our simulations and experiments predict that the best criterion to distinguish between learned and novel patterns is the duration of a pause in firing that occurs after presentation of a pattern, with shorter pauses in response to learned patterns [1].

Although our previous simulations have used a biophysically detailed PC model that has been tuned to generate realistic behaviours under in vitro and in vivo conditions, we have applied a simplified learning rule where the AMPA receptor conductance of an active PF synapse is halved every time a PF pattern is learned. Moreover, our previous simulations have not incorporated the LTD of inhibitory synapses that can be induced when the PC receives coincident CF input [2], and that could potentially counteract the effect of the depression of the excitatory PF synapses. Here, we study the effect of inhibitory synaptic plasticity on pattern recognition, and we explore a variation of our original learning rule that has been adapted to result in a better match to experimental data on LTD induction in slices [2,3].

To study the effect of plasticity at the synapses between inhibitory interneurons and PCs, we presented the model with feed-forward inhibitory input, which followed the excitatory input with a time delay of 1.4 ms [1,2]. Initially, we chose an inhibition/excitation ratio of one, in the range of experimental observations in vitro [2]. We then introduced LTD at the inhibitory synapses and evaluated the pattern recognition performance for varying numbers of learned patterns. We found that the performance was unaffected by the presence of inhibitory LTD, even in the extreme case when the inhibitory plasticity was restricted to the presentation of learned PF patterns. Our simulations predict that LTD based pattern recognition is very robust in the presence of LTD at inhibitory synapses.

By dividing the synaptic weights of active PFs by two for every pattern that was learned, our original learning rule could result in very small AMPA receptor conductances for large numbers of learned patterns. However, LTD induction in cerebellar slices hardly ever results in the depression of responses to less than 50% of the pre-induction baseline [2,3]. We studied the effect of saturating LTD in our simulations and found that the pattern recognition performance was very sensitive to the value at which the synaptic weights saturated. In contrast to a corresponding artificial neural network, which was unaffected by the value at which LTD saturated, pause based pattern recognition in the PC model deteriorated drastically in the presence of higher saturation values and therefore smaller amounts of LTD. To result in satisfactory pattern recognition, LTD had to depress the AMPA receptor conductances in the PC model down to at least 70% of their baseline values, and optimal performance resulted from setting the weights to zero and silencing the synapses completely. Interestingly, large numbers of silent PF synapses have been observed in another preparation [4]. Our simulation results suggest that it will be crucial to explore these differences to understand the connection between PF LTD and pattern recognition.

References

1. Steuber V, Mittmann W, Hoebeek FE, Silver RA, De Zeeuw CI, Hausser M, De Schutter E: **Cerebellar LTD and pattern recognition by Purkinje cells.** *Neuron* 2007, **54**:121-136.
2. Mittmann W, Hausser M: **Linking synaptic plasticity and spike output at excitatory and inhibitory synapses onto cerebellar Purkinje cells.** *J Neurosci* 2007, **27**:5559-5570.
3. Wang S-H, Denk W, Hausser M: **Coincidence detection in single dendritic spines mediated by calcium release.** *Nat Neurosci* 2000, **3**:1266-1273.
4. Isope P, Barbour B: **Properties of unitary granule cell – Purkinje cell synapses in adult rat cerebellar slices.** *J Neurosci* 2002, **22**:9668-9678.

Effect of LTD Saturation on Pattern Recognition by Cerebellar Purkinje Cells

Objectives

Many theories of cerebellar learning assume that long-term depression (LTD) of synapses between parallel fibres (PFs) and Purkinje cells (PCs) provides the basis for pattern recognition in the cerebellar cortex (1). Recent computer simulations and electrophysiological recordings in slices and awake mice have suggested that PCs can use a novel neural code based on the duration of silent periods, with shorter pauses in response to learned PF patterns (2). These simulations used a simplified learning rule, where the AMPA receptor conductance was halved each time a pattern was learned. However, experimental studies of LTD induction in cerebellar slices show that the mean AMPA receptor conductance saturates and is rarely reduced to less than 50% of its baseline value (3,4). Here we study the effect of LTD saturation at different minimal values on pattern recognition.

Methods

The simulations were performed using the GENESIS simulator. We used the multi-compartmental PC model with active dendrites and soma that is described in detail in (5,6). The pattern recognition performance was evaluated by calculating signal-to-noise ratios (2,7). Simulation data were analysed using C and MATLAB.

Results

We studied the effect of LTD saturation in our simulations and found that the pattern recognition performance was very sensitive to the value at which the synaptic weights saturated. The pause based pattern deteriorated drastically for smaller amounts of LTD and was affected strongly by the number of active PFs per pattern. In contrast, a corresponding artificial neural network (7) was unaffected by the value at which LTD saturated. To result in satisfactory pattern recognition, LTD had to depress the AMPA receptor conductances down to at least 70% of their baseline values, and optimal performance resulted from setting the weights to zero and silencing the synapses completely.

Conclusions

Our simulation results suggest that LTD based pattern recognition is improved by strong depression and optimized by setting the AMPA receptor conductances to zero. Interestingly, large numbers of silent synapses between PFs and PCs have been observed in cerebellar slices (8).

