# The Evolution, Analysis, and Design of Minimal Spiking Neural Networks for Temporal Pattern Recognition

Submitted to the University of Hertfordshire in Partial
Fulfilment of the Requirement of the Degree of

## Doctor of Philosophy

Muhammad Yaqoob

Supervisors:

Professor Volker Steuber

Professor Borys Wróbel

August 2022

UH Biocomputation Group
Centre for Computer Science and Informatics Research
School of Physics, Engineering and Computer Science
University of Herfordshire
Hatfield, UK

# Acknowledgements

First and foremost, I would like to express my profound gratitude to my supervisors Prof. Borys Wróbel and Prof. Volker Steuber for their consistent support, guidance, and encouragement throughout my studies and research. Their expertise and mentorship have been instrumental in shaping my academic and professional growth. I am grateful for the time they invested in me and I will always value their guidance and support. I extend my sincere gratitude to Prof. Rene te Boekhorst and Dr Reinoud Maex for taking the time to review my work and providing their valuable feedback.

I am obliged to the University of Hertfordshire for offering the means and financial support over the past three years. I am thankful to the staff members of the Doctoral College for creating a supportive and welcoming environment for graduate students. I also acknowledge the support of the KNOW RNA Research Center in Poznan (No. 01/KNOW2/2014). The initial experiments on the evolution of minimal spiking neural networks were conducted at the Evolving Systems Laboratory, led by Prof. Wróbel at Adam Mickiewicz University in Poznan, Poland.

I would like to thank all members of the Bio-computation Lab (Agi, Ankur, David, Deepak, Emil, Eleonora, Harpreet, Julia, Mahsa, Maria, Ming, Mohammad, Nathen, Nik, Rebecca, Reinoud, Retish, Ronak, Sam, and Shavika) for creating a pleasant and conducive learning environment.

Above all, I express my sincere gratitude to my family, especially my parents, for their guidance and unconditional support throughout my life. A special thanks to my siblings for their immense love, care and support in everything I do. Also, I truly appreciate the friendship of Aamir Khan and Tahir Mehmood, who I know I can always count on.

Finally, thanks to all those I have missed out on – you are not forgotten!

# Abstract

All sensory stimuli are temporal in structure. How a pattern of action potentials encodes the information received from the sensory stimuli is an important research question in neurosciencce. Although it is clear that information is carried by the number or the timing of spikes, the information processing in the nervous system is poorly understood. The desire to understand information processing in the animal brain led to the development of spiking neural networks (SNNs). Understanding information processing in spiking neural networks may give us an insight into the information processing in the animal brain. One way to understand the mechanisms which enable SNNs to perform a computational task is to associate the structural connectivity of the network with the corresponding functional behaviour. This work demonstrates the structure-function mapping of spiking networks evolved (or hand-crafted) for recognising temporal patterns. The SNNs are composed of simple yet biologically meaningful adaptive exponential integrate-and-fire (AdEx) neurons. The computational task can be described as identifying a subsequence of three signals (say ABC) in a random input stream of signals (”ABBBCCBABABCBBCAC”). The topology and connection weights of the networks are optimised using a genetic algorithm such that the network output spikes only for the correct input pattern and remains silent for all others. The fitness function rewards the network output for spiking after receiving the correct pattern and penalises spikes elsewhere.

To analyse the effect of noise, two types of noise are introduced during evolution: (i) random fluctuations of the membrane potential of neurons in the network at every network step, (ii) random variations of the duration of the silent interval between input signals. It has been observed that evolution in the presence of noise produced networks that were robust to perturbation of neuronal parameters. Moreover, the networks also developed a form of memory, enabling them to maintain network states in the absence of input activity. It has been demonstrated that the network states of an evolved network have a one-to-one correspondence with the states of a finite-state transducer (FST) – a model of computation for time-structured data.

The analysis of networks indicated that the task of recognition is accomplished by transitions between network states.

Evolution may overproduce synaptic connections, pruning these superfluous connections pronounced structural similarities among individuals obtained from different independent runs. Moreover, the analysis of the pruned networks highlighted that memory is a property of self-excitation in the network. Neurons with self-excitatory loops (also called autapses) could sustain spiking activity indefinitely in the absence of input activity. To recognise a pattern of length n, a network requires n+1 network states, where n states are maintained actively with autapses and the penultimate state is maintained passively by no activity in the network. Simultaneously, the role of other connections in the network is identified.

Of particular interest, three interneurons in the network are found to have a specialized role: (i) the lock neuron is always active, preventing the output from spiking unless it is released by the penultimate signal in the correct pattern, exposing the output neuron to spike for the correct last signal, (ii) the switch neuron is responsible for switching the network between the inter-signal states and the start state, and (iii) the accept neuron produces spikes in the output neuron when the network receives the last correct input. It also sends a signal to the switch neuron, transforming the network back into the start state

Understanding how information is processed in the evolved networks led to handcrafting network topologies for recognising more extended patterns. The proposed rules can extend network topologies to recognize temporal patterns up to length six. To validate the handcrafted topology, a genetic algorithm is used to optimise its connection weights. It has been observed that the maximum number of active neurons representing a state in the network increases with the pattern length. Therefore, the suggested rules can handcraft network topologies only up to length 6. Handcrafting network topologies, representing a network state with a fixed number of active neurons requires further investigation.

اسی کشمکش میں گزریں مری زندگی کی راتیں
کبھی سوز و سازِ رومی کبھی پیچ و تابِ رازی

*Allama Iqbal, Bal-e-Jibril-015*

# Contents

I Introduction 1

1 Introduction 2

1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Aim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1.3 Contributions to Knowledge . . . . . . . . . . . . . . . . . . 6

1.4 Structure of the Thesis . . . . . . . . . . . . . . . . . . . . 8

1.5 Publications . . . . . . . . . . . . . . . . . . . . . . . . . . 11

II Background 13

2 Neural Computation and Pattern Recognition 14

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.2 The Neuron . . . . . . . . . . . . . . . . . . . . . . . . . . 15

2.3 Neural coding . . . . . . . . . . . . . . . . . . . . . . . . . 18

2.3.1 Rate Coding . . . . . . . . . . . . . . . . . . . . . . . . 19

2.3.2 Temporal coding . . . . . . . . . . . . . . . . . . . . . . 21

2.3.3 Rank order code . . . . . . . . . . . . . . . . . . . . . . 23

2.4 Neural Networks . . . . . . . . . . . . . . . . . . . . . . . . 23

2.5 Temporal pattern recognition . . . . . . . . . . . . . . . . . 39

3 The GReaNs Platform 44
3.1 Linear Genome Representation 45
3.2 Artificial Gene Regulatory Network 46
3.3 Mapping of GRNs to SNNs 50
3.4 Structure of SNNs 51
3.5 Temporal Pattern Recognition Task 52
3.6 Genetic Algorithm for Evolving SNNs 54
3.7 Conclusion 56
III Results 57
4 Evolution of SNNs in the Absence of Noise 58
4.1 Experimental Setup 59
4.2 Analysis of the Evolved SNNs 63
4.2.1 Minimal FST for Recognising a String 64
4.2.2 Mapping Network States onto the FST States 66
4.2.3 Network State Transitions 69
4.2.4 Robustness 73
4.2.5 Longer silent intervals 74
4.3 Conclusion 78
5 Noise Promotes Robustness 79
5.1 Types of Noise 80
5.2 Experimental Setup I 81
5.2.1 Evolution 84
5.2.2 Robustness to Varying a Single Neuronal Parameter 86
5.3 Experimental Setup II 91

5.3.1 Robustness to Varying All Neuronal Parameters . . . . . . . . 92
5.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
**6 Dynamics of Evolved SNNs 100**
6.1 Experimental Setup . . . . . . . . . . . . . . . . . . . . . . . . . 101
6.2 Pruning Excessive Connections . . . . . . . . . . . . . . . . . . . . . 102
6.3 Understanding Dynamics of Pruned SNNs . . . . . . . . . . . . . . . 106
6.4 Conservation of Dale's rule . . . . . . . . . . . . . . . . . . . . . . 116
6.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118
**7 Handcrafting SNNs for longer Patterns 120**
7.1 Rules for Handcrafting Network Topologies . . . . . . . . . . . . . . . 121
7.2 Optimization of Connection Weights . . . . . . . . . . . . . . . . . . 122
7.3 Specialised Role of Neurons . . . . . . . . . . . . . . . . . . . . . . 124
7.4 Contribution of Connections to Pattern Recognition . . . . . . . . . . 126
7.5 Performance of the handcrafted Networks . . . . . . . . . . . . . . . . 127
7.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 130
**IV Conclusion 132**
**8 Conclusion 133**
8.1 Development of Ideas . . . . . . . . . . . . . . . . . . . . . . . . . 133
8.2 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
8.3 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 139
**Appendix A 143**
A.1 Networks Evolved in the Presence or Absence of Noise . . . . . . . . 143
A.2 Networks Evolved in the Presence of Intrinsic and Extrinsic Noise . . 148

A.3 Network Topologies Handcrafted for Recognizing a Pattern . . . . . . 151

A.4 Git Repositories . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155

**Appendix B Enclosed Papers 156**

B.1 IEEE SSCI 2017 M. Yaqoob and B. Wróbel . . . . . . . . . . . . . . . 156

B.2 ALIFE 2018 M. Yaqoob and B. Wróbel . . . . . . . . . . . . . . . . . . . 165

B.3 ICANN 2018 M. Yaqoob and B. Wróbel . . . . . . . . . . . . . . . . . 174

B.4 ICANN 2019 M. Yaqoob and B. Wróbel . . . . . . . . . . . . . . . . . 185

# Part I

# Introduction

# Chapter 1

# Introduction

Over the past 15 years, it has been established that the temporal activity of neurons represents information in the somatosensory, auditory, visual and olfactory systems [131, 63, 122, 94, 85, 47, 120, 137]. However, information processing in the brain is poorly understood. Due to its biological relevance and resemblance to sensory information processing, temporal pattern recognition using spiking neural networks (SNNs) is among the most exciting and widely studied topics in computational neuroscience. There are two common ways of learning to recognise temporal patterns using spiking neural networks: (i) adjusting conduction delays [119], and (ii) selecting conduction delays from a spectrum of existing delays [14]. This work provides a new way of learning to recognise temporal patterns by evolving (constructing following the rules borrowed from the natural evolution of population and selection) the topology and connections weights of SNNs, without changing or selecting conduction delays. An abstract way to get insight into how information is processed in spiking neural networks is to first evolve artificial spiking neural networks for a simpler computational task, and then understand the dynamics of the developed networks. In view of this, a computational task is designed in which bio-plausible (yet simple) spiking neural networks have to recognise a temporal pattern (a subsequence of signals) in a stream of input signals. In the first stage, a population of SNNs is evolved (both in noisy and noiseless conditions) such that the readout neuron of the SNNs responds only to a particular ordering of signals. Then the evolved networks are cleaned by pruning superfluous connections (if any exist) to ease the analysis. Pruning revealed structural similarities and commonalities among the evolved SNNs, which helped in understanding the mechanisms driving SNNs to perform temporal pattern recognition. Finally, the insight into the evolved SNNs enabled handcrafting network topologies for recognizing longer patterns.

## 1.1 Motivation

Our perception of information works both in time and space. Most biological nervous systems can detect relations and connections in patterns related to a certain time frame or space [3]. In general, the information we receive from all senses, including odour, [56] sight, [127] and sound [67] is temporal in structure. The brain must process these signals to represent the external world accurately. Processing of information in the brain and artificial spiking neural networks (SNNs) is represented by the temporal activity of neurons [66, 106, 126]. Understanding how temporal signals are processed by the brain and spiking neural networks is of great interest to the neuroscience community. Several studies have shown that information processing in the nervous systems is linked to the transition from one type of spiking behaviour to another [3, 10, 25, 39, 55, 78, 101]. However, despite extensive research, the relationship between the structural connectivity and functional behaviour of neural systems remains unclear. Unfolding the mechanisms that enable SNNs to process information is one of the most challenging problem in computational neuroscience

[8, 19]. A possible way to study the complex computational processing capabilities of the brain is to artificially evolve networks with biologically simple neurons that can learn to perform a specific task. Explaining the mechanisms driving the evolved networks performing temporal pattern recognition may provide an insight into the information processing in the nervous system.

More importantly, a solution to a computational task is considered reliable if its functionality can be explained. Explainable SNNs, even small ones, will open a new dimension in the field of computational neuroscience. In particular, the algorithms used by known SNNs to perform a task can be employed to handcraft network topologies for similar yet more complex computational tasks. Furthermore, these SNNs can be extended/wired together to perform even more complicated operations. This bottom-up approach is interesting for two reasons: smaller networks are easier to train and understand; second, the functionality of a large network made up of smaller interpretable building blocks can also be explained.

## 1.2 Aim

This work aims to understand spiking neural networks evolved to recognise a specific temporal pattern. How biological neuronal networks process information and maintain functionality in the presence of noise and damage is one of the primary interests in neuroscience. At the same time, understanding information processing in SNNs is a fundamental problem in computational neuroscience. The interpretation of simple yet biologically meaningful SNNs may give us an insight into information processing in their counterpart biological nervous systems. The work presented in this thesis can be divided into three parts: (i) first, the evolutionary setup for obtaining minimal networks for recognising temporal patterns is refined, (ii) then, the structural regularities and commonalities among the evolved networks are identified. These similarities can explain the mechanisms driving the evolved network to behave as a perfect-recogniser, an over-recogniser or a wrong-recogniser. With this knowledge, the specific role of each individual neuron in the computation is identified, and (iii) finally, after acquiring a complete insight into the evolved networks, rules are defined for handcrafting network topologies for recognising longer patterns in a particular order. This study aims to answer the following research questions:

- How can spiking neural networks learn to generate the desired output?
- How does the network process information?
- Why is the presence of noise important, and what are the potential benefits of noise in the system?
- What is the functional role of autapses (synapses a neuron makes with itself) and why they are essential? Is there any relationship between the length of the pattern being recognized and the number of autapses in the network?
- What is the relationship between the length of the pattern being recognized and the number of interneurons required in the minimal network?
- Is there any specific role for individual neurons in the network? If yes, how do they contribute to the recognition of a signal?
- Can we handcraft the topology of a network for recognising a pattern of arbitrary length?

## 1.3 Contributions to Knowledge

I have evolved the topology and connections weights of very small spiking neural networks for temporal pattern recognition, which has resulted in the following contributions to knowledge:

- I have demonstrated that both the topology and connection weights of a very small spiking neural network can be evolved to recognise temporal patterns in a continuous stream of input. This is true for both noisy and noiseless evolution; the noise is modelled as random fluctuations of the membrane potential of neurons in the network at every simulation step. Of particular interest, I observed that recognition happens with transitions between network states [147, 148, 149, 150].

- Networks evolved in the absence of noise are extremely fragile. A slight variation in the values of parameters or incoming input signals severely impaired the network's performance. In contrast, the networks evolved in the presence of noise showed a graceful degradation of performance to the disturbance level of parameters [149]. Moreover, I proposed a probabilistic algorithm to find the robustness range of all inter-dependent neuronal parameters of the AdEx model [150].

- In biological nervous systems, a neuron can either excite or inhibit other neurons but not both [15]. This property of a neuron is called Dale's principle. To keep the network small during evolution, I allowed the neurons to excite and inhibit other neurons simultaneously. To show that this violation is benign, I have demonstrated that an evolved network could be transformed to follow Dale's principle by splitting the violating neurons into an excitatory and an inhibitory part [147].

- Evolution may overproduce synaptic connections. While analysing the evolved network, I found that unlike noiseless evolution, SNNs evolved in the presence of noise are also robust to connection pruning. Pruning is essential for understanding the working mechanism of the networks [147].

- I have associated functional roles to the interneurons in the network. The interneurons have specialised behaviours that contribute to recognising the pattern. In particular, the *lock* neuron prevents the output from spiking unless it is released by the penultimate signal in the target pattern, the *switch* neuron is responsible for switching the network between the inter-signal state and the start state, and the *accept* neuron produces spikes in the output neuron when the network receives the last correct input. This neuron also sends a signal to the *switch* neuron, transforming the network back into the start state.

- I have suggested two possible functional roles for excitatory autapses: (i) networks with excitatory autapses are capable of maintaining network states in the absence of input activity, and (ii) autapses enable smooth transitions between network states.

- Based on the switching mechanism and structural commonalities of the evolved networks, I defined rules for handcrafting network topologies for recognising longer temporal patterns. The constructed topology is then validated by optimising connection weights for the target pattern.

## 1.4 Structure of the Thesis

This thesis consists of 8 chapters, starting with an introductory chapter (this one). The subsequent two chapters (2 & 3) provide a detailed background of the study. Then, the findings of the study comprise four chapters (4, 5, 6, and 7), and the last chapter (8) concludes the thesis with possible future directions.

I. Introduction

> **Chapter 1** starts with briefly introducing the topic, followed by the motivations behind the study, then the aim and the research questions are presented, the contributions to the knowledge made in this study are listed, the structure of the thesis is detailed, and a list of publications is given.

II. Background

> **Chapter 2** begins with the historical background of the neuron with its morphology and biophysical properties. Then two generally accepted neural codes (rate and temporal) are discussed. Neural coding refers to the study of how neurons represent information. In the subsequent section the three generations of neural networks are detailed. Then, three widely used computational neuron models are described: (i) the integrate-and-fire model (LIF), (ii) the adaptive exponential integrate and fire model (AdEx), (iii) the Hodgkin-and-Huxley model. Finally, a literature review on temporal pattern recognition is presented.
>
> **Chapter 3** begins with introducing the Gene Regulatory evolving artificial Networks (GReaNs) platform, originally developed to simulate the evolution of artificial gene regulatory networks (GRNs). It uses a genetic algorithm to evolve a population of artificial organisms in a variety of settings including

multi-cellular development, signal processing and foraging. The artificial organisms are encoded as a linear genome in GReaNs. In the next section the structure of the linear genome is adapted to represent a spiking neural network (SNN). The subsequent section illustrates GRN to SNN mapping, followed by a general sketch of the genetic algorithm used in GReaNs to evolve a population of linear genomes.

III. Results

**Chapter 4** starts with the basic experimental setup for evolving spiking neural networks using the genetic algorithms in GReaNs. The individuals presented in this chapter are evolved without noise. In order to understand the mechanism driving these networks, first the network states of a network are identified. Then, a correspondence is established between the identified network states and the state of a finite state transducer (FST). Finally, the robustness of the individual is assessed. The chapter concludes with findings and possible improvements for the experimental setup.

**Chapter 5** demonstrates the benefits of noise when introduced during evolution. Noise is modelled as random fluctuations of the membrane potential of neurons at every 1 ms network step. This chapter contains two independent experimental setups. The first setup updates genetic operators and introduces noise during evolution. Networks evolved in a noisy environment developed robustness to neuronal parameters. Then the robustness range of varying a single neuronal parameter (keeping others at their default values) is determined. The second experimental setup is a further refinement, especially in the way the fitness is calculated. This modification in the fitness function improved both yield and evolvability. Then, the robustness range to varying all inter-dependent neuronal (AdEx) parameters is obtained.

**Chapter 6** sheds light on the importance of pruning. Superfluous connections in the network are deceitful – making the analysis of the network difficult. Networks presented in this chapter are evolved in the presence of both noise and variation. Results obtained show that introducing variation on the duration of silence intervals during evolution favours the emergence of memory. Furthermore, the dynamics driving the minimal networks to recognise a target pattern are revealed. Finally, the transformation of a network to follow Dale's rule is described.

**Chapter 7** suggests rules for handcrafting network topology. The constructed topologies are then validated by optimising their connection weights. Then the specialized roles of neurons in the network are identified, followed by the contribution of connections to pattern recognition. The subsequent section demonstrates the performance of the handcrafted network for recognizing a pattern of 6 signals. The chapter is concluded with a list of findings.

IV. Conclusions

**Chapter 8** begins with the development of ideas, reviews the key findings, and concludes the thesis with a section detailing potential future directions.

## 1.5 Publications

1. M. Yaqoob, B. Wróbel, and V. Steuber (2022) Autapses Enable Temporal Pattern Recognition in Spiking Neural Networks (in preparation for Scientific Reports).

2. M. Yaqoob, V. Steuber, and B. Wróbel (2019) The Importance of Self-excitation in Spiking Neural Networks Evolved to Recognise Temporal Patterns, ICANN 2019 Munich, Germany, September 17-19, 2019.

3. M. Yaqoob and B. Wróbel (2018) Robust Very Small Spiking Neural Networks Evolved with Noise to Recognise Temporal Patterns, ALIFE 2018 Tokyo, Japan, July 23-27, 2018.

4. M. Yaqoob and B. Wróbel (2018) Very Small Spiking Neural Networks Evolved for Temporal Pattern Recognition and Robust to Perturbed Neuronal Parameters, ICANN 2018 Rhodes, Greece, October 5-7, 2018.

5. M. Yaqoob and B. Wróbel (2017) Very Small Spiking Neural Networks Evolved to Recognise a Pattern in a Continuous Input Stream, IEEE Symposium Series on Computational Intelligence (IEEE SSCI 2017) at Honolulu, Hawaii, USA, Nov. 27 to Dec. 1, 2017.

**Abstracts**

6. M. Yaqoob, V. Steuber and B. Wróbel (2022) Spiking Neural Networks as Finite State Transducers for Temporal Pattern Recognition, 31st Annual Computational Neuroscience Meeting CNS 2022, Melbourne, Australia, July 16-20, 2022.

7. M. Yaqoob, B. Wróbel, and V. Steuber (2022) Analysing Spiking Neural Networks Evolved for Temporal Pattern Recognition, PECS 2022, April 12, 2022,

University of Hertfordshire, Online.

8. M. Yaqoob, V. Steuber and B. Wróbel (2020) Autaptic Connections Can Implement State Transitions in Spiking Neural Networks for Temporal Pattern Recognition, CNS 2020 Organization for computational neurosciences, Online, July 18-22, 2020.

9. M. Yaqoob and B. Wróbel (2019) Robustness to the Removal of Connections in Spiking Neural Networks Evolved for Temporal Pattern Recognition, CNS 2019 Organization for computational neurosciences Barcelona, Spain, July 13-17, 2019.

10. M. Yaqoob and B. Wróbel (2018) Artificial Evolution of Very Small Spiking Neural Network Robust to Noise and Damage for Recognising Temporal Patterns, CNS 2018 Organization for computational neurosciences, Seattle, USA, July 13-18, 2018.

11. M. Yaqoob and B. Wróbel (2017) Evolving Spiking Neural Networks as Finite State Machine to Recognise Patterns in a Continuous Input Stream, Aspects of neuroscience, University of Warsaw, Poland, Nov. 26-27, 2017.

# Part II

# Background

# Chapter 2

# Neural Computation and Pattern Recognition

## 2.1 Introduction

This chapter describes the common terms and mechanisms in the neuroscience literature and in the present study. In particular, the structural unit of the brain, the neuron; the generation mechanism of spikes, the action potential; and the connections between neurons, the synapse. The aim is to understand the theoretical background and the composition of artificial spiking neural networks in general. Starting from a brief history of neurons, three state-of-the-art mathematical models of neurons with their system of differential equations are presented. Then, spiking neural networks are discussed, and finally, the significance of temporal pattern recognition is detailed.

## 2.2 The Neuron

The story began in 1832, when Johannes Purkinje observed the individual nerve cells for the first time in the cerebellum. Three decades later, in 1860, Otto Friedrich obtained the most precise picture of the nerve cell with two unusual protrusions. They were named "protoplasmatic processes" and "axis cylinder, " later becoming dendrites and axons. However, the structure of a nerve cell could not explain the organization of nerve cells in the brain. At that time, two opposing theories about the nervous system intrigued the researchers. The neuronists claimed that the brain comprises distinct cellular units – nerve cells, whereas the reticularists favoured the notion of a continuous network that can not be divided into discrete cells. In 1873 Camillio Golgi put an end to the debate by developing a staining technique called black reaction (Golgi's method) and demonstrated the gaps between two nerve cells. Later, Ramon y Cajal improved Golgi's method, confirmed the nervous system's cellular nature, and produced the famous drawing of neurons in the mammalian cortex in 1909 (Figure 2.1).

Moreover, the vocabulary we use today was established in the last decade of the 19th century. Heinrich Wilhelm used the term "neuron" for the first time in 1891. The name axon was given to "axis cylinder" by Rudolph Albert. Then Wilhelm His Sr. addressed "protoplasmic processes" as dendrites in his work, and in 1897 the junction between two neurons was called synapse by Sir Charles Sherrington. At the beginning of the 20th century, it was already believed that a neuron

- is the basic structure and processing unit of the brain,
- acts as an independent unit,
- consists of three parts: dendrites, soma and axon,

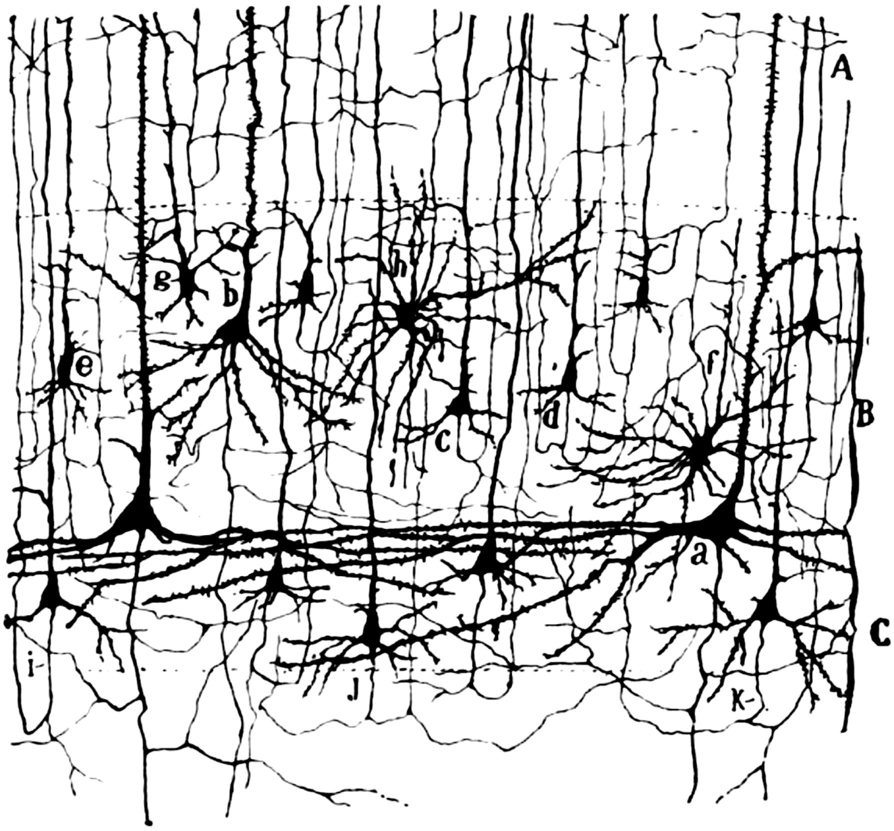


Figure 2.1: Drawing of the visual cortex of a human infant using Golgi's method by Ramon y Cajal (1909). A, B and C show layers 5, 6 and 7 respectively. With the staining procedure, only a small number (about 1%) of neurons are coloured. However, In reality, the neurons are extremely dense in the visual cortex. The dendrites extend vertically upwards whereas the axons branch out downwards in the left and right directions. For example label (a) shows a gigantic pyramidal neuron, (b) an average-sized pyramidal neuron.

- is uni-directional, the information flows from dendrites to axon via soma.

As stated above, the neuron – a unit of information processing in the brain, comprises three parts: dendrites, a cell body (soma) and an axon which can be considered as inputs, processing unit and output, respectively. The dendrites of a given neuron receive input information from other neurons in the form of spikes – action potentials. This information propagates in one direction from dendrites to axon through the cell body – soma. Both the integration and the process of generating a spike occur in the soma. If the total voltage crosses a certain threshold, the neuron generates an action potential (Figure 2.2) and sends it to other efferent neurons through the axon. This induces neurotransmitter release at synapses – connections between axon terminals and dendrites. This neurotransmitter release in turn causes current to flow in the dendrites of efferent neurons. The neuron sending the signal is referred to as a presynaptic neuron while the one at the receiving end is called a postsynaptic neuron.

Spikes are short (1-2 ms duration) electrical signals of amplitude roughly 100 mV. Since all spikes emitted by a given neuron have the same characteristic waveform, it is clear that the shape of spikes does not contain much information. Instead, the information is carried by the number and the precise timing of spikes. A chain of spikes generated by a given neuron is known as a spike train – a sequence of time-structured action potentials. A spike train is used to represent neurophysiological activity by recording only the timing of spikes and discarding their shape.

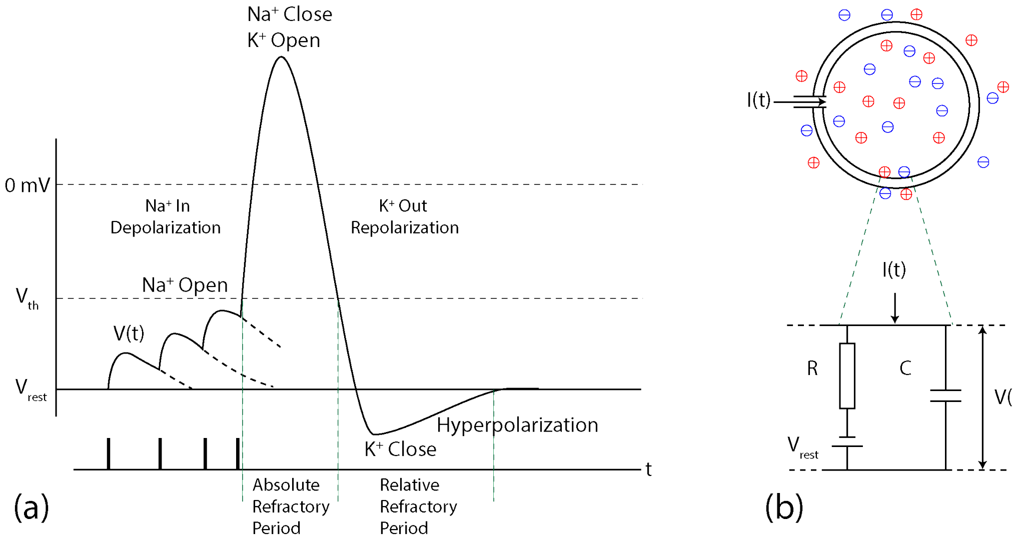


Figure 2.2: (a) spike generation mechanism: when the membrane potential V(t) reaches the firing threshold a spike is fired. Right after the spike, the voltage returns to a value below $V_{rest}$ – the resting potential. (b) Parallel RC circuit to describe cell membrane voltage when a neuron receives a spike at one of its synapses. The charge inside the cell charges the cell membrane capacitor C. Leakage of charge from the cell membrane is modelled as the leak resistance R.

The membrane potential remains constant at $V_{rest}$ in the absence of input, when a current pulse I is injected, the potential difference builds up across the cell membrane; if the potential difference crosses a certain level, the sodium channels open, allowing sodium ions $Na^+$ to enter the cell – called depolarization. This continues until the concentration of sodium ions $Na^+$ inside and outside the cell reaches equilibrium. Further depolarization activates the potassium channels and the $K^+$ ions leave the cell to repolarize it. Simultaneously, sodium channels are closed while the potassium channel remains open until the membrane potential drops to a value less than the resting potential $V_{rest}$ – called the hyperpolarization state. In this state the neuron is unable to generate another spike for a short while – called the refractory period.

Nowadays, the morphology and the biophysical mechanism (Figure 2.2) of a neuron are well understood, e.g. how a spike is generated? how does a sequence of spikes cause the release of the transmitter at the axon terminals? and how does this transmitter act on the connected neurons? However, it is less clear how a series of spikes affect other neurons. What do they represent? and how this information contributes to a certain behaviour?

## 2.3 Neural coding

Neural coding– the study of how neurons represent information – is the first step to understanding the brain. Neurons in the nervous system receive information (in the form of spikes) from sensory modalities as well as from other neurons. As a result, the membrane potential of the neuron fluctuates, and in many cases produces an irregular spike train. To describe a spike train, we study neural codes. Simply put, neuronal coding attempts to relate the neuronal response to the input stimuli. In contrast, neural decoding refers to reconstructing the stimulus from the sequence of spikes. For instance, the firing rate of motor neurons may represent the stretching of muscles, and the precise timing of spikes may describe the onset of a sensory stimulus. Due to fast processing capabilities, the notion that sensory information is carried by spike times rather than spike-rate gained attention in the past two decades [131, 90]. More importantly, it is possible to define a range of new coding schemes with spike times [101, 82]. In artificial systems, the feasibility of a coding scheme for a certain processing task is determined by its processing speed, resilience to noise, and simplicity (in terms of implementation) [98]. For more details please refer to the paper of Simon Thorpe et al. [128].

### 2.3.1 Rate Coding

The notion that spike trains represent information dates back to the very first recordings of sensory fibres in 1920 by Adrian [2]. In his experiments, he showed that stimulus intensity is positively correlated to the spike rate of the electrical activity of the fibre, hence representing information by a single real number. As the name suggests, rate coding is the average number of spikes in a time interval. There are three ways to define a rate code: temporal averaging, trial averaging, and population averaging.

**Temporal averaging** is recordings from one single neuron in the period during and after the presentation of the stimulus (say 10 ms); count the number of spikes and normalize it with the duration of the interval (equation 2.1). To analyze the data recorded from real neurons, interspike interval distribution measures the regularity of the spike train, whereas the Fano factor determines the repeatability across trials. Therefore temporal averaging is considered good for analyzing experimental data.

However, it is too slow to be used by real neurons in the brain.

$$V(t) = \frac{n^{spikes}}{T} \tag{2.1}$$

where $n^{spikes}$ is the number of spikes produced in the duration of the interval T.

**Trial averaging** repeats K "temporal averaging" trials, and then the spikes are counted in n tiny intervals $\Delta t$ (say 10 ms) across all repetitions. This time-dependent response is called peri-stimulus time histogram (PSTH) (equation 2.2). The rate measure across K repetitions is determined at a temporal resolution of $\Delta t$. PSTH is a powerful tool for experimental analysis. However, this is definitely not how the real neuron codes information, simply because it requires a repeated number of trials of the stimulus (An animal does not average across different trials before responding).

$$PSTH(t) = \frac{n(t; t + \Delta t)}{K \Delta t} \tag{2.2}$$

where K is the number of trials a stimulus is repeated and $\Delta t$ is the time resolution in which the spike rate is determined.

**Population averaging** is recording from a population of neurons simultaneously in a single trial. In this case the rate is determined across n neurons at temporal resolution $\Delta t$. This is a natural way of coding information. The average firing rate of a single neuron across different trials can be approximated as the firing rate of a population if the neurons in the population have similar properties.

$$A(t) = \frac{n(t; t + \Delta t)}{N \Delta t} \tag{2.3}$$

where A(t) is the average spike rate of a neuron in the population, N is the number of neurons in the population and $\Delta t$ is the time resolution at which the spike rate is determined.

For several decades the spike rate was considered as the only information encoding mechanism of neurons, due to ease of implementation and robustness to noise. However, in the past three decades, the possibility of alternative coding schemes gained attention because the processing of many behavioural situations is performed too fast to rely on the firing rate of neurons [10, 82, 99]. These behaviours include somatosensory, olfactory, auditory and visual responses [66, 106, 126, 131].

### 2.3.2 Temporal coding

Temporal coding uses temporal relations between spikes to represent information. Temporal codes can be divided into two sub-types: (i) time-of-arrival (TOA) codes and (ii) temporal pattern codes. Time-of-arrival (TOA) coding encodes information in the arrival time of spikes at a target neuron or a group of neurons. It refers to the idea that the timing of spikes relative to the arrival of a stimulus carries information about the stimulus. Moreover, the information about the location or direction of a stimulus is represented by the difference in the arrival time of spikes at different neurons [48].

**Time-to-first-spike** encodes information based on the delay between the onset of the stimulus and the time to the first spike (Figure 2.3a). Instead of counting the number of spikes in a window, ample information can be accumulated by recording the timing of the first spike. It has been documented that time-to-first-spike encodes the touch information in the tactile system [107, 64]. This code is simple and can be implemented with a single neuron. Moreover, the action is taken as soon as the

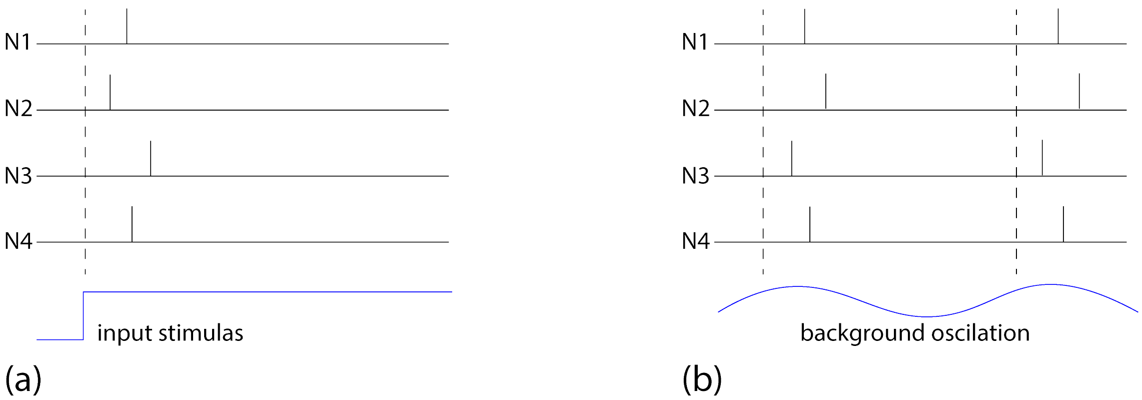


Figure 2.3: (a) Time-to-first-spike, neuron N2 is the first one to spike after the stimulus onset. (b) Phase code, the neurons fire relative to the ongoing background oscillation.

first spike is generated after the stimulus. Thus, the processing is particularly fast with this coding technique.

There is always some oscillation in the brain that can be recorded in the electroencephalogram (EEG). In **Phase coding** the neuronal spike patterns encode information in the phase of spikes relative to the ongoing internal oscillations in the nervous system. The neurons may repeat the spike pattern periodically if the stimulus remains the same from one cycle to the next (Figure 2.3b). Time-to-first-spike can also be employed to encode information with reference to the ongoing background oscillation instead of the input stimulus. It has been observed in the olfactory system and hippocampus [91, 78]. Early experiments on rat hippocampal place cells suggested a correlation between the receptive field and the phase of the oscillatory neural activity [53]. Furthermore, phase coding is also observed in electric fish [48].

**Temporal pattern code** encodes information based on the spike patterns and the relationship between spikes that are generated by a target neuron. It utilizes inter-spike interval (ISI) to convey information such that the information about temporally varying signals is conveyed by changes in the inter-spike interval [17, 136]. For example, a neuron might have a longer ISI in response to a weak stimulus and a shorter ISI in response to a stronger stimulus. In this way, the neuron conveys information about the intensity of the stimulus by changing its spike timing. It has been demonstrated that H1 neurons in the fly visual system can efficiently represent events in the temporally varying stimulus and duration of ISI before each spike [24]. Moreover, the distribution of inter-spike interval can rapidly convey information about the stimulus variance in an identified H1 neuron in the fly visual system as well as in a simulated Hodgkin-Huxley neuron model [81].

### 2.3.3 Rank order code

As the name suggests, this coding scheme looks at the order in which a set of neurons generate their first spikes. Each neuron is given a rank according to its first spike. This coding scheme only transmits the order in which the neurons fire in an observation window. Thus rank order code is robust to possible time jitters. Van Rullen and Thorpe [105] have demonstrated that the information between the retina and brain is represented by rank order coding. They have shown that the first few spikes of the retinal ganglion cells in response to a visual stimulus carry sufficient information to reconstruct a stimulus, even when the precise timing of spikes is subject to noise. Moreover, a single spike per neuron is sufficient for decoding [105].

## 2.4 Neural Networks

Neural networks process information in a brain-like fashion by incorporating individual computational units – neurons. Since information processing is accomplished by computations, neural networks are computational systems [96]. However, the parallel nature of these systems makes them substantially different from a typical digital

(Von Neumann) computer (Figure 2.5a). Furthermore, unlike digital computers, the memory is distributed throughout the system (in the network's parameters) and is readily available for computation (activation of the processing unit – neuron). Consequently, the functionality of the system degrades gracefully in the case of developing breakdown (missing units, damaged paths). Moreover, the event-driven nature of neural systems consumes less power as compared to the procedure-driven approach of digital computers [132]. There are no step-by-step instructions, thus, neural networks are trained (not programmed). Training a network requires a sample of input/output pairs, and an algorithm to adjust the weights of the connections in order to produce the desired output. This intensive process involves repetitive input feeding, output monitoring and connection-strength adjustments among neural units to obtain the desired output.

Both artificial intelligence and cognitive science (including computational neuroscience) employ the neural-network approach to build intelligent systems. Artificial intelligence concentrates on designing algorithms to develop intelligent systems. On the other hand, cognitive science studies the mechanism driving cognition.

### Feed-forward and Recurrent Neural Networks

According to the structure of the network, neural networks can be categorised as Feed-forward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs). FNNs consist of an input layer, one or more hidden layers, and an output layer. The information flows in one direction from the input layer to the output layer via the hidden layers. The network does not have any feedback connections and does not maintain any state from one trial to the next. Thus, FNNs do not keep memory and are generally used for regression and classification tasks.

On the other hand, Recurrent Neural Networks (RNNs) are designed to process time-series or sequential data. These networks keep a ”memory” of the previous hidden state and use that along with the current input to update the current hidden state. RNNs also have three layers; input layers, one or more hidden layers, and the output layers. However, unlike FNNs the networks have feedback connections such that each neuron in a hidden layer is connected not only to the inputs but also to the hidden layer outputs. This creates a feedback loop that allows information to be passed from one-time step to the next. RNNs are used for tasks such as language translation, image captioning, speech recognition, and natural language processing.

The feedback connections in RNN architecture enable them to process the previous state and the current input to update the current state. The traditional RNNs [32, 65, 18, 141, 102, 118] consisting of standard cells have shown remarkable performance in dealing with short-term dependencies. However, they cannot handle long-term dependencies because the magnitude of the error signals vanishes or exploded when back-propagated in time (multiplied several times with the same weight matrix), making it difficult to capture information from earlier steps. In order to overcome this issue LSTM networks were developed as an improvement over traditional recurrent neural networks.

### Long Short-Term Memory

In 1997 Hochreiter and Schmidhuber proposed LSTM (Long Short-Term Memory) to handle the problem of long-term dependencies by using gates and memory cells, allowing the network to selectively retain or discard the information for long durations, preventing the typical problems of gradient vanishing and exploding in standard recurrent networks [50]. To this date, several variants of LSTM have been proposed for a variety of machine learning problems [44]. The standard LSTM architecture

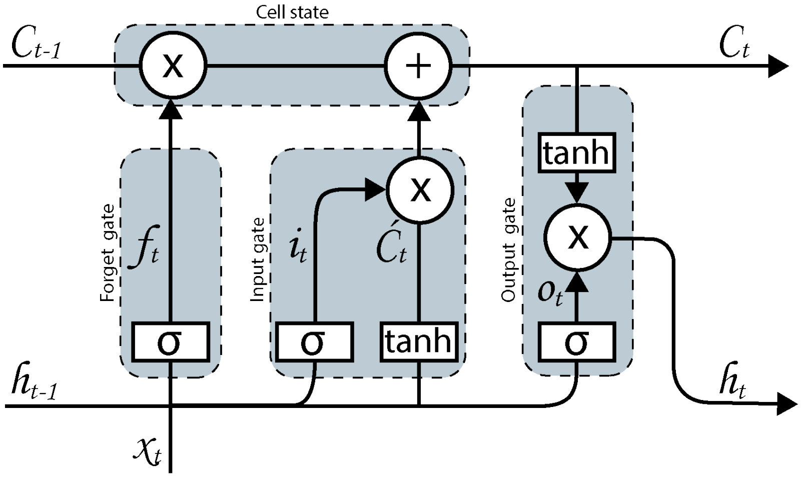


Figure 2.4: An LSTM cell consists of forget gate, input gate and output gate. The cell receives information from the current input $X_t$ and the previous hidden state $h_{t-1}$. The forget and the input gate update the cell state $C_t$ and the output gate determines the next hidden state $h_t$.

is given in Figure 2.4.

An LSTM cell comprised of three gates: (i) forget gate, (ii) input gate, (iii) output gate. These cells are implemented using a combination of the sigmoid and hyperbolic tangent (tanh) activation functions allowing the cell to selectively retain or discard information to deter the problem of vanishing gradient in the case of long-term dependencies.

The forget gate decides when to forget the information from the cell-state $C_{t-1}$. The information from the current input $X_t$ and the previous hidden state $h_{t-1}$ is passed through a sigmoid function, generating a value between 0 and 1 (Equation 2.4). A value close to 1 means that the old output is important.

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f) \tag{2.4}$$

where $f_t$ is the output of the forget gate at time t, $W_f$ is the weight matrix between forget and input gate, $h_{t-1}$ is the previous hidden state at t, $X_t$ is the current input, and $b_f$ is the connection bias with respect to $W_f$.

The input gate is implemented using a combination of sigmoid and tanh as activation functions. An input vector $i_t$ is generated by passing the current state $X_t$ and the previous hidden state $h_{t-1}$ through the sigmoid function (Equation 2.5). Simultaneously, the same information is passed through the tanh function, creating a vector $\hat{C}_t$ with values between -1 and 1 (Equation 2.6).