References

1- J. Physiol. 202:437, 1969
2 - Neuron 54:121, 2007
3- J Neurosci. 27:5559, 2007
4 - Nat Neurosci. 3:1266, 2000
5 - J. Neurophysiol. 71:375, 1994
6 - J. Neurophysiol. 71:401, 1994
7 - Neurocomputing 38:383, 2001
8 - J Neurosci. 22:9668, 2002

The Effect of Different Forms of Synaptic Plasticity on Pattern Recognition in the Cerebellar Cortex

Giseli de Sousa, Rod Adams, Neil Davey, Reinoud Maex, and Volker Steuber

Science and Technology Research Institute
University of Hertfordshire
Hatfield, Herts, UK
{G.Sousa, R.G.Adams, N.Davey, R.Maex1, V.Steuber}@herts.ac.uk

Abstract. Many cerebellar learning theories assume that long-term depression (LTD) of synapses between parallel fibres (PFs) and Purkinje cells (PCs) provides the basis for pattern recognition in the cerebellum. Previous work has suggested that PCs can use a novel neural code based on the duration of silent periods. These simulations have used a simplified learning rule, where the synaptic conductance was halved each time a pattern was learned. However, experimental studies in cerebellar slices show that the synaptic conductance saturates and is rarely reduced to less than 50% of its baseline value. Moreover, the previous simulations did not include plasticity of the synapses between inhibitory interneurons and PCs. Here we study the effect of LTD saturation and inhibitory synaptic plasticity on pattern recognition in a complex PC model. We find that the PC model is very sensitive to the value at which LTD saturates, but is unaffected by inhibitory synaptic plasticity.

Keywords: Associative memory, Long-term depression, Purkinje cell, Cerebellum

1 Introduction

The cerebellum is a part of the brain involved in a multitude of tasks, including motor control, and its functioning is responsible for the smoothness and precision of movements. These skills are improved by a process called motor learning, which is often assumed to be implemented by a form of synaptic plasticity known as long-term depression (LTD). LTD is a long-lasting decrease in synaptic strength due to a loss of AMPA receptors in the postsynaptic membrane[1]. In the cerebellum, LTD has been shown to occur at the synapses between Purkinje cells (PCs) and their excitatory inputs: climbing fibres (CFs) and parallel fibres (PFs). More specifically, cerebellar LTD is an associative process in which the strength of a PF synapse onto a PC is depressed when the CF and PF are activated at the same time.

Classical cerebellar learning theories suggest that a PC can learn to discriminate between different activity patterns presented by its thousands of afferent

PFs, due to LTD of the PF synapses [2]. It is assumed that as a result of LTD, the PC firing rate will be reduced when a learned pattern is presented again, and the PC will exert less inhibition on the deep cerebellar nuclei (Fig. 1). As a consequence, the cerebellar output should be increased, which could implement motor learning[1, 3]

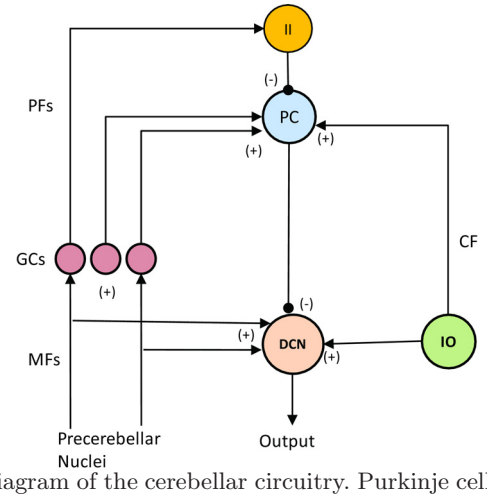


Fig. 1. Schematic diagram of the cerebellar circuitry. Purkinje cells (PCs) receive excitatory inputs (+) from 150,000 parallel fibres (PFs) and a single climbing fibre (CF), and inhibitory inputs (-) from inhibitory interneurons (II), and in turn inhibit the deep cerebellar nuclei (DCN). Also shown are: mossy fibres (MFs), granule cells (GCs) and the inferior olive (IO).

Recent work on cerebellar pattern recognition has demonstrated that this view is too simple. A combined theoretical and experimental study suggested that PCs can use a novel neural code based on the duration of their silent periods, where shorter pauses are produced in response to learned patterns [4] (Fig. 2A). This form of neural coding diverges from the classical view that uses the number or timing of individual spikes to distinguish between novel and learned patterns. In the computer simulations and experiments, the pause was compared with other spike response features like the number of spikes in a fixed time window after pattern presentation and the latency of the first spike in the response, and it was shown that the length of the pause was the best criterion for cerebellar PCs to identify learned patterns (Fig. 2B).

The previous simulations (see Methods) applied a simplified learning rule, where the AMPA receptor conductance was decreased by 50% each time a pattern was learned. After having stored a number of PF patterns, this could result in very small AMPA receptor conductances. However, experiments with LTD induction in cerebellar slices hardly ever result in mean AMPA receptor conductances of less than 50% of the pre-induction baseline [5, 6]. We have therefore investigated a different learning rule with AMPA receptor conductances that

saturate at varying values and have studied the effect of this learning rule in pattern recognition simulations.

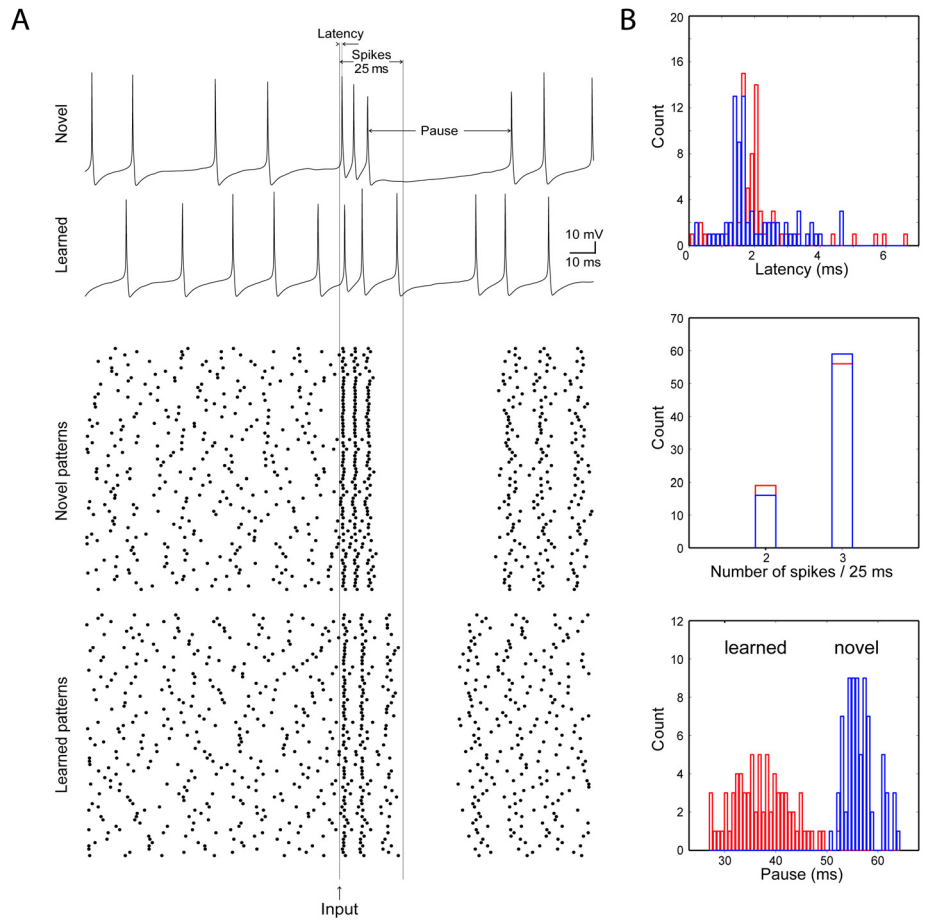


Fig. 2. Responses of a model Purkinje cell to novel and learned patterns of PF input. (A) Upper: The pause evoked by a novel pattern is longer than that for a learned pattern. Lower: Raster plot showing the responses to 75 learned and 75 novel patterns. (B) Response distribution for three different spike features. Upper: Latency of first spike after pattern presentation. Middle: Number of spikes in the first 25ms. Lower: Length of pause (modified with permission from [4]).

Another contribution of the present work is to study the effect of LTD at the inhibitory synapses made by interneurons onto PCs (Fig. 1). It has recently been described that this inhibitory synaptic plasticity results in a mean depression of inhibitory inputs down to 75% of their original values [5]. We have run computer simulations to investigate the effect of different amounts of inhibitory synaptic plasticity on pattern recognition.

2 Methods

2.1 Purkinje Cell Model

The simulations were performed using the GENESIS neural simulator [7], with additional routines implemented in C++ and MATLAB. We simulated a multicompartmental PC model with active dendrites and soma, as described in detail in references [8, 9]. The model morphology was based on a reconstruction of a guinea-pig Purkinje cell [10]. Ten different types of voltage-dependent channels were modelled using Hodgkin-Huxley-like equations. The soma compartment had a fast and persistent Na^+ conductance, a delayed rectifier, a transient A-type K^+ conductance, a non-inactivating M-type K^+ conductance, an anomalous rectifier and a low-threshold T-type Ca^{2+} conductance. The dendritic compartments contained a Purkinje-cell specific high-threshold P-type and a low-threshold T-type Ca^{2+} conductance, two different types of Ca^{2+}-activated K^+ (KCa) conductances and an M-type K^+ conductance. Each cell was originally modelled with 147,400 dendritic spines, which were activated randomly by a sequence of PF inputs at an average frequency of 0.28 Hz. The background excitation was balanced by tonic inhibition, which made the model fire simple spikes at an average frequency of 48 Hz. Due to the large number of dendritic spines, which made the simulations computationally expensive, a simplified version of the model was constructed by decreasing the number of spines to 1% of the original number. To compensate for this reduction, the rate of PF excitation was increased to an average frequency of 28 Hz. As this simplified model gave identical results as the full model, it was used in the simulations presented here.

To study the effect of plasticity at inhibitory synapses, the model was provided with feed-forward inhibitory input by activating a variable number of inhibitory synapses onto the soma and main dendrite. The inhibitory input followed the synchronous activation of excitatory PFs synapses with a delay of 1.4 ms. Inhibition/excitation ratios were measured as ratios of the mean inhibitory postsynaptic current (IPSC) peak to the mean excitatory postsynaptic current (EPSC) peak when the model was voltage clamped to -40 mV.