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i) \tag{2.5}$$

$$\hat{C}_t = tanh(W_c.[h_{t-1}, X_t] + b_c) \tag{2.6}$$

where $i_t$ is the input at time t, $W_i$ ($W_c$) is the weight matrix of the sigmoid (tanh) operator between the input and output gate, $b_i$ ($b_c$) is the connection bias with respect to $W_i$ ($W_c$), and $\hat{C}_t$ is the vector generated by tanh activation function.

To update the cell state $C_t$, the input vector $i_t$ and $\hat{C}_t$ are multiplied and the result is added to the product of forget vector $f_t$ and the previous cell state $C_{t-1}$ (Equation 2.7).

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.7}$$

The output gate determines the next hidden state $h_t$ by passing the current input $X_t$ and the previous hidden state $h_{t-1}$ through a sigmoid function and convolving the output with the new cell state $C_t$ after passing it through a tanh function (Equation 2.8 and 2.9).

$$o_t = \sigma(W_o.[h_{t-1}, X_t] + b_o) \tag{2.8}$$

$$h_t = o_t * tanh(C_t) \tag{2.9}$$

where $o_t$ is the output gate at time t, $W_o$ is the weight matrix at the output gate, $b_o$ is the connection bias with respect to $W_o$, and $h_t$ is the next hidden state.

**The Three Generations of Neural Networks**

According to the properties of the computational units, neural networks can be divided into three generations. The first generation employs McCulloch-Pitts [82] artificial neurons. These neurons accumulate the synaptic inputs and pass them to a simple step function to produce the output (Figure 2.5b). Thus, these neurons are digital (both input and output are digital) also referred to as threshold-gates or perceptron [103]. The Hopfield net [52, 54] and Boltzmann machine [49] are well-known examples of first-generation networks [82].

In the second generation of neural networks, the artificial neurons produce a continuous set of output values based on a non-linear function – the "activation function". Sigmoid and hyperbolic tangents are widely used activation functions. The sum of synaptic inputs is transformed using the neuron's activation function to produce the output – a real number. Therefore, second-generation artificial neural networks (ANNs) are more potent than first-generation networks and can deal with analogue inputs/outputs (Figure 2.5c). Furthermore, the output of the non-linear activation function can represent the firing rate of a real neuron. Therefore the second generation is considered more biological than the first [134].

It has been demonstrated that visual information in the brain is processed too swiftly to be encoded by firing rate [126, 125]. At the same time, several studies have indicated the significance of the temporal dimension of individual spikes in biological neurons to encode information [10, 131, 47, 101, 137, 34, 72]. To enable temporal and spatial aspects in the computation and transmission of information,

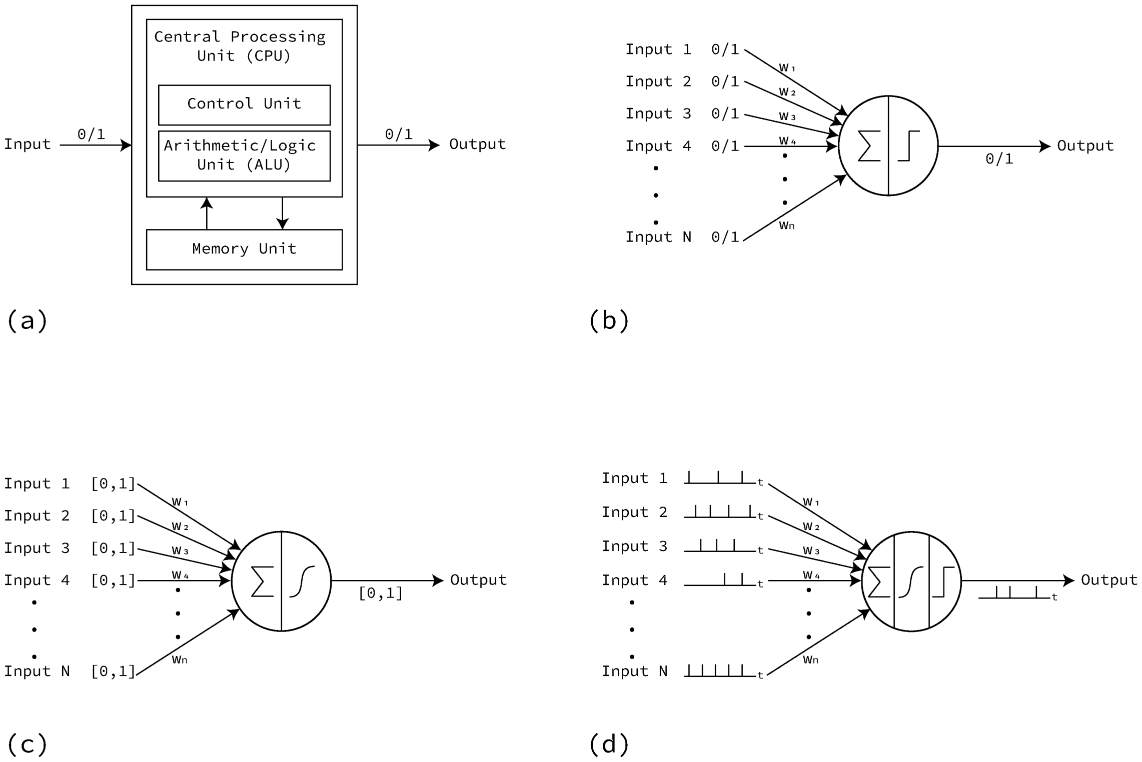


Figure 2.5: (a) Digital computer (Von Neumann architecture) performs digital computations (b) First-generation neural network (Perceptron): both inputs and outputs are digital. (c) Second-generation neural networks (feed-forward/ recurrent) are known for universal analogue computation. (d) Third-generation neural networks (integrate-and-fire model) incorporate temporal dimension by introducing individual spikes.

the third-generation neural networks incorporate a further level of realism by introducing individual spikes. These spiking neurons send/receive a series of spikes like biological neurons in the brain and networks employing spiking neurons are termed spiking neural networks (SNNs) (Figure 2.5d). The following section describes the mathematical formulation of three state-of-art neural models: (i) the integrate-and-fire (LIF) model, (ii) the Hodgkin and Huxley model, (iii) the adaptive exponential integrate-and-fire (AdEx) model.

### The Leaky Integrate-and-Fire Model

The early history of the integrate-and-fire model dates back to Louis Lapicque's work on the excitability of nerves [77, 13]. The simplest integrate-and-fire neuron can be defined with two essential components: (i) an equation to approximate the membrane potential of the neuron, and (ii) a mechanism to produce spikes [75, 114, 117, 138]. The leaky integrate-and-fire model is one of the simplest spiking neuron models. A fixed voltage threshold triggers the spike generation in LIF neurons. The membrane potential of a neuron increases with incoming spikes from other neurons; when it crosses a certain threshold, a spike is fired, and the membrane potential is reset to the resting potential $V_{reset}$. The membrane potential $V$ of a neuron integrates the input current $I(t)$ given by the following equation:

$$C\frac{dV}{dt} = I(t)$$

some charges may leak away; adding the leak term to the above equation, we get the simplest leaky integrate-and-fire neuron model.

$$C\frac{dV}{dt} + g_l V = I(t),$$

where $g_l$ is the leak conductance, and V is the membrane potential with respect to the resting potential, therefore:

$$C\frac{dV}{dt} + g_l(V - V_r) = I(t).$$

Adding excitatory and inhibitory conductance with respect to their specific reversal potential, gives:

$$C\frac{dV}{dt} + g_l(V - V_r) + g_{ex}(V - E_{ex}) + g_{in}(V - E_{in}) = I(t).$$

The above equation can be re-written as

$$C\frac{dV}{dt} = g_l(V_r - V) + g_{ex}(E_{ex} - V) + g_{in}(E_{in} - V) + I(t). \tag{2.10}$$

This equation models the membrane potential of a LIF neuron, where $V$ is the membrane potential, $g_l$ is the leak conductance, $V_r$ is the resting potential, $g_{ex}$ is the excitatory conductance, $E_{ex}$ is the excitatory reversal potential, $g_{in}$ is the inhibitory conductance, $E_{in}$ is the inhibitory reversal potential, and $C$ is the capacitance. When the membrane potential $V$ of a neuron exceeds the threshold $V_{th}$, a current impulse is fired, and $V$ is reset to $V_{reset}$. The values of the parameters for LIF neurons [1] are given in Table 2.1.

When the presynaptic neuron(s) spike, the excitatory ($g_{ex}$) or the inhibitory conductance ($g_{in}$) of the postsynaptic neuron is updated instantaneously by a value proportional to the synaptic weight according to the polarity of the synaptic connection. The excitatory and the inhibitory conductances decay exponentially:

$$\frac{dg_{ex}}{dt} = \frac{-g_{ex}}{\tau} \tag{2.11}$$

$$\frac{dg_{in}}{dt} = \frac{-g_{in}}{\tau} \tag{2.12}$$

where tau $\tau$ is the decay time constant, $\tau = 5$ ms is used in GReaNs for experiments [1, 144].

Table 2.1: LIF parameters

| Parameter | | Value |
|---|---|---|
| $V_r$ | resting potential | -65 mV |
| $E_{ex}$ | excitatory reversal potential | 0 mV |
| $E_{in}$ | inhibitory reversal potential | -70 mV |
| $V_{reset}$ | reset voltage | -58 mV |
| $g_l$ | leak conductance | 5 nS |
| $\tau$ | decay time constant | 5 ms |
| $C$ | total capacitance | 1 nF |

**The Adaptive Exponential Integrate-and-Fire Model**

The leaky integrate-and-fire model (Equation 2.10) can precisely generate spikes and can predict the spike timings of a real neuron. However, it has several limitations including the purely linear integration of input current, the fixed firing threshold, and the lack of memory of its own spiking history. Therefore extensions have been proposed in these three directions: (i) adding an exponential term enabling more realistic spike initiation [37], (ii) introducing adaptation of the membrane potential by adding another state variable [59], (iii) allowing conductance injection instead of current to mimic a real neuron [28]. Brette and Gerstner proposed the adaptive exponential integrate-and-fire (AdEx) model that extends the leaky integrate-and-fire model by introducing the above three generalizations [12].

An AdEx neuron in the network is governed by 4 state variables and 14 parameters. Differential equations of the state variables (membrane potential $V$, adaptation current $w$, excitatory conductance $g_{ex}$ and inhibitory conductance $g_{in}$) are given below:

$$
\begin{align}
C\frac{dV}{dt} &= g_{ex}(E_{ex}-V)+g_{in}(E_{in}-V)-w+g_l(E_l-V+\Delta_T e^{(\frac{V-V_T}{\Delta_T})}) \tag{2.13}\\
\tau_w\frac{dw}{dt} &= a(V-E_l)-w \tag{2.14}\\
\frac{dg_{ex}}{dt} &= \frac{-g_{ex}}{\tau_{ex}} \tag{2.15}\\
\frac{dg_{in}}{dt} &= \frac{-g_{in}}{\tau_{in}} \tag{2.16}
\end{align}
$$

Out of the 14 parameters, four bifurcation parameters are responsible for spiking behaviour: the adaptation conductance $a$, the spike-triggered adaptation $b$, the adaptation time constant $\tau_w$, and the resting potential $V_r$. The remaining scaling parameters are: the total capacitance $C$, the total leak conductance $g_l$, the effective rest potential $E_l$, the inhibitory $E_{in}$ and the excitatory $E_{ex}$ reversal potential, the threshold slope factor $\Delta_T$, the effective threshold potential $V_T$, and the two time constants: excitatory synapses $\tau_{ex}$, inhibitory synapses $\tau_{in}$.

The exponential term in equation 2.13 defines the spike generation mechanism and the ascent of the action potential. In the mathematical description of the model,

Table 2.2: AdEx parameters for tonic spiking

| Parameter | | Value |
|---|---|---|
| $E_l$ | effective rest potential | -70 mV |
| $E_{in}$ | inhibitory reversal potential | -70 mV |
| $E_{ex}$ | excitatory reversal potential | 0 mV |
| $V_r$ | reset voltage | -58 mV |
| $V_T$ | effective threshold potential | -50mV |
| $V_{th}$ | spike detection threshold | 0 mV |
| $\Delta_T$ | threshold slope factor | 2 mV |
| $C$ | total capacitance | 0.2 nF |
| $g_l$ | total leak conductance | 10 nS |
| $a$ | adaptation conductance | 2 nS |
| $b$ | spike-triggered adaptation | 0 pA |
| $\tau_w$ | adaptation time constant | 30 ms |
| $\tau_{ex}$ | excitatory time constant | 5 ms |
| $\tau_{in}$ | inhibitory time constant | 5 ms |

a spike is fired at time $t^f$ when the membrane potential crosses an arbitrary firing threshold value (much larger than $V_T$, say +30 mV). When this happens the integration of the differential equations (Equations 2.13 to 2.16) is stopped, the spike time $t^f$ is recorded, and the voltage is reset to a fixed value $V_r$. This reset describes the descent of the action potential, given by:

$$\text{at} \quad t = t^f \quad \text{reset} \quad V \longrightarrow V_r$$

Simultaneously, when a spike is recorded at time $t^f$, the adaptation current $w$ is increased by an amount $b$.

$$\text{at} \quad t = t^f \quad \text{reset} \quad w \longrightarrow w + b$$

Although we use the parameters (Table 2.2) for tonic spiking in all experiments in this work, the interaction between the differential equations of the AdEx model and the above two discrete resets can generate a variety of spiking behaviour [58, 88]. In future studies, it would be interesting to explore how a neuron in the network can represent other meaningful network states by changing its spiking behaviour (for instance, from tonic spiking to bursting behaviour).

### The Hodgkin-Huxley Model

The giant squid axon (diameter up to 1 mm) is the largest known axon, approximately 1000 times thicker than the human axon [92, 152]. Regardless of the size difference, the functionality of the nerve cell remains the same [135]. In 1952 Hodgkin and Huxley [51] observed three different ion currents in the giant axons of the squid. Together with measuring the three different ion currents (sodium, potassium and leak – mostly $Cl^-$), they described their dynamics using differential equations. This fantastic revolutionary work was awarded a Nobel Prize 11 years later in 1963. The elegance of Hodgkin and Huxley's experiment was that for the $Na^+$ and $K^+$ ion channels, they were able to fit (by hand) the gating variables (m, n, h), the stationary values $(m_0, n_0, h_0)$, and the time constants $(\tau_m, \tau_n, \tau_h)$. In short, they provided a general computational model for neurons with an arbitrary number of ion channels. Gating variables describe the state of activation or inactivation of a channel. The general equation of an ion channel is described as:

$$I_{ion} = -g_{ion} r^{n1} s^{n2} (V - E_{ion}) \tag{2.17}$$

where r (activation) and s (inactivation) are gating variables, n1 represents the number of activation gates, and n2 is the number of inactivation gates. The differential equations of gating variables r and s are given as:

$$\frac{dr}{dt} = -\frac{r - r_0(V)}{\tau_r(V)} \tag{2.18}$$

$$\frac{ds}{dt} = -\frac{s - s_0(V)}{\tau_s(V)} \tag{2.19}$$

Hodgkin and Huxley provided stationary values $(r_0, s_0)$ and time constants $(\tau_r, \tau_s)$ for gating variables by clamping the membrane at a range of voltage V.

In Figure 2.6, the capacitor C describes the cell membrane, the ion channels $(Na, K, leak)$ are modelled as conductances (resistors $R_{Na}, R_K, R_L$) and the drive of ions to move back and forth (the Nernst or reversal potential) is represented by batteries $(E_{Na}, E_K, E_L)$.

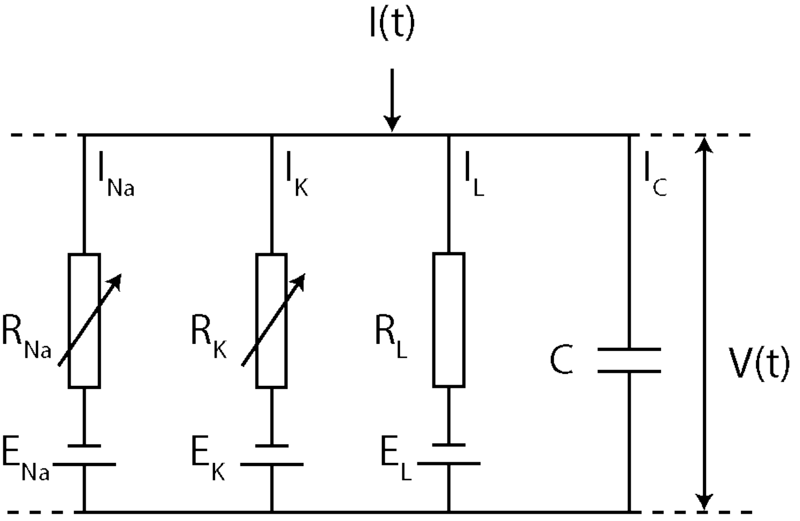


Figure 2.6: Parallel RC circuit diagram for Hodgkin and Huxley model

By the conservation of current:

$$I(t) = I_{Na} + I_K + I_L + I_C$$

where $I_{ion}$ is given by Ohm's law $I_{ion} = V_{ion}/R_{ion}$, the total voltage across the circuit for each ion $V_{ion} = V - E_{ion}$. If the membrane potential is equal to the Nernst potential, no current of that ion will flow.

$$I_C = -\frac{1}{R_{Na}}(V - E_{Na}) - \frac{1}{R_K}(V - E_K) - \frac{1}{R_L}(V - E_L) + I(t)$$

The capacitive current $I_C = dq/dt$, using the definition of capacitance $C = q/V$ the capacitive current can be re-written as: $I_C = CdV/dt$.

Therefore:

$$C\frac{dV}{dt} = -\frac{1}{R_{Na}}(V - E_{Na}) - \frac{1}{R_K}(V - E_K) - \frac{1}{R_L}(V - E_L) + I(t)$$

As conductance $g_{ion} = 1/R_{ion}$ therefore:

$$C\frac{dV}{dt} = -g_{Na}(V - E_{Na}) - g_K(V - E_K) - g_l(V - E_L) + I(t)$$

by introducing gating variables m, n, and h we get:

$$C\frac{dV}{dt} = -g_{Na}m^3.h(V - E_{Na}) - g_K n^4(V - E_K) - g_L(V - E_L) + I(t) \tag{2.20}$$

Equation 2.20 is the main differential equation for Hodgkin and Huxley's model, where the gating variables m, n, and h are probabilities between 0 and 1 associated to the activation of various channel subunits. For example, each potassium channel has 4 activation gates represented by $n^4$ (all 4 $K^+$ gates need to be open to allow the passage of $K^+$ ions), each sodium channel has 3 activation gates ($m^3$) and an inactivation gate $h$. The equation for gating variables m, n, and h is given by equation 2.18 and 2.19.

The choice of neuron model depends on the computational problem and the level of detail required. The LIF model is computationally efficient and used for simulations that require simplicity. The simplest LIF model assumes the neuron as a leaky

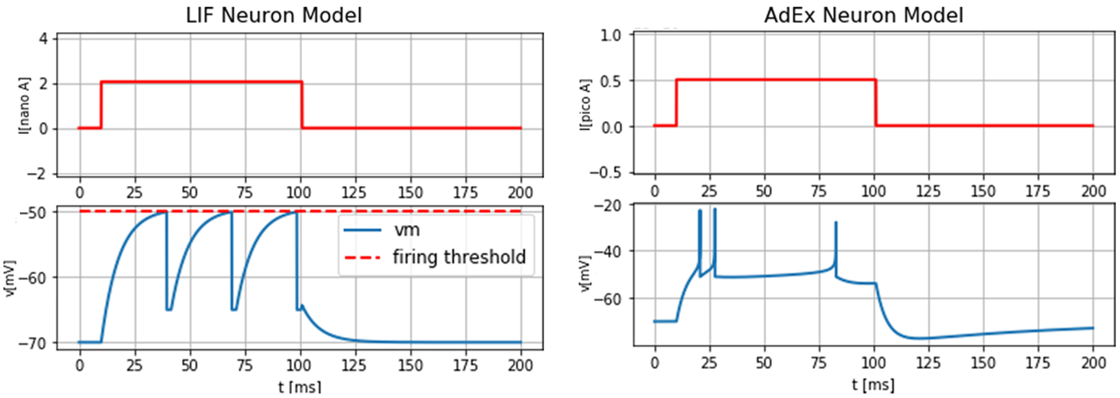


Figure 2.7: Step current injection into LIF (I = 2.1 nA), and AdEx (I = 0.5 pA) neuron model. Image source [40].

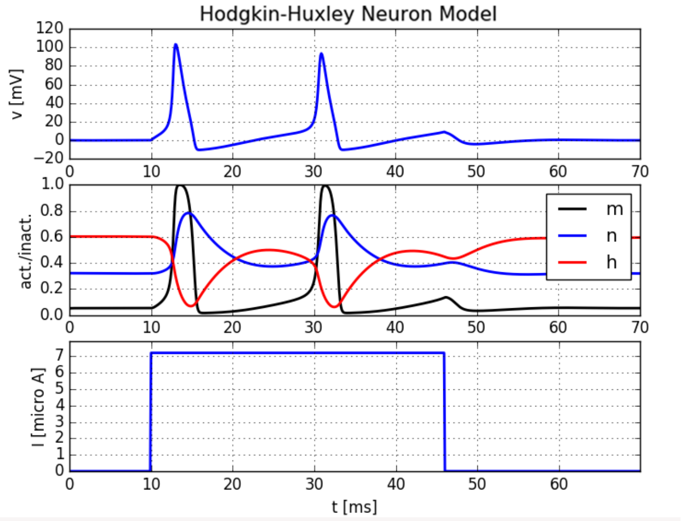


Figure 2.8: Step current injection (I = 7.2 $\mu$A) into Hodgkin-Huxley neuron model, where m, n, and h are gating variables associated with activation of various channel subunits. Image source [40].

capacitor that integrates input and generates spikes when the membrane potential reaches a fixed threshold (Figure 2.7). The AdEx model extends the LIF model by introducing exponential spike initiation and an adaptation mechanism which causes the threshold to increase over time, resulting in a decrease in the firing rate of the neuron. These extensions enable the AdEx model to mimic several spiking behaviours of biological neurons (Figure 2.7). Both LIF and AdEx models do not take the dynamics of ion channels of a biological neuron into account. Therefore, they are considered simple neuronal models. On the other hand, Hodgkin-Huxley model provides a general framework for neurons with an arbitrary number of ion channels. It includes a detailed description of the ionic mechanisms of biological neurons. A depolarizing current is injected into a neuron, resulting in an increase in the membrane potential, causing sodium channels to open. As a result, a spike is generated due to a rapid influx of sodium ions (Figure 2.8). After the spike, the potassium channels open and potassium ions leave the neuron which restores the membrane potential to its resting value. This is called repolarization.

## 2.5 Temporal pattern recognition

In the context of neuronal systems, the problem of temporal pattern recognition refers to identifying a sequence of spikes carrying information. The brain processes such sequences elegantly and responds in a timely manner [56, 67, 127]. Biological nervous systems can efficiently differentiate spike trains featuring time and space surprisingly well [64]. However, very little is known about how individual neurons contribute to processing temporal signals. Interpreting a sequence of spikes generated by a neuron (or a group of neurons) to determine the spatial or temporal structure of a stimulus is a fundamental problem in neuroscience [3].

Analogue sensory information from different modalities including olfactory, auditory, and visual is encoded in the form of spikes [53], and processed precisely by the brain with an incredible speed [70]. Spike timings capture the varying transient intensity of the stimulus, and even a single spike represents remarkable information after the stimulus onset [42, 131]. These results suggest that temporal features of spikes can precisely represent a stimulus, and convey information in artificial spiking neural networks [72].

From a computational perspective, processing temporal spike patterns is a general computational task performed by the brain [30, 23]. There are two common ways of learning to recognise temporal patterns: (i) adjusting conduction delays, (2) selecting conduction delays from a spectrum of existing delays. Spiking neural networks (SNNs) are documented to differentiate temporal patterns by exploiting different time delays and pathways in the network [53]. In the context of neural networks, time delays can be adjusted at the level of synapse, axon or soma (the cell body). Adjusting these delays at one or more levels in the network can uncover features of a signal [123]. On the other hand, it is possible that signals produced at different timings arrive together at the readout neuron – generating a maximal response. This phenomenon is used for edge detection in the visual system [100]. Moreover, delays in the network can also be used to identify keywords in a continuous speech [129].

In his influential paper, Hopfield suggested that artificial neural networks consisting of neurons with radial basis function (RBF) as activation function can recognize temporal patterns by using different latencies and coincidence detection [53]. An input signal encoded as the timings of spikes is processed irrespective of the stimulus intensity and scale. If the intensity of the stimulus is doubled the spikes will occur earlier in time by a proportional amount. Similarly using delays the pattern is recognized invariant of the scale – the stimulus is shifted in time. Moreover, it is possible to divide the stimulus into two halves, recognize them separately by independent networks and use coincidence detection to recognize the stimulus. If the two distinct readouts support each other, the input pattern is correct otherwise the pattern is wrong.

In another study, Steuber and Willshaw [119] presented a biophysical model of metabotropic glutamate receptors (mGluR) and showed that Purkinje cells could learn to perform pattern recognition by adjusting delays of the calcium response. The response time of the calcium influx is negatively correlated to the concentration of available receptors. Increasing the concentration of the receptors will result in decreased latency of the calcium response. Thus, learning is accomplished by adapting the concentration of the receptors [119].

The other way of learning to recognise temporal patterns is by selecting delays from a spectrum of existing delays – the spectral time model proposed by Bulloc et al. [14] for learning to select timed responses in the cerebellum. In their experiment on the conditioned nictating membrane response (NMR) of a rabbit, they showed that the cerebellum learns to adaptively delay responses by selecting conditioned delays. After repeated trials of pairing a conditional stimulus to an unconditional stimulus (corneal air-puff) the rabbit's NMR is conditioned to a conditional stimulus (light stimulus), anticipating the expected arrival of the air-puff by initiating eyelid closure before the onset of the corneal air puff. Studies have shown the involvement of the cerebellar cortex in the timing of behavioural responses [71, 95]. In the spectral time model, the axon of cerebellar granules cells divides into parallel fibres which excite both Golgi and Purkinje cells. The Golgi cells in turn inhibit the response of granule cells, attenuating their activation in time which is essentially learning to select timed responses [14].

Numerous studies have documented the integration of sensory information received from different modalities in the brain [16, 41, 133, 116]. Neurons responding to more than one modality perform this integration in the super-modal layer of the brain [31]. Also, experimental evidence of cross-modal coupling in the brain exists in which one modality interpolates in the regions belonging to other modalities [16]. According to this biological principle an SNN-based architecture has been proposed for integrating audio and visual modalities to authenticate a person[146]. Both modalities are temporal in structure, an independent SNN is employed for each modality, and the readout neurons of each modality represent a person. In the integration step, the output of both modalities is combined in the supramodal layer using the phenomenon of coincidence detection (Figure 2.9). The supramodal layer performs cross-modal coupling in which the neurons are able to process information received from different modalities [146].

Event-driven computation in SNNs makes them memory-and-energy efficient as compared to traditional ANNs. Several studies have indicated that SNNs can outperform conventional ANNs especially when processing event-based data both in terms

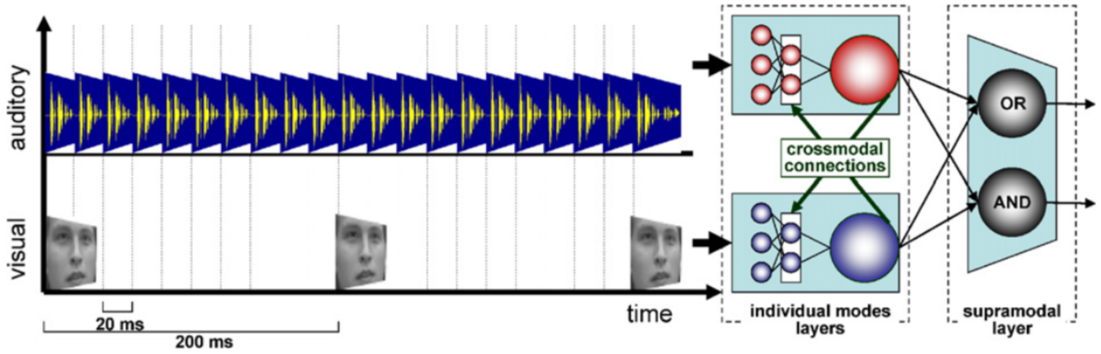


Figure 2.9: Integration of modalities in the supramodal layer. Both individual modes and supramodal layers employ spiking neurons. Image source [146].

of accuracy and efficiency (computational cost) [7, 26, 76, 82, 84, 135]. Moreover, the discrete nature of spiking neurons makes them fully parallel and highly energy efficient on neuromorphic hardware [4, 22, 38]. In addition to their biological relevance, the temporal encoding of SNNs naturally allows the processing of time-series data. Therefore, SNNs are considered appealing for investigating properties of time-structured data in general and spike trains in particular [115].

In this regard, efficient optimization techniques have been proposed to utilize the learning capabilities of SNNs. These techniques include unsupervised learning mechanisms such as Long-Term Potentiation (LTP) [124], Long-Term Depression (LTD) [57], Spike-Time-Dependent Plasticity (STDP) [9], Input-Time-Dependent Plasticity (ITDP) [110], and supervised learning mechanisms including Spike-Prop [11], Tempotron [45], ReSuMe [97], Chronotron [36], and SPAN [87]. In addition, based on evolving connectionist (ECoS) [69] methodology many algorithms are proposed for evolving SNNs for various computational tasks [29, 68, 108, 146]. Furthermore, two recent hybrid approaches filled the performance gap between SNNs and the typical ANNs on high-dimensional data: (i) the conversion approach transforms pre-trained ANNs into SNNs by mapping ReLU activation to integrate-and-fire activation [104, 109], (ii) the surrogate approach deals with the differentiation problem of leaky integrate-and-fire (LIF) networks by estimating the gradient function [89].

Although SNNs are complex and compute-intensive, advancement in optimization algorithms for SNNs has been remarkably successful. Even in some instances, SNNs are documented to outperform traditional ANNs both in terms of accuracy and speed [26, 76, 84, 135]. However, despite improvements in learning and optimization algorithms, very little is known about the computational mechanism of SNNs [20]. This study focuses on exploiting the computational abilities of SNNs for a computational task of temporal pattern recognition.

# Chapter 3

# The GReaNs Platform

The Gene Regulatory evolving artificial Networks (GReaNs) platform was originally developed to simulate the evolution of artificial gene regulatory networks (GRNs) – it uses a genetic algorithm to generate a population of artificial organisms in a variety of settings, including multi-cellular development (organisms in 2D & 3D, morphogenesis, soft-body animats), signal processing (pattern recognition and gain modulation) and robotic control (animat control, foraging, and movement of a robotic arm) [62, 73, 145, 147]. This chapter begins with the representation of the linear genome followed by the structure of artificial gene regulatory networks (GRNs) in GReaNs. Then the mapping of GRNs to SNNs is detailed. Subsequently, constraints on the structure of SNNs are given. Then, the computational task of pattern recognition is formulated. At the end, the genetic algorithm for the evolution of minimal SNNs is presented.

## 3.1 Linear Genome Representation

Due to scarce availability of resources, an organism expresses a gene only when it is required. Using this inspiration, the linear genome (representing an organism) in GReaNs consists of four genetic elements: external factors, cis-regulatory elements (CRE) a.k.a. promoters, trans-regulatory elements (TRE) a.k.a. products, and effectors. External factors are inputs to the cell, and effectors are the readouts, whereas CREs and TREs regulate the expression of the gene as per the requirement. In a real genome, TRE sites bind to CRE sites to activate or repress the transcription of genes into mRNA, which is then translated to proteins. This binding is termed an operon. An operon is abstracted in the artificial linear genome as a sequence of CREs followed by TREs. Thus, the number of operons is not fixed in a randomly created linear genome, where each operon has a variable number of cis and trans elements.

A genetic element in the linear genome has three attributes: a type (external factor, effector, CRE, TRE), a sign (positive, negative) and 2D (x, y) coordinates in $\mathbb{R}^2$. The affinity between elements is determined by the Euclidean distance between their x and y coordinates; the closer the elements in $\mathbb{R}^2$, the stronger the affinity, while the sign denotes activation (positive) and repression (negative) of transcription.

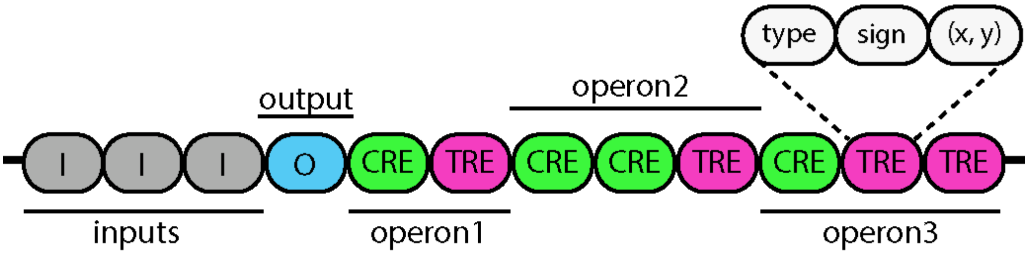


Figure 3.1: Encoding the gene regulatory network as a linear genome, an operon is a sequence of cis-regulatory elements (CRE) followed by a sequence of trans-regulatory elements (TRE).

## 3.2 Artificial Gene Regulatory Network

A gene regulatory network is a network of interactions between genes that explains how one gene promotes or suppresses the transcription of other genes. Here, a linear genome encodes the topology of an artificial gene regulatory network. In order to generate a GRN, the linear genome is first scanned for operons – a sequence of CREs followed by TREs. A multigraph is then created with the concept that products bind to the promoters with affinities calculated as a function of their Euclidean distance in $\mathbb{R}^2$. Each operon in the GRN consists of at least one product with a certain concentration in the cell (the concentration of the product is an abstraction of the amount of protein in the cell). A connection between two operons means that one gene regulates the other, that is the concentration of one influences (positively, or negatively) the synthesis of the other. These interactions are uni-directional from promoters to products with a cut off distance of 5 units. For example, if two elements are five or more units apart from each other in $\mathbb{R}^2$, there is no affinity between them, meaning they do not influence each other. This limits the full connectivity of the GRN. On the other extreme, if the distance between two elements is 0, an edge with the maximum (minimum) value of weight 10 (-10) is created in the GRN. The affinity between genetic elements is given by expression 3.1.

$$w_{i,j} = \begin{cases} s_i . s_j \frac{2(5-d_{i,j})\beta}{10d_{i,j}+\beta} \quad \text{when} \quad d_{i,j} \leq 5 \\ 0 \quad \text{when} \quad d_{i,j} > 5 \end{cases} \tag{3.1}$$

where $s_i$ and $s_j$ are the signs of genetic elements, the weight contribution is positive (negative) when both signs are the same (different). The distance between two elements is represented by $d_{i,j}$, and the beta $\beta$ parameter controls how fast the weight contribution drops with increasing distance. Figure 3.2 shows the distance conversion into affinity between elements for two values of beta ($\beta = 1$ and 5). All

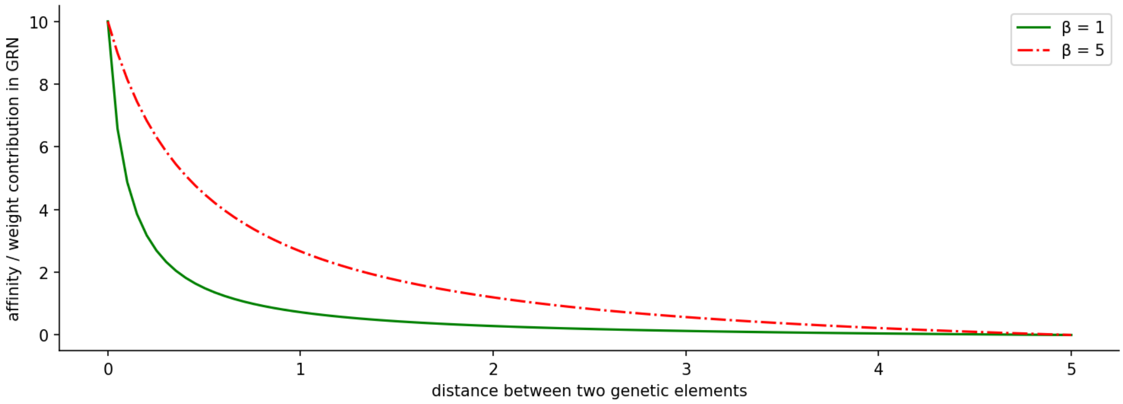


Figure 3.2: The distance affinity curve shows how the distance between genetic elements corresponds to the weight contribution in GRN.

experiments presented in this thesis use $\beta$=1. The weight contribution is given by the function $s_i s_j \frac{2(5-d_{i,j})\beta}{10d_{i,j}+\beta}$, when the distance between two elements is less than or equal to 5, otherwise the weight contribution is 0, thus, preventing full connectivity of the GRN by imposing a threshold value of 5 on the distance.

To sum up, if a product (TRE) is located inside the interaction distance of the promoter (CRE), an edge with weight equal to the computed affinity is added to the GRN. In the same way, the external factors (inputs) are connected to the CREs. The inputs are operated externally, and the network dynamics cannot control them. Finally, the effectors (outputs) are connected to the TREs according to the weight contribution given by Equation 3.1. Their concentration can be read externally. Furthermore, effectors cannot regulate the operations in the GRN, and the external factors cannot interact with them directly.

As an example, the decoding of a linear genome is given in Figure 3.3. After the identification of operons, the Euclidean distance between elements is translated to their corresponding affinities, and an edge is created in the network accordingly. Since an operon can have multiple CREs and TREs, the obtained GRN has multiple connections between operons. Moreover, external factors and effectors also connect

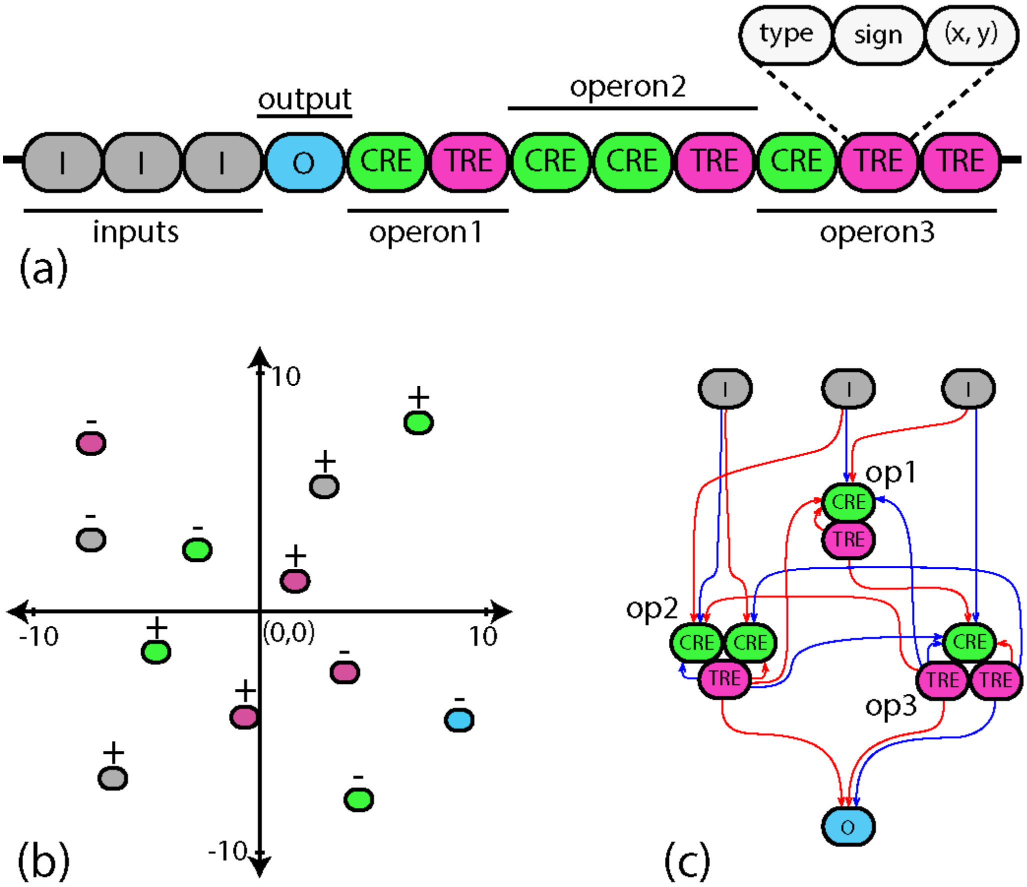


Figure 3.3: (a) Encoding of the linear genome into corresponding GRN. Structure of linear genome. (b) Position of genetic elements in 2D space. (c) Topology of the obtained GRN; red (blue) connections mean positive (negative) gene regulation.

to operons with more than one connection (Figure 3.3c). Multiple connections are then coalesced (by adding their weights) to form one connection. The combined effect of a coalesced connection is equal to the sum of individual connections.

All products in an operon have the same level of concentration (the amount of protein in the cell at a given time), biologically this abstraction can be interpreted as proteins with identical concentrations. The concentration L of each operon in the network is updated during the simulation in discrete time steps. The concentration level L is calculated as the sum of a degradation and a sigmoidal term given by Equation 3.2.

$$\Delta L = \left( \frac{1 - e^{-A}}{1 + e^{-A}} - L \right) \Delta t \tag{3.2}$$

where $\Delta L$ is the change in concentration, A is the activation level of the promoter (CRE) given by Equation 3.4, $L$ is the current level of concentration, and $\Delta t$ is the integration time. Forward Euler integration with a 1 ms time step is used to update concentrations. Since a single promoter can have $K$ binding factors, the activity of a promoter is calculated by the following Equation 3.3.

$$p_i = \sum_{k=1}^{K} L_k w_{k,i} \tag{3.3}$$

where $p_i$ is the activity of a given promoter, $L_k$ is the concentration level of $k^{th}$ factor and $w_{k,i}$ is the weight of the connection between factor k and promoter i. The activity of all promoters is then used to calculate the activation level of operons in the network.

$$A = \prod_{i=0}^{I} p_{m,i} \sum_{j=0}^{J} p_{a,j} \tag{3.4}$$

where $I$ and $J$ represent the number of multiplicative and additive promoters whereas $p_{m,1...i}$ and $p_{a,1...j}$ are described by the promoter activation function Equation 3.3 (note that $p_{m,0} = 1$ is the multiplicative identity and $p_{a,0} = 0$ is the additive identity). The presence of multiplicative promoters is required to ensure all-or-none regulation, that is a given promotor $p_m$ is multiplied by all multiplicative promoters (1 to I) in the GRN, thus the unit is expressed only if all products have an affinity to it, such all-or-none regulations are common in biological networks.

For more details on evolving gene regulatory networks, please refer to [60, 61, 62]. The GReaNs platform is extended for the evolution of spiking neural networks, SNNs [143]. The structure of a linear genome can be adapted to represent an SNN. The following section describes how the encoding of a GRN can be employed to represent an SNN as a linear genome.

## 3.3 Mapping of GRNs to SNNs

To adapt the linear genome representation for the encoding of spiking neural networks (SNNs), two essential modifications are required: (i) renaming the labels of genetic elements, (ii) updating the activation function. First, the cis and trans-regulatory elements (CRE and TRE) are renamed as dendrites Ds and axon terminals ATs. Like an operon, a sequence of dendrites followed by axon terminals encodes for a neuron (Figure 3.4). The strength of the connections (synapse) between the coordinates of genetic elements (inputs, outputs, dendrites and axon terminals) is determined by using the distance-affinity function (Equation 3.1) of the GRN. Similarly, the interactions (activation/ repression) between genes in the GRN map to the polarity (excitatory, inhibitory) of synaptic connections in the SNN [143].

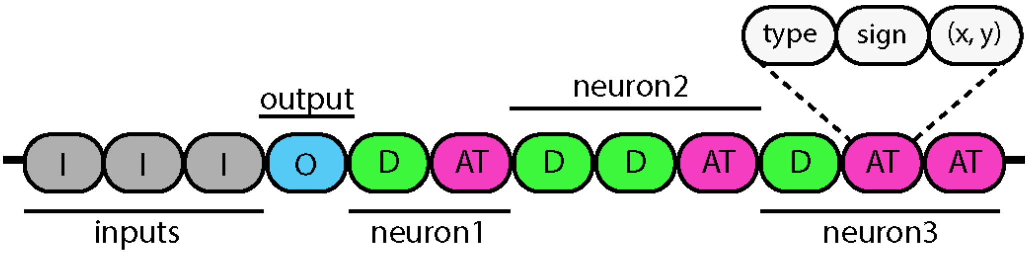


Figure 3.4: Encoding of the spiking neural network as a linear genome, a neuron is a sequence of dendrites (D) followed by a sequence of axon terminals (AT).

Second, the activation function (product concentration equation 3.4) is updated to a neuronal activation function. In a simple neuron model, when a presynaptic neuron spikes, an amount of conductance proportional to the synaptic strength is injected into the postsynaptic neurons. Consequently, the membrane potential of the postsynaptic neuron increases (for excitatory inputs) and emits a spike when it crosses a certain threshold.

Activation functions of two neuronal models "leaky integrate-and-fire – LIF" [117] and "adaptive exponential integrate-and-fire – AdEx" [12] are implemented in GReaNs (described in Chapter 2). Although the task of temporal pattern recognition can be accomplished with spiking networks composed of LIF neurons [143], the present work employs only AdEx neurons to exploit their ability to produce rich spiking patterns [88], which can be used in future studies to represent distinct network states.