2.2 Pattern Recognition

The pattern recognition simulations were performed in two steps. First, a number of random binary input patterns were generated, initially 200, and half of these patterns were learned by a corresponding artificial neural network (ANN). The ANN used was a modified version of an associative net with feed-forward connections between its inputs and output [11] and was trained by applying a modified version of the LTD learning rule [12](see below). The simulations of the ANN consisted of two phases: learning and recall.

In the learning phase, the weights of all synapses that received a positive input during the presentation of a pattern were set to a constant value. This LTD saturation value was kept constant and unaffected by further pattern presentations, different from the learning rule that had been used in the previous

simulations [4]. During the recall phase, the response of the ANN was given by the sum of the weights of all synapses that were associated with active inputs, which resulted in responses of the ANN to stored patterns that were lower than those to novel patterns (Fig. 3).

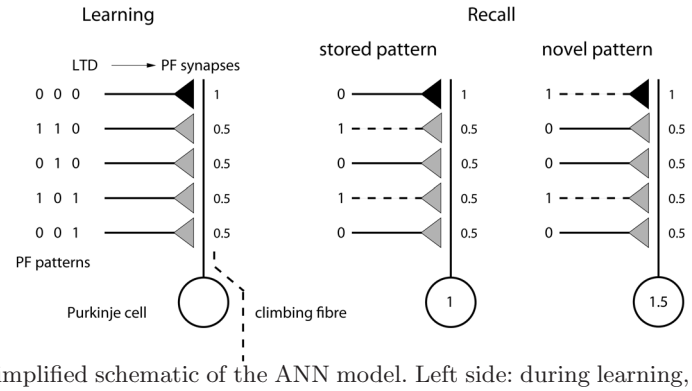


Fig. 3. Simplified schematic of the ANN model. Left side: during learning, three example PF patterns are stored by changing the synaptic weights that are associated with active input lines from their initial value of 1 to an LTD saturation value of 0.5 (this value is varied between different simulations). Right side: during recall, the responses to a stored and a novel pattern are calculated as dot product of input vector and weight vector, resulting in values of 1 and 1.5, respectively (note the difference to the original diagram in [12])

In the second phase of the pattern recognition simulations, the vector of synaptic weights was transferred from the ANN onto AMPA receptor conductances in the multi-compartmental PC model. This represents learning the PF patterns by depressing the corresponding AMPA receptor conductances during LTD induction. To test the recall of learned patterns, the PC model was then presented with a corresponding pattern of synchronous AMPA receptor activation at the PF synapses.

The discrimination between novel and learned pattern in the two models was evaluated by calculating a signal-to-noise ratio [13, 14]:

$$s/n = \frac{(\mu_s - \mu_n)^2}{0.5(\sigma_s^2 + \sigma_n^2)} \tag{1}$$

where μ_s and μ_n represent the mean values and σ_s^2 and σ_n^2 represent the variances of the responses to stored and novel patterns, respectively. In the PC model, three different features of the spike response were tested as criteria to distinguish stored from novel patterns: the latency of the first spike fired after pattern presentation, the number of spikes in a 25ms time window after pattern presentation, and the duration of a silent period that followed the pattern presentation (see response distributions for these three different metrics in Fig. 2B). In all cases studied, the pause duration was the best criterion, and only pause based signal-to-noise ratios are presented here.

3 Results

3.1 LTD Saturation and the Number of Active PF Inputs

We initially investigated the effect of varying two parameters that were expected to affect the pattern recognition performance: the value at which LTD saturated and the number of active PFs for each pattern.

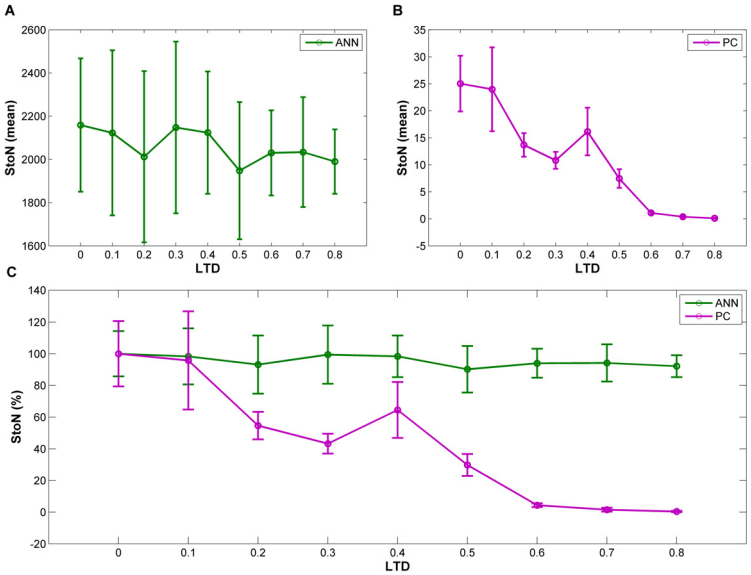


Fig. 4. Pattern recognition performance of the two models for a range of LTD values. The performance was evaluated by calculating s/n ratios for the ANN (A) and the PC model (B). The relative decreases in s/n ratio are compared in (C), showing that the PC model is more sensitive to LTD saturation than the ANN. Error bars indicate standard deviation (SD).