## 3.4 Structure of SNNs

The spiking neural networks (SNNs) constructed for the temporal pattern recognition task consist of three layers: the input layer receives the input signals, the hidden layer processes these signals, and the output layer indicates the network's response (Figure 3.5). The number of nodes in the input layer is equal to the length of the pattern to be recognised. Each input signal is given to the network through a dedicated input channel. The minimum duration for which a signal can be presented to the input channel is one network step (1 ms). However, it must be presented for a moderate amount of time because short signals of length 1 ms produce networks with stronger synaptic connections which causes inconsistent responses while long signals with no (or small) gaps produce stateless solutions. In this study, the length of a signal is kept between 4 and 6 ms. Furthermore, an interval of silence separates input signals. Without silent gaps during evolution, the obtained solutions cannot sustain a network state in the absence of an input signal. Also, silent intervals promote stable switching between network states. Moreover, variable intervals between input signals tend to form memory in the network. The networks are less likely to develop memory if the duration of intervals is fixed. In the initial experiments, the length of the silent interval was 8 ms which was later increased to 16 ms.

Recognising a temporal pattern of size n requires n input channels, m interneurons ($m \geq n$) and a single output neuron. At the beginning of evolution, the number of inputs and output are set beforehand according to the computational task. Also, the upper limit on the number of interneurons in the network is preset. The evolution cannot employ more than the allowed number of interneurons, however, it is possible to produce a solution with fewer interneurons. A neuron is allowed to be excitatory and inhibitory at the same time. Inputs are not allowed to connect to the output directly and the interneurons can form self-loops. The maximum number of connections in this structure recognising a pattern of length n is given by the expression $nm + m^2 + m$ (Figure 3.5). The strength of the connection is determined by a function of the Euclidean distance (Equation 3.1). Inputs connect to interneurons, interneurons connect to each other and to the output neuron. In the initial generation, all individuals are created at random. An individual is considered valid if the inputs and the output are connected to interneurons. Moreover, a solution is marked abnormal if an input or output node is removed from the network during evolution. Such solutions are not transferred to the next generation.

## 3.5 Temporal Pattern Recognition Task

The neural activity in the brain continuously recognises temporal patterns received from different sensory modalities. To explore the working mechanism of spiking neural networks, a computational task of identifying a specific sequence of signals in a continuous stream of inputs is designed. Spiking neural networks employed for this task are made up of recurrently connected artificial neurons. These neurons communicate with each other through spikes when an input signal is received and produces an output. For the pattern recognition task, the networks are optimised during evolution such that the readout neuron spikes only for the correct ordering of the input pattern and remains silent for others. The shortest possible temporal pattern is composed of two signals. The complexity of the task increases with the length of the pattern; recognising a shorter pattern is easier than a longer one because a shorter pattern has fewer possible orderings, and the network requires a smaller number of states. For example, the possible ordering of 2 signals (say A, B) is 4 (AA, AB, BA and BB). Similarly, the number of possible permutations of 3 signals is 27 (AAA to CCC).

Initially, the networks are evolved to recognise a temporal pattern of 3 signals. After analysis of the minimal networks obtained, the evolutionary algorithm is optimised

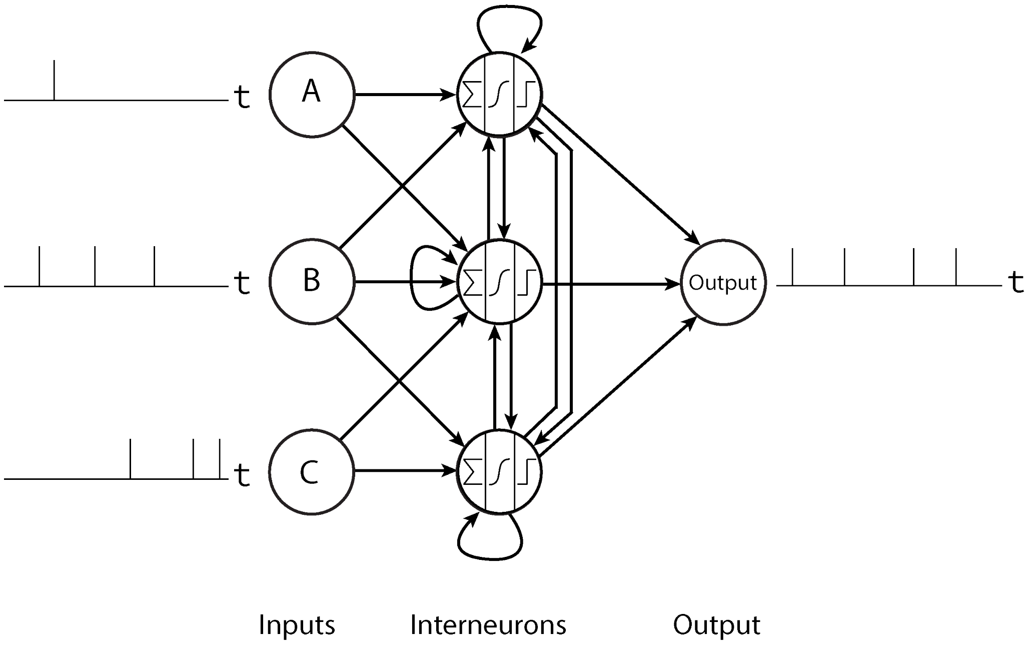


Figure 3.5: An example of a fully connected structure of a spiking neural network suggested for recognizing a temporal pattern ABC. The input layer receives the input signals, the interneurons process the incoming signals, and the output layer indicates the network's response. The arrows show synaptic connections (excitatory or inhibitory) between inputs, interneurons and the output.

to produce networks for recognising 4 signals in a particular order. Analysis of solutions obtained with independent evolutionary runs for recognising 3 signals revealed the recognition mechanism of SNNs. For recognizing 4 signals and above, the evolutionary algorithm could not produce a minimal solution (with less than 10 interneurons). Moreover, the problem of pattern recognition is analogous to processing a regular expression. Therefore, the paradigm of finite-state transducer (FST) is used to understand the switching mechanism of SNNs.

## 3.6 Genetic Algorithm for Evolving SNNs

The structure of the linear genome remains the same regardless of the network (GRN or SNN) being encoded. In the linear genome representation of a SNN, a neuron is encoded as a sequence of dendrites followed by axon terminals. The required numbers of inputs and outputs are inserted at the beginning of the genome. The product-promoter space is $\mathbb{R}^2$ and each genetic element (input, output, dendrites and axon terminal) is associated to a point (x, y) in $\mathbb{R}^2$. The distance between the coordinates of the genetic elements determines the strength of the synaptic connections. All the genetic operators are defined on the linear genome. A one-point mutation moves the coordinates associated with an element in $\mathbb{R}^2$ that corresponds to strengthening or weakening synaptic connections and re-wiring the structure of the neural network. The deletion (duplication) operator removes (copies) a random number of contiguous elements starting at a random site in the linear genome. The length of elements to be duplicated or deleted is drawn from a geometric distribution ($P(X = k) = p(1-p)^{k-1}$) with a mean of 10 (p = 0.1). To ensure inputs and outputs are intact during evolution, the first n+1 genetic elements (where n is the required number of inputs and 1 indicates a single output) in each genome are excluded from acting as the start elements for duplication or deletion, however, their coordinates can change – updating the weights connecting them to the network. A fixed number of individuals are subject to crossover (recombination), a crossover point is picked at random for each of the two parental genomes, and the right parts of the crossover points are exchanged, producing two new offsprings.

In the initial generation, a random population is created with a fixed number of inputs, outputs and an upper limit on the number of interneurons. A neuron in the linear genome has a random number of dendrites followed by a random number of axon terminals. These numbers are drawn from Gaussian distribution with a mean value of 3 and a standard deviation of 1. If the drawn number is less than 1, a single element (dendrite or axon terminal) is created. The coordinates of each element are drawn from a uniform distribution between [-10, 10] in $\mathbb{R}^2$, while the sign of the elements is either positive or negative determined by a coin flip (p = 0.5). As noted above, the strength of the synaptic connection between dendrites and axon terminals is determined by the distance function (Equation 3.1). The closer (further away) the genetic elements are in $\mathbb{R}^2$, the stronger (weaker) is the synaptic connection.

To extensively explore the search space for these minimal solutions the rate of duplications is kept higher than that of deletions. This results in lengthening of the genome (size of neuron) during evolution (but not the number of neurons in the network as they are set beforehand). Thus the dendrites and axon terminals tend to grow for a single neuron during evolution – precisely adjusting the weights of the connections in the network. Subsequent generations are created with a binary-tournament selection i.e. two individuals are chosen at random, and the best one according to the fitness value is transferred to the next generation after going through genetic operators (mutation, deletion, duplication and crossover). To ensure good solutions in the population, elitism is enabled during evolution. Fitness is defined as a function of reward and penalty. In this study of temporal pattern recognition, the fitness function rewards spiking of the output neuron for the correct pattern and penalises spikes elsewhere.

## 3.7 Conclusion

The structure of the linear genome in GReaNs can be adapted to describe the evolution of complex networks. Moreover, the linear encoding can efficiently explore the search space to produce optimal networks (with less than 10 interneurons) for various computational tasks in the presence of noise. However, the number of elements exposed to mutation increases multiple times with the network's size (several elements encode for a single neuron – the sequence of Ds followed by ATs). This causes an exponential increase in the search space with the number of neurons in the network. Therefore, the GReaNs platform is not suitable for evolving large networks.

# Part III

# Results

# Chapter 4

# Evolution of SNNs in the Absence of Noise

This chapter focuses on the evolution of spiking neural networks (SNNs) in the absence of noise for recognising a pattern of three signals. Noise can be introduced intrinsically or extrinsically during evolution. Intrinsic noise is random fluctuations of the membrane potential of each neuron in the network at every network step, while extrinsic noise is random variations of the duration of silent intervals in the input stream. An individual network consists of three input nodes (a dedicated channel for each input signal), up to five interneurons, and one output neuron. Using a genetic algorithm a population of SNNs is evolved, such that the output neuron spikes after receiving signals in the correct order (say ABC). During evolution, the networks are allowed to have up to five interneurons. However, the evolutionary algorithm produced perfect recognizers with only two interneurons. The network is presented with a continuous random sequence of signals (pulses) to the input channels (A, B and C). These signals are encoded as intervals of time. A signal's duration is 4 ms followed by a silent interval of 8 ms. Input signals are forwarded to the connected interneurons modelled as adaptive exponential integrate-and-fire (AdEx) neurons [2]. The recurrent interneurons successively process these signals and produce output. An AdEx neuron is governed by four state variables: membrane potential $V$, adaptation variable $w$, excitatory ($g_{ex}$) and inhibitory conductance ($g_{in}$). Their differential equations are given in Chapter 2, Equation 2.13 to 2.16. The values of initial parameters are chosen such that the interneurons generate tonic spiking for a constant input current (Chapter 2, Table 2.2). It is important to note that all the neurons in the network are conductance-based, such that when an input signal is received or a pre-synaptic neuron fires, the waveform of the synaptic conductance is convolved with the input and multiplied by a factor reflecting the strength of the connection (Chapter 2, Equation 2.13).

## 4.1 Experimental Setup

An independent evolutionary run consisted of a population of 300 individuals. The initial generation was created by generating random genomes, each coding for a network of up to five interneurons, three inputs and one output neuron. The length of a genome in the population was variable such that each region coding for a single neuron had an arbitrary number of dendrites and axon terminals. The genetic operators were defined by the linear genome; the structure of the linear genome is described in Chapter 3. To preserve good solutions in the population, the top 10 out of 300 individuals were transferred to the next generation without mutation, i.e. elite count = 10. Subsequent generations were created with binary tournament selection, i.e. two individuals were picked at random and the best one was transferred to the next generation after going through the three genetic operators simultaneously: (i) one point mutation, (ii) duplication, and (iii) deletion.

**One-point mutation.** Each genetic element (inputs I, output O, dendrites D, and axon terminals AT) of the genome had a probability of 0.01 to undergo a one-point mutation. If an element was chosen for one-point mutation, the x, y coordinates associated with the element were being moved in the affinity space in a random direction by a distance drawn from a normal distribution with mean 0 and standard deviation 3.

**Duplication.** The rate of duplication was 0.001 per genome. A random element was picked as a starting point on the chosen genome, and the number of elements to be duplicated was drawn from a geometric distribution with a mean value of 10. Then, the duplicated part of the genome was inserted at a randomly chosen insertion site.

**Deletion.** The rate of deletion was 0.0005 (half of the rate of duplication). A sequence of elements drawn from a geometric distribution with a mean value of 10 was deleted from a randomly picked site on the chosen genome.

The first four genetic elements (the three inputs and the output) were excluded from duplication and deletion. However, they were allowed to undergo a one-point mutation. Since the rate of duplication rate was higher than that of deletion, the genomes were likely to grow during evolution, but the size of the network stayed the same (set to five interneurons beforehand). Therefore, the neurons enlarged during evolution (number of dendrites and axon terminals), which accounted for fine-tuning of the synaptic connections.

**Fitness.** The fitness function represents error, 0 means the network has no error and 1 means maximum error. The fitness function rewards spiking of the output neuron in the interval after the correct pattern is received and penalises spike(s) in all other intervals (Figure 4.1). It is calculated by combining the following two functions (equation 4.1, 4.2). Theoretically, either of them can be used as a fitness function. In practice, however, using one of them did not work well; combining the two increased evolvability and resulted in a better yield (equation 4.3).

$$f_{fitness1} = 1 - (R - CP) \tag{4.1}$$

$$f_{fitness2} = \frac{R}{R + CP} \tag{4.2}$$

The two fitness functions are combined as follows

$$f_{fitness} = 1 - \frac{(R - CP) + \frac{R}{R+CP}}{2} \tag{4.3}$$

where R is the normalised reward: the number of correctly identified patterns divided by the total number of correct patterns in the sequence, and P is the normalised penalty: the number of intervals in which the output neuron spiked incorrectly divided by the total number of incorrect intervals, and the constant C is the penalty

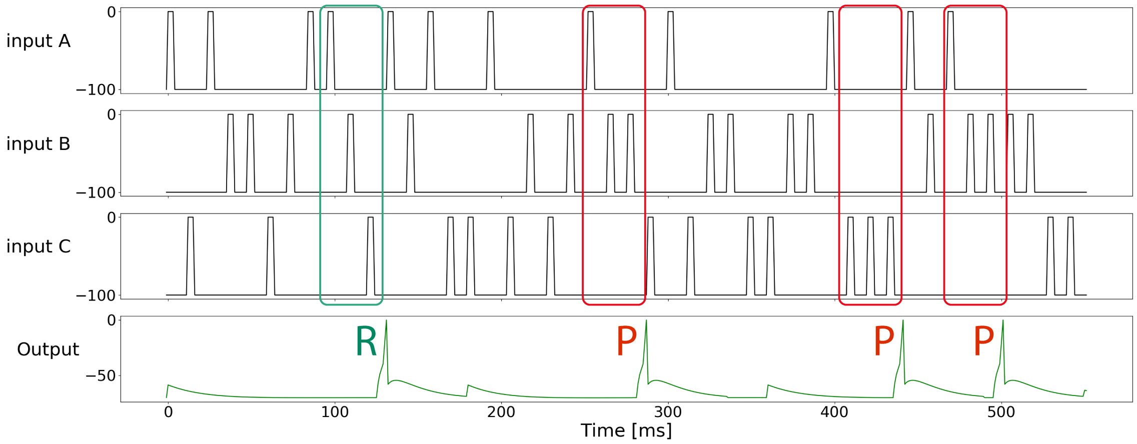


Figure 4.1: The fitness function rewards (R) spiking of the network output after receiving the correct input pattern ABC and penalizes (P) spikes elsewhere. In a random input stream, the number of intervals in which the output can spike incorrectly is a large number. Therefore the normalized penalty is amplified by a constant C.

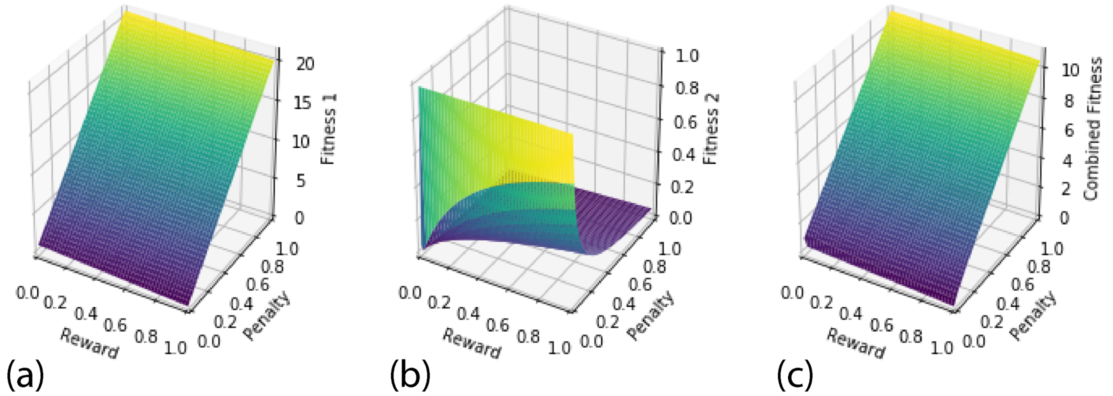


Figure 4.2: These plots show how reward and penalty correspond to the fitness value for C = 20. (a) The first fitness function defined by equation 4.1 can get a min value of 0 and a max of 21 (CP+1). (b) The second fitness defined by equation 4.2 can get a min value of 0 and a max of 1. (c) The combined fitness defined by equation 4.3 can get a min value of 0 and a max of 10.

amplifier. Since incorrect intervals are large in number, the normalised penalty P approaches zero when the number of false positives reduces. Therefore, a large value of C is required to penalise the wrong patterns in the later generations strongly. The value of C for experiments in this chapter is set to 20.

Employing only the first fitness function (Equation 4.1, Figure 4.2a) with a small value of C (below 10) produced over-recognizer networks (the punishment for responding to a wrong pattern gets negligible in the later generations). As a solution, when the value of C was increased to 20, the individuals preferred to remain silent because they were strongly penalized for responding to wrong patterns. To kick-start the evolutionary process the second fitness function (Equation 4.2, Figure 4.2b) was combined with the first fitness function. The combinations of two fitness functions (Equation 4.3, Figure 4.2c) not only encouraged evolution in the initial generations but also strongly penalized individuals responding to wrong patterns in the later generations.

Theoretically, for C = 20, the value of fitness (Equation 4.3) ranges from 0 to 10.

In practice, however, the fitness value is almost always between 0 and 1, because the normalised P is a very small number. The stopping criterion for the genetic algorithm is when the fitness value becomes zero or when the maximum number of generations (1000) is reached.

### Recognizing a pattern of three signals

In the absence of noise, a network recognising a pattern of three signals requires three input nodes, at least two interneurons, and one output neuron. A pattern of three signals (say A, B and C) has 27 possible permutations (AAA to CCC). Out of 27 patterns, only one pattern (say ABC) is considered correct. The output neuron of an evolved network responds only to the correct pattern ABC with at least one spike and remains silent for all other patterns. During evolution, every individual in the population is evaluated on six random sequences containing 5000 signals each, created with equiprobable occurrence of the three signals (A, B, and C). The pattern in the correct order (ABC) constitutes approximately 10% (1/27 * 3) of the continuous sequence of the form ABBACAABCCACBCAB... . Each signal is presented to the network on the designated channel for 4 ms (emitting a spike at every network/integration step 1 ms) followed by a silent interval of 8 ms.

## 4.2 Analysis of the Evolved SNNs

Out of 200 independent evolutionary runs, each optimising 300 SNNs up to 1000 generations, the top 10 individuals based on their fitness value are analysed in this chapter. These champions were tested with a new random sequence of 500,000 signals. Four out of 10 individuals behaved as perfect recognisers. The output neuron of a perfect recogniser spiked only after receiving the correct pattern ABC.

All the perfect recognisers had a fitness value of 0 (Table 4.1). The next three individuals by rank (5, 6, and 7) did spike incorrectly after receiving certain patterns ending with ..CBC, ..BBC and ..BBA. However, the networks did not respond when the same patterns were given to the network again with different preceding signals. This clearly indicates that the networks responded to these wrong patterns due to preceding signals in a specific order – a form of history. In the next two individuals (no. 8 and 9), the output neuron always responded to wrong patterns (CBC and BBC) with 2 (BBCBC) and 3 (ACBBBC) preceding signals, respectively. The last individual can be classified as an under-recogniser. This individual failed to recognise all ABCs preceded by CCB, and couldn't identify any pattern of the form ...CCBABC. However, it recognised all other ABCs and never responded to any wrong pattern. Therefore, the normalised penalty for this individual was zero (Table 4.1).

Table 4.1: The number of states and performance of the top individuals evolved for recognising a pattern of 3 signals with short intervals of 8 ms.

| | | | | number of network states | | | | |
|---|---|---|---|---|---|---|---|---|
| Champion | Reward | Penalty | Problematic sequences | Start | hA | hAB | hABC | Total |
| 1 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 2 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| 3 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| 4 | 1.0000 | 0.0000 | none | 4 | 2 | 1 | 1 | 8 |
| 5 | 0.9976 | 0.0000 | BAACBC sometimes | 3 | 2 | 1 | 1 | 7 |
| 6 | 0.9952 | 0.0002 | CBBBBC sometimes | 4 | 2 | 1 | 1 | 8 |
| 7 | 0.9502 | 0.0002 | ABABBA sometimes | 2 | 2 | 1 | 1 | 6 |
| 8 | 0.9774 | 0.0014 | BBCBC always | 3 | 3 | 1 | 1 | 8 |
| 9 | 0.9317 | 0.0022 | ACBBBC always | 4 | 4 | 2 | 1 | 11 |
| 10 | 0.6454 | 0.0000 | CCBABC never | 4 | 2 | 2 | 1 | 9 |

### 4.2.1 Minimal FST for Recognising a String

A finite state transducer (FST) is a formal model of computation for time-structured data [113]. The FST paradigm is used to understand the working mechanism of the networks evolved to recognise a temporal pattern of 3 signals.

In language processing FST has several applications including string recognition, generation, and translation [113]. A recognizer FST encodes the internal structure of a string and computes a function that maps the input string to the output string {0, 1}. Formally, an FST for recognizing a string of three letters ABC is described by 6-tuple $FST = (Q, \Sigma, \Delta, q_0, F, \sigma)$ where, $Q$ is the finite set of states = {S (start), hA (had A), hAB, hABC}, $\Sigma$ is the finite set of input symbols = {A, B, C}, $\Delta$ is the finite set of output symbols = {0,1}, $q_0$ is the designated start state = start, $F$ is the set of final states = {hABC}, and $\sigma$ is the transition function from input to output ($\sigma \subseteq Q \times \Sigma \times \Delta \times Q$). The set of all possible transitions is given by:

$$\begin{aligned}\sigma =\{&(start, A, 0, hA), (start, B, 0, start), (start, C, 0, start),\\ &(hA, A, 0, hA), (hA, B, 0, hAB), (hA, C, 0, start),\\ &(hAB, A, 0, hA), (hAB, B, 0, start), (hAB, C, 1, hABC),\\ &(hABC, A, 0, hA), (hABC, B, 0, start), (hABC, C, 0, start)\}\end{aligned}$$

where each 4-tuple represents a transition, for example, the first 4-tuple (start, A, 0, hA) descibes that input symbol A transforms the transducer (currently in the start state) in to hA state, and produces no ouput. The minimal FST for recognising a string of three letters can be constructed by hand (Figure 4.3), where S is the start state of the FST. The FST transitions from S to state hA (had A) when A is received, then to state hAB (had AB) only when this A is followed by a B, and lands in the state hABC (had ABC) when AB is followed by a C. When the FST transitions to the state hABC, it produces an output symbol 1, indicating that the correct string has been received. The preceding signal – history determines the current state of the FST, for example, if the symbol just received is B preceded by C, the FST must be in the start state (S), as this is the only state that could be reached after a sequence of symbols that ends with CB.

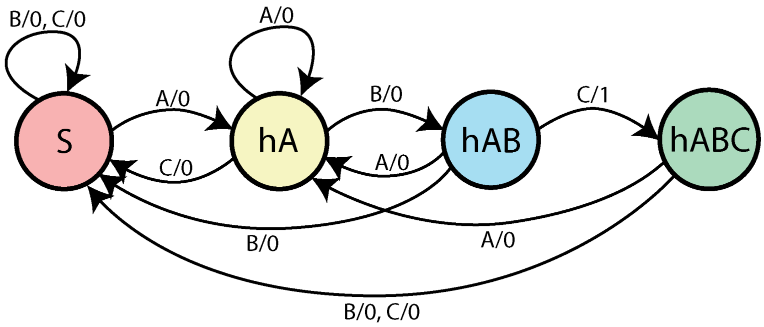


Figure 4.3: The minimal FST for recognising a string of three letters has four states. The circles represent states, and the edges show the transition from one state to another state. An output symbol (0 or 1 after "/") is produced upon each transition (0 or no output means reject the input, 1 means accept the input).

### 4.2.2 Mapping Network States onto the FST States

The behaviour and the constituents of the evolved SNNs are found to have a one-to-one correspondence with the 6-tuple of $FST$ that maps the input string ABC to the output string {0, 1}. Therefore, a spiking neural network recognising a temporal pattern of three signals can be formalised as a 6-tuple $FST = (Q, \Sigma, \Delta, q_0, F, \sigma)$, described in the previous section. Here, $\Delta$ represents the finite set of spiking behaviours of the output neurons = {quiet, spiking}.

Information processing in SNNs can be understood by mapping the spiking activity in the network onto the states of an FST. This is accomplished by observing the spiking behaviour of the interneurons and the output neuron in the silent interval (8 ms). A **neuron state** refers to the spiking pattern of the neuron in the silent interval after receiving an input signal, whereas a **network state** refers to the state of all neurons in the network in a given silent interval. Input signals have many-to-one correspondences with the network states, i.e. several input signals may transform the network into the same network state.

### Identification of Network States

In order to identify the possible network states of an evolved SNN, the network was given all possible orderings of signals A, B, and C (from AAA to CCC), and the network states were identified. It was observed that all networks converged to a single final network state upon receiving the last signal C in the correct pattern ABC. However, it is possible that the network produces different spiking patterns (due to preceding signals – history) upon receiving A and AB. Consequently, a network can have multiple hA or hAB states. Similarly, a network can have more than one start state for signals received in the wrong order.

Out of 4 perfect recognisers, two were chosen for analysis as an example of the least (champion 1 evolved 5 network states) and the most (champion 4 evolved 8 network states) number of network states (Table 4.1).

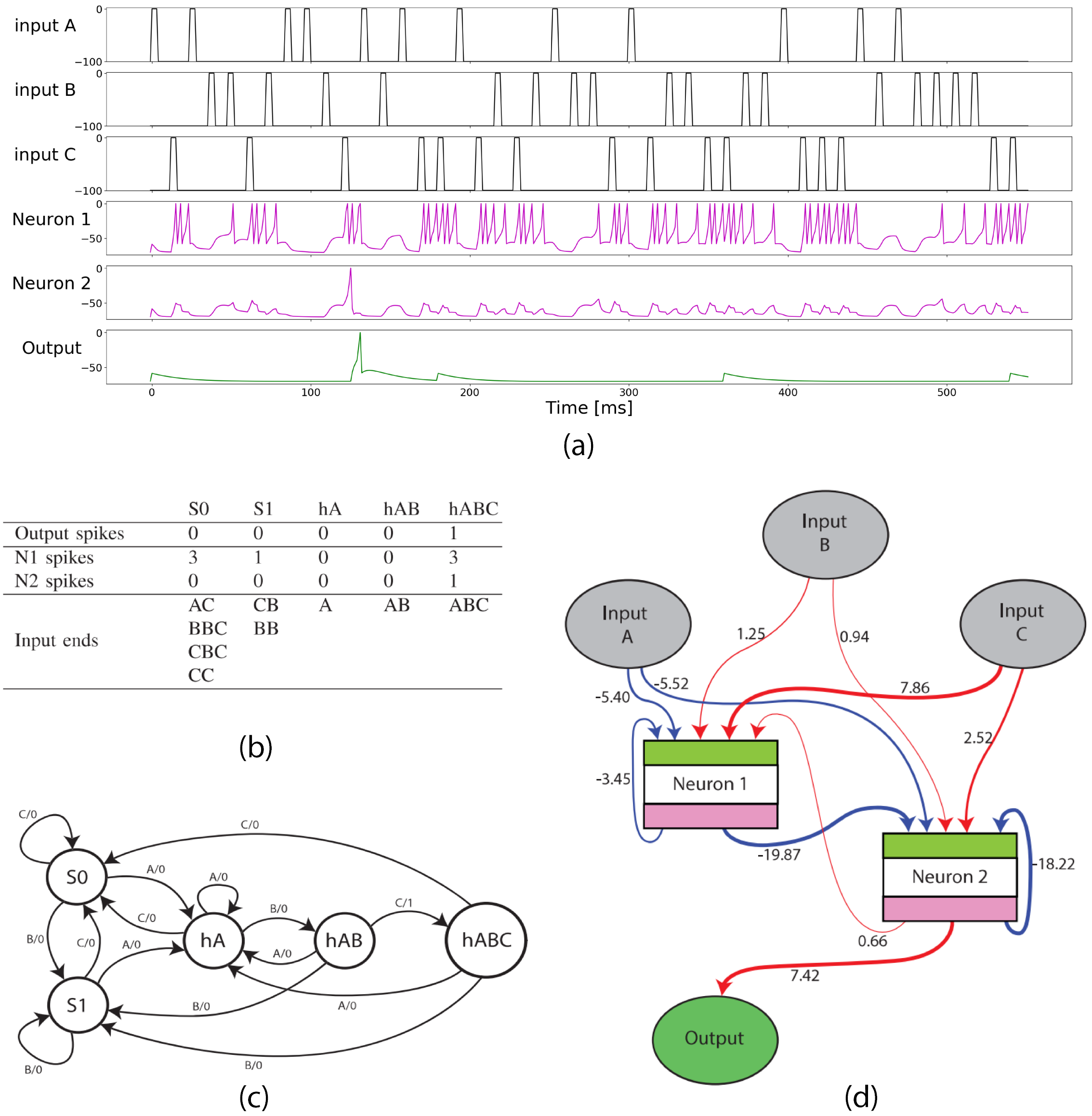
**Champion 1.** The network states of champion 1 are given in Figure 4.4c. When the network receives A, both interneurons and the output neuron turn off, transforming the network to hA state, represented by 000 (meaning the output, N1, and N2 do not spike – no activity in the network). When this A is followed by a B, the network goes into state hAB represented by 000, indicating no activity in the network. At this stage, if the network receives C, the network transforms to state hABC, denoted by 131 (output spikes once, N1 spikes 3 times, and N2 spikes once). In a similar fashion spiking patterns for the start state S are identified. For network states hA, hAB, and hABC, the spiking behaviour of all neurons remained consistent regardless of the preceding symbols. However, the start state S is represented by two different spiking patterns assigned to S0 and S1 (Figure 4.4b). When C is preceded by A, BB, BC or C the network goes in state S0, represented by spiking pattern 030. Likewise, if B is preceded by B or C, the network transforms to state S1, denoted by spike pattern 010.

An evolved SNN recognising pattern ABC should have at least 4 network states equal to the number of computationally different states in the minimal FST (Figure 4.3). However, it is possible to have more than one network state for each computational state of the minimal FST. Therefore, an evolved SNN has 4 groups of states, where each group can have multiple states except the last group which contains only one state. On the other hand, it is also possible that two computationally different network states have the same spiking pattern but obviously different neuron variables. Therefore, the spiking behaviour of a neuron is not sufficient to represent its current state. In the example of Champion 1, two computationally different states hA and hAB are represented by no activity in the network even though they can be differentiated based on the momentary membrane potential of neurons. When the network receives B after A the membrane potential V of both neurons rises, and if C arrives at this point, both neurons spike (N1 three times, N2 once) followed by the output neuron emitting a spike. This does not happen, if the order of the signals AB is reversed (BA) or B is preceded by any other signal B or C.

**Champion 4.** Network states of Champion 4 were identified in the same way as for Champion 1, and are illustrated in Figure 4.5. The spiking pattern of both interneurons and the output neuron in the silent interval (8 ms), after receiving input signals, determined the number of network states for recognising ABC. This individual accomplished the recognition task with 8 well-defined network states. The network states for this individual also converged to a single final state (hABC), represented by a unique spiking pattern. According to the input signals, this network generates 4 different spiking patterns for the start state S, represented by S0, S1, S2 and S3, and two different spiking patterns for state hA represented by states hA1 and hA2 in Figure 4.5 c.

### 4.2.3 Network State Transitions

A transformation of the network from one network state to another is referred to as a network state transition. Given the network topology, connections weights, the



| | S0 | S1 | hA | hAB | hABC |
|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 3 | 1 | 0 | 0 | 3 |
| N2 spikes | 0 | 0 | 0 | 0 | 1 |
| Input ends | AC<br>BBC<br>CBC<br>CC | CB<br>BB | A | AB | ABC |

Figure 4.4: Analysis of champion 1 which recognises a pattern of 3 signals in order ABC with 5 network states. (a) Spiking activity of the network. (b) Identified network state for each spiking pattern when the network receives different input signals. (c) Corresponding FST based on the network states identified in panel (b) for recognising a string of three letters ABC. (d) Evolved spiking network for recognising ABC.

spiking behaviour of the neurons, and the possible network states (Figure 4.4), it is straightforward to describe network state transitions when an external input signal is received.

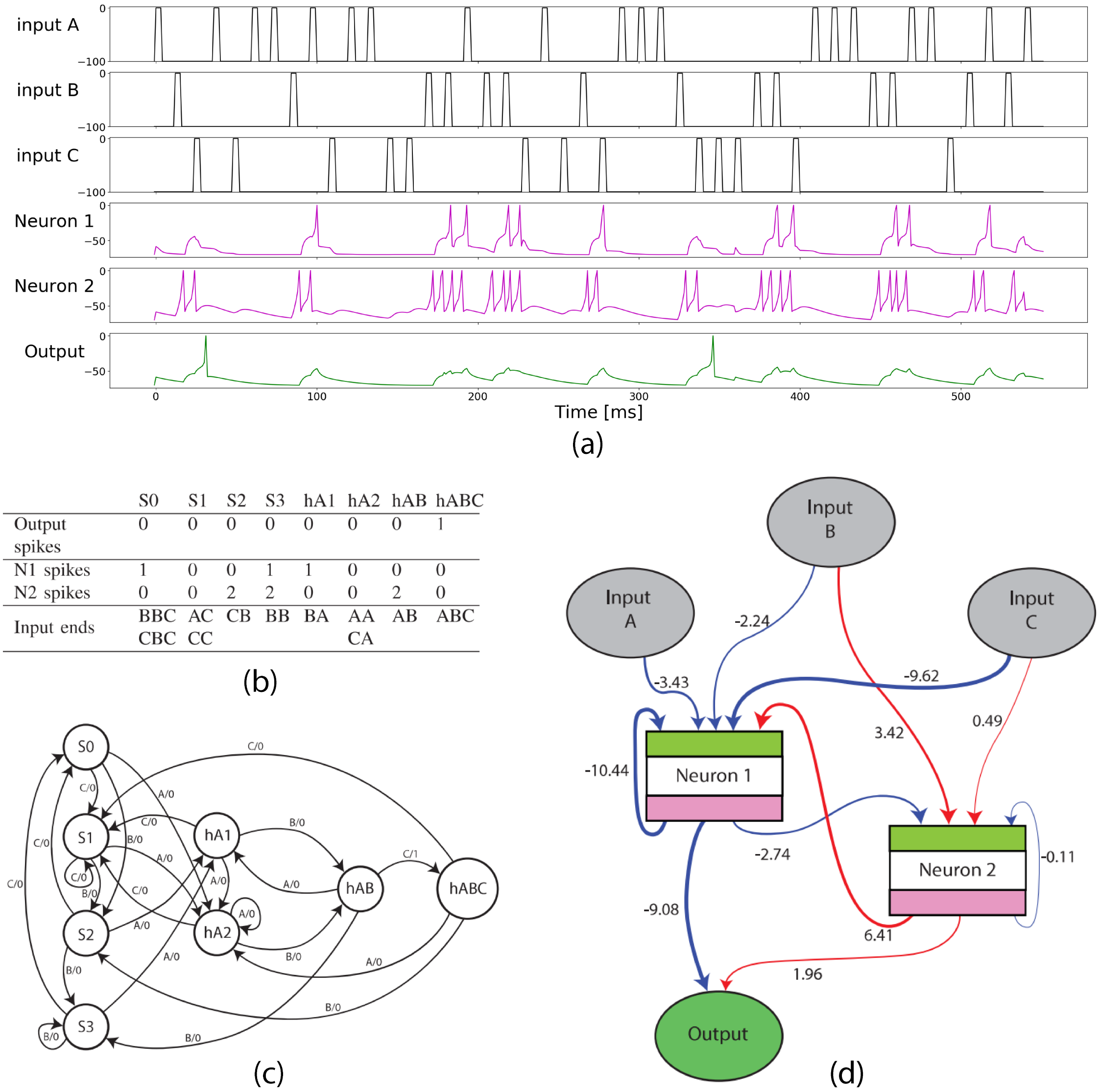
**Champion 1.** Looking at the network topology (Figure 4.4d), it is obvious that only N2 can activate the output as indicated by the strong excitatory connection from N2 to the output neuron, so the output neuron can only spike after N2 has activated. The moment when signal C is received, N2 can be activated only if two conditions are met: (i) N1 is inactive because N1 is strongly inhibiting N2 with a strong negative connection of weight -19.87, (ii) C is preceded by B, this is important because the connection from input B to N2 brings N2 to a higher sub-threshold value, ready to spike when C is received. Input signal B also excites N1 with a connection weight of 1.25 but cannot activate it due to the previous inhibition from input signal A in the correct order. Input signals received in orders other than ABC cannot activate N2 either because N1 is active (inhibiting N2) or N2 is at a lower sub-threshold value (when C is preceded by A) and therefore signal C cannot activate N2.

For a wrong pattern, when BC is preceded by B, the excitatory connection from B to N1 raises the membrane potential of N1 to a higher sub-threshold value, thereby, upon receiving the second B, N1 spikes and in turn strongly inhibits N2. Receiving a C at this moment will not activate N2, which would be required for the output neuron to spike. As a result, the output neuron remains silent for input pattern BBC. On the other hand, if BC is preceded by C, the strong excitatory connection from input C to N1, activates N1 which in turn suppresses N2, a B arriving at this point will maintain the spiking activity of N1, which results in extending the inhibition of N2. Due to the strong inhibition from N1, the last signal C cannot activate N2, and therefore the output neuron does not spike for CBC. Another possibility is when A follows C. The inhibitory connections from input A to N1 and N2 drop the membrane potential of both interneurons to a lower sub-threshold value. If a C is received after A, N1 activates faster than N2 (thanks to the strong excitatory connection from C to N1), and inhibits N2 before it manages to spike due to the excitatory connection from C to N2. Consequently, N2 never spikes for AC and the output neuron gracefully remains silent for input AC. It is important to note that the direct connection from C to N2 (2.52) is not enough to make it spike before the inhibition arrives through N1.

**Champion 4.** The perfect recogniser with the greatest number of states accomplished the pattern recognition task in a different way than the one with the least number of states (described above). Here, the output neuron receives a strong negative input from N1 and a positive connection from N2 (Figure 4.5). As a result, the output spikes only when N2 spikes but N1 does not spike. The positive connection from input B to N2 always induces two spikes in N2 after receiving B, whereas N2 is solely responsible for activating N1. Consequently, the membrane potential of N1 rises slowly, regardless of the following input signal, so N1 always spikes when the penultimate signal is B. However, if B is preceded by A in the correct pattern ABC then N1 does not spike due to the precise extra inhibition received from A in such a way that the positive contribution from N2 is unable to produce a spike in N1. On the other hand, the spiking of N1 in the case of BBC and CBC could be explained by the precise extra excitation received by N2. As a result, N2 spikes earlier in time, and activates N1 before the sub-threshold voltage drops.

Let us now consider two input sub-sequences, one ending with ...AABC and the other ending with ...ACBC. In the case of ...AABC, the network will transform to hA2 (000) state when AA is received, followed by hAB (002) state, finally the network transforms to hABC (100) state, and the ouput neuron spikes for receiving the correct input. On the contrary, a similar input ending with ...ACBC, upon receiving A, the network will transform to hA1 or hA2 state depending on the previous signal, followed by S1 (000) state AC, then the network switches to S2 (002) state ACB, and finally the network transforms to S0 (010) state ACBC.

In this example, it is important to note that regardless of the same preceding signal



|  | S0 | S1 | S2 | S3 | hA1 | hA2 | hAB | hABC |
|---|---|---|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| N2 spikes | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 |
| Input ends | BBC<br>CBC | AC<br>CC | CB | BB | BA | AA<br>CA | AB | ABC |

Figure 4.5: Analysis of champion 4, recognising a pattern of 3 signals in order ABC with 8 network states. (a) Spiking activity of the network. (b) Identified network state for each spiking pattern when the network receives different input signals. (c) Corresponding FST based on the network states identified in panel (b) for recognising a string of three letters ABC (d) Evolved spiking network for recognising ABC.

B in the sub-sequences AABC and ACBC, the network transformed to different network states when the final C is received i.e. hABC and S0. This shows that only the spiking pattern in the silent interval (signal length (4 ms) + silence length (8 ms) = 12 ms) is not enough to explain the transition from one network state to another. In addition to the spiking pattern, the network also relies on the history (preceding signals), the precise timing of spikes, and the momentary values of the two-state variables of a neuron, i.e. membrane potential $V$ and adaptation $w$. A feasible way to confirm the dependency on the precise timing of spikes is to investigate the robustness of these individuals to a noisy membrane potential and a variable duration of silent intervals.

### 4.2.4 Robustness

In order to assess the ability of the networks to function in a disturbed environment, the perfect recognisers were tested for slightly different lengths of signals and silent intervals than what they were evolved for (signal 4 ms and silent interval 8 ms). As expected, the networks (evolved without noise) were not robust to a small variation of 1 ms in the duration of either signals or silent intervals.

**Variation of silent intervals.** When the silent intervals were reduced from 8 ms to 7 ms, all the top 10 individuals stopped working. On the other hand, when silent intervals were extended to 9 ms, only two out of four perfect recognisers showed some robustness. Champion 4 was able to function almost perfectly with a reward outcome of 0.9979 and a penalty of 0.0000. Similarly, champion-1 continued to recognise ABC with a reward of 0.9995 but started responding to BBC with a penalty of 0.0002. Among the other six sub-optimal solutions three (champion 5, 6, and 9) were able to maintain functionality for extended intervals with rewards 0.9881, 0.9993, 0.8470 and penalties 0.0000, 0.0027, 0.0100, respectively.

**Variation of signals.** None of the top 10 individuals was robust to shortening (lengthening) of signals to 3 ms (5 ms). However, when a random step (1 ms) was turned off in this 5 ms duration of signals (01111, 10111, 11011, 11101, 11110), two individuals (champion 1 and 6) demonstrated some robustness with rewards 0.8938, 0.9246 and penalties of 0.0025, 0.0069, respectively.

**Disturbed neuronal parameters.** The functionality of the individuals broke down completely with a small variation of AdEx neuronal parameters, for example, a slight variation (-69.1 or -70.1 instead of -70 mV) of the $E_L$ parameter impaired the performance of all individuals. This abrupt performance degradation of individuals explains the necessity of introducing noise during evolution. It is therefore concluded that the networks evolved without variation (on input) or noise (in the network) are not robust to perturbation. Moreover, the performance of these networks dropped with extending the duration of silence which confirms that these networks are unable to maintain the network states when the silent intervals are prolonged. Before introducing noise during evolution in Chapter 5, the state maintenance of individuals is investigated in the following section.

### 4.2.5 Longer silent intervals

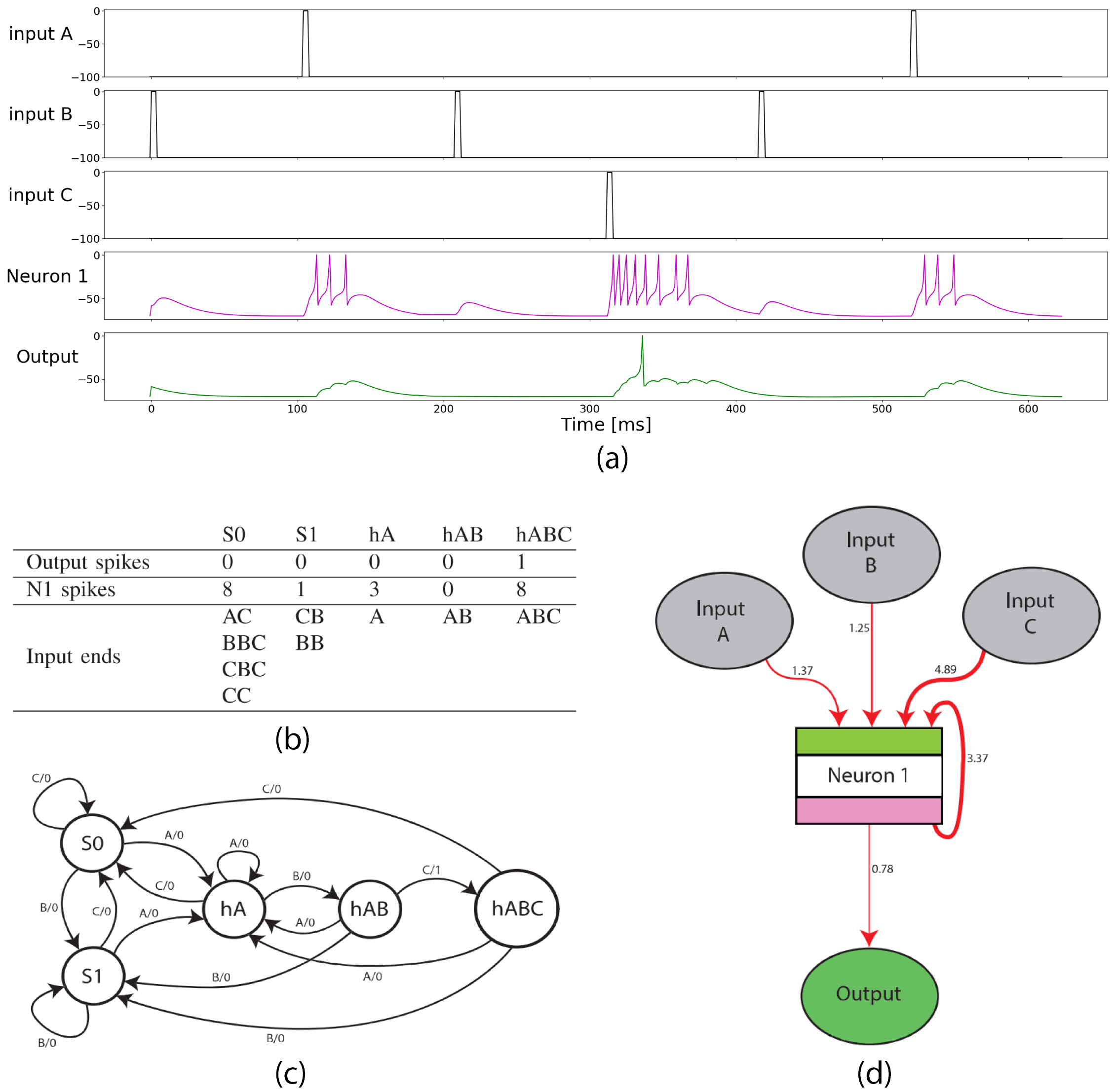
In order to promote state maintenance in the evolved networks, another set of experiments was performed with exactly the same settings except for an increased length of the silent interval from 8 ms to 100 ms. In addition to state maintenance, it was expected that the obtained solutions will mature unique network states for computational states of FST because each neuron will stabilise in the long period of silence. Out of 40 independent evolutionary runs with longer intervals of 100 ms, 10 ended with producing perfect recognisers (Table 4.2). This yield is much higher than the previous experiments with short intervals of 8 ms (where there were only 4 perfect recognisers). The network states were identified in a similar way as for short intervals: the spikes were counted in the silent interval of 100 ms window after all possible patterns of 3 signals (AAA to CCC).

Table 4.2: The number of states and top performing individuals evolved for recognising a pattern of 3 signals with longer intervals of 100 ms.

| | | | | network states | | | | |
|---|---|---|---|---|---|---|---|---|
| Champion | Reward | Penalty | Problematic sequences | Start | hA | hAB | hABC | Total |
| 1 | 1.0000 | 0.0000 | none | 3 | 1 | 1 | 1 | 6 |
| 2 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 3 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 4 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| 5 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 6 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 7 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| 8 | 1.0000 | 0.0000 | none | 3 | 1 | 1 | 1 | 6 |
| 9 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| 10 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |

In contrast to my expectations, the evolved solutions were unable to maintain network states, and a small variation of the silent interval substantially compromised the performance of the individuals. For example, 5 out of 10 individuals (champion 1, 4, 6, 7 and 10 ) sustained shortening the length of the interval to 99 ms with a reward of 1.0 and a penalty below 0.0005. The other 5 broke down completely down with reward 0, meaning none of the correct patterns was recognised. When the duration of the interval was further reduced to 90 ms only one individual (champion 10) showed some robustness with a reward 1.0 and a penalty 0.0008. All other individuals stopped working for 90 ms intervals except Champion 6 (with only one interneuron) which was able to recognise almost half of the ABCs in the sequence with a reward 0.45 and a penalty 0.0005. On the other hand, when intervals were extended from 1 ms to 101 ms, 8 out of 10 individuals showed robustness and were able to recognise all ABCs in the sequence. However, only 4 (champion 1, 6, 7, and 10) out of these 8 were flawless and never responded in any wrong interval with a penalty of 0.0. The other 4 (champion 2, 4, 5 and 7) started responding to other patterns with penalty 0.002. This result prevailed even when the intervals were increased to 110 ms. All 8 champions recognised all correct patterns in the input sequence with a reward 1.0 but responded to wrong patterns with penalty <0.002 as well. The individuals were then tested for disturbed input signals of the form (01111, 10111, 11011, 11101, 11110) – dropping a signal at random in the input sequence. Four out of ten individuals continued to recognise almost all ABC with rewards above 0.9980 but the output spiked wrongly after other patterns BBC and CBC with penalties below 0.0055. Moreover, the performance of all individuals dropped abruptly when the duration of input signal increased (decreased) to 5 ms (3 ms).

Thus, the hypothesis that introducing longer intervals during evolution will produce solutions that could maintain network states, invariant of the length of the silence intervals, did not hold. The networks were not very robust to the duration of the interval. Hence, the recognition happens based on the precise timing of spikes and momentary state variables in such a way that the output neuron spikes in the interval (100 ms) after receiving C in the correct input pattern ABC. Although introducing longer intervals between signals during evolution did not produce the expected solutions, an interesting solution (champion 6) was obtained with only one interneuron, which was not possible with a short interval between signals. Furthermore, this individual accomplished the recognition task with only 5 network stats (S0, S1, hA, hAB, and hABC ), where each state was represented by a unique spiking pattern (Figure 4.6c). All connections in the network were excitatory (Figure 4.6d). After receiving the correct first signal A, N1 spikes 3 times (hA 03). If this A is followed by input signal B, the network transforms to a quiescent state hAB represented by no activity in the network 00. Finally, if AB is followed by the last correct signal C, the only interneuron N1 spikes 8 times and activates the output neuron midway during spiking, recognising the correct pattern. However, if C is preceded by other patterns (AC, BBC, CBC, CC), N1 still spikes 8 times but does not activate the output neuron. This shows a strong dependence on the interplay between the state variables of neurons and the spike timings. Therefore, a very small change in the reversal potential $E_L$ from -70 mV to 70.1 or 69.9 mV broke down the network completely.



(a)

|  | S0 | S1 | hA | hAB | hABC |
|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 8 | 1 | 3 | 0 | 8 |
| Input ends | AC<br>BBC<br>CBC<br>CC | CB<br>BB | A | AB | ABC |

(b)

(c)

(d)

Figure 4.6: Analysis of champion 6, recognising a pattern ABC with 8 network states. (a) Spiking activity of the network. (b) Identified network state for each spiking pattern when the network receives different input signals. (c) Corresponding FST based on the network states identified in panel (b) for recognising a string of three letters ABC. (d) Evolved spiking network for recognising ABC.

## 4.3 Conclusion

The individuals generated with longer intervals were slightly robust to input variation, while the individual with short intervals could sustain functionality to small disturbance in parameters. However, both short and long intervals between the signals during evolution could not produce robust individuals that could maintain network states and sustain performance to a disturbed set of parameters. A previous study with a similar model suggested that evolving gene regulatory networks (GRNs) in the presence of noise produce robust individuals [145]. Although the authors in [145] evolve only GRNs in the presence of noise, it would be interesting to explore the benefits of introducing noise in SNNs during evolution. The next chapter investigates the evolution of spiking neural networks (SNNs) in the presence of noise on the membrane potential and variation of silent intervals in the input stream. The aim is to obtain networks that are: (i) robust to a disturbed set of parameters, and (ii) capable of maintaining network states indefinitely.

# Chapter 5

# Noise Promotes Robustness

Noise permeates biological nervous systems, yet they continue to function reliably in a disturbed environment. How biological neurons process information in the presence of noise and variability is a fundamental question in computational neuroscience. Experimental evidence exists that noise contributes to variability–a unique feature of behaviour. In fact, ”trial-to-trial” variability is the only trait that distinguishes behaviour from script. In the nervous system, variability emanates from both deterministic and non-deterministic sources. An example of the deterministic source is the initial state of the system (neuronal circuitry), i.e. a slight variation in the initial parameters results in a different response of the system. Noise – the non-deterministic source of variability in the nervous system – originates from many sources, including transduction of sensory input, synaptic transmission, and the number of ion channels [33]. This chapter addresses how introducing noise during evolution affects the evolvability of SNNs for a temporal pattern recognition task. Why is the presence of noise essential during evolution? And what are the potential benefits of noise? The aim is to obtain robust spiking networks that are resilient to break down in a disturbed environment.

This chapter contains two independent experimental setups. The first setup updates the genetic operators and introduces noise during evolution. Networks evolved in a noisy environment developed robustness to neuronal parameters. Then the robustness range of varying a single neuronal parameter (keeping others at their default values) is determined. The second experimental setup is a further refinement, especially in the way the fitness is calculated. This modification in the fitness function improved both yield and evolvability. Finally, the robustness range to varying all inter-dependent neuronal (AdEx) parameters is obtained.

## 5.1 Types of Noise

Noise originates from many sources in biological networks, including unreliable synapses, thermal variation and stochastic opening and closing of membrane channels. Biological networks not only maintain their functionality in the presence of noise and perturbation, but studies have also shown that some computational tasks are accomplished better in the presence of noise. This phenomenon is known as stochastic resonance [83]. Furthermore, the ability to operate in a noisy environment and trial-to-trial variability of biological neurons is intriguing and a salient feature of the nervous system.

In order to introduce stochasticity, two types of disturbance are introduced during evolution in the networks: intrinsic noise is introduced at the level of the neuron, modelled as random fluctuations of the membrane potential of neurons, whereas extrinsic noise is modelled as variations of the duration of silent intervals in the input stream. Although the evolutionary algorithm took longer to converge in the presence of noisen, the obtained networks were robust to perturbation and could maintain network states in the case of longer silent intervals.

This chapter focuses on evolution in the presence of intrinsic noise only, where noise is introduced during evolution as a random disturbance of the membrane potential of each neuron at every 1 ms network step. Since the membrane potential determines whether a neuron should spike or not (adaptive threshold) at a given network step, adding randomness to the membrane potential may prevent a neuron from spiking and defer the next spike. Similarly, a noisy membrane potential may trigger a spike earlier in time. Since information is carried by precise timing of spikes, it is important to investigate the impact of spike timing variability on the performance and evolvability of the networks.

During evolution, a random value of noise drawn from a normal distribution with mean 0 and standard deviation of 2 mV is added (at every 1ms step) to the membrane potential $V$ of each neuron in the network. This moderate level of noise is also observed in biological nervous systems [93, 27, 5, 35]. Moreover, another commonly used noise model, "Ornstein-Uhlenbeck", was also investigated and introduced in the membrane potential of each neuron during evolution. However, the evolutionary algorithm could not converge and no optimal solution (perfect recognizer) was produced. Moreover, the obtained sub-optimal solutions were neither robust to intrinsic noise nor extrinsic noise. In future studies, it would be interesting to explore the feasibility of introducing Ornstein-Uhlenbeck noise during evolution. All experiments in this work employ Gaussian noise on the membrane potential $V$ of neurons in the network.

## 5.2 Experimental Setup I

The artificial-evolution setup described in the previous chapter is updated for evolving SNNs in the presence of intrinsic noise. The initial generation is created in the same way from random genomes with a population size of 300. The population size is kept fixed with a binary tournament selection, and the top 10 out of 300 individuals are transferred to the next generation without undergoing genetic operators – elite count = 10. To obtain recognizer solutions in the presence of noise, the genetic operators are updated for better evolvability. The previous setup in Chapter 4 was unable to explore the search space due to a very low rate of mutation (0.01) and a high strength of mutation. Therefore, the rate of one-point mutation is increased from 0.01 to 0.1; if a genetic element is chosen for one-point mutation, its x, and y coordinates are moved by a distance drawn from a normal distribution with mean 0 and standard deviation 1 instead of 3 (in Chapter 4). The length (drawn from a geometric distribution with a mean of 10) and the probability of deletion (duplication) are kept the same at 0.001 (0.0005). A new genome level operator – multi-point crossover is introduced. In each generation, 10% (30 out of 300) individuals are subject to multi-point crossover.

**Multi-point crossover.** Two parental genomes A and B are chosen for recombination with binary tournament selection. The crossover begins at the first genetic element of the parents. Consider 2 pointers, each pointing to the first genetic element of parental genomes A and B. Elements are copied to the offspring with one of the two following schemes. The schemes are implemented by generating a uniform random number between 0 and 1. Scheme 1: the pointer advances on both parents; A (B) is chosen if the number is in the interval [0 0.03] ([0.03 0.06]). Scheme 2: the pointer advances on the chosen parent: A (B) is chosen if the number is in the interval [0.06 0.18] ([0.18 0.30]). The scheme remains the same if the number lies in [0.30 1]. Otherwise, one of the above two schemes is re-chosen according to the obtained random number.

In the absence of noise, two neurons were sufficient to recognize a pattern of 3 signals in a particular order. However, the evolutionary algorithm could not produce any solutions with two interneurons in the presence of noise. The minimal solution obtained had three interneurons. Therefore, we can say that evolution in a noisy environment requires more computational units (neurons) in the network.

**Fitness.** In the previous setup, the simple fitness (Equation 5.1) could not fully explore the search space due to two explanations: (i) the low mutation rate with an aggressive mutation strength, (ii) the lack of crossover operator may result in a homogeneous population. Thus, limiting the search space. As expected, when the multi-point crossover operator was introduced, and the mutation rate and strength were adjusted (described in Section 5.2), the complex fitness function (combining the two fitnesses used in Chapter 4, Equation 4.3) was no longer required. Therefore, the fitness function was updated to a simpler one (Equation 5.1), which was again based on reward and penalty. If the output neuron of the network spikes in the silence interval of 16 ms after the last signal of the pattern-to-be-detected, the individual was rewarded. Otherwise, the individual was penalized. A normalized reward R and penalty P were then calculated as the number of rewards (penalties) divided by the number of correct (incorrect) 16 ms silence intervals in the sequence.

$$f_{fitness} = 1 - (R - CP) \tag{5.1}$$

where

$$0 \leq R \leq 1$$

$$0 \leq P \leq 1$$

P is amplified by an arbitrary constant C (set to 4, optimized with grid search [1, 10]) because the number of incorrect intervals is a large number as compared to the number of target patterns in a randomly generated input sequence. Without the crossover operator and a high rate of mutation with low strength, C = 4 did not work in the noise-free setup.

In the absence of the penalty coefficient, the evolutionary algorithm is biased and produced over-recognizers that respond to other patterns in addition to the target pattern. In later generations during evolution, when the normalized P approaches 0 (the number of incorrect 16 ms silence intervals in the sequence remains the same), the evolutionary algorithm could not penalize the remaining wrong patterns enough. Therefore, the penalty P needs to be amplified. Although in theory, the normalized P can get a value of 1, in practice P is always a very small number, so even with amplification, the fitness value (Equation 5.1) ranges between 0 and 1, where 0 means a perfect individual.

### 5.2.1 Evolution

Each individual in the population was evaluated for 6 input sequences of 500 signals each. Four out of six sequences were generated randomly with an equally likely occurrence of A, B and C. The remaining 2 sequences were created by concatenating four patterns ABA, ABB, ABC, and BBC (3 closer to the target and one target pattern). In this set of experiments, the duration of a signal was increased to 6 ms followed by a silence of 16 ms, with the intuition that in the presence of noise with increased silent intervals the individuals may transform smoothly from one network state to the next.

In addition to the above modifications, to obtain networks with small weights of the connections, the excitatory and inhibitory gains were increased to 9 nS (instead of 5 nS in Chapter 4). Moreover, the offset current was removed from the output neuron because now the noise will account for sub-threshold activity in the output neuron. With these adjustments in the experimental setup, the yield and the evolvability improved: out of 100 independent runs without noise 33 ended up with fitness value 0 (perfect recognizers) in less than 500 generations. However, in the presence of intrinsic noise (mean 0 and SD 2 mV), at least 1000 generations were required to obtain 10 champions in 100 evolutionary runs. Due to noise, when the champions were re-evaluated on a large sequence (100000 signals), the fitness remained close to zero. Moreover, when re-evaluated without noise, the champions proved to be perfect recognizers by identifying the correct patterns and remaining silent for others.

### True Positive Rate

The true-positive rate (TPR) is defined as how often a network responds to the target pattern in a random sequence. The TPR is 1 if all occurrences of the target patterns are correctly identified (spiking of the output neuron in the correct interval) . On the other hand if the output neuron does not respond to any of the target patterns in the sequence then the TPR is 0. The TPR is the same as the normalized reward R in the fitness equation 5.1.

$$TRP \quad = \quad \frac{identified \quad target \quad patterns}{number \quad of \quad occurrences \quad of \quad the \quad target \quad pattern} \tag{5.2}$$

### False Discovery Rate

The false discovery rate (FDR) is the ratio between the number of times the network responded incorrectly and the total number of times the network responded. The FDR is given by the following expression:

$$FDR = \frac{number \quad of \quad times \quad the \quad network \quad responded \quad incorrectly}{total \quad number \quad of \quad times \quad the \quad network \quad responded} \tag{5.3}$$

The false discovery rate can be re-written as:

$$FDR = \frac{FP}{FP + TP} \tag{5.4}$$

where FP false positive is the number of times the network responds to incorrect patterns, and TP true positive is the number of times the network responds to the target patterns. The FDR is 0 if the network does not respond to any incorrect pattern whereas FDR is 1, if the network does not identify the target pattern and responds to at least one incorrect pattern.

### 5.2.2 Robustness to Varying a Single Neuronal Parameter

The individuals produced in the noisy evolutionary setup could sustain functionality to a disturbed set of neuronal parameters. To assess the robustness, a range around the default values of the neuronal parameters was explored in which the individual was robust. First, the range of robustness was obtained for changing a single neuronal parameter at a time while keeping all others at their default value. The range of robustness was defined as the largest range around the default value of each parameter for which the true positive rate (TPR) was above 0.99 and the false discovery rate (FDR) was below 0.05. Next, the range of robustness for varying all neuronal parameters at once was obtained such that an individual was robust to a random set of initial parameters drawn from the robustness range. As expected, the networks evolved in the presence of intrinsic noise were more robust to disturbed

Table 5.1: **Robustness of networks evolved without noise**. Robustness range of neuronal parameters, gain (E and I), properties of input and the noise itself. The values in square brackets show the range of robustness, the values above them show relative robustness.

| | | Top 10 individuals **evolved without noise** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P | Default | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $E_L$ | -70 mV | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 | 0.00 | 0.04 | 0.00 | 0.07 |
| | | [-70, -70] | [-70, -70] | [-70, -70] | [-70, -70] | [-70, -70] | [-70, -69] | [-70, -70] | [-70, -69] | [-70, -70] | [-70, -68] |
| $V_r$ | -58 mV | 0.00 | 0.00 | 0.13 | 0.04 | 0.08 | 0.00 | 0.13 | 0.13 | 0.00 | 0.13 |
| | | [-58, -58] | [-58, -58] | [-58, -55] | [-58, -57] | [-59, -57] | [-58, -58] | [-60, -57] | [-60, -57] | [-58, -58] | [-59, -56] |
| $V_T$ | -50 mV | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.20 |
| | | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -50] | [-50, -49] |
| $\tau_m$ | 20 ms | 0.01 | 0.00 | 0.04 | 0.03 | 0.01 | 0.03 | 0.01 | 0.06 | 0.01 | 0.02 |
| | | [19, 20] | [20, 20] | [19, 23] | [19, 22] | [19, 20] | [19, 22] | [19, 20] | [16, 21] | [20, 21] | [19, 21] |
| $\Delta_T$ | 2 mV | 0.07 | 0.00 | 0.07 | 0.07 | 0.00 | 0.00 | 0.00 | 0.13 | 0.00 | 0.20 |
| | | [1.9, 2.0] | [2.0, 2.0] | [1.9, 2] | [1.9, 2.0] | [2.0, 2.0] | [2.0, 2.0] | [2.0, 2.0] | [2.0, 2.2] | [2.0, 2.0] | [1.9, 2.2] |
| $C$ | 0.2 nF | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.22 |
| | | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.2, 0.2] | [0.19, 0.21] |
| $a$ | 2 nS | 0.00 | 0.00 | 0.03 | 0.00 | 0.00 | 0.03 | 0.03 | 0.03 | 0.00 | 0.10 |
| | | [2, 2] | [2, 2] | [1, 2] | [2, 2] | [2, 2] | [1, 2] | [1, 2] | [2, 3] | [2, 2] | [2, 5] |
| $b$ | 0 pA | 0.02 | 0.00 | 0.04 | 0.13 | 0.13 | 0.06 | 0.11 | 0.13 | 0.02 | 0.13 |
| | | [0, 1] | [0, 0] | [0, 2] | [0, 7] | [0, 7] | [0, 3] | [0, 6] | [0, 7] | [0, 1] | [0, 7] |
| $\tau_E$ | 5 ms | 0.00 | 0.00 | 0.18 | 0.00 | 0.00 | 0.00 | 0.00 | 0.18 | 0.00 | 0.18 |
| | | [5.0, 5.0] | [5.0, 5.0] | [5.0, 5.2] | [5.0, 5.0] | [5.0, 5.0] | [5.0, 5.0] | [5.0, 5.0] | [4.9, 5.1] | [5.0, 5.0] | [5.0, 5.2] |
| $\tau_I$ | 5 ms | 0.60 | 0.00 | 0.30 | 0.25 | 0.10 | 0.00 | 0.45 | 1.00 | 0.05 | 0.20 |
| | | [4.6, 5.8] | [5.0, 5.0] | [4.5, 5.1] | [4.8, 5.3] | [4.9, 5.1] | [5.0, 5.0] | [4.5, 5.4] | [4.1, 6.1] | [4.9, 5.0] | [4.7, 5.1] |
| $E_E$ | 0 mV | 0.00 | 0.00 | 0.08 | 0.00 | 0.00 | 0.00 | 0.00 | 0.08 | 0.00 | 0.25 |
| | | [0, 0] | [0, 0] | [0, 1] | [0, 0] | [0, 0] | [0, 0] | [0, 0] | [-1, 0] | [0, 0] | [0, 3] |
| $E_I$ | -70 mV | 0.56 | 0.00 | 0.11 | 0.22 | 0.00 | 0.00 | 0.44 | 1.00 | 0.11 | 0.11 |
| | | [-73, -68] | [-70, -70] | [-70, -69] | [-71, -69] | [-70, -70] | [-70, -70] | [-75, -69] | [-74, -65] | [-70, -69] | [-70, -69] |
| $gain_E$ | 9 nS | 0.00 | 0.00 | 0.05 | 0.10 | 0.00 | 0.00 | 0.00 | 0.05 | 0.05 | 0.30 |
| | | [9.0, 9.0] | [9.0, 9.0] | [9.0, 9.1] | [8.9, 9.1] | [9.0, 9.0] | [9.0, 9.0] | [9.0, 9.0] | [8.9, 9.0] | [9.0, 9.1] | [9.0, 9.6] |
| $gain_I$ | 9 nS | 0.28 | 0.00 | 0.11 | 0.14 | 0.12 | 0.01 | 0.26 | 1.00 | 0.05 | 0.09 |
| | | [8.6, 11.1] | [9.0, 9.0] | [8.4, 9.3] | [8.3, 9.6] | [8.4, 9.4] | [9.0, 9.1] | [8.6, 10.8] | [7.1, 15.6] | [8.6, 9.0] | [8.3, 9.1] |
| *noise* | 0 mV | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 |
| | | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] | [0, 0.1] |
| *silence* | 16 ms | 0.00 | 0.00 | 0.11 | 0.05 | 0.00 | 0.11 | 0.21 | 0.16 | 0.00 | 0.05 |
| | | [16, 16] | [16, 16] | [15, 18] | [16, 17] | [16, 16] | [15, 17] | [16, 20] | [15, 18] | [16, 16] | [15, 16] |
| *signal* | 6 ms | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| | | [6, 6] | [6, 6] | [6, 6] | [6, 6] | [6, 6] | [6, 6] | [6, 6] | [6, 8] | [6, 6] | [6, 6] |
| ARR | | 0.09 | 0.00 | 0.07 | 0.06 | 0.03 | 0.02 | 0.10 | 0.30 | 0.02 | 0.14 |

neuronal parameters than the ones evolved without noise (Table 5.1 and 5.2, see Figure A.1.1 and Figure A.1.2 in Appendix A.1 for information on the connection weights of individuals, see Appendix A.4 for network topologies of all individuals in Table 5.1 and 5.2).

The asymmetric robustness range for a single parameter was determined by extending the range at both sides while keeping all other parameters at their default values. The largest range around the default value in which the TPR was above 0.99 and FDR was below 0.05 for each parameter is given in Table 5.2. Similarly, the ro-

Table 5.2: **Robustness of networks evolved with noise**. Robustness range of neuronal parameters, gain (E and I), properties of input and the noise itself. The values in square brackets show the range of robustness, the values above them show relative robustness.

| | | Top 10 individuals **evolved with noise** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P | Default | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $E_L$ | -70 mV | 0.77 | 1.00 | 0.48 | 0.61 | 0.68 | 0.77 | 0.58 | 0.29 | 0.48 | 0.32 |
| | | [-81, -57] | [-91, -60] | [-76, -61] | [-78, -59] | [-86, -65] | [-88, -64] | [-78, -60] | [-75, -66] | [-77, -62] | [-73, -63] |
| $V_r$ | -58 mV | 0.42 | 0.79 | 0.46 | 0.71 | 0.50 | 0.46 | 0.21 | 0.79 | 0.58 | 1.00 |
| | | [-63, -53] | [-63, -44] | [-63, -52] | [-62, -45] | [-60, -48] | [-60, -49] | [-60, -55] | [-64, -45] | [-61, -47] | [-65, -41] |
| $V_T$ | -50 mV | 0.40 | 1.00 | 0.40 | 0.80 | 0.80 | 0.80 | 0.80 | 0.40 | 0.60 | 0.20 |
| | | [-51, -49] | [-53, -48] | [-51, -49] | [-52, -48] | [-51, -47] | [-51, -47] | [-52, -48] | [-52, -50] | [-51, -48] | [-51, -50] |
| $\tau_m$ | 20 ms | 0.39 | 1.00 | 0.95 | 0.55 | 0.35 | 0.42 | 0.53 | 0.17 | 0.96 | 0.22 |
| | | [8, 44] | [6, 99] | [12, 100] | [8, 59] | [6, 39] | [7, 46] | [8, 57] | [10, 26] | [11, 100] | [9, 29] |
| $\Delta_T$ | 2 mV | 0.28 | 0.89 | 1.00 | 0.83 | 0.83 | 0.83 | 0.83 | 0.44 | 0.89 | 0.28 |
| | | [1.7, 2.2] | [1.3, 2.9] | [1.5, 3.3] | [1.3, 2.8] | [1.6, 3.1] | [1.7, 3.2] | [1.5, 3.0] | [1.4, 2.2] | [1.6, 3.2] | [1.6, 2.1] |
| $C$ | 0.2 nF | 0.40 | 1.00 | 0.50 | 0.50 | 0.50 | 0.60 | 0.30 | 0.40 | 0.60 | 0.20 |
| | | [0.17, 0.21] | [0.15, 0.25] | [0.17, 0.22] | [0.18, 0.23] | [0.17, 0.22] | [0.17, 0.23] | [0.19, 0.22] | [0.16, 0.20] | [0.17, 0.23] | [0.19, 0.21] |
| $a$ | 2 nS | 0.60 | 0.87 | 0.37 | 0.87 | 1.00 | 0.87 | 0.90 | 0.27 | 0.40 | 0.73 |
| | | [-4, 14] | [-7, 19] | [-4, 7] | [-5, 21] | [-2, 28] | [-5, 21] | [-8, 19] | [-1, 7] | [-3, 9] | [-9, 13] |
| $b$ | 0 pA | 0.60 | 0.74 | 0.28 | 0.79 | 1.00 | 0.68 | 0.82 | 0.28 | 0.47 | 0.39 |
| | | [0, 34] | [0, 42] | [0, 16] | [0, 45] | [0, 57] | [0, 39] | [0, 47] | [0, 16] | [0, 27] | [0, 22] |
| $\tau_E$ | 5 ms | 0.73 | 0.91 | 0.64 | 1.00 | 0.64 | 0.73 | 0.91 | 0.45 | 0.55 | 0.64 |
| | | [4.7, 5.5] | [4.7, 5.7] | [4.6, 5.3] | [4.4, 5.5] | [4.5, 5.2] | [4.6, 5.4] | [4.4, 5.4] | [4.9, 5.4] | [4.7, 5.3] | [4.9, 5.6] |
| $\tau_I$ | 5 ms | 0.43 | 0.57 | 1.00 | 0.95 | 1.00 | 0.76 | 0.43 | 0.29 | 0.71 | 0.43 |
| | | [4.5, 5.4] | [4.6, 5.8] | [4.5, 6.6] | [4.5, 6.5] | [4.8, 6.9] | [4.8, 6.4] | [4.7, 5.6] | [4.8, 5.4] | [4.6, 6.1] | [4.5, 5.4] |
| $E_E$ | 0 mV | 0.50 | 0.83 | 0.75 | 0.92 | 1.00 | 0.83 | 0.75 | 0.58 | 0.67 | 0.50 |
| | | [-2, 4] | [-4, 6] | [-4, 5] | [-4, 7] | [-8, 4] | [-6, 4] | [-5, 4] | [-1, 6] | [-4, 4] | [0, 6] |
| $E_I$ | -70 mV | 0.33 | 0.56 | 0.89 | 0.78 | 0.67 | 0.56 | 0.33 | 0.33 | 0.44 | 0.22 |
| | | [-71, -68] | [-72, -67] | [-76, -68] | [-74, -67] | [-75, -69] | [-74, -69] | [-72, -69] | [-71, -68] | [-73, -69] | [-70, -68] |
| $gain_E$ | 9 nS | 0.50 | 1.00 | 0.75 | 0.95 | 0.95 | 0.85 | 0.75 | 0.65 | 0.80 | 0.50 |
| | | [8.6, 9.6] | [8.2, 10.2] | [8.2, 9.7] | [8.3, 10.2] | [7.7, 9.6] | [7.9, 9.6] | [8.1, 9.6] | [8.8, 10.1] | [8.2, 9.8] | [8.9, 9.9] |
| $gain_I$ | 9 nS | 0.21 | 0.40 | 0.92 | 0.46 | 0.46 | 0.35 | 0.28 | 0.26 | 0.40 | 0.15 |
| | | [8.0, 9.8] | [7.2, 10.6] | [7.9, 15.7] | [7.1, 11] | [8.2, 12.1] | [8.2, 11.2] | [7.7, 10.1] | [7.7, 9.9] | [7.7, 11.1] | [8.0, 9.3] |
| $noise$ | 2 mV | 0.85 | 1.00 | 0.93 | 0.85 | 0.78 | 0.78 | 0.89 | 0.74 | 0.93 | 0.74 |
| | | [0, 2.5] | [0, 2.9] | [0, 2.7] | [0, 2.5] | [0, 2.3] | [0, 2.3] | [0, 2.6] | [0, 2.2] | [0, 2.7] | [0, 2.2] |
| $silence$ | 16 ms | 0.95 | 0.25 | 0.25 | 1.00 | 0.20 | 0.20 | 0.70 | 0.55 | 0.20 | 0.60 |
| | | [13, 32] | [14, 19] | [13, 18] | [10, 31] | [12, 16] | [13, 17] | [11, 25] | [12, 23] | [14, 18] | [10, 22] |
| $signal$ | 6 ms | 0.50 | 0.50 | 0.50 | 1.00 | 1.00 | 0.50 | 0.50 | 0.00 | 1.00 | 0.00 |
| | | [6, 7] | [5, 6] | [6, 7] | [5, 7] | [4, 6] | [5, 6] | [5, 6] | [6, 6] | [6, 8] | [6, 6] |
| ARR | | 0.52 | 0.78 | 0.65 | 0.80 | 0.73 | 0.65 | 0.62 | 0.41 | 0.63 | 0.42 |

bustness range was obtained for the top 10 individuals without noise (Table 5.1). The value above the range is the relative robustness of a given parameter across the top 10 champions of both categories (evolution in the presence of noise and without noise). To calculate the relative robustness for a given parameter, the difference between the maximum and minimum value of each parameter (row) was divided by the maximum difference value across the 20 individuals (top 10 champions in each category). The last row of Table 5.1 and 5.2 shows the average relative robustness (ARR).

A comparison of the ARR values of top individuals in both categories clearly indicates that the individuals evolved in the presence of noise were more robust to the disturbed neuronal parameters and variation of the input signal. In the absence of noise individual number 8 (as the only individual) showed the highest robustness with an ARR value of 0.30. In contrast, the least (most) robust individual evolved in the presence of noise secured an ARR value of 0.41 (0.80). Likewise, the second-best individuals without noise obtained an ARR value of 0.13 while the remaining 8 were below 0.1.

Furthermore, unlike the networks evolved in the absence of noise, those evolved in the presence of intrinsic noise did not break abruptly beyond the robustness range. Instead, their performance degraded slowly. For example, Figure 5.1 shows the performance comparison of varying $E_L$ – the effective resting potential for the best individual with noise and without noise. Champion 4, the best individual evolved with noise secured an ARR value of 0.80. This network was robust to different values of $E_L$ between -78 and -59 (default -70 mV). Interestingly, the performance of the network degraded gracefully beyond this range (Figure 5.1 left). On the other hand, the functionality of the best individual (Champion 8) evolved without noise

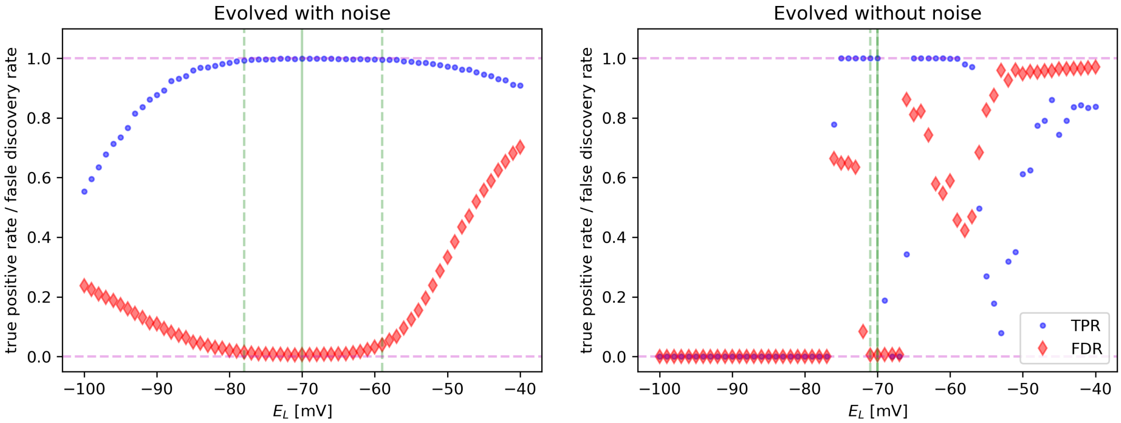


Figure 5.1: Performance comparison of the network evolved with noise and without noise to changes of parameter $E_L$ (Vrest). The range of robustness is shown by the dashed green lines, the default value is -70 mV. The network evolved (and tested) with noise (left; the network of champion 4) shows graceful degradation of network performance and a larger range of robustness than the network evolved (and tested) without noise (right; the most robust network (Champion 8) evolved without noise).

(ARR 0.30) broke down completely beyond the robustness range [-70, -69] (Figure 5.1 right).

A comparison of the performance to varying the silence interval between the signals showed that individuals obtained in the presence of intrinsic noise were also robust to longer silent intervals between signals. The default duration of silent intervals was 16 ms; the range was extended asymmetrically on both sides around the default value. The left(right) panel of Figure 5.2 shows the best individual obtained in the presence(absence) of noise. The robustness range was obtained by lengthening and shortening the silence interval around the default value of 16 ms such that the TPR(FDR) remained above(below) 0.99(0.05). The best individual evolved with noise (Champion 4) was robust to silent intervals between 10 and 31 ms. Furthermore, beyond this range, this individual showed robustness to indefinite lengthening of silent intervals with a slightly low value of TPR = 0.96, while the FDR remained below 0.05 (Figure 5.2 left). In contrast, the best individual evolved without noise

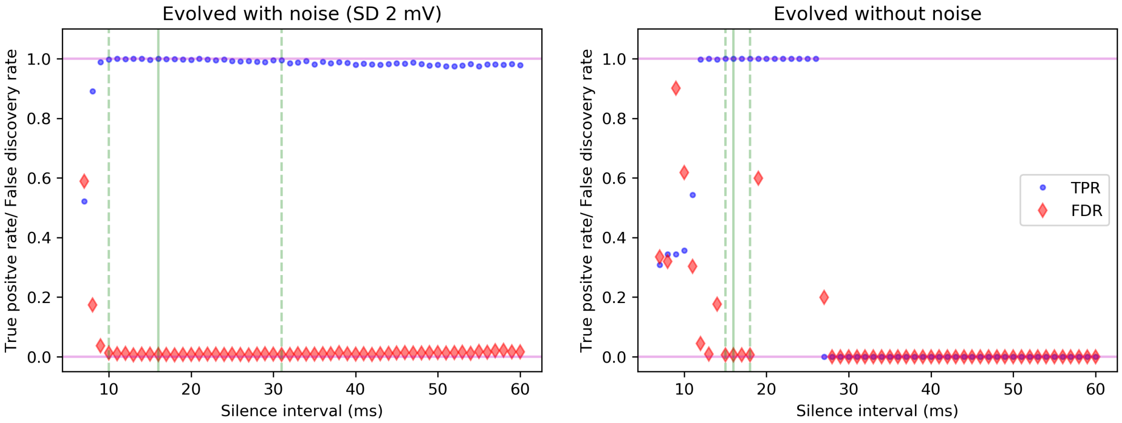


Figure 5.2: Performance comparison of the network evolved with noise and without noise to variation of silence interval *Silence*. The range of robustness is shown by the dashed green lines, the default duration is 16 ms. The network evolved (and tested) with noise (left; the network of champion 4) shows the performance of the network is hardly affected by increasing the duration of silence interval. The network evolved (and tested) without noise (right; the most robust network (Champion 8) evolved without noise is not robust to variation of intervals).

(Champion 8) was robust only to the duration of silences between [15, 18] ms and beyond this range, its performance deteriorated abruptly (Figure 5.2 right).

Robustness to silent intervals is essential for network states' maintenance. A network is capable to maintain states if it can recognize patterns with longer silent intervals between input signals. The presence of recurrent excitatory connections in the network explains the state maintenance – a form of memory. In short, evolution in the presence of noise also leads to the emergence of memory in spiking neural networks (see Figure A.1.3 in Appendix A.1 for information on the robustness of all top 10 individuals to all parameters).

## 5.3 Experimental Setup II

The analysis of the individuals obtained with the settings of Section 5.2 fostered three important modifications in the evolutionary algorithm. The most important modification is the way normalised reward (R) and penalty (P) are calculated in the fitness function (Equation 5.1). As the length of a signal is 6 ms followed by a silence interval of 16 ms, R is now calculated as the spiking of the output neuron in the interstimulus interval (ISI) after the onset of the last signal C in the target pattern ABC, divided by the total number of target patterns in the sequence. The ISI is the interval between the onsets of two consecutive signals (ISI = length of the signal + length of the following interval = 6 ms + 16 ms = 22 ms). Similarly, P is the number of interstimulus intervals (signal + silence = 22 ms) in which the output neuron spikes incorrectly, divided by the total number of such intervals (ISIs) in the sequence. This is in contrast to the previous fitness function where R and P were calculated only in the 16 ms silence interval. The second important update is reducing the length of the duplicated and deleted elements. In this setup the length of elements to be deleted or duplicated is drawn from a geometric distribution with a mean value of 6, previously it was 10 which resulted in unwanted lengthening or shortening of the genome; deleting a large portion of the genome may remove useful connections and neurons. Similarly duplicating a large portion can strengthen excessive connections. Furthermore, the excitatory and the inhibitory gains are fine-tuned to 7 nS with a grid search [5, 9] nS.

Since an input signal lasts for 6ms, it is possible for the output neuron of the network to spike while receiving the input. The previous fitness functions (used in Chapter 4 and Chapter 5, Section 5.2) did not take this into account. These modifications in the evolutionary setup, especially the way fitness is calculated, had a pronounced impact on the evolvability and yield. Without noise, out of 100 independent runs of 300 individuals, 81 ended up being perfect recognizers as compared to 33 in the previous setting (Section 5.2). In the presence of intrinsic noise, however, 100 independent runs yielded only 13 perfect recognizers with a maximum limit of 3 interneurons in the network during evolution. When this limit was increased to 4 interneurons, the number of perfect recognizers was raised to 19. Conversely, no solution was produced when the limit was reduced to 2 interneurons.

### 5.3.1 Robustness to Varying All Neuronal Parameters

In Section 5.2.2, the robustness range was obtained for varying a single parameter while keeping all others at their default values. However, this range did not provide any information about the robustness of the network when more than one parameter was disturbed. Therefore, the robustness range for varying all neuronal parameters was investigated in which the network was robust to any random set of parameters. Since the neuronal parameters are inter-dependent, changing the value of one parameter could constrict or relax room for other parameters. Due to this interdependence among neuronal parameters, finding a robustness range for varying all parameters at once was not straightforward.

### Range-extension Algorithm

A probabilistic algorithm is proposed to obtain the robustness range for varying all parameters at once. This algorithm runs in 2 steps: (i) the range of all parameters is extended in both directions around their default value by a small amount (1% of the default value), (ii) the extended range is evaluated, if the extension is accepted (TPR above 0.95 and FDR below 0.05), step 1 is repeated. Otherwise, the extension is reversed, and the parameter causing the break is identified (Algorithm 1).

To **evaluate the robustness-range**, one hundred random sets of neuronal parameters are drawn from the extended range and copies of the evolved network are created such that all neurons in a network are given one set of parameters. The range is accepted only if 90 out of 100 same networks, but with different parameter values have a TPR above 0.90 and FDR below 0.10 for a random input sequence of 50000 signals with equiprobable occurrence of signals.

To find the **parameter-causing-the-break**, the range of all parameters is extended one by one in the left or right direction, and the extended robustness range is evaluated. The extension of the parameter(s) rejecting the extended range in one of the two directions is stopped.

The algorithm stops when all parameters are excluded from extension in both directions, producing the robustness range. The network is robust to any set of neuronal parameters drawn from this range (Table 5.3).