To study the effect of LTD saturation, we varied the LTD saturation value over a range from zero to 0.8, while keeping the same numbers of active PFs (1000) and PF patterns (100 novel and 100 stored) as in previous work [4]. We found that the ANN was insensitive to the amount of LTD induced (Fig. 4A). In contrast, the pattern recognition capacity based on the duration of silent periods in the PC model improved when the LTD saturation value decreased, with an optimal performance when the synaptic weights of active PFs were set to zero (Fig. 4B). The relative sensitivities of the ANN and the PC model to the amount of LTD induced are compared in Fig. 4C. While the ANN was unaffected by varying the amount of LTD, increasing the LTD saturation value to 0.8 in the PC model reduced the signal-to-noise ratio down to $0.4 \pm 0.4\%$ ($n = 10$) of the optimal value obtained by switching off the synapses completely. For LTD saturation values below 0.5, the PC model performed as well as or better than the previous model with a non-saturating learning rule [4].

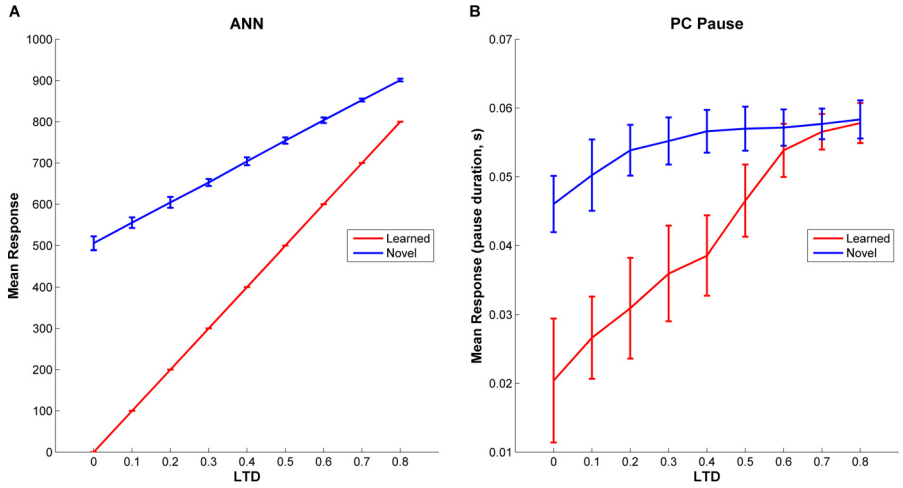


Fig. 5. Relationship between the LTD saturation value and the mean responses to stored and novel patterns in the ANN and the PC model. Although the difference between the mean responses to stored and novel patterns decreases with increasing LTD saturation values in both cases, in the ANN the variance of responses to novel patterns also decreases. This results in s/n ratios in the ANN that are independent of the LTD saturation value. Same simulation parameters as in Fig. 4. Error bars indicate SD.

The reason for the difference in sensitivity of the ANN and the PC model to varying amounts of LTD became apparent when the mean responses of the two models to stored and novel patterns were plotted against the LTD saturation value (Fig. 5). In the PC model, increasing LTD saturation values reduced the difference in pause duration between stored and novel patterns, with standard deviations that were affected to a much lesser extent (Fig. 5B). This led to the drastic reduction in s/n ratio for weak LTD shown in Figure 4. In the ANN, the difference between the mean responses to stored and novel patterns was affected much less by the LTD saturation value, while the standard deviation of responses to novel patterns decreased with increasing LTD saturation values (Fig. 5A). Based on Equation (1), the constant signal-to-noise ratio of the ANN in the presence of varying amounts of LTD can be explained by a linear relationship between the squared difference of the mean responses to stored and novel patterns $(\mu_s - \mu_n)^2$ and the variance of the responses to novel patterns σ_n^2.

In a second set of simulations, we measured the effect of varying the number of active PFs in each pattern for a range of LTD values. As expected, the performance of the ANN deteriorated for larger numbers of activated PFs, while being independent of the amount of LTD induced over the whole range of numbers of active PFs tested (500-5000, Fig. 6A). In contrast, the PC model showed the best pattern recognition capacity for a range between 1000 and 2000 active PFs and performed consistently worse for higher LTD saturation values (Fig. 6B).

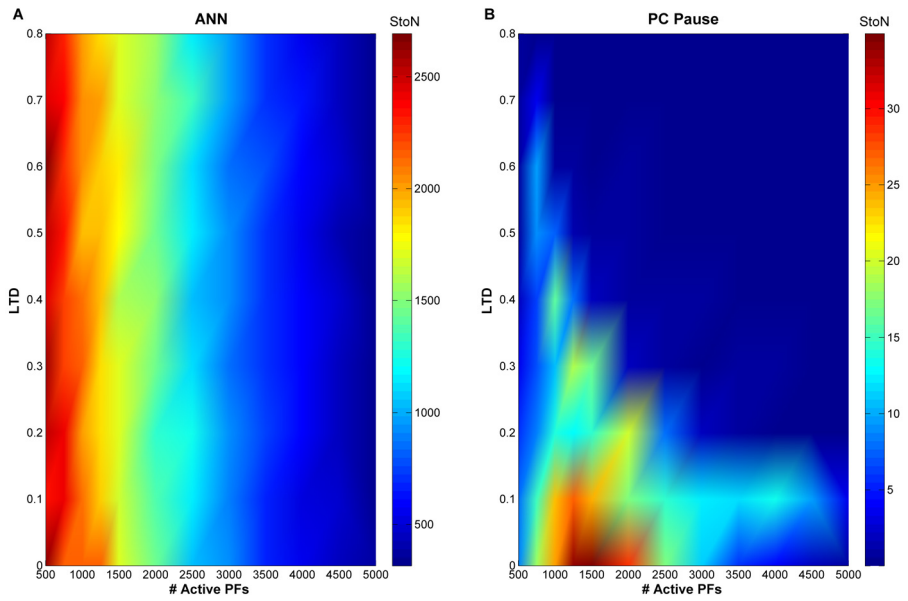


Fig. 6. Pattern recognition performance of the ANN (A) and PC model (B). The colour represents the resulting s/n ratio for each combination of a number of active PFs for each pattern (indicated on the x-axis) and an LTD saturation value (y-axis).

3.2 Inhibitory Synaptic Plasticity

To investigate the effect of plasticity at the synapses between inhibitory interneurons and PCs, we initially used an inhibition/excitation ratio of one (see Methods), which is in the range of experimentally observed data from cerebellar slices [5]. We then introduced LTD at the inhibitory synapses and evaluated the pattern recognition performance of the PC for different numbers of patterns. The effect of inhibitory LTD was examined by depressing the inhibitory conductance to values between 25% and 75% of their pre-depression baseline. We used four different simulation setups (Fig. 7): no inhibition, plasticity at inhibitory synapse for stored and novel PF patterns, plasticity for stored patterns only and no plasticity for both patterns, that is, maintaining the baseline amplitude value for the original inhibition/excitation ratio [5].

We found that the pattern recognition performance of the PC model was unaffected by the presence of inhibitory LTD, even in the extreme case where the inhibitory plasticity was restricted to learned PFs patterns.

4 Conclusion

Previous computer simulations and experiments in cerebellar slices and awake behaving rats suggested that the cerebellum can use a novel neural code that is based on the duration of silent periods in neuronal activity [4]. These simulations

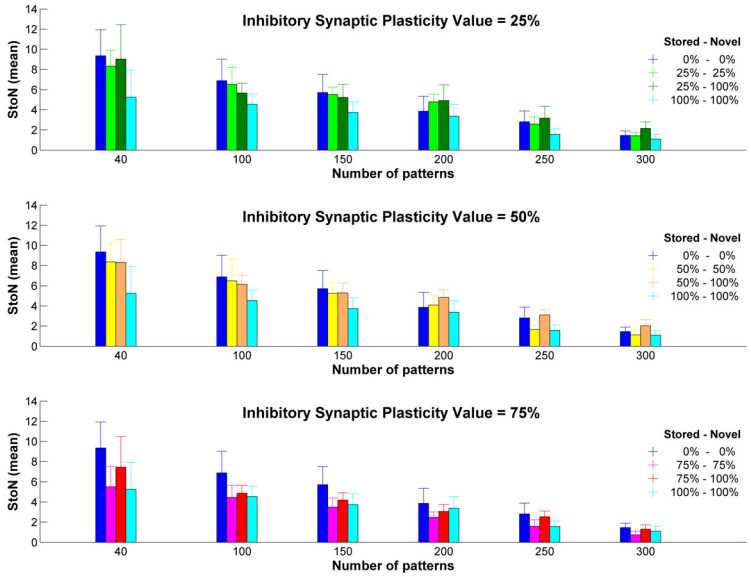


Fig. 7. Depression at inhibitory synapses. Three different inhibitory synaptic plasticity rules were applied for varying numbers of patterns. The first bar of each graph shows the s/n ratio when no inhibition is applied for both stored and novel patterns, resulting in the best pattern recognition performance. The other bars represent cases with inhibition present, with from left to right: plasticity for both stored and novel patterns, plasticity for stored patterns only and no plasticity for either type of patterns, using the original inhibitory conductances. Error bars indicate SD.

used a complex multi-compartmental model of a cerebellar Purkinje cell that had been tuned to replicate a wide range of behaviours in vitro and in vivo [8, 9], but they applied a simplified LTD learning rule, which involved dividing the synaptic weights of active PF inputs by two every time a PF pattern was learned. This could result in very small synaptic weights and does not fit experimental data on LTD induction in cerebellar slices, where the mean AMPA receptor conductances saturate and are hardly ever depressed to less than 50% of their pre-depression baseline values [5, 6].Moreover, the previous simulations did not include the plasticity at synapses between inhibitory interneurons and PCs that has recently been characterised [5].

We have studied the effect of inhibitory synaptic plasticity and saturating LTD in the complex PC model. We found that the ability of the PC model to discriminate between learned and novel PF input patterns was unaffected by the presence of inhibitory plasticity for a wide range of parameter values.

However, the pattern recognition performance of the PC model was very sensitive to the value at which LTD saturated. In contrast to a corresponding ANN, which was unaffected by the amount of LTD induced, the performance of the PC model was improved by lower LTD saturation values. The best performance resulted from LTD saturation values of zero, which corresponds to silencing the

PF synapses completely. Interestingly, large numbers of silent PF synapses have been observed by monitoring microscopically identified PF-PC connections in cerebellar slices [15]. Our simulation results indicate that the discrepancy between the existence of these silent synapses and the apparent saturation of LTD in induction experiments needs to be resolved to understand the connection between LTD and cerebellar learning.