**Algorithm 1** Robustness range algorithm for varying all neuronal parameters

**Input:** An evolved SNN for recognizing temporal patterns

**Output:** Robustness range for all neuronal parameters

```
procedure EXTEND RANGE
    while !converged do                          ▷ initially converged = $false$
        while extend_range do                  ▷ initially extend_range = $true$
            extend range in both directions
            sample 100 sets of random parameters within the range
            robust_sets_count=0
            for each set do
                if corresponding network is robust then
                    increment robust_sets_count
                end if
            end for
            if robust_sets_count<90 then
                shrink range of all parameters in both directions
                extend_range = $false$
            end if
        end while
                                                 ▷ find parameter causing breaking
        for each parameter do
            for each direction left or right do
                extend the range of parameter
                sample 100 sets of random parameters within the range
                robust_sets_count=0
                for each set do
                    if corresponding network is robust then
                        increment robust_sets_count
                    end if
                end for
                if robust_sets_count<90 then
                    shrink range of parameter
                    parameter extension_flag = $false$
                    extend_range = $true$
                end if
            end for
        end for
                                                              ▷ check convergence
        check_convergence = true
        for each parameter do
            for each direction left or right do
                if extension_flag = $true$ then
                    check_convergence = $false$
                end if
            end for
        end for
        if check_convergence == "true" then
            converged = $true$
        end if
    end while
end procedure
```

Table 5.3: Robustness range obtained by varying neuronal parameters at once. The values in square brackets show the range of robustness in the left and right directions. The table below shows the performance of individuals with 100 different sets of neuronal parameters, sampled from the ranges of robustness specific for each individual.

| | | Top 10 individuals | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P | Default | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| $E_L$ | -70 mV | [-71, -65] | [-74, -64] | [-72, -66] | [-72, -67] | [-73, -65] | [-73, -67] | [-72, -68] | [-71, -67] | [-72, -66] | [-72, -66] |
| $V_r$ | -58 mV | [-59, -56] | [-60, -57] | [-60, -56] | [-59, -55] | [-61, -56] | [-60, -56] | [-60, -56] | [-59, -55] | [-60, -54] | [-60, -55] |
| $V_T$ | -50 mV | [-51, -48] | [-52, -49] | [-52, -48] | [-51, -48] | [-52, -47] | [-52, -47] | [-52, -48] | [-51, -48] | [-51, -48] | [-52, -48] |
| $\tau_m$ | 20 ms | [16, 25] | [17, 21] | [18, 22] | [19, 22] | [18, 23] | [16, 22] | [18, 22] | [19, 22] | [18, 23] | [18, 22] |
| $\Delta_T$ | 2 mV | [1.5, 2.2] | [1.8, 2.3] | [1.8, 2.2] | [1.6, 2.4] | [1.8, 2.2] | [1.8, 2.3] | [1.8, 2.1] | [1.9, 2.2] | [1.8, 2.3] | [1.8, 2.3] |
| $C$ | 0.2 nF | [0.19, 0.22] | [0.16, 0.21] | [0.18, 0.22] | [0.19, 0.22] | [0.18, 0.22] | [0.18, 0.22] | [0.18, 0.21] | [0.19, 0.22] | [0.17, 0.23] | [0.17, 0.22] |
| $a$ | 2 nS | [-2, 7] | [-2, 5] | [0, 4] | [-2, 4] | [0, 5] | [0, 4] | [-1, 4] | [0, 5] | [1, 6] | [0, 5] |
| $b$ | 0 nA | [0, 1] | [0, 0] | [0, 2] | [0, 7] | [0, 7] | [0, 3] | [0, 6] | [0, 7] | [0, 1] | [0, 7] |
| $\tau_E$ | 5 ms | [4.9, 5.2] | [4.7, 5.3] | [4.8, 5.2] | [4.8, 5.2] | [4.8, 5.3] | [4.7, 5.2] | [4.8, 5.2] | [4.9, 5.3] | [4.9, 5.3] | [4.8, 5.2] |
| $\tau_I$ | 5 ms | [4.6, 5.6] | [4.5, 5.1] | [4.8, 5.2] | [4.9, 5.2] | [4.7, 5.6] | [4.7, 5.3] | [4.8, 5.2] | [4.9, 5.2] | [4.9, 5.3] | [4.8, 5.2] |
| $E_E$ | 0 mV | [-1, 2] | [-2, 3] | [-2, 2] | [-2, 2] | [-2, 2] | [-2, 5] | [-4, 1] | [-4, 4] | [-2, 4] | [-2, 3] |
| $E_I$ | -70 mV | [-71, -68] | [-73, -69] | [-72, -68] | [-71, -67] | [-72, -68] | [-72, -68] | [-72, -68] | [-71, -67] | [-73, -68] | [-72, -68] |
| $gain_E$ | 7 nS | [6.9, 7.2] | [6.7, 7.1] | [6.8, 7.2] | [6.9, 7.3] | [6.8, 7.5] | [6.7, 7.3] | [6.8, 7.1] | [6.9, 7.3] | [6.9, 7.3] | [6.8, 7.3] |
| $gain_I$ | 7 nS | [6.9, 7.3] | [6.6, 7.1] | [6.8, 7.2] | [6.8, 7.3] | [6.8, 7.4] | [6.9, 7.5] | [6.8, 7.6] | [6.9, 7.2] | [6.8, 7.4] | [6.8, 7.3] |
| | | Test: when each neuron in the network has a different set of parameters. | | | | | | | | | |
| TPR>0.99 & FDR<0.01 | | 37 | 37 | 27 | 80 | 19 | 53 | 47 | 52 | 67 | 14 |
| TPR>0.95 & FDR<0.05 | | 71 | 79 | 75 | 97 | 68 | 99 | 93 | 98 | 93 | 80 |
| TPR>0.90 & FDR<0.10 | | 84 | 86 | 86 | 100 | 85 | 99 | 97 | 99 | 99 | 91 |
| | | Test: when the silence interval between signals is increased. | | | | | | | | | |
| *Silence* | 16 ms | $\geq 100$ | 35 | $\geq 100$ | 28 | $\geq 100$ | $\geq 100$ | 48 | $\geq 100$ | $\geq 100$ | $\geq 100$ |

The last row in Table 5.3 shows robustness to the duration of silence intervals between signals. Out of 10 individuals, 7 were robust to an indefinite increase in the silence interval. In other words, these 7 networks were capable of maintaining network states indefinitely in the absence of input activity (see Figure A.1.4 in Appendix A.1 for more information on the connections weights of the in the individuals in Table 5.3, see Appendix A.4 for network topologies of all individuals presented in Table 5.3).

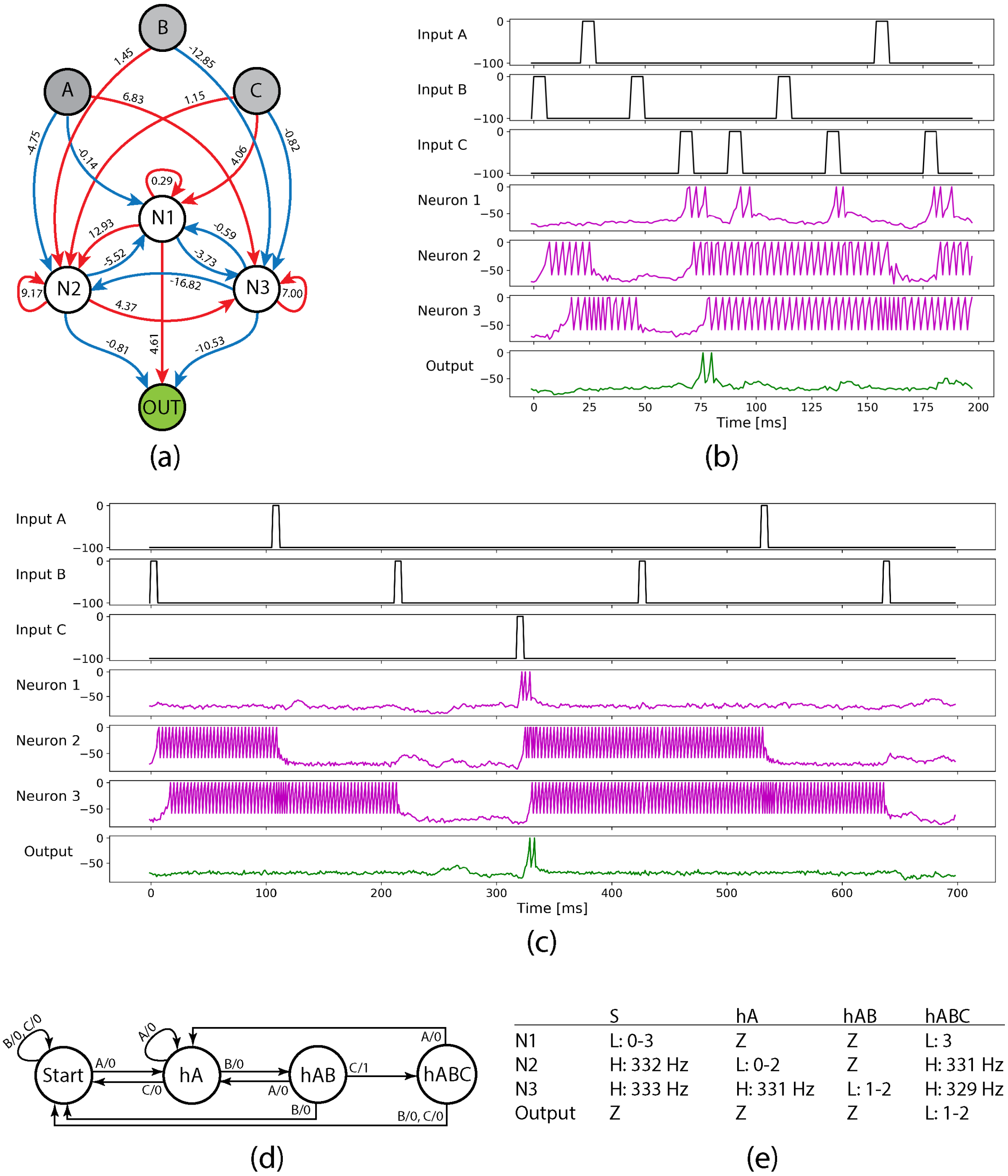
To further test the robustness of individuals, each neuron in the network was given a different set of neuronal parameters drawn from the obtained robustness range. Afterwards, the TPR and FDR were reported for a random sequence of 50000 signals. One hundred such evaluations were made for the top-10 networks and the TPR and FDR were reported (Table 5.3). Individual numbers 4, 6, 7, 8 and 9 showed robustness to this change.

To understand the working mechanism of the networks, the most robust individual was explored further. Although individual no. 4 was the most resilient to the disturbed neuronal parameters, it could not maintain network states when the silent intervals between signals were increased. Therefore, individual 9 was chosen for further analysis. This individual was robust to disturbed set of neuronal parameters, and it could maintain performance to an indefinite increase in the silent intervals.

A minimal finite-state transducer FST for recognizing a pattern of three signals requires at least four network states (Figure 5.3c). To explore how the evolved network accomplished the task in the presence of intrinsic noise, first the network states were identified. Unlike individuals obtained without noise in Chapter 4, all individuals (top 10) performed the task with only 4 network states that could be mapped onto the states of minimal FST. Furthermore, 7 out of 10 individuals were able to maintain network states when the interval between signals was increased.

With the initial analysis, it was observed that in the presence of intrinsic noise all the networks evolved self-excitatory loops. For example, all interneurons of individual no. 9 (Figure 5.3a) had self-excitatory loops (two strong and one weak) and were fully connected. However, this was not a sufficient condition for the emergence of memory (state maintenance) in the network. As a counter-example, individual no. 7 shared a similar topology with two strong self-excitatory loops but could not maintain network states. The network topology of individual no. 7 is provided in Appendix A.1, Figure A.1.5.

To explain the working mechanism of an evolved network, the spiking behaviour of the network is observed in one of the four (start, hA, hAB, hABC) possible network states. The spiking state of neurons is classified as one of the three identified states: a low spiking state (denoted by L) 0 to 3 spikes, a high spiking state (denoted by H) approx. 330 Hz, no spiking activity (denoted by Z). Using this notation the four network states (start, hA, hAB, hABC) of individual no. 9 are described as (LHH, ZLH, ZZL, LHH). In each triplet, a symbol (Z, L, or H) represent the state of a



| | S | hA | hAB | hABC |
|---|---|---|---|---|
| N1 | L: 0-3 | Z | Z | L: 3 |
| N2 | H: 332 Hz | L: 0-2 | Z | H: 331 Hz |
| N3 | H: 333 Hz | H: 331 Hz | L: 1-2 | H: 329 Hz |
| Output | Z | Z | Z | L: 1-2 |

Figure 5.3: Analysis of individual no. 9 evolved to recognize ABC in the presence of intrinsic noise. (a) Topology and connection weights of the network. (b) The activity of the network for short 16 ms, and (c) long silent intervals. (d) Minimal FST for recognizing a pattern of 3 letters. (e) The identified network states.

neuron, and the order corresponds to the order of interneurons in the network.

The network remains in the start state unless the first correct signal A is received. The start state of the network is represented by high spiking of N2 and N3, and low spiking of N1. The high spiking of N2 and N3 can be explained by strong excitatory connections from B and C to N2, which in turn excites N3. Both N2 and N3 sustain the spiking activity by exciting themselves with strong excitatory loops. Therefore, the network remains in the start state even in the case of long silent intervals. When the network receives the first signal A in the correct order, the negative connection from input A to N2 inhibits the high state of N2, switching it to a low state. At the same time, the excitatory connection from A to N3 speeds up N3 for the input duration which gets back to the normal H state. Hence, the hA state is maintained by continuous spiking of N3 even if the silent intervals are increased to 100 ms (Figure 5.3d), thanks to the strong self-excitatory connection on N3.

The next correct state hAB is maintained by an absence of activity in the network that can be explained by the topology of the network (Figure 5.3a). Assume the network transforms to the hA state after receiving A, represented by continuous spiking of N3. If the network receives an input signal B while in state A, the inhibitory connection to N3 shuts down N3 and transforms it to the L state. On the other hand, the weak excitatory connection from B to N2 is unable to produce any spikes in N2. However, a second B can activate N2 which in turn actives N3, and transforms the network into the start state. Furthermore, if input A is followed by B, the network goes into a quiescent state (hAB state) and the inhibition from the output neuron is released. At this stage, if the network gets a C, the strong excitatory connection from C to N1 activates N1 which triggers the output neuron to fire for the correct pattern ABC. Moreover, N1 also excite N2 which activates N3, putting the network back into the start state (H state of N2 and N3 neurons).

## 5.4 Conclusion

Noise induces robustness. Introducing intrinsic noise during evolution as random fluctuations of membrane potential of neurons, is beneficial and has a computational role. Consequently, the obtained networks are robust to a disturbed set of neuronal parameters. Furthermore, the networks are more likely to maintain network states, which implies that noise also facilitates the emergence of memory in these minimal networks. The memory appears to be stored in the self-excitatory loops. Thus the formation of self-loops during evolution is linked to the emergence of memory in the network. To quantify the robustness, first, the robustness range is obtained for changing a single parameter and keeping all others at their default values. Furthermore, a probabilistic algorithm is proposed to find the robustness range for varying all parameters at once. It has been demonstrated that the individuals evolved in the presence of intrinsic noise are robust to disturbed neuronal parameters sampled from the obtained range.

# Chapter 6

# Dynamics of Evolved SNNs

In Chapter 4 and Chapter 5, the experimental setup for the evolution of SNNs to perform temporal pattern recognition was refined with fine-tuning of the genetic operators and the fitness function. As a result, robust SNNs were obtained. However, the working mechanism of the evolved networks is not yet obvious. For example, in the presence of noise, why are some networks capable of maintaining network states but not all? Why do two networks with the same structure but different connection weights exhibit inconsistent behaviour when the silent intervals between signals are increased? What is the contribution of each connection in recognition? Are there any superfluous connections in the network? Is a weaker connection less significant than a stronger one? What is the role of self-loops (autapses)? This chapter focuses on understanding the working mechanism of the evolved networks.

In this chapter, a connection pruning algorithm is proposed which removes superfluous connections from the network to pronounce the essential connections. Afterwards, the network states of the pruned network are mapped onto the states of a finite state transducer (FST). Successively, the emergence of memory and the switching mechanism of the pruned networks are explored.

## 6.1 Experimental Setup

To understand, why memory emerges in some networks but not all, two different experiments were conducted each with 100 independent evolutionary runs of 300 individuals. The setup for the first one was kept similar to the one used in Chapter 5, Experimental setup II. In the second experiment, the duration of silent intervals between input signals was kept variable in the interval [16 32] ms during evolution instead of fixed intervals of 16 ms. The intuition behind introducing variable silent intervals during evolution was to obtain state-maintaining individuals. Moreover, the level of intrinsic noise remained the same for both experiments; a random value drawn from Gaussian distribution with a mean of 0 and a standard deviation of 2 mV, was added to the membrane potential of each neuron at every network step. Out of 100 independent runs, the experiment with constant silent intervals of 16 ms produced 15 perfect recognisers. In the case of variable silent intervals [16 32] ms, 12 out of 100 runs ended with producing perfect recognisers. Although the experimental setup was capable of deleting neurons during evolution, the champions of both experiments had three interneurons. Furthermore, the individuals evolved with variable silent intervals had a slightly higher number of connections as compared to the ones evolved with constant silent intervals (mean 19.20, standard deviation 0.54 vs mean 18.83, standard deviation 1.333) as shown in Table 6.1. Despite the structural dissimilarities among the individuals obtained with the two setups, the individuals evolved with variable silent intervals were more likely to maintain network states. Out of 12 champions, 11 were capable of keeping memory, in contrast to fixed silent intervals, where only 4 out of 15 were able to maintain states for longer duration of silent intervals. Furthermore, regardless of the experiment, all the state-maintaining individuals had exactly 2 self-excitatory loops. However, this was not a sufficient condition for state maintenance. As a counter-example, 5 individuals (4, 7, 11, 12,

14) formed 2 self-excitatory loops yet they were unable to sustain states in the absence of input activity. This question will be revisited after examining the evolved networks for excessive connections. The connections weights of all individuals presented in Table 6.1 are provided in Appendix A.2, Figure A.2.1 and Figure A.2.2.

## 6.2 Pruning Excessive Connections

Evolution may produce superfluous connections which can be pruned without impairing the performance of the network [147]. Evolved networks are pruned by removing a random connection and evaluating the network for a random sequence of length 10000. If the TPR value drops below 0.95 and FDR rises above 0.05, the connection is restored and is labelled as vital. Otherwise the excessive connection is removed from the network. This process is repeated until all connections in the network are labelled as vital. With this procedure, all the top individuals (15 + 12) are pruned. It is important to mention that connections are pruned irrespective of their weight contribution. A connection with a large weight can be pruned if it is not vital i.e. the removal of this connection from the network does not affect the network's performance ($TPR \geq 0.95 and FDR \leq 0.05$).

Although the pruning algorithm allowed the deletion of nodes, none of the top individuals was left with less than 3 interneurons. Hence, in the presence of noise, at least 3 interneurons were required to recognise a pattern of three signals. Furthermore, the individuals capable of maintaining network states continued to maintain network states after pruning. This indicated that the superfluous connections were neither needed for recognition nor for state maintenance. Both networks (evolved and the pruned) performed equally well when they were re-evaluated for a random sequence of length 100000, where the duration of each signal was 6 ms signal followed

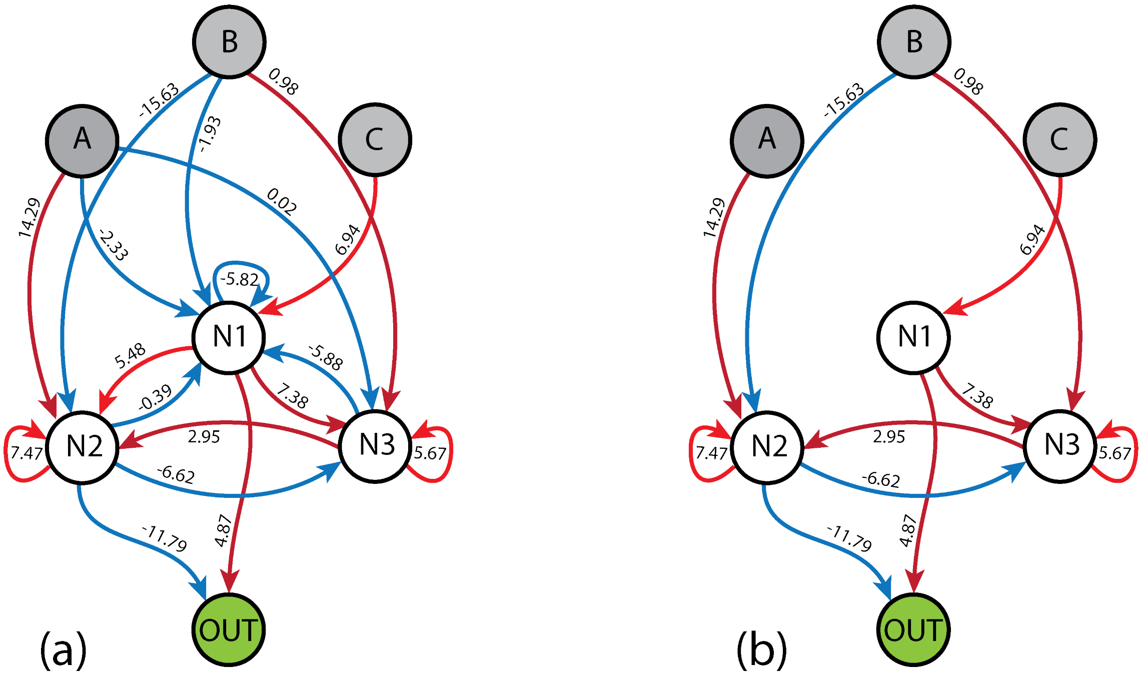


Figure 6.1: Individual no. 5 evolved with noise and fixed silent intervals of 16 ms. This individual cannot maintain a network state. (a) the topology of the evolved network. (b) the simplified network after pruning.

by a silent interval of 100 ms.

Table 6.1: The number of edges and self-excitatory loops in perfect recognisers evolved in the presence of noise. Top: evolved with constant silent intervals between signals (15 champions). Bottom: evolved with variable silent intervals between signals (12 Champions). The last two rows show their robustness to increased silent intervals from 16 ms to 100 ms.

Champions evolved in the presence of constant (16 ms) silent intervals

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evolved edges | 19 | 21 | 20 | 19 | 18 | 19 | 19 | 18 | 21 | 16 | 20 | 20 | 19 | 20 | 19 |
| edges after pruning | 10 | 14 | 15 | 13 | 11 | 14 | 16 | 15 | 16 | 16 | 17 | 16 | 15 | 13 | 13 |
| self ex-loops | 1 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 2 | 2 | 2 | 2 |
| 100 ms TPR | 0.02 | 0.98 | 0.98 | 0.97 | 0.96 | 0.95 | 0.99 | 0.55 | 0.99 | 0.00 | 0.00 | 0.00 | 0.96 | 0.99 | 0.96 |
| 100 ms FDR | 0.99 | 0.37 | 0.01 | 0.12 | 0.56 | 0.01 | 0.03 | 0.90 | 0.20 | 1.00 | 1.00 | 0.99 | 0.58 | 0.01 | 0.56 |

Champions evolved in the presence of noise and variable (16-32 ms) intervals of silence

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evolved edges | 19 | 21 | 20 | 19 | 20 | 18 | 16 | 20 | 18 | 19 | 18 | 18 |
| edges after pruning | 13 | 12 | 14 | 14 | 14 | 13 | 11 | 13 | 15 | 12 | 13 | 13 |
| self ex-loops | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 100ms TPR | 0.99 | 0.99 | 0.99 | 0.98 | 0.90 | 0.98 | 0.99 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 |
| 100ms FDR | 0.04 | 0.01 | 0.01 | 0.01 | 0.40 | 0.04 | 0.01 | 0.01 | 0.04 | 0.01 | 0.02 | 0.01 |

Network topologies of individual no. 5 before and after pruning are presented in Figure 6.1. In this network, out of 19 connections, 7 were superfluous and removing them from the network did not impair the performance of the network. The presence of superfluous connections in the evolved networks was misleading which made the analysis of the networks more challenging. After pruning, the network (Figure 6.1b) was left with only 11 connections ( 8 excitatory and 3 inhibitory) and three interneurons. This not only helped to understand the network in more depth but was also efficient in terms of both memory consumption and computational power.

It was interesting to observe that all the state-maintaining networks formed exactly 2 self-excitatory loops, while the individuals that could not maintain network states had 2.42 self-excitatory loops on average. In contrast to that, the average sum of the weights of the self-loops was higher in the networks that could maintain states 14.6 vs 12.3. Furthermore, all the memory keepers had exactly 2 active neurons. All these observations indicated that memory is a property of self-excitatory loops.

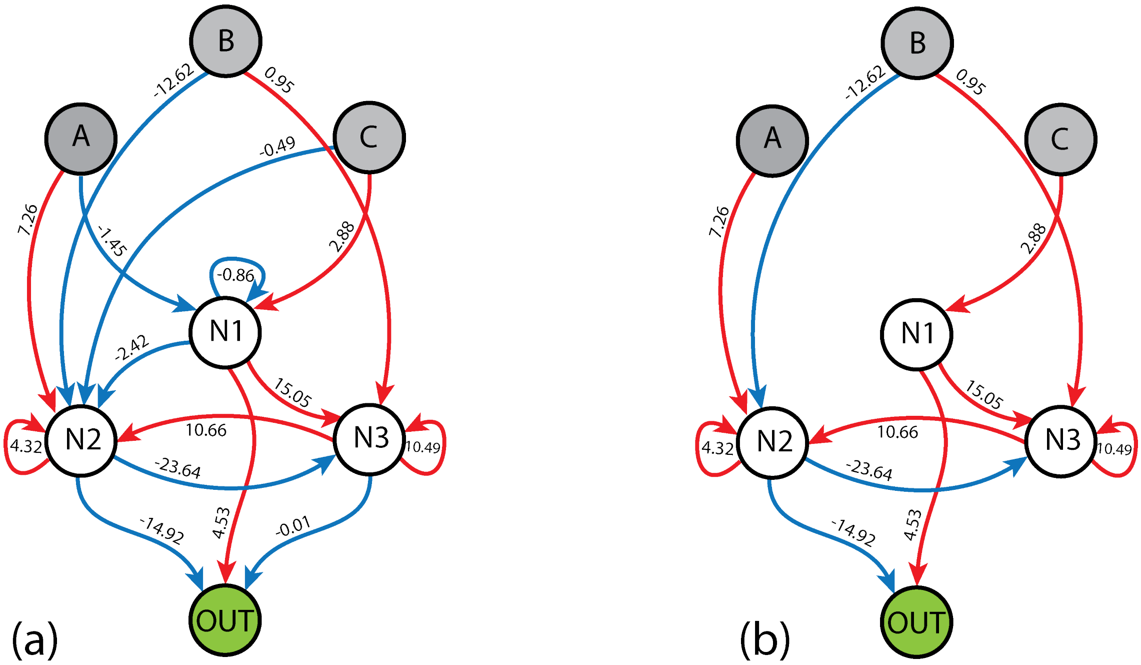


Figure 6.2: Individual no. 7 evolved with noise and variable silent interval [16, 32] ms. This individual maintains network states when the intervals between signals are increased. (a) the topology of the evolved network. (b) the simplified network after pruning.

Figure 6.2 shows the topologies of a memory keeper before and after pruning. This individual was obtained in the presence of intrinsic and extrinsic noise. Out of 16 connections in the evolved network 5 were marked superfluous by the pruning algorithm and therefore removed from the network (Figure 6.2b). The performance of the pruned network with only 11 connections remained intact; both the recognition ability and the property of state maintenance were not affected.

Despite being obtained from independent evolutionary runs in the presence of noise, the network topology of the two pruned networks (Figure 6.1 and 6.2) were exactly the same. However, one was a memory keeper and the other was not. Since both networks shared the same topology, it was the weights of the connections that maintained the states of the network. Without pruning, it was difficult to make this observation in the evolved networks.

To reveal the working mechanism and the dynamics that enable the property of state maintenance, the network states of the pruned individuals are identified in Section 6.3. The same formal model of finite-state transducer (FST) is used to demonstrate that the recognition task is accomplished by transitions between network states. In Chapter 4, the network states of individuals evolved without noise were mapped onto the states of an FST. Subsequently, chapter 5 described the importance of noise during evolution and established that SNNs evolved in the presence of noise developed robustness to perturbed neuronal parameters. In this chapter, the focus is on understanding the dynamics of pruned networks and mapping their network states onto the states of the FST.

## 6.3 Understanding Dynamics of Pruned SNNs

The pruned individuals were reevaluated for a random sequence of 100000 signals with 100 ms silent intervals between signals. In the case of short intervals of 16 ms, all 27 individuals (15 + 12) performed as perfect recognisers. However, 12 out of 27 individuals broke down when silent intervals between signals were increased, resulting in a different true positive rate (TPR) and false discovery rate (FDR). According to the reevaluated TPR and FDR values, the individuals were classified into one of the four categories as follows:

**Perfect-recognisers.** Also termed memory keepers, they secured a high TPR and low FDR value. Out of 27 individuals 15 were classified as perfect-recognizers. All individuals evolved in the presence of both types of noise performed as perfect-recognizers except individual no. 5. In addition, four individuals evolved in the presence of intrinsic noise only performed as perfect-recognizers: no. 3, 6, 7, 13.

**Over-recognisers.** Individuals with high TPR and high FDR values were classified as over-recognizers. Out of 27 individuals 7 behaved as over-recognizers, out of which 6 were evolved with intrinsic noise: no. 2, 4, 5, 9, 13, 15, and the last one was evolved with both types of noise: no. 5.

**Wrong-recognisers.** These individuals obtained a low TPR and a high FDR when reevaluated for a large sequence. Only 4 individuals evolved in the presence of only intrinsic noise operated as wrong recognizers: no. 1, 8, 11, 12.

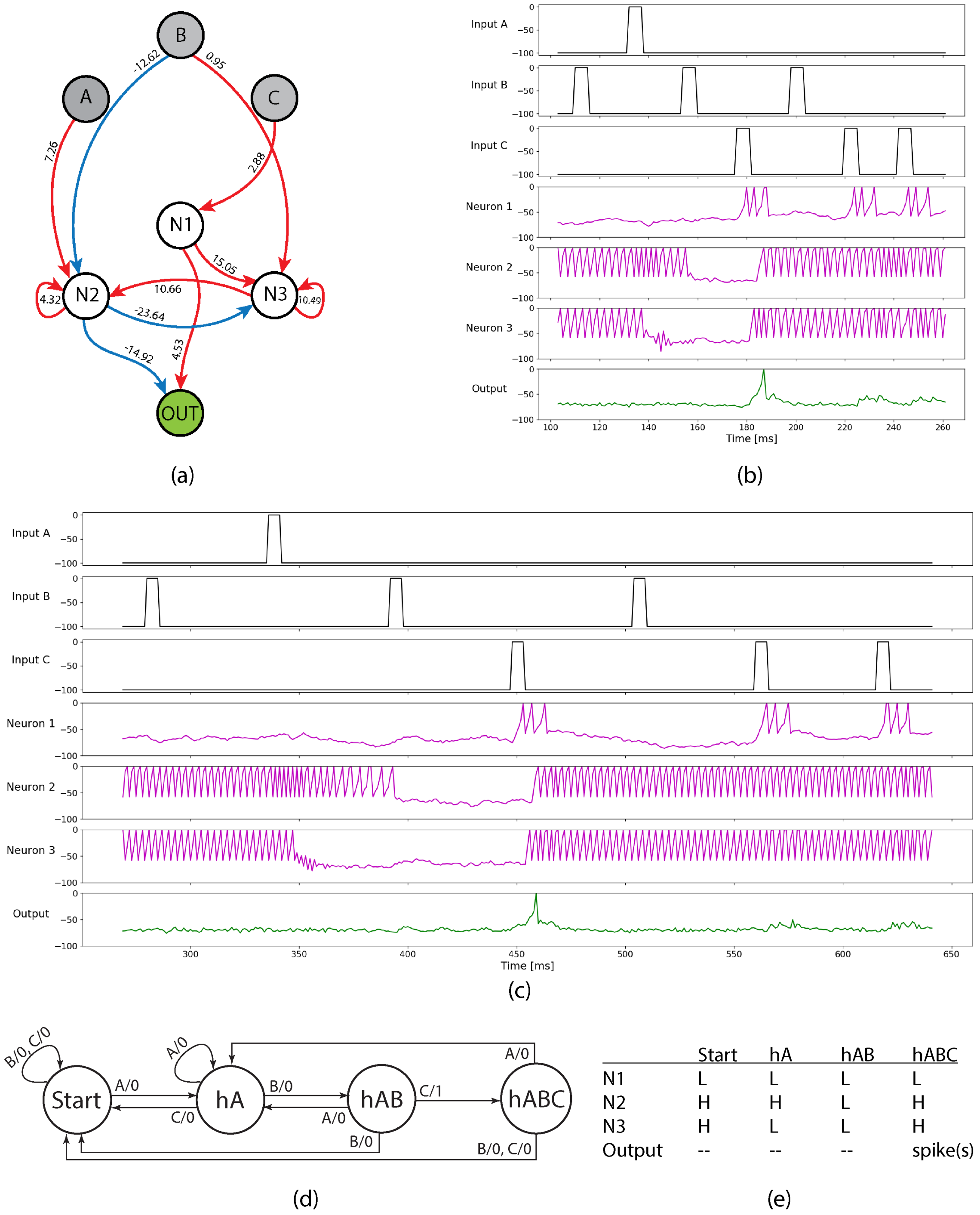
**Aphonic-networks.** These individuals could not keep the spiking activity of the output neuron, resulting in low TPR and low FDR values. The output neuron of an aphonic network never spikes irrespective of the input pattern. Individual no 10, evolved with only intrinsic noise acted as an aphonic network.

To sum up, 11 out 12 individuals evolved with both types of noise could maintain network states when the silent intervals were increased to 100 ms. In contrast, just 4 out of 15 networks evolved with intrinsic noise could maintain states. This indicated that extrinsic noise played an essential role in producing memory keepers. To understand why the performance of some individuals deteriorated when silent intervals between signals were increased, the individuals from each category are explored further.

Individuals that broke down to an increase of silent intervals to 100 ms also experienced impaired performance for 50 ms intervals. Therefore, the network activity (Figure 6.3c) is presented with silent intervals of 50 ms only. Moreover, considering the fact that excessive connections are misleading (in terms of understanding the mechanism driving a network to recognize a pattern) and make the analysis more difficult, only pruned networks are discussed in the next section. For simplicity, a neuron can be either in a low (L) or high (H) spiking state at a time. The state of a neuron is determined in the interstimulus interval (ISI). In the case of short intervals, a network state represents the state of all neurons in 6 + 16 = 22 ms intervals (Figure 6.3b), while the length of the ISI is increased to 6 + 50 = 56 ms (Figure 6.3c) when the silent intervals are increased to 50 ms. Furthermore, a network state LHL means that neurons N1 and N3 are in the L state, and N2 is in the H state. The order of the network state LHL follows the numbering (N1, N2, N3) of the interneurons in the network.

### Perfect-recognizer

Individual no. 7 (Figure 6.3), evolved in the presence of both types of noise is analysed as an example of a perfect recogniser. The topology of the pruned network has only 11 connections (8 excitatory and 3 inhibitory) and three interneurons (Figure



| | Start | hA | hAB | hABC |
|---|---|---|---|---|
| N1 | L | L | L | L |
| N2 | H | H | L | H |
| N3 | H | L | L | H |
| Output | -- | -- | -- | spike(s) |

Figure 6.3: Individual no. 7 evolved with both intrinsic and extrinsic noise. (a) The topology of the pruned network. (b) The activity of the network for short 16 ms, and (b) for long 50 ms silent intervals, which indicates that this individual can maintain network states when the silence prolongs. (d) The FST for accepting a string of 3 letters ABC. (e) the identified network states.

6.3a). Before receiving the first correct signal A, the network is in the start state represented by LHH, where N1 is in a low spiking state, while N2 and N3 are spiking with a high rate. As soon as, the network receives the first correct signal A, it transforms to the hA state. The strong excitatory connection from A to N2, along with the self-excitatory loop on N2, speeds up N2, which in turn inhibits N3. Consequently, N3 shuts down, and the network goes to the hA state denoted by LHL. If the next input signal is B, N2 shuts down, and the network transforms to the hAB state represented by LLL. This transition is explained by the strong inhibitory connection from B to N2. In addition, input B excites N3 weakly with a connection weight of 0.95. This connection can activate N3 only if the first B shuts down N2, releasing inhibition to N3. Therefore, getting a second B can activate N3 which in turn activates N2, transforming the network back into the start state. Intuitively, when this connection (from B to N3) was removed the network started responding to ABBC, ABBBC, ... in addition to the target pattern ABC. Suppose the network is in the hAB state, upon receiving the last correct signal C, the network goes into the hABC state denoted by LHH with the spiking of the output neuron. The only connection from C to N1 activates N1,and N1 activates N3 and the output neuron. The output neuron spikes for receiving the correct pattern ABC. N3 also passes the activity to N2, transforming the network back to the start state LHH – maintained actively by continuous spiking of N2 and N3.

The analysis of the 15 perfect-recognizers uncovered topological and behavioural commonalities among pruned networks. All memory keepers were left with exactly 2 self-excitatory loops after pruning. This stood out as one of the most salient topological features which accounts for state maintenance in these minimal networks. Moreover, the output neuron of all memory keepers received both inhibition and excitation from interneurons. This common structural observation could be explained by the correct spiking behaviour of the output neuron. Out of 4, three network states (start, hA, hABC) inhibited (prevent from spiking) the output neuron. In the hAB state the inhibition from the output neuron was released, making it ready to spike for the last input signal C in the correct order. The hAB state was maintained passively by no activity in the network.

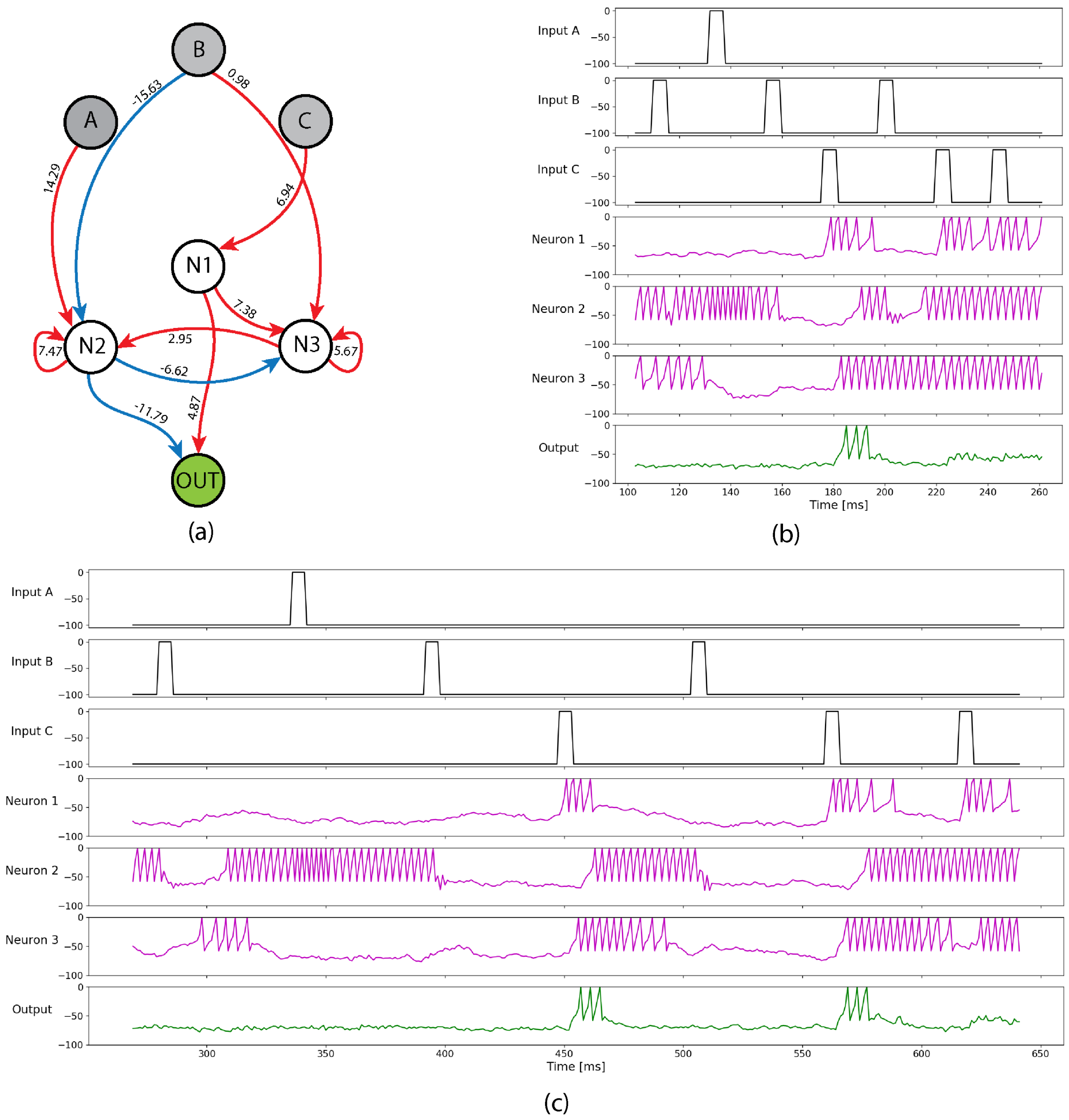


Figure 6.4: Individual no. 5 evolved with intrinsic noise and fixed silent intervals of 16 ms. (a) The topology of the pruned network. (b) The activity of the network for short 16 ms, and (b) for long 50 ms silent intervals, which indicates that this individual can maintain network states when the silence prolongs.

### Over-recognizer

Out of 27 individuals, 7 start responding to other patterns in addition to the correct pattern ABC, when the silent intervals between the signals are increased to 100 ms. Here, individual no. 5, evolved with only intrinsic noise is chosen for further analysis as an example of an over-recogniser network. This individual shares the same topology with the memory keeper (Figure 6.3 and 6.4). In the case of short silent intervals between signals, both individuals perform as perfect recognizers (a high TPR and a low FDR) where the start and the hABC states are maintained by high spiking of N2 and N3 neurons denoted as LHH. Unfortunately, when the silent intervals are prolonged, the activity of N3 dies out during the interval (due to the weaker self-loop on N3 in individual no. 5, Figure 6.4). This unwanted transition transforms the network to the hA state (LHL) while A is not actually received. The arrival of B at this stage turns off N2, transforming the network into a passively maintained state. Thus, receiving an input signal C, while in state hB would produce spike(s) wrongly in the output neuron. In this way, the network starts responding to all patterns ending with ...BC. Consequently, the FDR increases and the TPR remains high as the network continues to recognize the correct pattern ABC.

The network topology of the over-recognizer and the perfect-recognizer (Figure 6.3) are similar. Thus, it is obvious that because of the weaker strength of the self-excitatory loop on N3 in individual no. 5, the network could not maintain the start/hABC state when the silent intervals are increased. As a result, the activity of N3 dies out during the silence (state forgetting), transforming the network wrongly to the hA state. It can be deduced that the weights of the self-excitatory loops play an important role in maintaining network states.

### Wrong-recognizer

The four wrong recognizers are the outcome of the evolution in the presence of only intrinsic noise. In the case of short intervals (16 ms) they performed perfectly

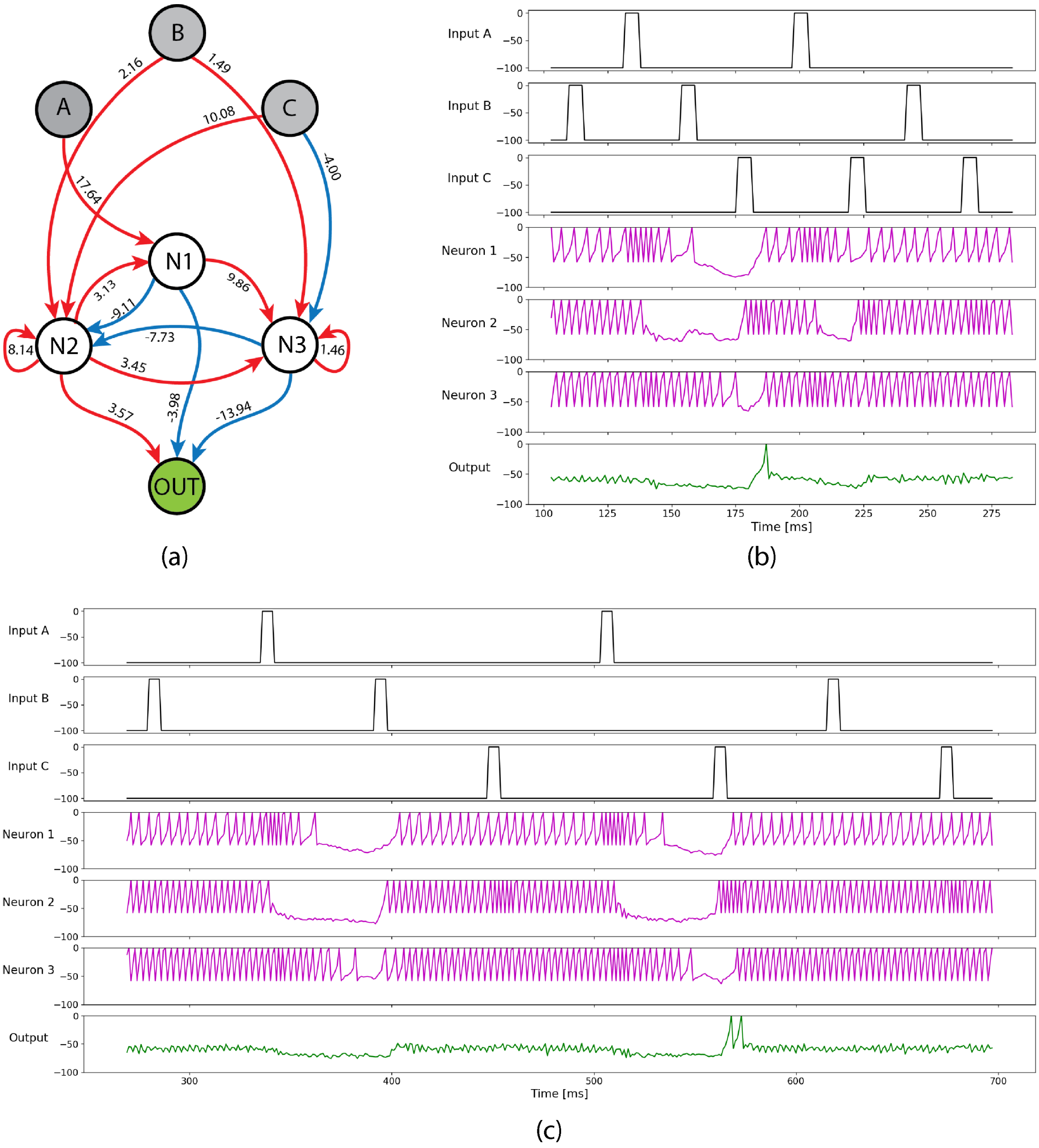


Figure 6.5: Individual no. 8 evolved with intrinsic noise and fixed silent intervals of 16 ms. (a) The topology of the pruned network. (b) The activity of the network for short 16 ms, and (b) for long 50 ms silent intervals, which indicates that this individual can maintain network states when the silence prolongs.

well. However, when silent intervals between signals were increased, the network experienced unstable transitions between network states. The working mechanism of individual no. 8 (Figure 6.5a) is explored as an example of a wrong-recognizer. This network is left with 15 (10 excitatory and 5 inhibitory) connections after pruning. The start state is represented by high spiking of all the 3 interneurons HHH (Figure 6.5b) with short silent intervals of 16ms. When the network receives an A, N1 accelerates and shuts down N2, transforming the network into the hA (HLH) state. Since N1 does not have a loop, therefore the activity dies out precisely as soon as B arrives. Thus the network goes into LLH state. The weak excitatory loop on N3 cannot prevent the spiking activity from dying out. Consequently, the network switches to the LLL state, releasing inhibition from the output neuron and exposing it to spiking for the last correct signal C.