Acknowledgment

This work was supported by a BBSRC-ANR Systems Biology Fellowship to V.S..

References

1. Ito, M.: Cerebellar long-term depression: Characterization, signal transduction, and functional roles. Physiol. Rev. **81**(3) (2001) 1143–1195
2. Marr, D.: A theory of cerebellar cortex. Journal of Physiology (London) **202** (1969) 437–470
3. Ito, M.: The cerebellum and neural control. Raven Press, New York (1984)
4. Steuber, V., Mittmann, W., Hoebeek, F.E., Silver, R.A., De Zeeuw, C.I., Häusser, M., De Schutter, E.: Cerebellar LTD and pattern recognition by purkinje cells. Neuron **54**(1) (2007) 121–136
5. Mittmann, W., Häusser, M.: Linking synaptic plasticity and spike output at excitatory and inhibitory synapses onto cerebellar purkinje cells. J. Neurosci. **27**(21) (2007) 5559–5570
6. Wang, S.S.H., Denk, W., Häusser, M.: Coincidence detection in single dendritic spines mediated by calcium release. Nat Neurosci **3**(12) (2000) 1266–1273
7. Bower, J., Beeman, D.: The book of GENESIS: Exploring realistic neural models with the general neural simulation system (2003)
8. De Schutter, E., Bower, J.: An active membrane model of the cerebellar purkinje cell. I. simulation of current clamps in slice. Journal of Neurophysiology **71**(1) (1994) 375–400
9. De Schutter, E., Bower, J.: An active membrane model of the cerebellar purkinje cell: II. simulation of synaptic responses. Journal of Neurophysiology **71**(1) (1994) 401–419
10. Rapp, M., Segev, I., Yarom, Y.: Physiology, morphology and detailed passive models of guinea-pig cerebellar purkinje cells. Journal of Physiology (London) **474**(1) (1994) 101–118
11. Willshaw, D., Buneman, O., Longuet-Higgins, H.: Non-holographic associative memory. Nature **222** (1969) 960–962
12. Steuber, V., De Schutter, E.: Long-term depression and recognition of parallel fibre patterns in a multi-compartmental model of a cerebellar purkinje cell. Neurocomputing **38** (2001) 383–388
13. Dayan, P., Willshaw, D.J.: Optimising synaptic learning rules in linear associative memories. Biol Cybern **65**(4) (1991) 253–265
14. Graham, B.P.: Pattern recognition in a compartmental model of a ca1 pyramidal neuron. Network: Computation in Neural Systems **12**(4) (2001) 473 – 492
15. Isope, P., Barbour, B.: Properties of unitary granule cell-purkinje cell synapses in adult rat cerebellar slices. Journal of Neuroscience **22**(22) (2002) 9668–9678

Optimization of Neuronal Morphologies for Pattern Recognition

Giseli de Sousa, Reinoud Maex, Rod Adams, Neil Davey, Volker Steuber
Science and Technology Research Institute, University of Hertfordshire, Hatfield, Herts, AL10 9AB, UK
E-mail: g.sousa@herts.ac.uk

Previous studies have shown that the morphology of a neuron can affect its firing pattern [1, 2]. Specifically, some neuronal morphologies tend to favour bursting, where short sequences of spikes are interspersed with pauses in firing [1, 2]. This type of bursting behaviour has been observed in cerebellar Purkinje cells (PCs), and previous work on associative memory in PCs has shown that the generation of burst-pause sequences can be important for information storage in the cerebellum [3]. These results have implications for the coding of information in the brain, but they are specific to one particular neuron with a highly specialised morphology. In this study we therefore use a general approach to optimise generic neuronal structures for pattern recognition, while analysing how their morphology influences their firing pattern.

To study how the ability of a neuron to perform pattern recognition depends on morphology, we have built a genomic representation of neuronal models, focusing as a first objective on optimising dendritic architectures. The optimization process uses an evolutionary algorithm and involves four steps. Firstly, genotypes are generated, which specify binary tree structures [4]. Secondly, the genotype is expressed as a model neuron phenotype, in which the branching pattern is derived from the genotype, and which is then converted to a multi-compartmental model written in NEURON simulation code. Thirdly, the fitness values are assessed by evaluating the pattern recognition performance. Finally, genetic variation is introduced, using a process where the genes are modified by crossover and mutation operators. Unlike previous work that focussed on generating a subset of realistic neuronal morphologies for specific computational tasks [5], our representation ensures that the algorithm can generate the set of all possible morphologies for a specific number of terminal branches. The fitness function evaluates pattern recognition performance as described previously [3, 6], by storing a number of input patterns based on changing synaptic weights and quantifying the ability of the model to distinguish the set of stored patterns from a set of novel patterns. The discrimination between stored and novel patterns is evaluated for different features of the spike response and quantified by calculating a signal-to-noise ratio. The evolved artificial neuronal morphologies are compared with reconstructed morphologies from real neurons. An extension of the work involves optimising other neuronal features such as types and distributions of ion channels and the spatial structure of inputs in patterns.