In contrast, the network forgets the hA state (HLH) when the silent intervals are increased. During the silent interval, both N1 and N3 die out. The reason is that there is no self-loop on N1 while the self-excitatory loop on N3 is too weak to prevent it from dying out for long. As a result, the network relaxes to the LLL state during the silent interval, leaving the output neuron exposed. If a C arrives at this moment, the output neuron spikes and the network transforms back into the start state HHH. Similarly, when the network is in LLL state (has forgotten A), receiving a B at this point transforms the network back to the start state – continuous spiking of all interneurons denoted as HHH. Due to this reason the network remains silent for the correct pattern ABC, resulting in a low TPR.

A perfect recognizer behaves as a wrong-recognizer if it forgets the hA state during silent intervals. On the other hand, a perfect network operates as an over-recognizer if it cannot maintain the start or the hABC state when the silence continues.

### Aphonic-recognizer

A perfect-recognizer for short intervals can act aphonic to long silences. The working mechanism of the only aphonic-recognizer individual no. 10 (Figure. 6.6) is discussed here. The pruned network has 16 connections (9 excitatory and 7 in-

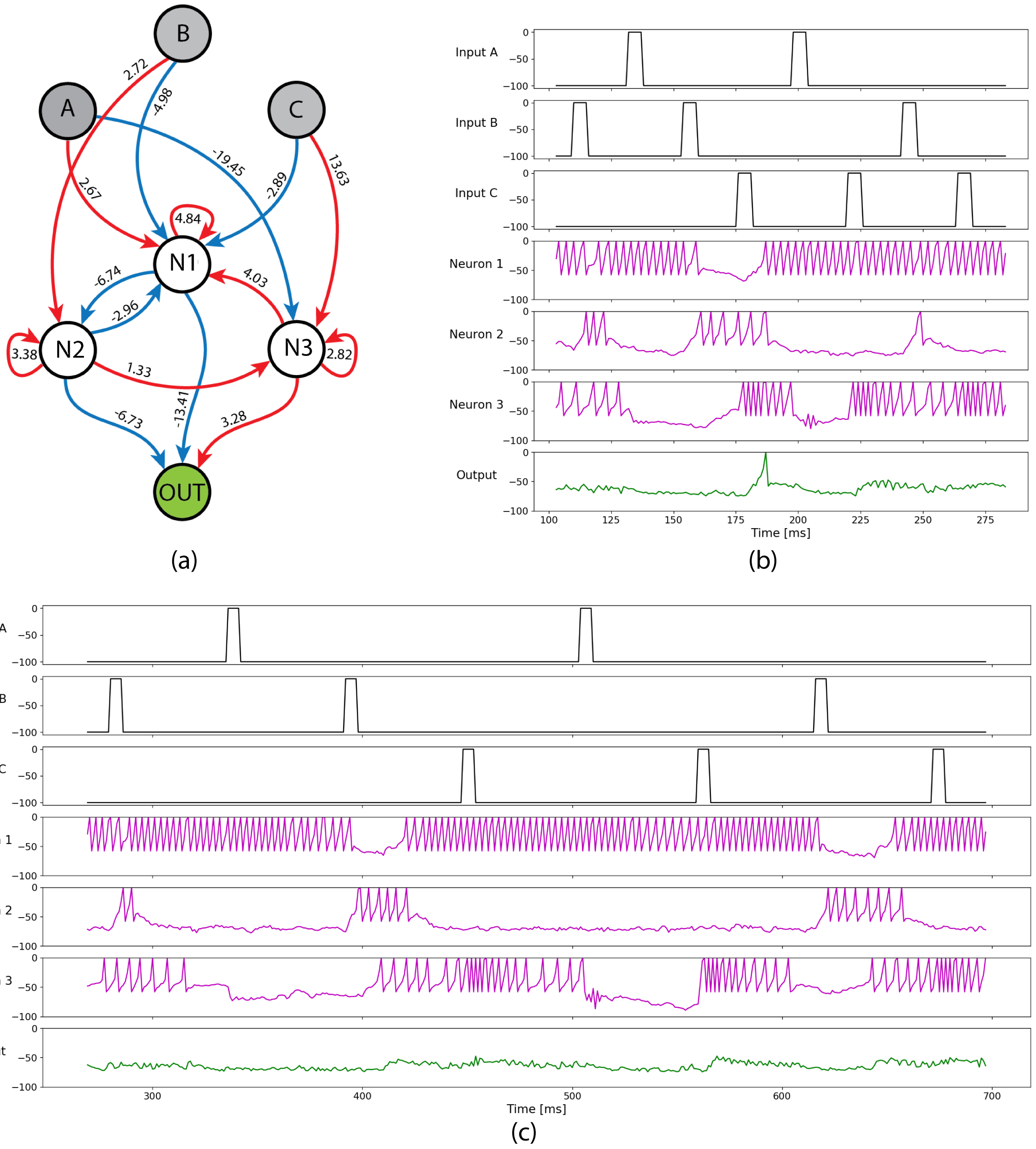


Figure 6.6: Individual no. 10 evolved with noise and fixed silent interval 16 ms. (a) The topology of the pruned network. (b) The activity of the network for short 16 ms, and (b) for long 50 ms silent intervals, which indicates that this individual can maintain network states when the silence prolongs.

hibitory). In comparison with other individuals the increased number of inhibitory connections in this solution indicates its aphonic nature. For short intervals, the recognition is accomplished with unstable state transitions. When the network gets an A, it goes into the hA state denoted by HLL – high spiking of N1 which in turn inhibits the output neuron (preventing it from spiking for wrong inputs). If this A follows a B, the inhibitory connection from B to N1 shuts down N1, while the excitatory connection from B to N2 activates N2, which now inhibits the output neuron. This handing over the responsibility of preventing output from spiking with the transition from one network state to another is intriguing. The hAB state is represented by continuous slow spiking of N2. Successively N2 excites N3. If an input signal C is received at this point, N3 accelerates and the output spikes for the correct pattern ABC.

On the other hand, when the silent intervals between signals are prolonged, the network goes into the only stable state hA (HLL) after receiving A. If B follows A, it shuts down N1 and activates N2 which in turn activates N3 slowly with a delay. However, N2 dies out before the silence ends, thus releasing inhibition from N1. Consequently, N3 activates the exposed N1. After the cessation of activity in N2, N3 cannot sustain itself for long with a weak self-excitatory loop. In this situation, if a C arrives N3 speeds up but it can never activate the output due to strong inhibition from N1. In short, regardless of the input pattern, the precise conditions for the output neuron to spike never occur in the case of prolonged silent intervals. Therefore, the network is termed aphonic.

## 6.4 Conservation of Dale's rule

The motivation behind producing minimal networks that could recognize a temporal pattern is two-fold. First, small networks are easier to evolve (due to smaller search space). Second and more important, it is easier to explain the dynamics of a small network. In order to roughly halve the number of neurons in a network, Dale's rule [15] is relaxed during evolution. According to Dale's rule, a neuron cannot inhibit and excite other neurons at the same time.

All the champions presented in this chapter have at least one neuron that violates Dale's rule - exciting and inhibiting other neurons at the same time. This violation is benign because an evolved network can be transformed to conform to Dale's rule. The transformation requires splitting the violating neuron into two new neurons; one excitatory and one inhibitory. Both neurons receive the same input connections with the same weight as the original one. The outgoing connections are also kept the same. If the original neuron has an excitatory self-loop, the new excitatory neuron inherits this loop, and an excitatory connection with the same weight is made from the excitatory neuron to the inhibitory neuron. Similarly, in the case of an inhibitory self-loop, the inhibitory neuron keeps the loop and an inhibitory connection is created from the inhibitory to the excitatory neuron. This new connection could affect the performance of the network due to introducing an extra delay of 1 ms. However, the networks evolved in the presence of noise were robust to this structural perturbation. Furthermore, the networks also showed robustness to the extra noise incurred by splitting a neuron into two neurons. For example, individual no. 7, the perfect recognizer, has one neuron N2, which violates Dale's rule. The splitting of this neuron is demonstrated in Figure 6.7a, and the activity of the transformed network for short and long intervals is shown in Figure 6.7. The performance of the network is not affected by Dale's transformation. Moreover, the network not only remained perfect but also continued to maintain states in the case of prolonged silent intervals. It is important to note that individual no. 7 is evolved in the presence of both types of noise which explains the robustness of the network to such a transformation. On the contrary, without noise the networks were hardly

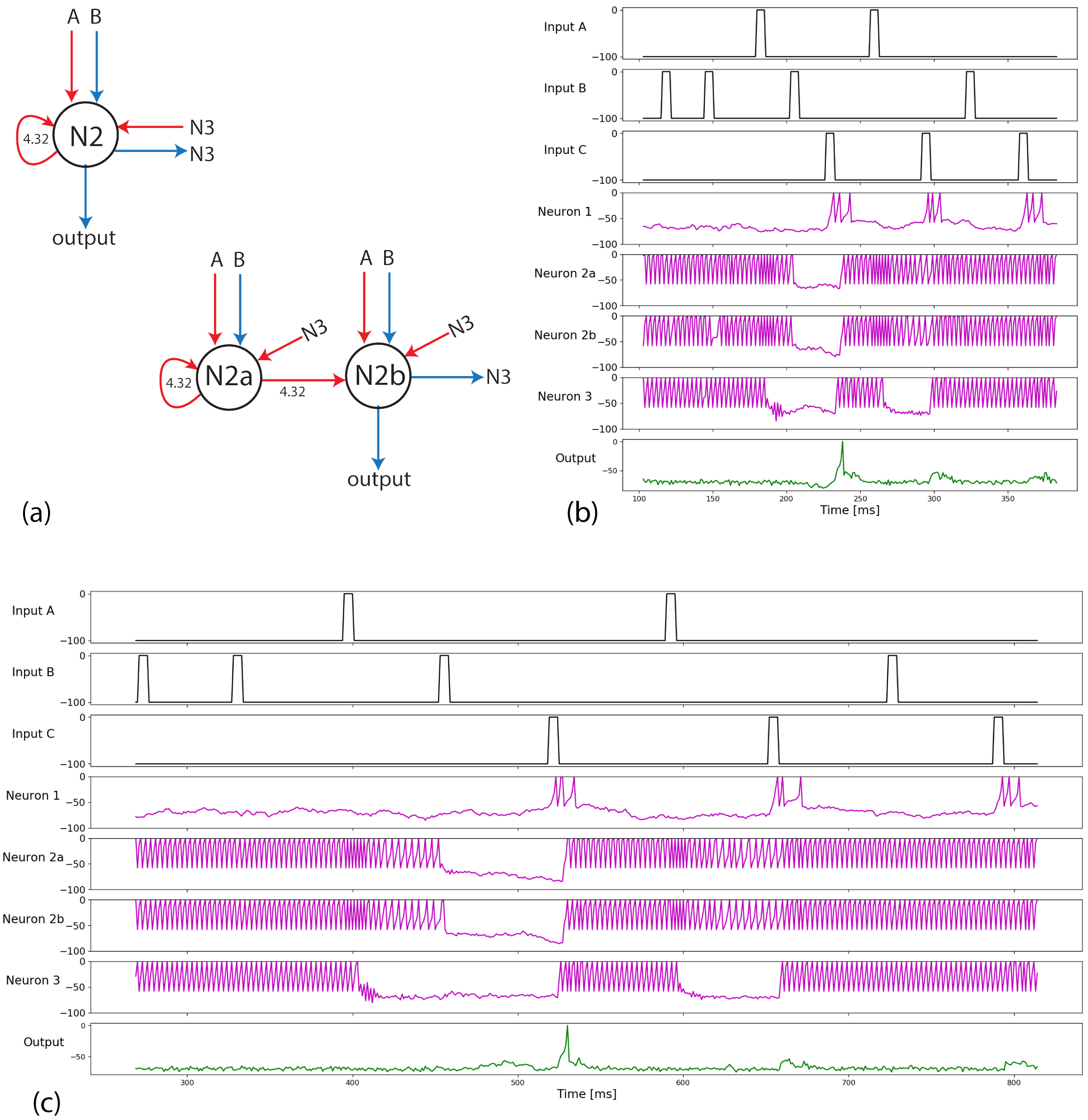


Figure 6.7: Transforming individual no 7 (Figure 6.3) to confirm to Dale's rule. (a) The only violating neuron N2 is divided into two neurons, one excitatory and one inhibitory. (b) The activity of the network for short 16 ms, and (c) for long 50 ms silent intervals, which indicate that an evolved network can be transformed to follow Dale's rule.

robust to any type of perturbation.

## 6.5 Conclusion

It is very likely for evolution to over-produce connections which can be pruned without impairing the performance of the network. Superfluous connections are misleading and make the analysis of the networks more challenging. It is much easier to observe commonalities among the pruned networks. Furthermore, the role of each connection stood out in the pruned networks which helped greatly in understanding how these minimal networks function.

Self-excitatory loops are more likely to form in the presence of both intrinsic and extrinsic noise which play an important role in state maintenance when the silent intervals are longer. In the presence of noise, recognizing a pattern of three signals requires at least 3 interneurons. A state-maintaining individual must form at least 2 self excitatory loops with sufficient weights to keep the spiking activity from dying out. A perfect-recognizer accomplishes the recognition task by switching between network states. A network-state is defined as the state of all neurons in the inter-stimulus interval (ISI) – the duration between the onsets of two consecutive signals (ISI = length of the signal + length of the following interval). The minimal FST accomplishes the task of recognizing a string of three letters by 4 network states, and so do the networks evolved in the presence of noise. Mapping network states onto the states of an FST explains the basis of the recognition mechanism, i.e. switching from one network state to another. Furthermore, it has been observed that the penultimate state hAB (had AB) is always maintained passively by no activity in the network. On the other hand, the start and the hABC states are maintained actively in a similar way, with the only difference of intermittent activity in the output neuron.

In the case of prolonged silent intervals, over-recognition happens when a network cannot maintain the start or the terminal (hABC) state. On the other hand, if the network cannot maintain the inter-pattern state hA or hAB, it stops recognizing the correct pattern and starts responding to the wrong patterns. Another possibility is becoming completely quiet regardless of the input signal.

Finally, the results have demonstrated that the network evolved in presence of noise can be transformed to obey Dale's rule – a neuron can either excite or inhibit other neurons at the same time but not both.

# Chapter 7

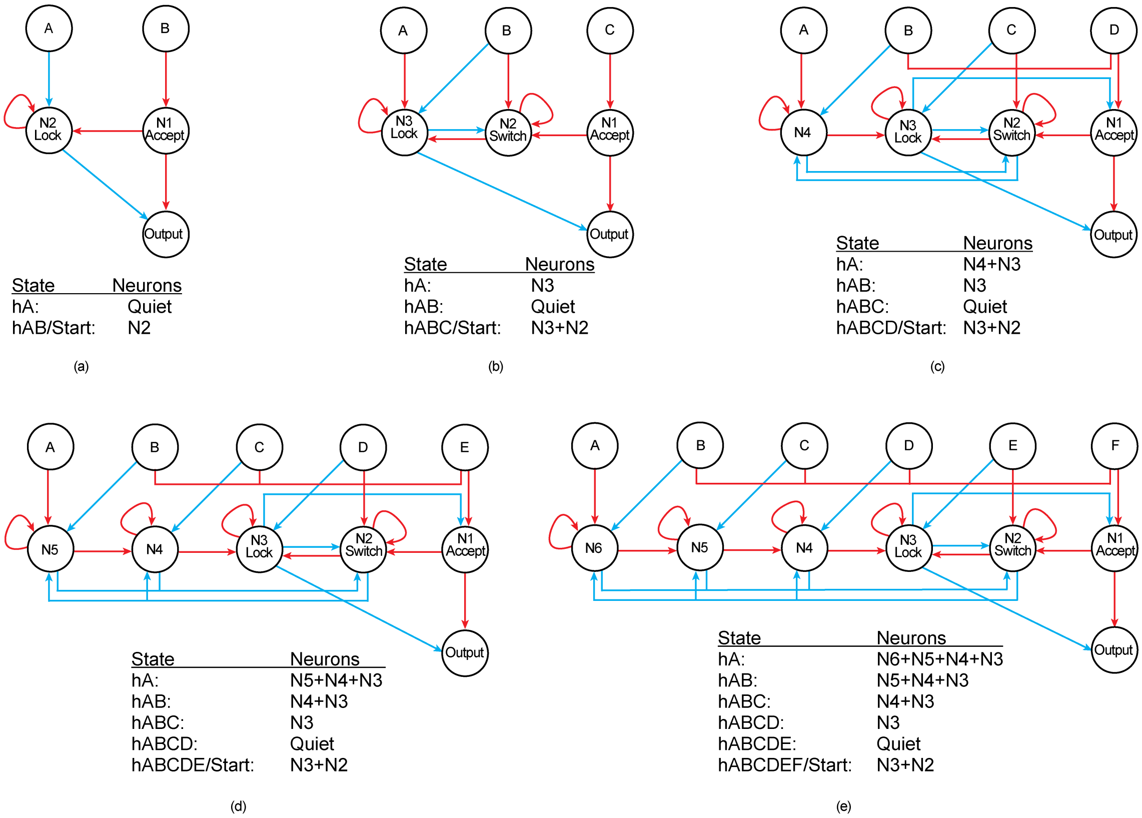
# Handcrafting SNNs for longer Patterns

In Chapter 6, we analysed the behaviour of the evolved networks in the presence or absence of noise during evolution. This chapter uses that knowledge to define rules for handcrafting network topologies. Analysis of the pruned networks revealed several commonalities among the evolved networks obtained from different evolutionary runs. An important observation is that the network topologies of perfect recognisers evolved in the presence of noise are similar in structure. Other commonalities include: (i) the network states of an evolved network can be mapped onto the states of a finite state transducer (FST), accepting a string of letters, (ii) the recognition happens with transitions from one network state to another, (iii) for state maintenance, the weights of the self-excitatory loops should be strong enough to keep the spiking activity from dying out in the absence of input activity, (iv) the output neuron is prevented from spiking in the start and in the hA state by continuous inhibition, (v) the inhibition from the output is released when the network transforms to the penultimate state, passively maintained by no activity in the network, exposing the output to spike for the last signal in the correct order. These findings led to handcrafting network topologies for recognising longer patterns for which the evolutionary algorithm failed to produce minimal solutions.

The following section suggests rules for handcrafting network topology. The next section validates the constructed topologies by optimising their connection weights. Then the specialized roles of neurons in the network are identified, followed by the contribution of connections to pattern recognition. The subsequent section demonstrates the performance of the handcrafted network for recognizing a pattern of 6 signals. The chapter is concluded with a list of findings.

## 7.1 Rules for Handcrafting Network Topologies

The switching mechanism of the evolved three-signal networks is employed to handcraft network topologies for recognising patterns of lengths four and above. For example, a network topology recognising a pattern composed of three signals can be extended by adding a new input A, an inter-neuron N4, and six synaptic connections (Figure 7.1b-c). In the extended network, the input stimuli are renamed as $ABCD$. The newly added input A excites the neuron N4. The input B (previously named input A) connects to neuron N2 (the switch neuron) with an excitatory connection. It also inhibits the newly added neuron N4. Neuron N4 connects to N3 and to itself with excitatory connections. Furthermore, N4 and N2 (the switch neuron) inhibit each other and thus are mutually exclusive. For seamless transitions, two more connections are essential for switching the network back to the start state after receiving the last correct input signal. Therefore, these connections are introduced in the network to recognise patterns of length four and above: (i) the last input excites the switch neuron (N2), (ii) the lock neuron (N3) inhibits the accept neuron (N1).



| State | Neurons |
|---|---|
| hA: | Quiet |
| hAB/Start: | N2 |

| State | Neurons |
|---|---|
| hA: | N3 |
| hAB: | Quiet |
| hABC/Start: | N3+N2 |

| State | Neurons |
|---|---|
| hA: | N4+N3 |
| hAB: | N3 |
| hABC: | Quiet |
| hABCD/Start: | N3+N2 |

| State | Neurons |
|---|---|
| hA: | N5+N4+N3 |
| hAB: | N4+N3 |
| hABC: | N3 |
| hABCD: | Quiet |
| hABCDE/Start: | N3+N2 |

| State | Neurons |
|---|---|
| hA: | N6+N5+N4+N3 |
| hAB: | N5+N4+N3 |
| hABC: | N4+N3 |
| hABCD: | N3 |
| hABCDE: | Quiet |
| hABCDEF/Start: | N3+N2 |

Figure 7.1: Networks recognising patterns of lengths 2 to 6: The network in panel b is obtained with artificial-evolution. The table below each network shows the network transitions between states, represented by the number of active neurons. The topology is extended to recognise patterns up to length 6 (c-e).

These rules can extend the topology of a four-signal network to recognise patterns composed of five and six signals (Figure. 7.1d-e, see Figure A.3.1 in Appendix A.3 for connections weights of the handcrafted topologies).

## 7.2 Optimization of Connection Weights

The structural topology and the polarity of connections were predefined in the handcrafted networks. To optimise connection weights, I used a genetic algorithm where an individual's genome was the adjacency list of the handmade network. The number of individuals (100) in the population was kept constant during the evolution, with an elite count of 10 individuals. In the first generation, the synaptic weights were drawn from a uniform distribution between 0 and 10 (-10 and 0) for excitatory (inhibitory) connections. The only mutation was adding a random number drawn from a normal distribution with mean = 0 and SD = 1 to the connection weights with a probability of 0.1 for choosing a connection. Subsequent generations were created with size-two tournament selection; two individuals were picked randomly from the population and the best one (according to the fitness value) was transferred to the next generation after undergoing the genetic operator. The fitness function rewarded spikes in the correct inter-stimulus interval and penalised spikes elsewhere.

The number of possible patterns increases exponentially with the length of the pattern to be recognised; there are $n^n$ possible orderings for n signals. For examples, a pattern of 4 signals has $4^4 = 256$ (AAAA to DDDD) possible permutations, and a pattern of 6 signals has $6^6 = 46656$ (AAAAAA to FFFFFF) permutations. This implies that when n gets large, the patterns occur less frequently in the continuous random input stream of signals, or some patterns may not occur at all. Since individuals are evaluated for a random sequence in every generation, a simple genetic algorithm could not optimize connection weights. Therefore, a genetic algorithm is proposed that runs in two stages. Consider evolving networks for recognising a pattern of four signals, n = 4. In the first stage, the individuals are optimised only for the patterns similar to the target pattern of the form AXXX, XXXD, where XXX in AXXX (XXXD) is replaced by all possible 27 patterns of BCD (ABC). A sequence of 10000 signals is created by randomly concatenating 54 patterns (ABBB to ADDD and AAAD to CCCD). When the genetic algorithm converges (finds a network that only responds to the correct pattern ABCD and remains silent for all other patterns), in the second stage the hard to recognise patterns are identified by evaluating the champion individual for all possible patterns of signals A, B, C and D with a different preceding history. A pattern is considered hard if the network fails to identify it correctly (at least 10% of its total number of occurrences in the input stream). The hard patterns are then added to the input sequence, and the penalty coefficient in the fitness function is increased to 50 (which remains 50 in the subsequent iterations of stage 2). The evolution is then continued and the population is further evolved with the new fitness function. In the subsequent iterations of stage 2, each individual in the population is evaluated for a new training sequence of 10000 signals created randomly by concatenating the 54 patterns of the form (AXXX + XXXD) and the identified hard patterns with equal probability. The genetic algorithm halts when a perfect individual is obtained for the new training sequence. The second stage is repeated until no hard patterns remain.

## 7.3 Specialised Role of Neurons

The network topologies of the pruned networks recognising a pattern of 3 signals were similar in their structure [150, 147]. In particular, these networks accomplished the task with four neurons (3 interneurons and an output neuron) and 11 connections (3 inhibitory and 8 excitatory). Moreover, these networks preserved three network states actively; the $hA$ state by persistent spiking of the *lock* neuron and $hABC/start$ state by tonic spiking of the *lock* and the *switch* neurons (Figure 7.2a). The only difference between the *start* and the $hABC$ state was the intermittent spiking of the output neuron. During evolution, both the *lock* and the *switch* neurons formed self-excitatory loops with sufficient weights to prevent the spiking activity from dying out. The formation of autapses in these minimal networks explained their ability to maintain network states when silent intervals between signals were increased [149, 147].

The interneurons of an evolved network are observed to have specialised roles of

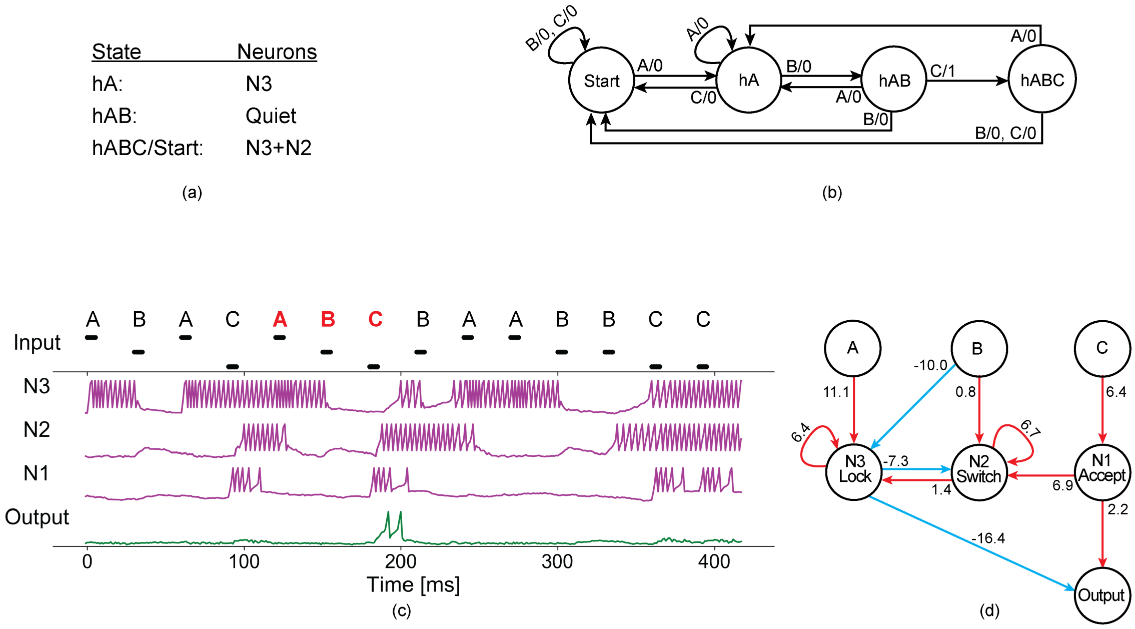


Figure 7.2: (a) Network states with corresponding active neurons. (b) Minimal finite state transducer for recognizing a string of three letters ABC. (c) The behaviour of each neuron in the network. (d) Handcrafted network for recognising a pattern of length 3.

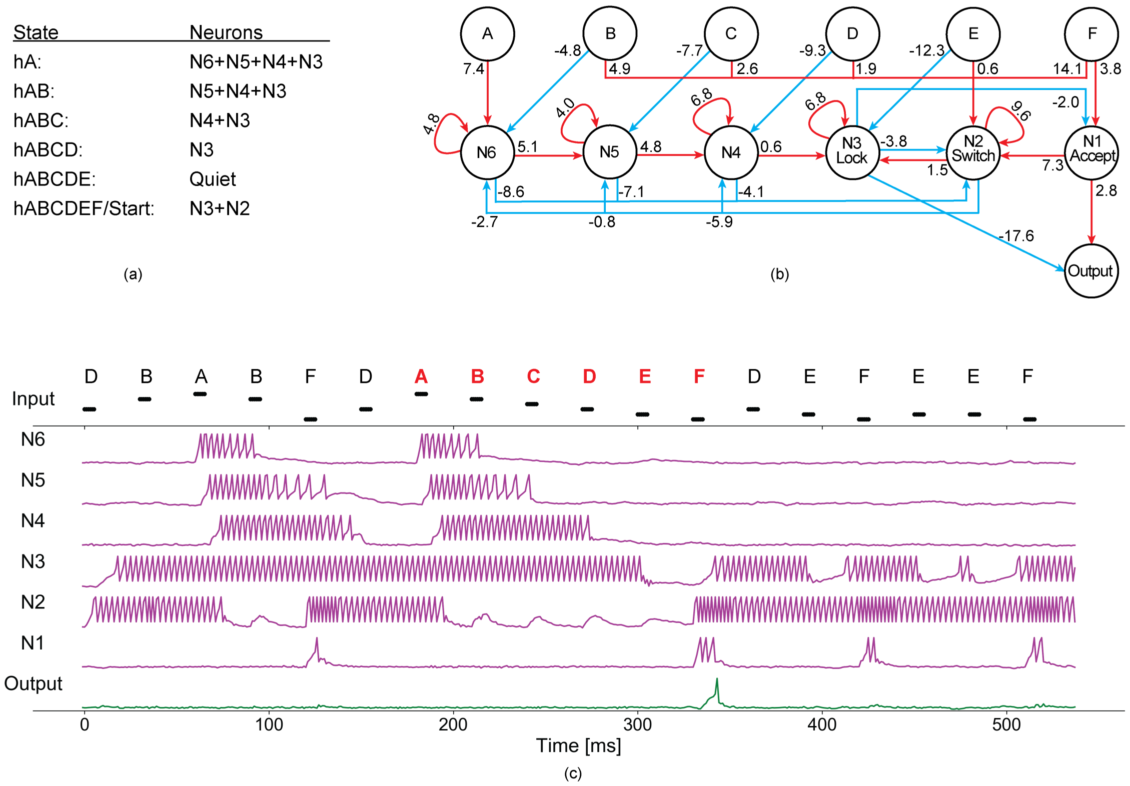
*locking*, *switching* and *accepting*. The self-excitation of the *lock* neuron prevents the output neuron from spiking, except when the second to the last correct input signal shuts down the *lock* neuron, enabling the output neuron to spike for the last correct input signal. If the *lock* is released by the penultimate correct input signal, the *accept* neuron activates the output neuron on receiving the last correct input. The *accept* neuron also sends a signal to the *switch* neuron, transforming the network back into the start state. The *switch* neuron is responsible for the transition between the network's start state and inter-signal states; it is active in the *start* state of the network and shuts down as soon as the network goes into an inter-signal state (*hA* and *hAB*).

## 7.4 Contribution of Connections to Pattern Recognition

The contribution of each synaptic connection to pattern recognition is determined by the behaviour of postsynaptic neurons. Consider the network recognising a pattern of 3 signals (Figure 7.2): the excitatory connection from input A to neuron N3 ($A \rightarrow N3$) activates the N3 neuron, N3 excites itself with an autaptic connection ($N3 \rightarrow N3$) and inhibits N2 with an inhibitory connection ($N3 \rightarrow N2$). The persistent spiking of $N3$ represents the $hA$ state of the network. In addition, the spiking of N3 also prevents the output neuron from spiking with a strong inhibitory connection from $N3$ to the *output* neuron. The second input B connects N3 ($B \rightarrow N3$) with an inhibitory connection. When A is followed by B, this connection shuts down the N3 neuron that maintained the $hA$ state, and the network goes into a quiescent state $hAB$. The weak excitatory connection from input B to N2 prevents repeated input signal B. The weight of this connection is adjusted such that the first B cannot activate the switch (N2) neuron due to continued inhibition from $N3$. However, when $N3$ is deactivated by the first B in the correct order, the second B can activate the switch (N2) neuron which in turn activates the $N3$. As a result, the network goes back into the *start* state (continuous spiking of N3 and N2). As could be expected, when this connection is removed, the output neuron of the network starts responding to ABBC, ABBBC, etc. Furthermore, the positive connection from C to $N1$ triggers several spikes in $N1$ neurons, $N1$ passes the activity to the output neuron which spikes for receiving the last correct input C. Note that a C received in the wrong order (after A or C) cannot activate the output due to continued inhibition from the lock neuron, which is released only if A is followed by B. In addition, $N1$ activates $N2$–the *switch* neuron which in turn activates $N3$–the *lock* neuron, transforming the network back into the start state.

## 7.5 Performance of the handcrafted Network for Recognising a Pattern of 6 Signals

The analysis of a handcrafted network for identifying a pattern of 6 signals showed that the network accomplished the task of recognition with seven well-defined network states (Figure 7.3a). As the network received the first correct input signal A around 180 ms (Figure 7.3c), the network transformed to the *hA* (had A) state represented by four active neurons N6, N5, N4 and N3. The input A connects to N6 directly with a strong excitatory connection, causing N6 to spike. The spiking continues on N6 due to the self-excitatory loop (Figure 7.3b), which in turn activates N5 and inhibits N2 (the switch neuron) from spiking. The spiking activity persists



| State | Neurons |
|---|---|
| hA: | N6+N5+N4+N3 |
| hAB: | N5+N4+N3 |
| hABC: | N4+N3 |
| hABCD: | N3 |
| hABCDE: | Quiet |
| hABCDEF/Start: | N3+N2 |

Figure 7.3: (a) Network states with corresponding active neurons. (b) Handcrafted network for recognising a pattern of length 6. (c) The behaviour of each neuron in the network.

in N5 (due to the self-excitatory loop on N5). N5 passes the activity to N4, which in turn activates N3 (the lock neuron) in a similar fashion. Meanwhile, if a B is received, the inhibitory connection from input B to N6 shuts down N6, transforming the network into the $hAB$ state, which is maintained by continuous spiking of the remaining three neurons (N5, N4 and N3). Input B also excites the switch neuron with a precise connection weight such that only a second B can activate the switch neuron and transform the network back into the start state. Similarly, if AB is followed by input signal C in the correct order, the inhibitory connection from input C to N5 shuts down N5, transforming the network into the $hABC$ state, denoted by continuous spiking of N4 and N3 neurons. Next, if the network receives an input D, the inhibitory connection from D to N4 switches off N4, transforming the network into the $hABCD$ state, maintained by continuous spiking of N3 only. All the four states from $hA$ to $hABCD$ are maintained actively by persistent spiking of 4, 3, 2, and 1 neuron(s), respectively. It is important to note that N3 (the lock neuron) is always active except when the network receives the second to the last input signal in the correct order, allowing the output neuron to spike for the correct last input signal F. The strong inhibitory connection from N3 (the lock neuron) to the output neuron secures the output from spiking for incorrect patterns. If the network is in the $hABCD$ state (represented by continuous spiking of N3–the lock neuron) and it receives the penultimate input signal E, the inhibitory connection from E to the lock neuron deactivates the lock, exposing the output neuron to spike for the last correct input signal F. The input F activates N1 with a positive connection, which in turn passes the activity to both the output and the switch neuron. The output spikes for receiving the correct pattern, and the switch neuron activates the lock neuron, putting the network back into the start state where the switch (N2) and the lock (N3) spike continuously (network topologies for recognizing patterns of length 4 and 5 are provided in Appendix A.3, Figure A.3.2 and Figure A.3.3).

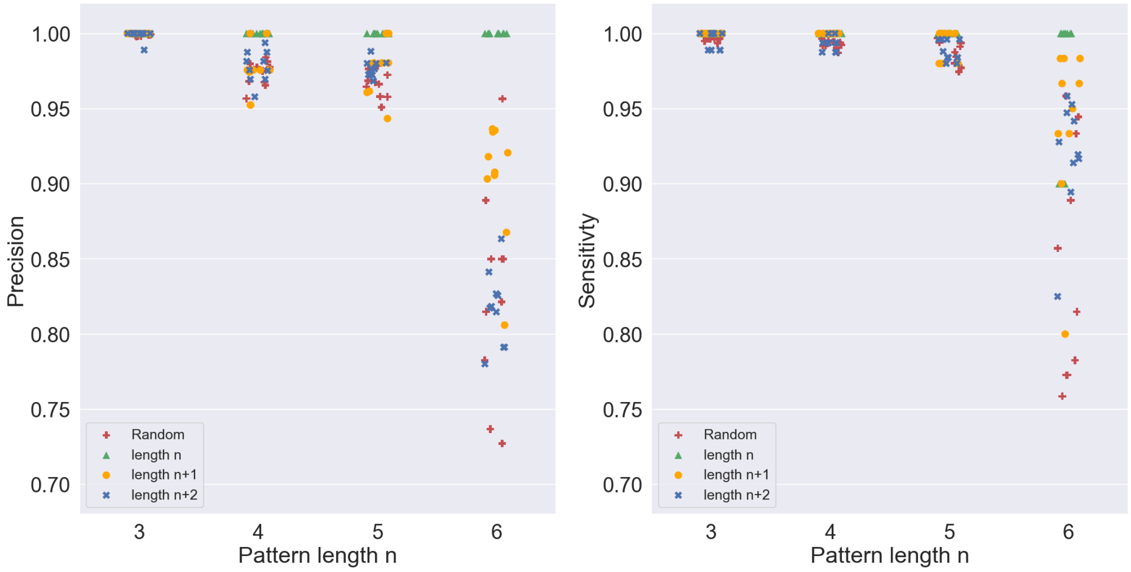


Figure 7.4: Performance degradation of handcrafted networks with increasing pattern length. Precision (left) and sensitivity (right) of the top 10 networks for each pattern size (3 to 6) are evaluated for a random sequence of length 1 million and all possible patterns of length n, n+1 and n+2.

The performance of the handcrafted networks is evaluated in terms of precision and sensitivity. Precision is defined as how often the output neuron spikes correctly, i.e. the number of true positives divided by the total number of times the output spiked (Equation 7.1), whereas sensitivity is the average number times of the output neuron spiked correctly, i.e. the number of true positives divided by the actual number of correct patterns in the sequence (Equation 7.2). The top 10 networks for patterns of lengths 3, 4, 5 and 6 are re-evaluated for a random sequence of 1 million signals (Figure 7.4). The performance of the networks degrades with increasing length of the pattern. For example, the precision of the top 10 networks obtained for recognising $ABCDEF$ is between 0.73 and 0.96, whereas for length 5 ($ABCDE$) and below the precision is always above 0.94. The networks are then tested with all possible permutations of 6 signals in 6 spots with replacement $^6P_6$ (from $AAAAAA$ to $FFFFFF$), $^6P_7$ ( from $AAAAAAA$ to $FFFFFFF$) and $^6P_8$ ( from $AAAAAAAA$ to $FFFFFFFF$). The testing of the networks with all possible patterns of lengths 7 and 8 ensures that the network's performance is not affected by the preceding signals – history. Similarly, all possible permutations of lengths 3, 4, and 5 are evaluated up to 2 preceding signals (Figure 7.4).

$$precision = \frac{TP}{TP + FP} \tag{7.1}$$

$$sensitivity = \frac{TP}{P} \tag{7.2}$$

To demonstrate that all false positives were caused by history or noise, each pattern was given to the network 10 times. All the perfect networks responded to the correct pattern $XXABCDEF$ at least 8 out of 10 times, and they responded to a very small number of false positives not more than 4 out of 10 times. This clear margin indicated that handcrafted networks could perfectly recognise correct patterns up to 6 signals.

## 7.6 Conclusion

This chapter revealed the relationship between the structural connectivity and the functional behaviour of spiking neural networks for the pattern recognition task. In addition, rules are proposed for handcrafting network topologies for recognising longer patterns. The constructed topologies are validated by optimising their connection weights. Moreover, the functional role of excitatory autapses is highlighted both in memory maintenance and network state transition.

To conclude, handcrafting SNNs adds a novel investigative tool to the field of computational neuroscience. Understanding the functioning of minimal spiking neural networks can greatly help in designing low-power intelligent solutions. Furthermore, explainable SNNs may also give an insight into information processing in the animal brain. In the future work, we expect to see handcrafted spiking networks for other computational tasks.

# Part IV

# Conclusion

# Chapter 8

# Conclusion

## 8.1 Development of Ideas

This study suggested a new way of learning to recognise distinct temporal patterns by optimising the topology and connections weights of small spiking neural networks (SNNs) without changing or selecting conduction delays in the network. In the initial experimental setup, the genetic algorithm (GA) in GReaNs was adapted to optimise a population of SNNs such that the output neuron of the network spikes only for the correct pattern and remains silent for all others. The focus was on fine-tuning the GA parameters to obtain minimal SNNs that perform temporal pattern recognition [148]. However, despite the success in obtaining such minimal SNNs, they were found to be rigid in nature, and even a slight change in the parameters could result in a complete breakdown of the network's performance. A previous study on the evolution of SNNs for animate control has found that introducing noise in the membrane potential of neurons during evolution allows for efficient control and promotes robustness to disturbed neuronal parameters [142]. Therefore, I conducted another set of experiments in the presence of two types of noise during evolution: (i) modelled as random fluctuations of the membrane potential, and (ii) modelled as random variations of the silent intervals in the input stream [150, 147]. Interestingly, the networks obtained in presence of noise were not only robust but were also simpler, which helped greatly in understanding the operation mechanism of these minimal networks.

The task of identifying a subsequence of signals is analogous to string recognition, as both involve recognizing a pattern within a large sequence. Therefore, one way to understand the properties of an evolved SNN is to map the activity in the network (network states) onto the states of a finite state transducer (FST) – a general model of computation for time-structured data. Since the networks in this work contain less than 10 interneurons, I was able to construct FSTs for the evolved SNNs by hand. It was interesting to see that all of the perfect recognizer networks could be mapped onto an FST. This intriguing observation prompted a deeper investigation of understanding how these minimal networks operate at the level of synaptic connections and individual neurons.

The analysis of the evolved networks uncovered the possibility of overproduced synaptic connections during evolution. Removal of these superfluous connections eased the analysis without impairing the network's performance. More importantly, it pronounced structural similarities and commonalities among the evolved networks. It soon became apparent that self-excitatory loops in the networks have a functional role in state maintenance – a form of memory. Moreover, the special role of each neuron in the evolved networks was identified, which led to the idea of constructing the networks by hand for recognizing longer and more complex patterns. Handcrafted networks are important for two reasons: (i) a handcrafted network confirms a complete understanding of the network, (ii) the GA could not produce any minimal network that could perfectly recognise patterns of length 4 and above.

## 8.2 Results

The aim of this study was to understand information processing in artificial spiking neural networks evolved for temporal pattern recognition. The key focus was on the evolution, analysis, and handcrafting of very small spiking neural networks for temporal pattern recognition. The main results can be summarised as follows:

- The genetic algorithm implemented in GReaNs can be adopted to optimise the network topologies and connection weights of a population of SNNs for a temporal pattern recognition task in the presence or absence of noise. Noise can be introduced intrinsically or extrinsically during evolution. Intrinsic noise is modelled as random fluctuations of the membrane potential of neurons in the network at every network step, while extrinsic noise is random variations of silent intervals in the input stream.

- In the absence of noise during evolution, the networks' performance breakdown completely with a slight perturbation in the values of parameters or with a small variation in the duration of the silent interval between input signals.

- Noise promotes robustness, introducing an optimal noise level during evolution is beneficial and has a computational role [33, 142]. Minimal SNNs evolved in the presence of noise developed robustness to:

  - disturbed neuronal parameters,
  - prolonged silent intervals,
  - removal of superfluous connections,
  - transformation of the network to follow Dale's principle by splitting the violating neurons into an excitatory and an inhibitory half.

- The network states of an evolved network that recognizes a temporal pattern can be mapped onto the states of a finite state transducer (FST) [113], accepting a string of letters. As a matter of fact, the network accomplishes the task of pattern recognition by transitioning between different network states, where a network state refers to the state of all neurons in the network during the inter-stimulus interval (ISI), while a neuron state refers to the spiking pattern of a neuron during a given ISI. The inter-stimulus interval is the interval between the onsets of two consecutive signals.

  To see how an evolved SNN transitions from one network state to the next, first the network states were identified (by hand) by giving all possible orderings of signals A, B, and C (from AAA to CCC) to the network and observing the spiking patterns of the neurons in the ISI. It is observed, that the network exhibits a similar network state when it receives either B or C, which corresponds to the start state of the FST. However, if the network receives an A, the network state changes, representing the ”hA” (had A) state of the FST. While in state hA, the network remains in hA state if it receives another A but the network state changes if it receives either B or C. If the network receives a B, the network state transforms into ”hAB” state, representing the next state of the FST. If the network receives a C, the network goes back to the start state. By following this process, the network successfully recognizes the correct pattern ABC. This mechanism can be extended to recognize longer patterns.

  In addition, it is observed that the network states of the SNNs that are evolved in the presence of noise have a one-to-one correspondence with the states of the FST [147]. Whereas in the absence of noise, the network states have many-to-one correspondences with the states of the FST. i.e. more than one network states map on the same FST state [148].

Furthermore, the network states of all perfect recognizers could be mapped onto the state of the deterministic FST; that is for each input signal the current network state has only one unique next state. On the contrary, the imperfect recognizer networks failed to map on the deterministic FST because for a given input signal the network may transform to multiple next states based on the preceding history or variation in the network. Consequently, the networks respond to incorrect patterns.

- The shortest possible recurrent connection observed in the nervous system is a self-connection. Self-connections (autaptic connections or autapses) are recurrent synaptic connections between the axon and dendrites or soma of a single neuron (either excitatory or inhibitory). Autapses were discovered five decades ago [130], in the mammalian brain. They are common in the brain and have been observed in the neocortex, hippocampus and cerebellum [6]. Recent studies suggest possible roles of autapses in synchronisation of the networks [139], flexible working memory networks [74] and coherence resonance [151]. However, their functional role remains unknown [140]. This study demonstrates a functional role of autapses (self-excitatory loops) in maintaining network states in the absence of input activity – a form of memory. Moreover, the relationship between the pattern length and the number of autapses in the network is identified i.e. at least n-1 self-excitatory loops are required in the minimal network to perform temporal pattern recognition in the presence of noise, where n is the length of the pattern. Without noise, the relationship between the number of self-excitatory loops and the length of the pattern cannot be determined.

- An SNN recognising a pattern of length n requires at least n interneurons in the minimal SNN. Three interneurons in the network are found to have a specialised role based on their behaviour: (i) the *lock* neuron prevents the output from spiking unless it is released by the penultimate signal in the correct pattern, (ii) the *switch* neuron is responsible for switching the network between the inter-signal states and the start state, and (iii) the *accept* neuron produces spikes in the output neuron when the network receives the last correct input, and sends a signal to the *switch* neuron, transforming the network back into the start state.

- Although the primary focus of this study is on evolving SNNs to recognize patterns composed of distinct signals, it is interesting to note that both the evolved and handcrafted networks can be readily adapted to recognize the regular expression of the form $AB^+C$, $AB^+C^+D$, $AB^+C^+D^+E$ and $AB^+C^+D^+E^+F$. For example, in the three-signal SNN, the excitatory connection from input channel B to the switch neuron prevents the repeated signal B, when this connection was removed, the network starts responding to ABBC, ABBBC, etc. Similarly in four-single SNN, the excitatory connection from input channels B and C to the switch neuron prevents the repeated signal B and C, respectively (Figure 7.1 b-c). Thus, these networks can be easily adapted to recognize patterns with repetitions.

- SNNs can be handcrafted to recognize a temporal pattern. Rules are defined for handcrafting network topologies to identify temporal patterns up to length 6, in a random stream of input signals.

- An incremental genetic algorithm can optimise connection weights of the handcrafted topology such that the output neuron of the network spikes only for the correct pattern received by the network inputs.

## 8.3 Future Work

The preliminary setup for evolving minimal SNNs to perform temporal pattern recognition presented in Chapter 4, and Chapter 5, can be adapted for temporal pattern classification tasks. In the case of pattern classification, a network requires more than one output neuron, depending on the number of classes. Each output neuron is designated for one class. The GA can be adapted to evolve a population of SNNs such that an output neuron spikes only for one class of patterns and remains silent for all others. Understanding how a classifier network functions is challenging, because a single connection in the network may contribute to the recognition of more than one pattern. Similarly, a neuron might have several roles in the network.

An adaptive exponential integrate-and-fire (AdEx) neuron can generate several spiking behaviours based on different values of initial parameters, e.g. tonic spiking, bursting, spike once, delayed spiking etc. [88]. This study is limited to the parameters generating tonic spiking behaviour for a constant input current. It is not clear how to exploit the rich spike generation mechanism of AdEx neurons in these minimal networks. This will involve transforming a neuron into a different state by changing its behaviour from one spiking state to another.

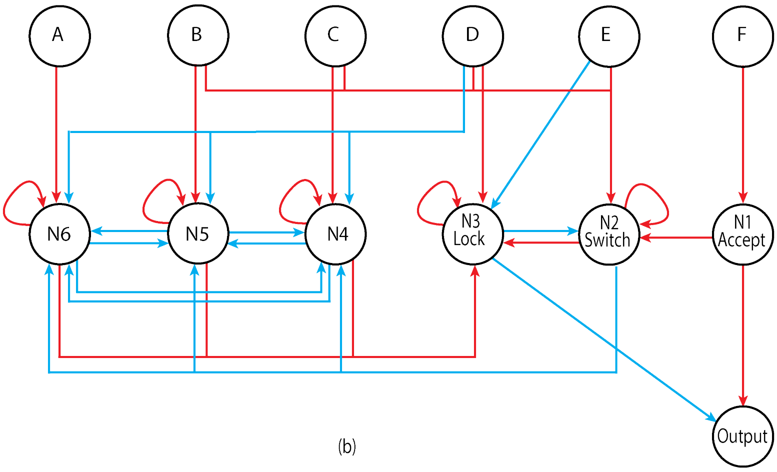
In the presence of noise (both intrinsic and extrinsic), the SNNs obtained using the artificial-evolution in GReaNs had a similar structure, i.e. (i) all perfect recognisers maintained network states with self-excitatory loops (autapses), (ii) the penultimate state was always maintained by an absence of activity in the network. (ii) all networks were found to have a lock, a switch and an accept neuron. It would be interesting to explore different solutions with less number of active neurons by tuning the parameters of the genetic algorithm to exhaustively explore the search space.

The rules for handcrafting the network topologies defined in Chapter 7 are limited to recognising patterns up to length 6. This is because the maximum number of interneurons representing a state in the network increases with the length of the pattern. In future studies, it would be interesting to explore the possibility of handcrafting networks with an upper limit on the number of active neurons representing network states, invariant of the length of the pattern. Networks with a small number of active neurons are energy efficient. Furthermore, the connection weights in the handcrafted topology strengthen with the number of active neurons. SNNs with strong synaptic connections between neurons are less likely to optimise due to the maximum limit on the neuronal spike rate.

Processing more extended patterns with the proposed solution suffers from strong synaptic weights in the network which results in highly active neurons. It could be solved by incorporating short-term potentiation, i.e. a short-term increase (facilitation) or decrease (depression) in synaptic strengths. Facilitation refers to a short-term increase in the synaptic strength of a pre-synaptic neuron when it fires.

| State | Neurons |
|---|---|
| hA: | N6+N3 |
| hAB: | N5+N3 |
| hABC: | N4+N3 |
| hABCD: | N3 |
| hABCDE: | Quiet |
| hABCDEF/Start: | N3+N2 |

(a)



(b)

Figure 8.1: (a) The table shows the network states, e.g. when the network receives an input signal A, it goes to the $hA$ state, represented by only two active neurons $N6 + N3$. (b) Handcrafted network topology for recognising signal of length 6 with a maximum of 2 active neurons representing a state.

Hence, improving the likelihood of the postsynaptic neuron to elicit spikes in response to successive inputs. On the contrary, depression refers to a short-term decrease in the synaptic strength of a pre-synaptic neuron when it fires with high frequency. Thus, reducing the likelihood of postsynaptic neurons to fire spike(s) in response to successive inputs [121].

A preliminary handcrafted topology for recognising a pattern of length 6 with a maximum of 2 active neurons expressing a network state is presented in Figure 8.1b. The connection weights of this topology can be optimised using the incremental genetic algorithm (described in Chapter 7, Section 7.2) to identify the correct pattern ABCDEF in an input stream created randomly by concatenating patterns of the form ABCXXX and XXXDEF, where XXX is DEF and ABC, respectively. However, the algorithm could not find any perfect recogniser that could identify the correct pattern in a random input stream of signals (A, B, C, D, E, and F). Handcrafting network topology with an upper limit on the number of active neurons representing a network state requires further investigation.

LSTM (long short-term memory) is widely regarded as the preferred choice for processing time series data. They have shown remarkable performance to handle large sequential data with long-term dependencies [112]. The standard LSTM model employs three gates: the forget gate, the input gate, and the output gate [50]. Since its inception, many LSTM variants have been proposed [153]. A recent study has introduced a non-linear spiking neural P system that employs a modified LSTM model for processing time series data (LSTM-SNP). Unlike the standard LSTM model, the LSTM-SNP employs three non-linear gates: the reset gate, the consumption gate, and the generation gate. These gates determine the extent to which the previous state needs to be reset, consumed, and the number of output spikes generated, respectively [79].

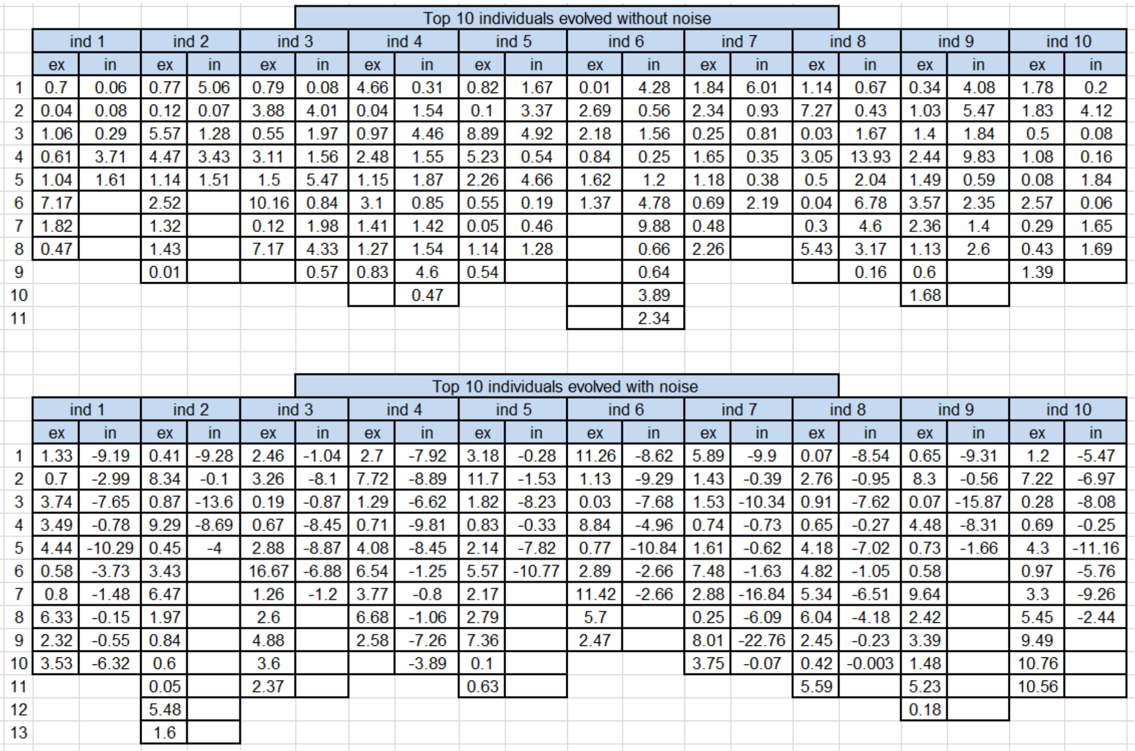
Due to the complex nature of recurrent SNNs, a large amount of the literature focuses on training one-layer SNNs [46, 86]. However, there has been a growing interest in recent years in training multi-layer SNNs [111]. Moreover, the recent exciting work on the spiking LSTM models has gained the interest of the computational neuroscience community [21, 43, 79, 80]. In biological neurons, the mechanism of spike generation is controlled by different ions that move through specific gates in the cell membrane [51]. Similarly, in the standard LSTM model, the information is controlled by a gating mechanism [50]. Therefore, I believe that this work can be extended to a multi-layer architecture, where the identified properties of neurons such as locking, switching, and accepting a network state can be adapted to develop a new novel variant of spiking LSTM. In future, it would be interesting to investigate the possibility of wiring up the lock, switch, and accept gate for low-cost training and processing of large sequential data.

This work indicates the functional role of excitatory autapses in maintaining network states when the input activity is absent – a form of memory. However, the role of inhibitory autapses (self-inhibitory connections) in these minimal networks yet needs to be explored. In future work, it would interesting to identify the role of inhibitory autapses. One possible way is to encourage the formation of inhibitory autapses during evolution.

# Appendix A

## A.1 Networks Evolved in the Presence or Absence of Noise in the Membrane Potential of each Neuron in the Network

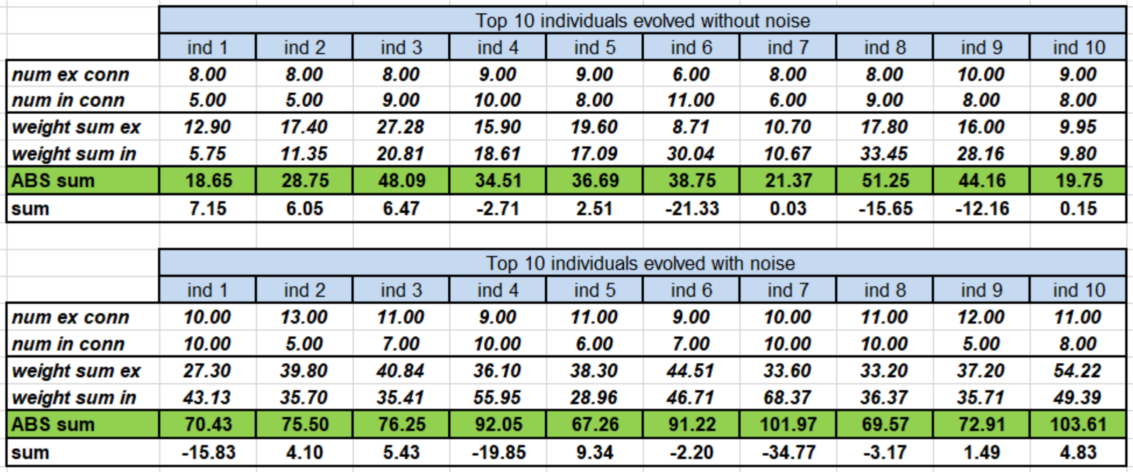
Network files and topologies of all individuals presented in Chapter 5 are available on GitHub: `https://github.com/yaqoobcs/greans-snns-tpr`.



| | Top 10 individuals evolved without noise | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ind 1 | | ind 2 | | ind 3 | | ind 4 | | ind 5 | | ind 6 | | ind 7 | | ind 8 | | ind 9 | | ind 10 | |
| | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in |
| 1 | 0.7 | 0.06 | 0.77 | 5.06 | 0.79 | 0.08 | 4.66 | 0.31 | 0.82 | 1.67 | 0.01 | 4.28 | 1.84 | 6.01 | 1.14 | 0.67 | 0.34 | 4.08 | 1.78 | 0.2 |
| 2 | 0.04 | 0.08 | 0.12 | 0.07 | 3.88 | 4.01 | 0.04 | 1.54 | 0.1 | 3.37 | 2.69 | 0.56 | 2.34 | 0.93 | 7.27 | 0.43 | 1.03 | 5.47 | 1.83 | 4.12 |
| 3 | 1.06 | 0.29 | 5.57 | 1.28 | 0.55 | 1.97 | 0.97 | 4.46 | 8.89 | 4.92 | 2.18 | 1.56 | 0.25 | 0.81 | 0.03 | 1.67 | 1.4 | 1.84 | 0.5 | 0.08 |
| 4 | 0.61 | 3.71 | 4.47 | 3.43 | 3.11 | 1.56 | 2.48 | 1.55 | 5.23 | 0.54 | 0.84 | 0.25 | 1.65 | 0.35 | 3.05 | 13.93 | 2.44 | 9.83 | 1.08 | 0.16 |
| 5 | 1.04 | 1.61 | 1.14 | 1.51 | 1.5 | 5.47 | 1.15 | 1.87 | 2.26 | 4.66 | 1.62 | 1.2 | 1.18 | 0.38 | 0.5 | 2.04 | 1.49 | 0.59 | 0.08 | 1.84 |
| 6 | 7.17 | | 2.52 | | 10.16 | 0.84 | 3.1 | 0.85 | 0.55 | 0.19 | 1.37 | 4.78 | 0.69 | 2.19 | 0.04 | 6.78 | 3.57 | 2.35 | 2.57 | 0.06 |
| 7 | 1.82 | | 1.32 | | 0.12 | 1.98 | 1.41 | 1.42 | 0.05 | 0.46 | | 9.88 | 0.48 | | 0.3 | 4.6 | 2.36 | 1.4 | 0.29 | 1.65 |
| 8 | 0.47 | | 1.43 | | 7.17 | 4.33 | 1.27 | 1.54 | 1.14 | 1.28 | | 0.66 | 2.26 | | 5.43 | 3.17 | 1.13 | 2.6 | 0.43 | 1.69 |
| 9 | | | 0.01 | | | 0.57 | 0.83 | 4.6 | 0.54 | | | 0.64 | | | | 0.16 | 0.6 | | 1.39 | |
| 10 | | | | | | | | 0.47 | | | | 3.89 | | | | | 1.68 | | | |
| 11 | | | | | | | | | | | | 2.34 | | | | | | | | |

| | Top 10 individuals evolved with noise | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ind 1 | | ind 2 | | ind 3 | | ind 4 | | ind 5 | | ind 6 | | ind 7 | | ind 8 | | ind 9 | | ind 10 | |
| | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in |
| 1 | 1.33 | -9.19 | 0.41 | -9.28 | 2.46 | -1.04 | 2.7 | -7.92 | 3.18 | -0.28 | 11.26 | -8.62 | 5.89 | -9.9 | 0.07 | -8.54 | 0.65 | -9.31 | 1.2 | -5.47 |
| 2 | 0.7 | -2.99 | 8.34 | -0.1 | 3.26 | -8.1 | 7.72 | -8.89 | 11.7 | -1.53 | 1.13 | -9.29 | 1.43 | -0.39 | 2.76 | -0.95 | 8.3 | -0.56 | 7.22 | -6.97 |
| 3 | 3.74 | -7.65 | 0.87 | -13.6 | 0.19 | -0.87 | 1.29 | -6.62 | 1.82 | -8.23 | 0.03 | -7.68 | 1.53 | -10.34 | 0.91 | -7.62 | 0.07 | -15.87 | 0.28 | -8.08 |
| 4 | 3.49 | -0.78 | 9.29 | -8.69 | 0.67 | -8.45 | 0.71 | -9.81 | 0.83 | -0.33 | 8.84 | -4.96 | 0.74 | -0.73 | 0.65 | -0.27 | 4.48 | -8.31 | 0.69 | -0.25 |
| 5 | 4.44 | -10.29 | 0.45 | -4 | 2.88 | -8.87 | 4.08 | -8.45 | 2.14 | -7.82 | 0.77 | -10.84 | 1.61 | -0.62 | 4.18 | -7.02 | 0.73 | -1.66 | 4.3 | -11.16 |
| 6 | 0.58 | -3.73 | 3.43 | | 16.67 | -6.88 | 6.54 | -1.25 | 5.57 | -10.77 | 2.89 | -2.66 | 7.48 | -1.63 | 4.82 | -1.05 | 0.58 | | 0.97 | -5.76 |
| 7 | 0.8 | -1.48 | 6.47 | | 1.26 | -1.2 | 3.77 | -0.8 | 2.17 | | 11.42 | -2.66 | 2.88 | -16.84 | 5.34 | -6.51 | 9.64 | | 3.3 | -9.26 |
| 8 | 6.33 | -0.15 | 1.97 | | 2.6 | | 6.68 | -1.06 | 2.79 | | 5.7 | | 0.25 | -6.09 | 6.04 | -4.18 | 2.42 | | 5.45 | -2.44 |
| 9 | 2.32 | -0.55 | 0.84 | | 4.88 | | 2.58 | -7.26 | 7.36 | | 2.47 | | 8.01 | -22.76 | 2.45 | -0.23 | 3.39 | | 9.49 | |
| 10 | 3.53 | -6.32 | 0.6 | | 3.6 | | | -3.89 | 0.1 | | | | 3.75 | -0.07 | 0.42 | -0.003 | 1.48 | | 10.76 | |
| 11 | | | 0.05 | | 2.37 | | | | 0.63 | | | | | | 5.59 | | 5.23 | | 10.56 | |
| 12 | | | 5.48 | | | | | | | | | | | | | | 0.18 | | | |
| 13 | | | 1.6 | | | | | | | | | | | | | | | | | |

Figure A.1.1: Experimental setup I (chapter 5). The top (bottom) table shows the weights of the excitatory and inhibitory connections in the top 10 networks that were evolved without (with) noise. The network topologies of all individuals are available on GitHub.



| | Top 10 individuals evolved without noise | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ind 1 | ind 2 | ind 3 | ind 4 | ind 5 | ind 6 | ind 7 | ind 8 | ind 9 | ind 10 |
| ***num ex conn*** | ***8.00*** | ***8.00*** | ***8.00*** | ***9.00*** | ***9.00*** | ***6.00*** | ***8.00*** | ***8.00*** | ***10.00*** | ***9.00*** |
| ***num in conn*** | ***5.00*** | ***5.00*** | ***9.00*** | ***10.00*** | ***8.00*** | ***11.00*** | ***6.00*** | ***9.00*** | ***8.00*** | ***8.00*** |
| ***weight sum ex*** | ***12.90*** | ***17.40*** | ***27.28*** | ***15.90*** | ***19.60*** | ***8.71*** | ***10.70*** | ***17.80*** | ***16.00*** | ***9.95*** |
| ***weight sum in*** | ***5.75*** | ***11.35*** | ***20.81*** | ***18.61*** | ***17.09*** | ***30.04*** | ***10.67*** | ***33.45*** | ***28.16*** | ***9.80*** |
| **ABS sum** | **18.65** | **28.75** | **48.09** | **34.51** | **36.69** | **38.75** | **21.37** | **51.25** | **44.16** | **19.75** |
| **sum** | **7.15** | **6.05** | **6.47** | **-2.71** | **2.51** | **-21.33** | **0.03** | **-15.65** | **-12.16** | **0.15** |

| | Top 10 individuals evolved with noise | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ind 1 | ind 2 | ind 3 | ind 4 | ind 5 | ind 6 | ind 7 | ind 8 | ind 9 | ind 10 |
| ***num ex conn*** | ***10.00*** | ***13.00*** | ***11.00*** | ***9.00*** | ***11.00*** | ***9.00*** | ***10.00*** | ***11.00*** | ***12.00*** | ***11.00*** |
| ***num in conn*** | ***10.00*** | ***5.00*** | ***7.00*** | ***10.00*** | ***6.00*** | ***7.00*** | ***10.00*** | ***10.00*** | ***5.00*** | ***8.00*** |
| ***weight sum ex*** | ***27.30*** | ***39.80*** | ***40.84*** | ***36.10*** | ***38.30*** | ***44.51*** | ***33.60*** | ***33.20*** | ***37.20*** | ***54.22*** |
| ***weight sum in*** | ***43.13*** | ***35.70*** | ***35.41*** | ***55.95*** | ***28.96*** | ***46.71*** | ***68.37*** | ***36.37*** | ***35.71*** | ***49.39*** |
| **ABS sum** | **70.43** | **75.50** | **76.25** | **92.05** | **67.26** | **91.22** | **101.97** | **69.57** | **72.91** | **103.61** |
| **sum** | **-15.83** | **4.10** | **5.43** | **-19.85** | **9.34** | **-2.20** | **-34.77** | **-3.17** | **1.49** | **4.83** |

Figure A.1.2: Experimental setup I (chapter 5). The top (bottom) table shows the number and weight-sum of the excitatory and inhibitory connections evolved without (with) noise.

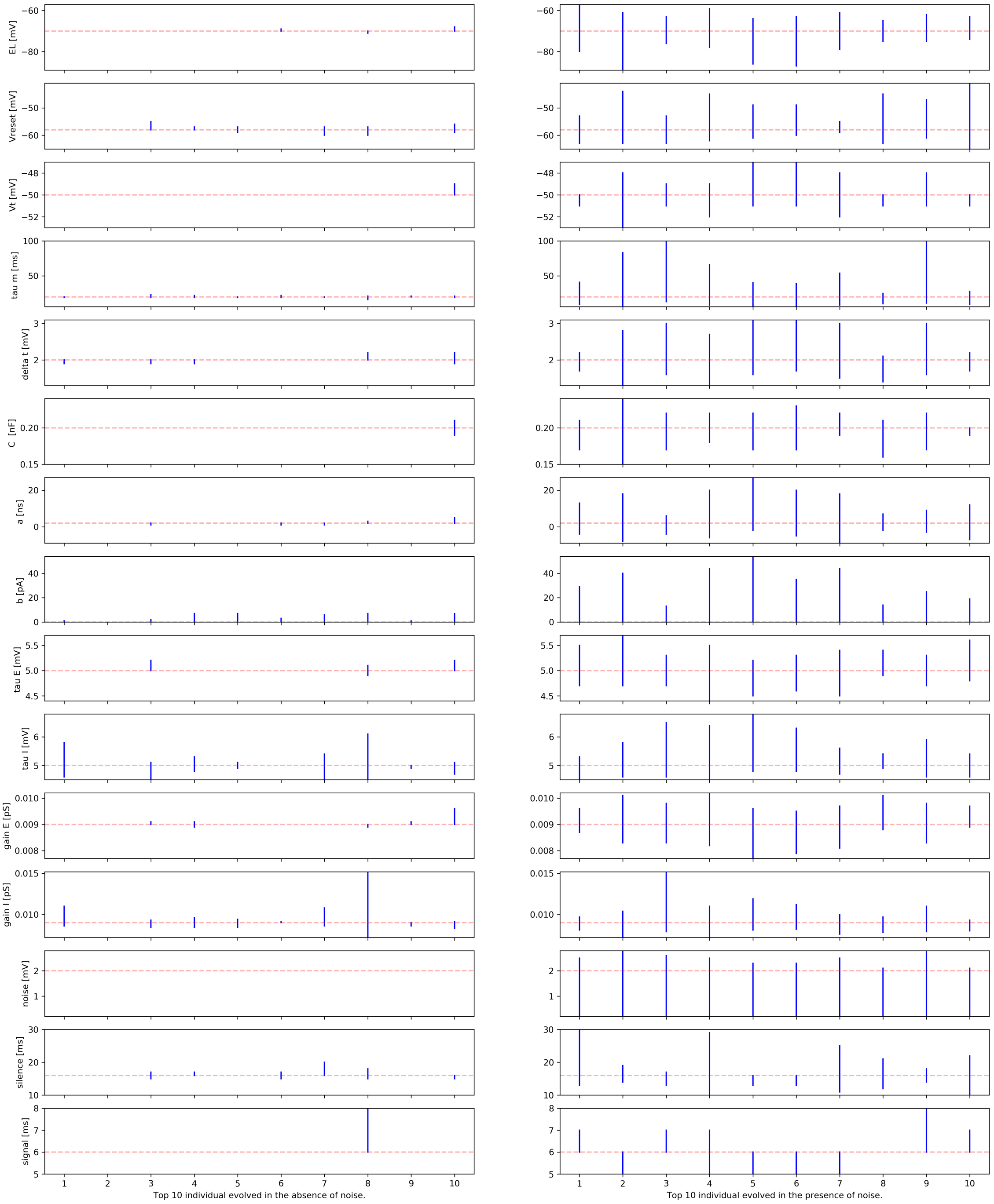
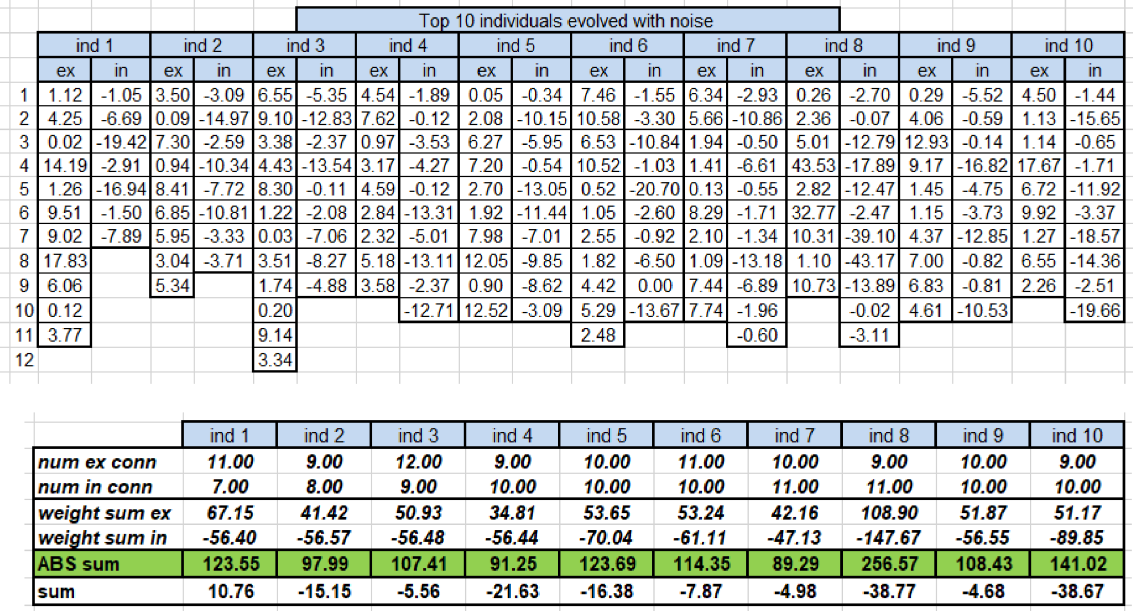


Figure A.1.3: Performance comparison of the top 10 networks evolved with and without noise to disturbed neuronal parameters and input conditions. The left column shows the top 10 individuals evolved without noise, numbered from 1 to 10 on the x-axis. The red dashed line in each subplot indicates the default value of the parameter for which the networks were evolved. For example, individual number 8 (the most robust individual evolved without noise) obtained robustness to signal length between 6 and 8 (default at 6). The right column shows the top 10 individuals evolved with noise, also numbered from 1 to 10. The plot clearly demonstrates that introducing noise during evolution promotes robustness in these minimal networks.



| | Top 10 individuals evolved with noise | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ind 1 | | ind 2 | | ind 3 | | ind 4 | | ind 5 | | ind 6 | | ind 7 | | ind 8 | | ind 9 | | ind 10 | |
| | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in | ex | in |
| 1 | 1.12 | -1.05 | 3.50 | -3.09 | 6.55 | -5.35 | 4.54 | -1.89 | 0.05 | -0.34 | 7.46 | -1.55 | 6.34 | -2.93 | 0.26 | -2.70 | 0.29 | -5.52 | 4.50 | -1.44 |
| 2 | 4.25 | -6.69 | 0.09 | -14.97 | 9.10 | -12.83 | 7.62 | -0.12 | 2.08 | -10.15 | 10.58 | -3.30 | 5.66 | -10.86 | 2.36 | -0.07 | 4.06 | -0.59 | 1.13 | -15.65 |
| 3 | 0.02 | -19.42 | 7.30 | -2.59 | 3.38 | -2.37 | 0.97 | -3.53 | 6.27 | -5.95 | 6.53 | -10.84 | 1.94 | -0.50 | 5.01 | -12.79 | 12.93 | -0.14 | 1.14 | -0.65 |
| 4 | 14.19 | -2.91 | 0.94 | -10.34 | 4.43 | -13.54 | 3.17 | -4.27 | 7.20 | -0.54 | 10.52 | -1.03 | 1.41 | -6.61 | 43.53 | -17.89 | 9.17 | -16.82 | 17.67 | -1.71 |
| 5 | 1.26 | -16.94 | 8.41 | -7.72 | 8.30 | -0.11 | 4.59 | -0.12 | 2.70 | -13.05 | 0.52 | -20.70 | 0.13 | -0.55 | 2.82 | -12.47 | 1.45 | -4.75 | 6.72 | -11.92 |
| 6 | 9.51 | -1.50 | 6.85 | -10.81 | 1.22 | -2.08 | 2.84 | -13.31 | 1.92 | -11.44 | 1.05 | -2.60 | 8.29 | -1.71 | 32.77 | -2.47 | 1.15 | -3.73 | 9.92 | -3.37 |
| 7 | 9.02 | -7.89 | 5.95 | -3.33 | 0.03 | -7.06 | 2.32 | -5.01 | 7.98 | -7.01 | 2.55 | -0.92 | 2.10 | -1.34 | 10.31 | -39.10 | 4.37 | -12.85 | 1.27 | -18.57 |
| 8 | 17.83 | | 3.04 | -3.71 | 3.51 | -8.27 | 5.18 | -13.11 | 12.05 | -9.85 | 1.82 | -6.50 | 1.09 | -13.18 | 1.10 | -43.17 | 7.00 | -0.82 | 6.55 | -14.36 |
| 9 | 6.06 | | 5.34 | | 1.74 | -4.88 | 3.58 | -2.37 | 0.90 | -8.62 | 4.42 | 0.00 | 7.44 | -6.89 | 10.73 | -13.89 | 6.83 | -0.81 | 2.26 | -2.51 |
| 10 | 0.12 | | | | 0.20 | | | -12.71 | 12.52 | -3.09 | 5.29 | -13.67 | 7.74 | -1.96 | | -0.02 | 4.61 | -10.53 | | -19.66 |
| 11 | 3.77 | | | | 9.14 | | | | | | 2.48 | | | -0.60 | | -3.11 | | | | |
| 12 | | | | | 3.34 | | | | | | | | | | | | | | | |

| | ind 1 | ind 2 | ind 3 | ind 4 | ind 5 | ind 6 | ind 7 | ind 8 | ind 9 | ind 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| ***num ex conn*** | ***11.00*** | ***9.00*** | ***12.00*** | ***9.00*** | ***10.00*** | ***11.00*** | ***10.00*** | ***9.00*** | ***10.00*** | ***9.00*** |
| ***num in conn*** | ***7.00*** | ***8.00*** | ***9.00*** | ***10.00*** | ***10.00*** | ***10.00*** | ***11.00*** | ***11.00*** | ***10.00*** | ***10.00*** |
| ***weight sum ex*** | ***67.15*** | ***41.42*** | ***50.93*** | ***34.81*** | ***53.65*** | ***53.24*** | ***42.16*** | ***108.90*** | ***51.87*** | ***51.17*** |
| ***weight sum in*** | ***-56.40*** | ***-56.57*** | ***-56.48*** | ***-56.44*** | ***-70.04*** | ***-61.11*** | ***-47.13*** | ***-147.67*** | ***-56.55*** | ***-89.85*** |
| **ABS sum** | **123.55** | **97.99** | **107.41** | **91.25** | **123.69** | **114.35** | **89.29** | **256.57** | **108.43** | **141.02** |
| **sum** | **10.76** | **-15.15** | **-5.56** | **-21.63** | **-16.38** | **-7.87** | **-4.98** | **-38.77** | **-4.68** | **-38.67** |

Figure A.1.4: Experimental setup II (chapter 5). The top table shows the weights of the excitatory and inhibitory connections in the top 10 networks that were evolved with noise. The bottom table shows the number and weight-sum of the excitatory and inhibitory connections. The network topologies of all individuals are available on GitHub.

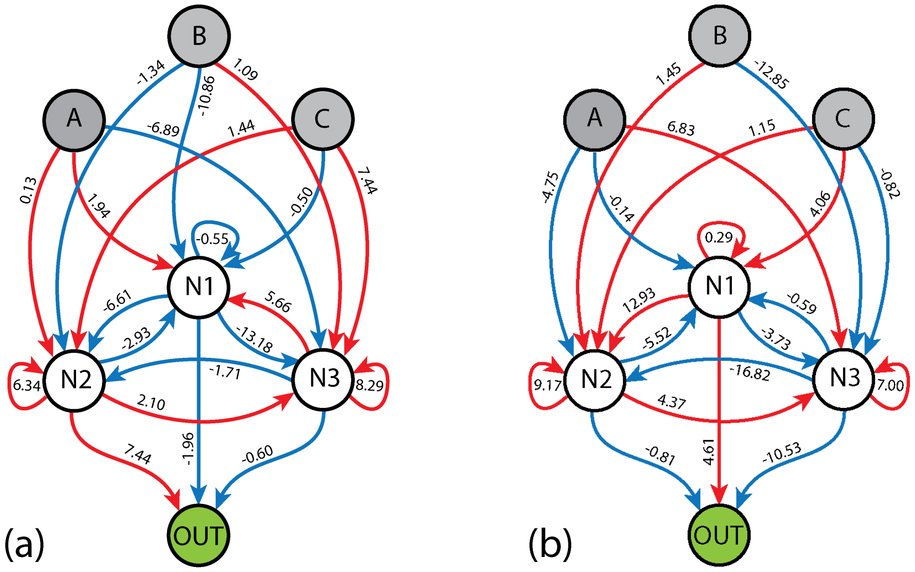
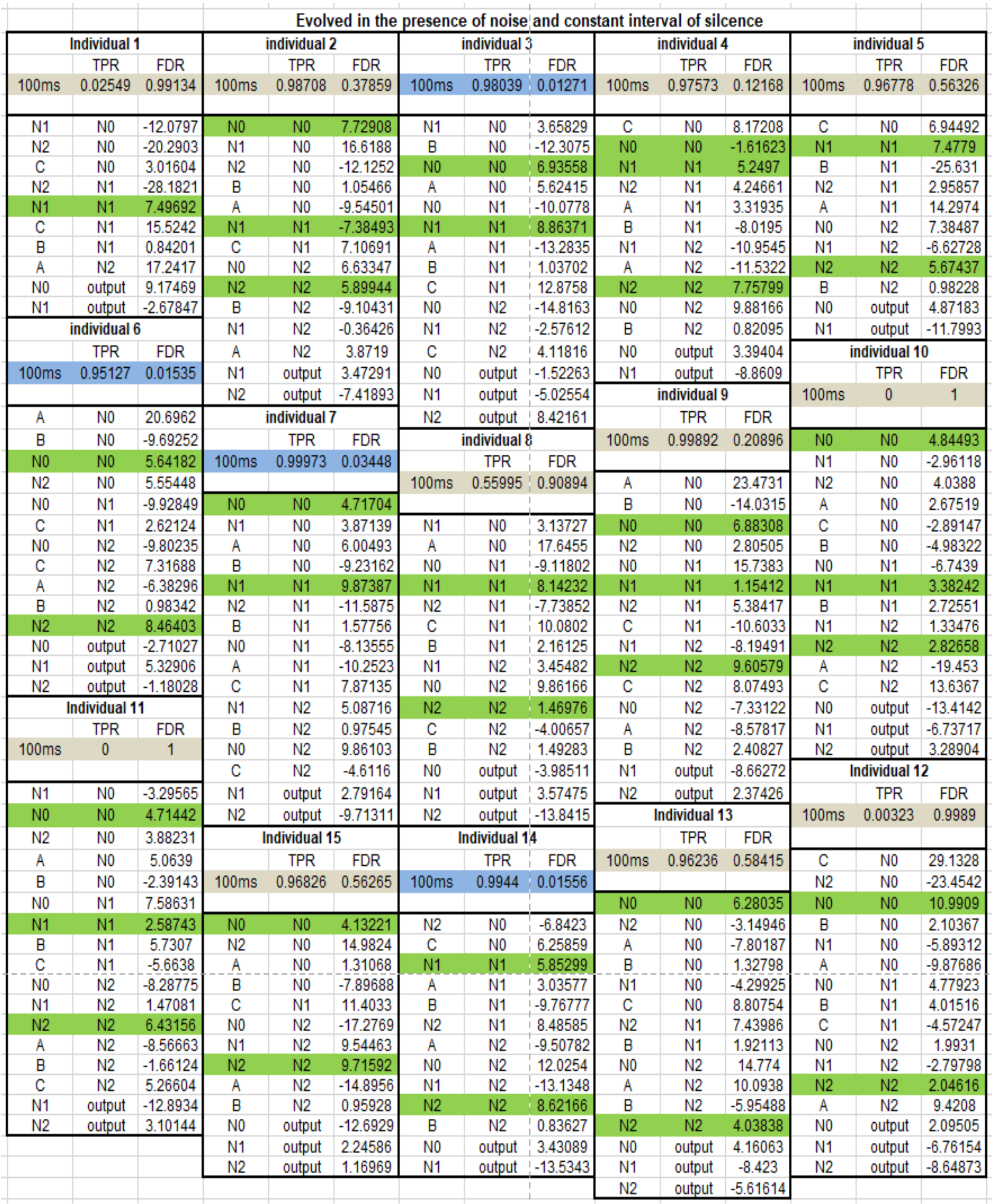


Figure A.1.5: (a) Network topology of individual no. 7 which is unable to maintain network states in the absence of input activity. (b) Network topology of individual no. 9 which is capable of maintaining network states in the absence of input activity. It is not clear whether network state maintenance is solely a property of self-excitatory loops in the network, because both networks have two strong excitatory loops, yet one is capable of maintaining network states while the other is not.

## A.2 Networks Evolved in the Presence of Noise in the Membrane Potential of each Neuron in the Network and random variation of silence interval between the input signals

Network files and topologies of all individuals presented in Chapter 6 are available on GitHub: `https://github.com/yaqoobcs/greans-snns-tpr`.