References

1. Mainen ZF, Sejnowski TJ: **Influence of dendritic structure on firing pattern in model neocortical neurons**. *Nature* 1996, **382**:363-366.
2. van Ooyen A, Duijnhouwer J, Remme M, van Pelt J: **The effect of dendritic topology on firing patterns in model neurons**. *Network: Computation in Neural Systems* 2002, **13**:311-325.
3. Steuber V, Mittmann W, Hoebeek FE, Silver RA, De Zeeuw CI, Häusser M, De Schutter E: **Cerebellar LTD and Pattern Recognition by Purkinje Cells**. *Neuron* 2007, **54**(1):121-136.
4. Van Pelt J, Uylings HBM, Verwer RWH, Pentney RJ, Woldenberg MJ: **Tree asymmetry--A sensitive and practical measure for binary topological trees**. *Bulletin of Mathematical Biology* 1992, **54**(5):759-784.
5. Stiefel KM, Sejnowski TJ: **Mapping Function Onto Neuronal Morphology**. *J Neurophysiol* 2007, **98**(1):513-526.
6. de Sousa G, Adams R, Davey N, Maex R, Steuber V: **The Effect of Different Forms of Synaptic Plasticity on Pattern Recognition in the Cerebellar Cortex**. In *Adaptive and Natural Computing Algorithms*. 2009:413-422.

The effect of dendritic morphology on pattern recognition in the presence of active conductances

Giseli de Sousa, Reinoud Maex, Rod Adams, Neil Davey, Volker Steuber
Science and Technology Research Institute, University of Hertfordshire, Hatfield, Herts, AL10 9AB, UK
E-mail: g.sousa@herts.ac.uk

In previous experiments [1], we showed that dendritic morphology affects the ability of passive neurons to recognise spatial patterns of synaptic inputs. In particular, the most symmetric morphologies outperformed the most asymmetric ones based on the measure of signal-to-noise ratio between stored and novel patterns. In the present study, we analyse how the dendritic morphology affects the pattern recognition performance in active models.

To evaluate pattern recognition performance, we ran a set of simulations using a large sample of neuronal morphologies each consisting of 128 terminal points and the same set of ion channel conductances, defined in previous models [2]. The model response was evaluated by calculating the signal-to-noise ratio over the number of spikes after presenting a pattern, differently from the experiments with passive models where the EPSP size was used [1, 3]. For all experiments, we investigated whether the pattern recognition performance correlated with different morphometric parameters, including the asymmetry index [4], and the average and variance of path length. We also investigated suitable ranges for model parameters such as dendritic compartment length and synaptic strength, among other properties related to the pattern presentation. The results achieved in active models were then compared with the ones from passive models, using the same set of parameters.

The experiments confirmed that there are different pattern recognition abilities associated with a range of different morphologies from the most symmetric to the most asymmetric ones. The initial results suggested a strong anti-correlation between pattern recognition performance and neuronal asymmetry in the presence of active conductances (see Figure 1). The same correlation was also observed in passive models, however with a less accentuated performance difference when compared with active ones. The results also show that average path length is the best morphometric parameter tested to predict pattern recognition, where a more linear correlation was found when compared with other metrics (Figure 1B).

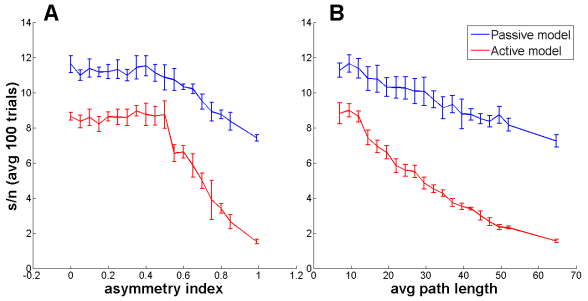


Figure 1. Pattern recognition performance in active and passive models. Results were obtained by generating a population of 150K random trees, and then averaging over 5 randomly selected ones in each bin, using two metrics: asymmetry index (**A**) and average path length (**B**).

Currently we are investigating the optimization of active neuronal morphologies for pattern recognition, using an evolutionary algorithm, previously presented in [1]. In addition to the dendritic topology, different parameters were added to the genomic representation, such as dendritic compartment length and tapering, using the parameter ranges found in the previous experiments on these models. With these results, we want to confirm the more pronounced effect observed in active models (Figure 1), as they suggest the evolutionary algorithm may also be more successful in finding optimal morphologies for active as compared to passive dendrites.

References

1. de Sousa G, Maex R, Adams R, Davey N, Steuber V: **Optimization of neuronal morphologies for pattern recognition**. *BMC Neuroscience* 2010, **11**(Suppl 1):P80.
2. van Ooyen A, Duijnhouwer J, Remme M, van Pelt J: **The effect of dendritic topology on firing patterns in model neurons**. *Network: Computation in Neural Systems* 2002, **13**:311-325.
3. Graham BP: **Pattern recognition in a compartmental model of a CA1 pyramidal neuron**. *Network: Computation in Neural Systems* 2001, **12**(4):473 - 492.
4. van Pelt J, Uylings HBM, Verwer RWH, Pentney RJ, Woldenberg MJ: **Tree asymmetry--A sensitive and practical measure for binary topological trees**. *Bulletin of Mathematical Biology* 1992, **54**(5):759-784.