Evolved in the presence of noise and constant interval of silcence

| Individual 1 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.02549 | 0.99134 |
| N1 | N0 | -12.0797 |
| N2 | N0 | -20.2903 |
| C | N0 | 3.01604 |
| N2 | N1 | -28.1821 |
| N1 | N1 | 7.49692 |
| C | N1 | 15.5242 |
| B | N1 | 0.84201 |
| A | N2 | 17.2417 |
| N0 | output | 9.17469 |
| N1 | output | -2.67847 |

| individual 2 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.98708 | 0.37859 |
| N0 | N0 | 7.72908 |
| N1 | N0 | 16.6188 |
| N2 | N0 | -12.1252 |
| B | N0 | 1.05466 |
| A | N0 | -9.54501 |
| N1 | N1 | -7.38493 |
| C | N1 | 7.10691 |
| N0 | N2 | 6.63347 |
| N2 | N2 | 5.89944 |
| B | N2 | -9.10431 |
| N1 | N2 | -0.36426 |
| A | N2 | 3.8719 |
| N1 | output | 3.47291 |
| N2 | output | -7.41893 |

| individual 3 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.98039 | 0.01271 |
| N1 | N0 | 3.65829 |
| B | N0 | -12.3075 |
| N0 | N0 | 6.93558 |
| A | N0 | 5.62415 |
| N0 | N1 | -10.0778 |
| N1 | N1 | 8.86371 |
| A | N1 | -13.2835 |
| B | N1 | 1.03702 |
| C | N1 | 12.8758 |
| N0 | N2 | -14.8163 |
| N1 | N2 | -2.57612 |
| C | N2 | 4.11816 |
| N0 | output | -1.52263 |
| N1 | output | -5.02554 |
| N2 | output | 8.42161 |

| individual 4 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.97573 | 0.12168 |
| C | N0 | 8.17208 |
| N0 | N0 | -1.61623 |
| N1 | N1 | 5.2497 |
| N2 | N1 | 4.24661 |
| A | N1 | 3.31935 |
| B | N1 | -8.0195 |
| N1 | N2 | -10.9545 |
| A | N2 | -11.5322 |
| N2 | N2 | 7.75799 |
| N0 | N2 | 9.88166 |
| B | N2 | 0.82095 |
| N0 | output | 3.39404 |
| N1 | output | -8.8609 |

| individual 5 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.96778 | 0.56326 |
| C | N0 | 6.94492 |
| N1 | N1 | 7.4779 |
| B | N1 | -25.631 |
| N2 | N1 | 2.95857 |
| A | N1 | 14.2974 |
| N0 | N2 | 7.38487 |
| N1 | N2 | -6.62728 |
| N2 | N2 | 5.67437 |
| B | N2 | 0.98228 |
| N0 | output | 4.87183 |
| N1 | output | -11.7993 |

| individual 6 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.95127 | 0.01535 |
| A | N0 | 20.6962 |
| B | N0 | -9.69252 |
| N0 | N0 | 5.64182 |
| N2 | N0 | 5.55448 |
| N0 | N1 | -9.92849 |
| C | N1 | 2.62124 |
| N0 | N2 | -9.80235 |
| C | N2 | 7.31688 |
| A | N2 | -6.38296 |
| B | N2 | 0.98342 |
| N2 | N2 | 8.46403 |
| N0 | output | -2.71027 |
| N1 | output | 5.32906 |
| N2 | output | -1.18028 |

| individual 7 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.99973 | 0.03448 |
| N0 | N0 | 4.71704 |
| N1 | N0 | 3.87139 |
| A | N0 | 6.00493 |
| B | N0 | -9.23162 |
| N1 | N1 | 9.87387 |
| N2 | N1 | -11.5875 |
| B | N1 | 1.57756 |
| N0 | N1 | -8.13555 |
| A | N1 | -10.2523 |
| C | N1 | 7.87135 |
| N1 | N2 | 5.08716 |
| B | N2 | 0.97545 |
| N0 | N2 | 9.86103 |
| C | N2 | -4.6116 |
| N1 | output | 2.79164 |
| N2 | output | -9.71311 |

| individual 8 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.55995 | 0.90894 |
| N1 | N0 | 3.13727 |
| A | N0 | 17.6455 |
| N0 | N1 | -9.11802 |
| N1 | N1 | 8.14232 |
| N2 | N1 | -7.73852 |
| C | N1 | 10.0802 |
| B | N1 | 2.16125 |
| N1 | N2 | 3.45482 |
| N0 | N2 | 9.86166 |
| N2 | N2 | 1.46976 |
| C | N2 | -4.00657 |
| B | N2 | 1.49283 |
| N0 | output | -3.98511 |
| N1 | output | 3.57475 |
| N2 | output | -13.8415 |

| individual 9 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.99892 | 0.20896 |
| A | N0 | 23.4731 |
| B | N0 | -14.0315 |
| N0 | N0 | 6.88308 |
| N2 | N0 | 2.80505 |
| N0 | N1 | 15.7383 |
| N1 | N1 | 1.15412 |
| N2 | N1 | 5.38417 |
| C | N1 | -10.6033 |
| N1 | N2 | -8.19491 |
| N2 | N2 | 9.60579 |
| C | N2 | 8.07493 |
| N0 | N2 | -7.33122 |
| A | N2 | -8.57817 |
| B | N2 | 2.40827 |
| N1 | output | -8.66272 |
| N2 | output | 2.37426 |

| individual 10 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0 | 1 |
| N0 | N0 | 4.84493 |
| N1 | N0 | -2.96118 |
| N2 | N0 | 4.0388 |
| A | N0 | 2.67519 |
| C | N0 | -2.89147 |
| B | N0 | -4.98322 |
| N0 | N1 | -6.7439 |
| N1 | N1 | 3.38242 |
| B | N1 | 2.72551 |
| N1 | N2 | 1.33476 |
| N2 | N2 | 2.82658 |
| A | N2 | -19.453 |
| C | N2 | 13.6367 |
| N0 | output | -13.4142 |
| N1 | output | -6.73717 |
| N2 | output | 3.28904 |

| Individual 11 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0 | 1 |
| N1 | N0 | -3.29565 |
| N0 | N0 | 4.71442 |
| N2 | N0 | 3.88231 |
| A | N0 | 5.0639 |
| B | N0 | -2.39143 |
| N0 | N1 | 7.58631 |
| N1 | N1 | 2.58743 |
| B | N1 | 5.7307 |
| C | N1 | -5.6638 |
| N0 | N2 | -8.28775 |
| N1 | N2 | 1.47081 |
| N2 | N2 | 6.43156 |
| A | N2 | -8.56663 |
| B | N2 | -1.66124 |
| C | N2 | 5.26604 |
| N1 | output | -12.8934 |
| N2 | output | 3.10144 |

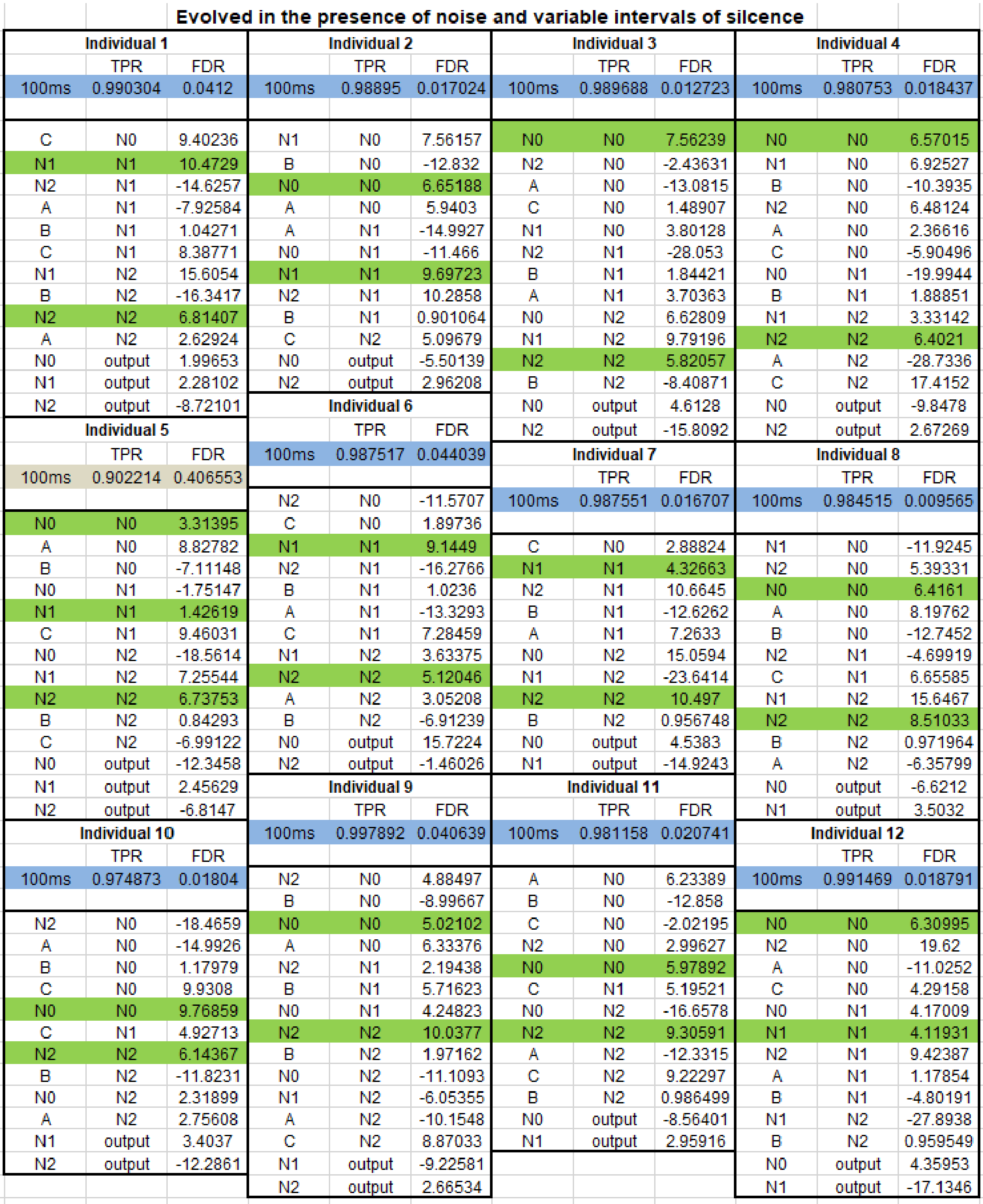
| Individual 12 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.00323 | 0.9989 |
| C | N0 | 29.1328 |
| N2 | N0 | -23.4542 |
| N0 | N0 | 10.9909 |
| B | N0 | 2.10367 |
| N1 | N0 | -5.89312 |
| A | N0 | -9.87686 |
| N0 | N1 | 4.77923 |
| B | N1 | 4.01516 |
| C | N1 | -4.57247 |
| N0 | N2 | 1.9931 |
| N1 | N2 | -2.79798 |
| N2 | N2 | 2.04616 |
| A | N2 | 9.4208 |
| N0 | output | 2.09505 |
| N1 | output | -6.76154 |
| N2 | output | -8.64873 |

| Individual 13 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.96236 | 0.58415 |
| N0 | N0 | 6.28035 |
| N2 | N0 | -3.14946 |
| A | N0 | -7.80187 |
| B | N0 | 1.32798 |
| N1 | N0 | -4.29925 |
| C | N0 | 8.80754 |
| N2 | N1 | 7.43986 |
| B | N1 | 1.92113 |
| N0 | N2 | 14.774 |
| A | N2 | 10.0938 |
| B | N2 | -5.95488 |
| N2 | N2 | 4.03838 |
| N0 | output | 4.16063 |
| N1 | output | -8.423 |
| N2 | output | -5.61614 |

| Individual 14 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.9944 | 0.01556 |
| N2 | N0 | -6.8423 |
| C | N0 | 6.25859 |
| N1 | N1 | 5.85299 |
| A | N1 | 3.03577 |
| B | N1 | -9.76777 |
| N2 | N1 | 8.48585 |
| A | N2 | -9.50782 |
| N0 | N2 | 12.0254 |
| N1 | N2 | -13.1348 |
| N2 | N2 | 8.62166 |
| B | N2 | 0.83627 |
| N0 | output | 3.43089 |
| N1 | output | -13.5343 |

| Individual 15 | | |
|---|---|---|
| | TPR | FDR |
| 100ms | 0.96826 | 0.56265 |
| N0 | N0 | 4.13221 |
| N2 | N0 | 14.9824 |
| A | N0 | 1.31068 |
| B | N0 | -7.89688 |
| C | N1 | 11.4033 |
| N0 | N2 | -17.2769 |
| N1 | N2 | 9.54463 |
| N2 | N2 | 9.71592 |
| A | N2 | -14.8956 |
| B | N2 | 0.95928 |
| N0 | output | -12.6929 |
| N1 | output | 2.24586 |
| N2 | output | 1.16969 |

Figure A.2.1: Experimental setup chapter 6. Networks were evolved in the presence of noise and constant (16 ms) silent intervals. The green bar shows the number of self-loops in the network. The color blue indicates the network's ability to maintain network states, while the color grey means the network is unable to maintain network state. Only individuals 3, 6, 7, and 14 could maintain network states. The network topologies of all individuals are available on GitHub.



Evolved in the presence of noise and variable intervals of silcence

**Individual 1**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.990304 | 0.0412 |
| C | N0 | 9.40236 |
| N1 | N1 | 10.4729 |
| N2 | N1 | -14.6257 |
| A | N1 | -7.92584 |
| B | N1 | 1.04271 |
| C | N1 | 8.38771 |
| N1 | N2 | 15.6054 |
| B | N2 | -16.3417 |
| N2 | N2 | 6.81407 |
| A | N2 | 2.62924 |
| N0 | output | 1.99653 |
| N1 | output | 2.28102 |
| N2 | output | -8.72101 |

**Individual 2**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.98895 | 0.017024 |
| N1 | N0 | 7.56157 |
| B | N0 | -12.832 |
| N0 | N0 | 6.65188 |
| A | N0 | 5.9403 |
| A | N1 | -14.9927 |
| N0 | N1 | -11.466 |
| N1 | N1 | 9.69723 |
| N2 | N1 | 10.2858 |
| B | N1 | 0.901064 |
| C | N2 | 5.09679 |
| N0 | output | -5.50139 |
| N2 | output | 2.96208 |

**Individual 3**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.989688 | 0.012723 |
| N0 | N0 | 7.56239 |
| N2 | N0 | -2.43631 |
| A | N0 | -13.0815 |
| C | N0 | 1.48907 |
| N1 | N0 | 3.80128 |
| N2 | N1 | -28.053 |
| B | N1 | 1.84421 |
| A | N1 | 3.70363 |
| N0 | N2 | 6.62809 |
| N1 | N2 | 9.79196 |
| N2 | N2 | 5.82057 |
| B | N2 | -8.40871 |
| N0 | output | 4.6128 |
| N2 | output | -15.8092 |

**Individual 4**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.980753 | 0.018437 |
| N0 | N0 | 6.57015 |
| N1 | N0 | 6.92527 |
| B | N0 | -10.3935 |
| N2 | N0 | 6.48124 |
| A | N0 | 2.36616 |
| C | N0 | -5.90496 |
| N0 | N1 | -19.9944 |
| B | N1 | 1.88851 |
| N1 | N2 | 3.33142 |
| N2 | N2 | 6.4021 |
| A | N2 | -28.7336 |
| C | N2 | 17.4152 |
| N0 | output | -9.8478 |
| N2 | output | 2.67269 |

**Individual 5**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.902214 | 0.406553 |
| N0 | N0 | 3.31395 |
| A | N0 | 8.82782 |
| B | N0 | -7.11148 |
| N0 | N1 | -1.75147 |
| N1 | N1 | 1.42619 |
| C | N1 | 9.46031 |
| N0 | N2 | -18.5614 |
| N1 | N2 | 7.25544 |
| N2 | N2 | 6.73753 |
| B | N2 | 0.84293 |
| C | N2 | -6.99122 |
| N0 | output | -12.3458 |
| N1 | output | 2.45629 |
| N2 | output | -6.8147 |

**Individual 6**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.987517 | 0.044039 |
| N2 | N0 | -11.5707 |
| C | N0 | 1.89736 |
| N1 | N1 | 9.1449 |
| N2 | N1 | -16.2766 |
| B | N1 | 1.0236 |
| A | N1 | -13.3293 |
| C | N1 | 7.28459 |
| N1 | N2 | 3.63375 |
| N2 | N2 | 5.12046 |
| A | N2 | 3.05208 |
| B | N2 | -6.91239 |
| N0 | output | 15.7224 |
| N2 | output | -1.46026 |

**Individual 7**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.987551 | 0.016707 |
| C | N0 | 2.88824 |
| N1 | N1 | 4.32663 |
| N2 | N1 | 10.6645 |
| B | N1 | -12.6262 |
| A | N1 | 7.2633 |
| N0 | N2 | 15.0594 |
| N1 | N2 | -23.6414 |
| N2 | N2 | 10.497 |
| B | N2 | 0.956748 |
| N0 | output | 4.5383 |
| N1 | output | -14.9243 |

**Individual 8**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.984515 | 0.009565 |
| N1 | N0 | -11.9245 |
| N2 | N0 | 5.39331 |
| N0 | N0 | 6.4161 |
| A | N0 | 8.19762 |
| B | N0 | -12.7452 |
| N2 | N1 | -4.69919 |
| C | N1 | 6.65585 |
| N1 | N2 | 15.6467 |
| N2 | N2 | 8.51033 |
| B | N2 | 0.971964 |
| A | N2 | -6.35799 |
| N0 | output | -6.6212 |
| N1 | output | 3.5032 |

**Individual 9**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.997892 | 0.040639 |
| N2 | N0 | 4.88497 |
| B | N0 | -8.99667 |
| N0 | N0 | 5.02102 |
| A | N0 | 6.33376 |
| N2 | N1 | 2.19438 |
| B | N1 | 5.71623 |
| N0 | N1 | 4.24823 |
| N2 | N2 | 10.0377 |
| B | N2 | 1.97162 |
| N0 | N2 | -11.1093 |
| N1 | N2 | -6.05355 |
| A | N2 | -10.1548 |
| C | N2 | 8.87033 |
| N1 | output | -9.22581 |
| N2 | output | 2.66534 |

**Individual 10**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.974873 | 0.01804 |
| N2 | N0 | -18.4659 |
| A | N0 | -14.9926 |
| B | N0 | 1.17979 |
| C | N0 | 9.9308 |
| N0 | N0 | 9.76859 |
| C | N1 | 4.92713 |
| N2 | N2 | 6.14367 |
| B | N2 | -11.8231 |
| N0 | N2 | 2.31899 |
| A | N2 | 2.75608 |
| N1 | output | 3.4037 |
| N2 | output | -12.2861 |

**Individual 11**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.981158 | 0.020741 |
| A | N0 | 6.23389 |
| B | N0 | -12.858 |
| C | N0 | -2.02195 |
| N2 | N0 | 2.99627 |
| N0 | N0 | 5.97892 |
| C | N1 | 5.19521 |
| N0 | N2 | -16.6578 |
| N2 | N2 | 9.30591 |
| A | N2 | -12.3315 |
| C | N2 | 9.22297 |
| B | N2 | 0.986499 |
| N0 | output | -8.56401 |
| N1 | output | 2.95916 |

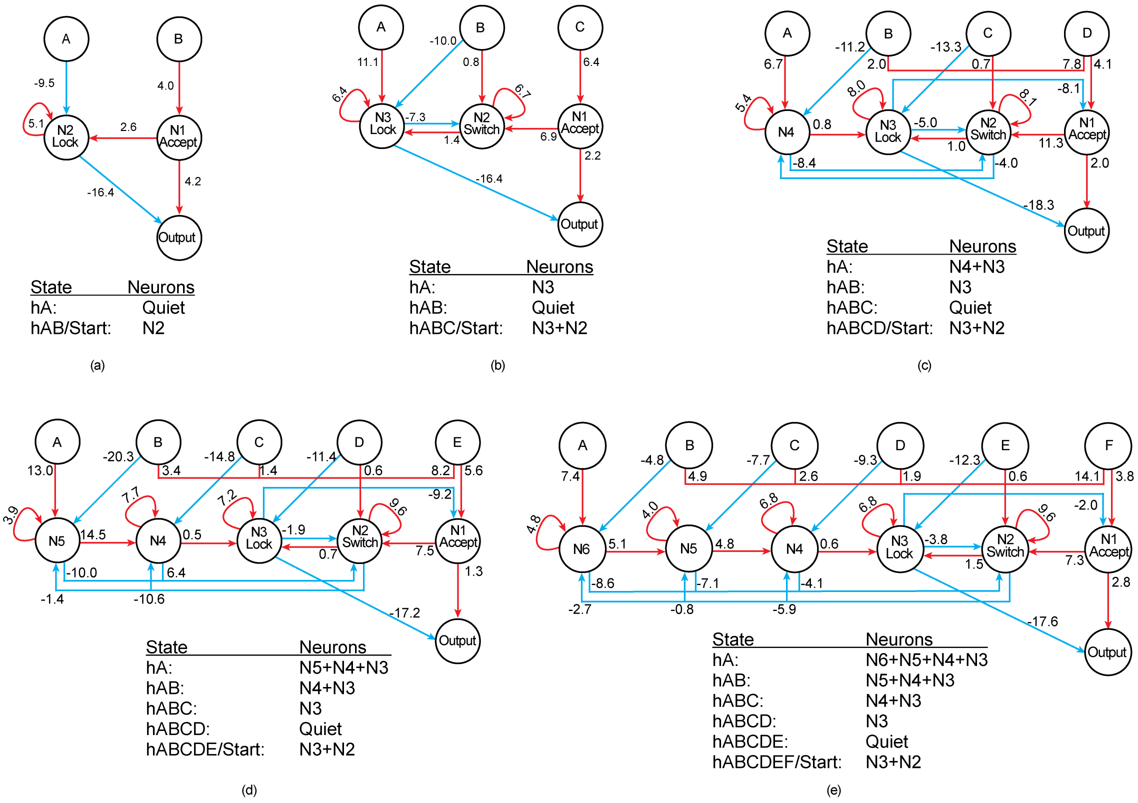
**Individual 12**

| | TPR | FDR |
|---|---|---|
| 100ms | 0.991469 | 0.018791 |
| N0 | N0 | 6.30995 |
| N2 | N0 | 19.62 |
| A | N0 | -11.0252 |
| C | N0 | 4.29158 |
| N0 | N1 | 4.17009 |
| N1 | N1 | 4.11931 |
| N2 | N1 | 9.42387 |
| A | N1 | 1.17854 |
| B | N1 | -4.80191 |
| N1 | N2 | -27.8938 |
| B | N2 | 0.959549 |
| N0 | output | 4.35953 |
| N1 | output | -17.1346 |

Figure A.2.2: Experimental setup chapter 6. Networks were evolved in the presence of noise and variable (16-32 ms) silent intervals. The green bar shows the number of self-loops in the network. The color blue indicates the network's ability to maintain network states, while the color grey means the network is unable to maintain network state. On;y individual 5 could not maintain network states. The network topologies of all individuals are available on GitHub.

## A.3 Network Topologies Handcrafted for Recognizing a Pattern of Length 3, 4, 5 and 6

Network topologies of all individuals presented in Chapter 7 are available on GitHub: `https://github.com/yaqoobcs/greans-snns-tpr`.



(a)

| State | Neurons |
|---|---|
| hA: | Quiet |
| hAB/Start: | N2 |

(b)

| State | Neurons |
|---|---|
| hA: | N3 |
| hAB: | Quiet |
| hABC/Start: | N3+N2 |

(c)

| State | Neurons |
|---|---|
| hA: | N4+N3 |
| hAB: | N3 |
| hABC: | Quiet |
| hABCD/Start: | N3+N2 |

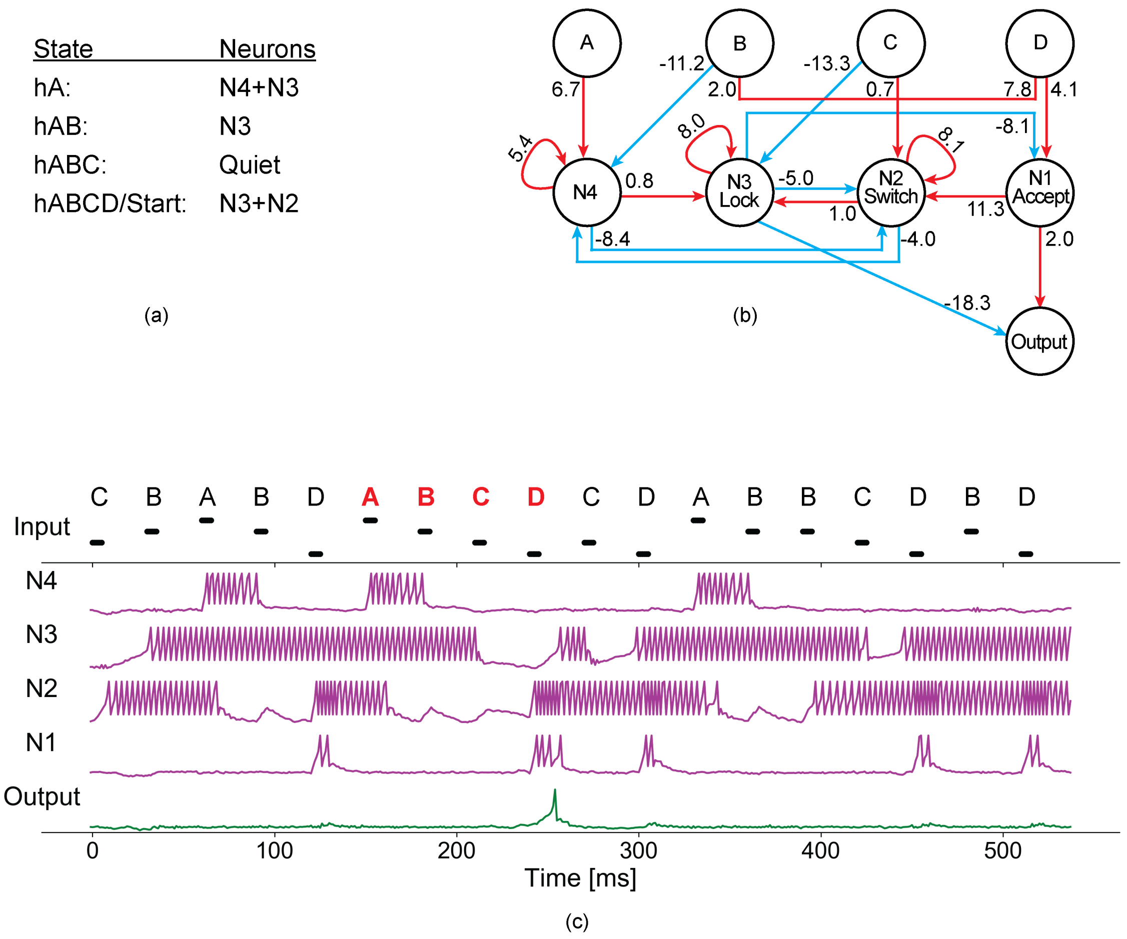
(d)

| State | Neurons |
|---|---|
| hA: | N5+N4+N3 |
| hAB: | N4+N3 |
| hABC: | N3 |
| hABCD: | Quiet |
| hABCDE/Start: | N3+N2 |

(e)

| State | Neurons |
|---|---|
| hA: | N6+N5+N4+N3 |
| hAB: | N5+N4+N3 |
| hABC: | N4+N3 |
| hABCD: | N3 |
| hABCDE: | Quiet |
| hABCDEF/Start: | N3+N2 |

Figure A.3.1: Handcrafted network topologies recognising patterns of lengths 2 to 6. The red (blue) connections indicate excitatory (inhibitory) connections. The number next to each connection shows the weight of the connection. The table below each network shows the network transitions between states, represented by the number of active neurons. The network in panel b is obtained with artificial-evolution. The topology is extended to recognise patterns up to lengths 4 (c), 5 (d), and 6 (e).



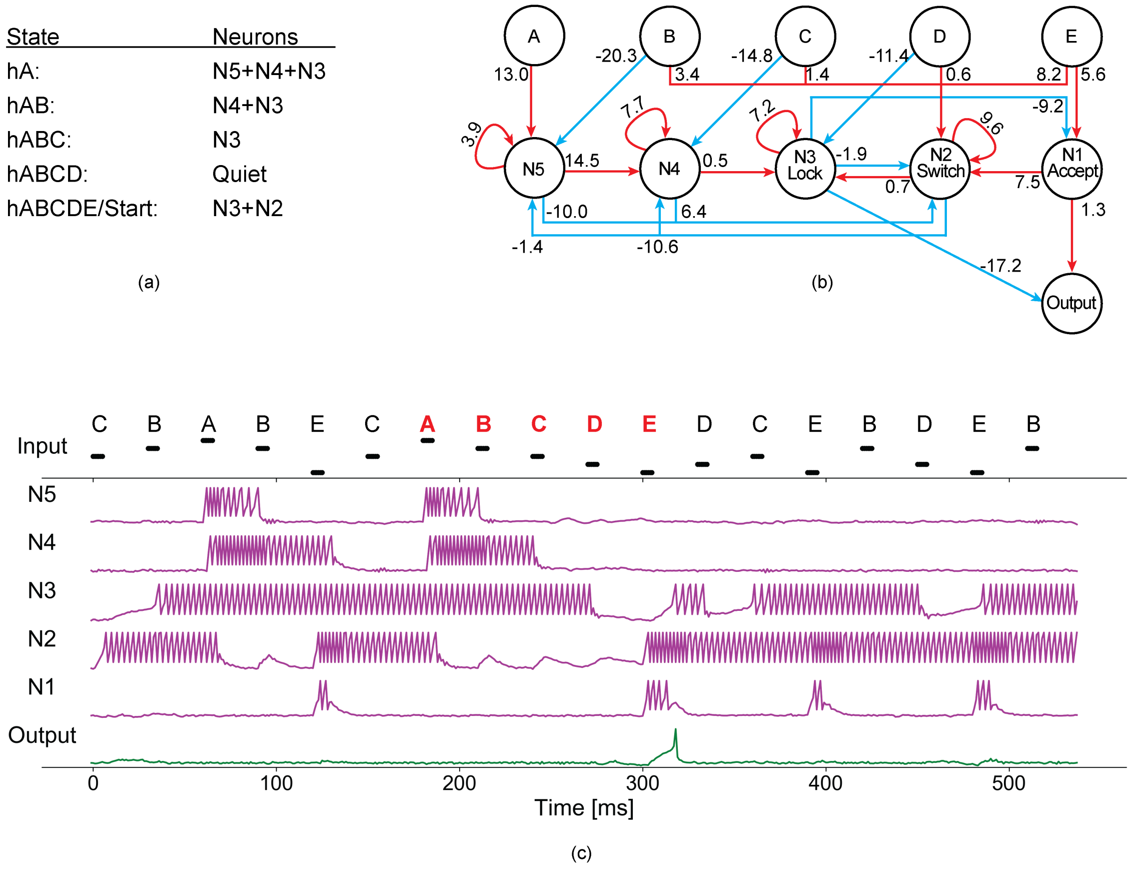
| State | Neurons |
|---|---|
| hA: | N4+N3 |
| hAB: | N3 |
| hABC: | Quiet |
| hABCD/Start: | N3+N2 |

(a)

(b)

(c)

Figure A.3.2: (a) Network states with corresponding active neurons. (b) Handcrafted network for recognising a pattern of length 4. (c) Behaviour of each neuron in the network.



| State | Neurons |
|---|---|
| hA: | N5+N4+N3 |
| hAB: | N4+N3 |
| hABC: | N3 |
| hABCD: | Quiet |
| hABCDE/Start: | N3+N2 |

Figure A.3.3: (a) Network states with corresponding active neurons. (b) Hand-crafted network for recognising a pattern of length 5. (c) Behaviour of each neuron in the network.

## A.4 Git Repositories

Code for optimizing connections weights of handcrafted network topologies is available on GitHub: `https://github.com/yaqoobcs/evosnns`

Network files and topologies of the evolved SNNs presented in Chapters 5, 6, and 7 are available on Github: `https://github.com/yaqoobcs/greans-snns-tpr`

# Appendix B

# Enclosed Papers

## B.1 M. Yaqoob and B. Wróbel (2017) Very Small Spiking Neural Networks Evolved to Recognize a Pattern in a Continuous Input Stream, IEEE Symposium Series on Computational Intelligence, IEEE SSCI 2017, Honolulu, Hawaii, USA, Nov. 27 to Dec. 1, 2017.

# Very Small Spiking Neural Networks Evolved to Recognize a Pattern in a Continuous Input Stream

Muhammad Yaqoob
Evolving Systems Laboratory,
Adam Mickiewicz University,
Poznan, Poland
email: yaqoob@evosys.org

Borys Wróbel
Evolving Systems Laboratory,
Adam Mickiewicz University,
Poznan, Poland,
and IOPAN, Sopot, Poland
email:wrobel@evosys.org

***Abstract*—We obtained, with artificial evolution, very small (one or two interneurons, one output neuron) spiking neural networks (SNNs) recognizing a simple temporal pattern in a continuous input stream. The patterns the network evolved to recognize consisted of three different signals. In other words, the task was equivalent to searching in a stream (sequence) of three symbols (say, ABBCACBC..) for a specific subsequence (ABC). The fitness function we used rewarded spiking after the occurrence of the correct pattern (subsequence), and penalized spikes elsewhere. We found out that the networks did not go below two interneurons when they evolved to solve this task with a brief interval of silence between signals. However—surprisingly—for a longer interval of silences between signals the task could be accomplished with just one interneuron. We then analyzed how the spiking networks work by mapping the states of the network onto states of Finite State Machines—a general model of computation on time series. Our long term goal is to understand the mechanisms governing the neural networks that accomplish computational tasks in a way that is robust to noise and damage.**

***Index Terms*—artificial evolution, complex networks, evolutionary algorithm, minimal cognition, spiking neural networks, temporal pattern recognition**

## I. Introduction

All animals need to process signals with a temporal structure, and this is true for all senses, including smell [1], sight [2], and perhaps especially hearing [3]. The animal neural circuits must process these signals in an online manner, and understanding how biological neural networks accomplish this task is one of the central interests of neuroscience. At the same time, the neuroscientific insights inform bio-inspired systems in artificial life/intelligence (robotics, computer vision, and machine learning) that also process continuous signals in real time.

In animal neuronal circuits and in artificial spiking neural networks information about the stimuli (input) is encoded in timing of discrete events (spikes). It is the temporal activity of neurons that is central for informational processing in the brain [4]–[10].

A formal computational model, finite state machine (FSM) is frequently used for analyzing computations on time series—to be more precise, finite state transducer, FSM that operates on sequences of symbols in an online manner, constantly producing an output (for a formal definition of FSMs, see [11]). Previous work on creating FSMs based on recurrent spiking neural networks (SNNs) [12]–[14] considered large multilayer networks (an alternative to induction of FSMs based on SNNs directly is to consider rate-based recurrent neural networks, for which literature abounds [13], and convert such a network to a spiking one [15]).

Understanding how the network transforms temporal information into its state is not essential per se for building bio-inspired systems. It is sufficient that readout neurons can extract this information. But how these networks 'work' is of course intellectually vexing, and properly within the scope of artificial life. An understanding of computing properties of spiking networks inspired by biology, even if much simplified, is necessary for the possible understanding of the mechanisms in biological circuits.

In this paper, we evolve both the topology and the weights in very small SNNs, hoping that the analysis of a variety of solutions will allow us to derive general rules on how such a simple, but not trivial, computational task can be accomplished by networks with just a few spiking neurons.

Recognition of temporal patterns requires temporal storage of the stimulus or delays [16]–[20]. Our model allows for recurrent connections in the networks we evolve, and for delays caused by synaptic delays. However, the time scales we use in this paper are such that long-term storage of states is not necessary—and the information what signals were received persists in the network only as long as the neurons' variables (such as voltages) do not change closer to their steady-state equilibrium values (in the case of voltages, the effective rest potential, see below).

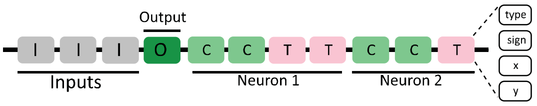


Fig. 1. The encoding of a network in a linear genome. See text for details.

This work was supported by the Polish National Science Center (project EvoSN, UMO-2013/08/M/ST6/00922). MY acknowledges the support of the KNOW RNA Research Center in Poznan (No. 01/KNOW2/2014)

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

## II. The model

### A. SNNs and their encoding in the genomes; the input and output to the networks

The evolving SNNs [21], [22] in our model are encoded in linear genomes in a way that is inspired by biological gene networks, and in principle permits evolving networks with unlimited number of links and nodes. In this paper, however, we limit the number of interneurons to two to allow for faster evolution. Firstly, simulation of smaller networks takes less time. Secondly, limiting network size reduces the search space. Thirdly, we would like to understand how the networks work, which is much easier for smaller networks. (Limiting the networks to one interneuron prevented evolvability.) Nodes in the network function not like genes but like biological neurons, sending discrete events (spikes) to the neurons they are connected to. Received spike(s) may elicit a spike in a receiving (that is, postsynaptic) neuron. The neurons we use in this paper have four hidden variables each (voltage, $V$; adaptation, $w$; excitatory, $g_E$, and inhibitory conductance, $g_I$) whose values are determined using Euler integration with 1 ms time step (the names and values of the parameters are in Table I):

$$\begin{aligned} C\frac{dV}{dt} &= -g_I(V-E_I) - g_E(V-E_E) \\ &\quad -g_L(V-E_L) + g_L\Delta_T e^{(\frac{V-V_T}{\Delta_T})} - w + I \quad (1) \\ \tau_w\frac{dw}{dt} &= a(V-E_L) - w \quad (2) \\ \frac{dg_E}{dt} &= \frac{-g_E}{\tau_E} \quad (3) \\ \frac{dg_I}{dt} &= \frac{-g_I}{\tau_I} \quad (4) \end{aligned}$$

.

A spike is generated when the voltage ($V$) crosses a threshold ($V > V_{th}$). After $V > V_{th}$, $V$ changes to $V_r$, and $w$ to $w + b$).

In a neuron that receives a spike, $g_E$ (for excitatory connections) or $g_I$ (for inhibitory ones) is increased by the value of the synaptic gain (5 nS in this paper) multiplied by the absolute value of the synaptic weight. The offset current was non-zero only for the Output neuron, bringing it closer to spiking than the interneurons.

TABLE I: AdEx Parameters in This Paper

| Parameter | Value |
|---|---|
| $\tau_{E/I}$ excitatory/inhibitory time constant | 5 ms |
| $g_L$ total leak conductance | 10nS |
| $E_L$ effective rest potential | -70 mV |
| $E_I$ inhibitory reversal potential | -70 mV |
| $E_E$ excitatory reversal potential | 0 mV |
| $\Delta_T$ threshold slope factor | 2 mV |
| $V_T$ effective threshold potential | -50mV |
| $C$ total capacitance | 0.2 nF |
| $a$ adaptation conductance | 2 nS |
| $b$ spike-triggered adaptation | 0 pA |
| $\tau_w$ adaptation time constant | 30 ms |
| $V_r$ reset voltage | -58 mV |
| $V_{th}$ spike detection threshold | 0 mV |
| $I$ offset current for the Output neuron | 0.1 nA |

Such adaptive exponential integrate-and-fire (AdEx) neurons [23] (with conductance-based synapses) can show a variety of behaviors [23]; we have chosen the parameters (Table I) that respond with tonic spiking to constant input current higher than a certain threshold [23].

The genome in our model is a vector of elements that can be either $input$, $output$, $cis$, or $trans$ (Fig. 1); the term cis, taken from biology, refers to the fact that changing a $cis$ element can affect directly the spiking of only the neuron encoded by this element and the other ($cis$ and $trans$) elements next to it; while changing a $trans$ element can have a direct effect on the neurons encoded anywhere in the genome. In each linear genome, the first four elements are three $inputs$, and one $output$ elements (this facilitates implementation of deletions and duplications, see below). Each element has three numbers: a flag $s$ (+1 or -1), and two $coordinates$ (defining a point corresponding to the element in an abstract 2-D affinity space).

In order to obtain the connections in the network, first an intermediate multigraph is constructed. During this construction, only the links from $input$ to $cis$, from $trans$ to $cis$, and from $trans$ to $output$ are allowed. They are present if the Euclidean distance, $d_{i,j}$, between the points in the affinity space associated with elements $i$ and $j$ is below 5. The associated contribution to the weight between two neurons is $s_i s_j \frac{2(5-d_{i,j})}{10d_{i,j}+1}$ (a contribution has a maximum value when the two points overlap in the abstract affinity space, and is positive if both signs are the same, negative if different). In other words, since a particular interneuron can be encoded by more than one $cis$ and more than one $trans$ elements, there can be parallel edges in the multigraph connecting (i) a particular Input node (encoded by a particular $input$ element each) to an interneuron, (ii) two interneurons, and (iii) an interneuron to the Output (encoded by one $output$ element).

Then, the multigraph is converted to a simple graph of the SNN, by converting each set of parallel edges to single edges, summing their weight contributions to obtain the synaptic weight. A connection in the SNN is excitatory if the synaptic weight is positive, and inhibitory if the weight is negative. Self-connections can occur if one of the $trans$ element coding a given interneuron connects to one of the $cis$ elements coding this neuron.

We will consider detection of temporal patterns—subsequence of signals or symbols, ABC—in a continuous input stream—a long sequence of symbols (say, BCAC**ABC**CABCC...). Each symbol (say, A) will make a corresponding Input node (encoded by a corresponding $input$ element) active for 4 ms (a spike is emitted by Input node in every step), and each symbol is followed by 8 ms silence (so 500 symbols in the input sequence will give 6000 ms of sensory input) in the input sequence with brief intervals or 100 ms (resulting in 52000 ms of input for 500 symbols) for the input sequence with long intervals. The task of the network will be to respond with at least one spike in the silence (8 ms or 100 ms) following C after ABC is received.

To decode the two interneurons, any $trans$ elements after $output$ are ignored, then the following group of $cis$ elements

TABLE II: The Recognition Efficiency for the Top 10 SNNs Evolved for Input with 8 ms Intervals Between Symbols, and the Number of the Network States

| | | | | number of network states | | | | |
|---|---|---|---|---|---|---|---|---|
| Champion | Reward | Penalty | Problematic subsequences | S | hA | hAB | hABC | Total |
| B0 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| B1 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| B2 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| B3 | 1.0000 | 0.0000 | none | 4 | 2 | 1 | 1 | 8 |
| B4 | 0.9976 | 0.0000 | BAACBC sometimes | 3 | 2 | 1 | 1 | 7 |
| B5 | 0.9952 | 0.0002 | CBBBBC sometimes | 4 | 2 | 1 | 1 | 8 |
| B6 | 0.9502 | 0.0002 | ABABBA sometimes | 2 | 2 | 1 | 1 | 6 |
| B7 | 0.9774 | 0.0014 | BBCBC always | 3 | 3 | 1 | 1 | 8 |
| B8 | 0.9317 | 0.0022 | ACBBBC always | 4 | 4 | 2 | 1 | 11 |
| B9 | 0.6454 | 0.0003 | CCBABC never | 4 | 2 | 2 | 1 | 9 |

followed by $trans$ encodes the first interneuron (Fig. 1). Elements following the last $trans$ element of the second interneuron are again ignored.

### B. Artificial evolution

Each independent evolutionary run had 1000 generations with 300 individuals in a population, elitism (10 individuals) and size-two tournament. The population in the initial generation was created by generating random genomes coding for two interneurons. Each region in the genome coding one neuron had a random number of $cis$ and $trans$ elements (a number was drawn from $N[0,3]$, the normal distribution with mean 0 and standard deviation 3, and rounded to the closest positive integer), three $input$ elements and one $output$. The coordinates were generated so that each element was at a random distance (drawn from $N[0,10]$) from point $(0,0)$, in a random direction (drawn from a uniform distribution).

If a genome won the tournament, it was placed in the next generation after the following genetic operators were applied (elite genomes were not changed): with probability 0.008 an element could be chosen for mutation of coordinates (this mutation could move the point in the abstract affinity space associated to the element in a random direction, drawn from a uniform distribution, by a distance drawn from $N[0,3]$), and with probability 0.001 (0.00005) a genome could be chosen for duplication (deletion, respectively). The start element for duplication (deletion) and the insertion site for duplication were chosen randomly (with equal probability for all elements), excluding four elements: $output$ and $input$ elements. The number of consecutive elements to be duplicated (deleted) was drawn from a geometric distribution with mean 11. While $output$ and $input$ elements could not be duplicated (deleted), their coordinates could change, resulting in changes of the connections between the Input nodes and the interneurons, and between the interneurons and the Output neuron.

Because of the high duplication rate, genomes grew during evolution, but always encoded two interneurons. To save memory, a neutral operator (an operator that did not affect fitness) was applied at the end—elements from the first $trans$ or $cis$ element to the last $trans$ coding interneuron 2 were counted. The resulting number was used to determine how many elements beyond this last coding $trans$ element were kept; the rest were deleted.

Each individual in the population was evaluated on six sequences, each with 500 symbols: four containing all symbols (A, B, C) with equal probability (and thus about 7-15% of the input stream was taken up by ABC), and two sequences constructed by concatenating ABA and ABB (with equal probability), two triplets we found the most problematic to discriminate from ABC (since the sequence is continuous, it contains substrings BAA, AAB, BBA, BAB).

For each input, we have calculated $R$ (for reward)—fraction of intervals of silence in the input following the C in ABC during which the Output spiked (each such silence was 8 ms when we evolved for recognition with brief intervals or 100 ms for long intervals). We also calculated $P$ (penalty)—number of Output spikes at all other times, divided by the maximum number of spikes that could be produced then (our model neurons can spike at most every 2 ms when the integration step is 1 ms). We then calculated two values: $f_{fitness1} = 1-(R-20P)$ and $f_{fitness2} = 1 - \frac{R}{R-20P}$. We found that while using one of them as a fitness function did not quite work (it resulted in a very low yield), averaging the two numbers allowed for reasonable evolvability (in other words, the fitness function we used was $f_{fitness} = (f_{fitness1} + f_{fitness2})/2$; the fitness function was minimized by the artificial evolution).

## III. Results and Discussion

### A. Evolution of SNNs for input with brief intervals between symbols

The yield of perfect recognizers (networks who responded perfectly to the input) for 8 ms intervals between symbols in the input sequence was very low. We ran 200 independent runs, and then taken the best individual from each run (the champion) and ranked these 200 champions by fitness. We then chose ten best champions for further analysis. Out of these ten, four were perfect recognizers (2% yield)—they never spiked outside the interval of silence following ABC (Table II; all analyses after evolution were done using 1000 random input sequences with 500 symbols, and results averaged).

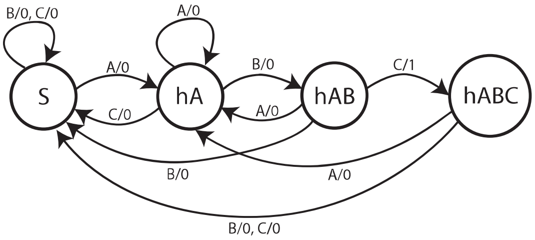


Fig. 2. Minimal FSM for recognizing ABC

The other champions did spike at other times. Surprisingly, the three champions following the perfect recognizers by rank had sometimes spikes after very specific subsequences (ending in ..BAACBC, ..CBBBBC, ..ABABBA; Table II) when some 500-symbol sequences containing them were given as an input stream. However, the networks did not spike after these subsequences always (as did the next two by rank, B7 and B8; Table II). In particular, champions B4, B5 and B6 did not spike in response to input streams that consisted of 16-symbol subsequences (six symbols listed in Table II and up to ten preceding symbols) of the 500-symbol sequences for which they spiked incorrectly. We plan to investigate this issue further in future work. B9, as B7 and B8 made mistakes consistently: B7 and B8 responded always after specific subsequences (ACBBBC, BBCBC), regardless of previous history, and B9 consistently failed to recognize only those ABCs that were preceded by CCB (Table II).

### *B. Analysis of the SNNs evolved for brief intervals, mapping the states of the SNNs to the states of a Finite State Machine*

The minimal possible FSM for recognizing the pattern ABC in a stream of symbols A, B and C has four states (Fig. 2). The nodes represent states of the FSM, and edges represent transition from one state to another state on receiving a symbol (A, B, or C); since we are considering transducers, an output symbol (0 or 1 after ”/”) is produced upon a transition (1 means the Output neuron spikes, 0 means no spike).

In this FSM, S is the state of the FSM after receiving a character that is not (potentially) a part of the pattern ABC. The FSM transitions from S to hA (had A) when A occurs, then to hAB (had AB) only when this A is followed by B, and lands in the state hABC (had ABC) when such AB is followed by a C. Only when the FSM transitions to the state hABC is the output of the FSM different (1) than for the other transitions (0). In order to know which state the FSM is in, one needs to look back in history (for example, if the symbol just received is B preceded by C, the FSM is in state S, as this is the only state that could be reached after a sequence of symbols that ends with CB).

Since we want to use the FSM model as an aid to understanding how the SNNs accomplish the task, the first step in the analysis was to analyze the spike patterns of the interneurons 1 and 2 (N1, N2), and Output neuron following a particular symbol and the symbols preceding it. To do so, we looked at the spiking pattern of these three neurons in windows of 12 ms (4 ms for the symbol followed by 8 ms silence; shifted by 1 ms for interneurons and 2 ms for Output—because each synapse introduces a delay). If two different subsequences of symbols result in two different spiking patterns, this indicates that the network is in a different computationally relevant state after each of these subsequences. However, two computationally relevant states can have the same spiking pattern (but, obviously, different values of neuron variables).

All of the networks we analyzed had only one spiking pattern upon receiving the final C in ABC. We constructed the FSM first going backwards in the input stream from numerous ABC patterns. For example, if there was always only one spiking pattern upon receiving AB, then it indicated the network had only one state hAB, and so on. Then we traced the input streams going forwards, confirming if each symbol resulted in the expected spiking pattern, following the transitions of the model FSM.

Let us take champion B0 (Table III, Fig. 3,4,5) as an example (a perfect recognizer for whom we have found the smallest number of states; Table II). When B may belong to ABC (when B follows A), the pattern of spiking is 000; when B follows C or B, the spiking pattern is 010. We can thus assign hAB to 000, and a state of type S to 010. Similarly for C—we can assign hABC to the pattern 131 (observed upon receiving the C in ABC), and another state of type S to 030 (observed for all the other inputs ending with C). Since there are two states of type S, the FSM is not minimal (Fig. 4).

However, it is not enough to consider only spikes. Both after A following any symbol or when B following A is received (states hA and hAB), neither neuron spikes. However, hA and hAB can be differentiated. When network receives B after A, the voltage of both interneurons rises (Fig. 3; this is caused by positive connections from Input B to both; Fig. 5). This higher voltage allows both interneurons to spike when C arrives (state hABC)—Input C connects very strongly to N1, so N1 spikes three times, and less strongly to N2, so N2 spikes once.

It is relatively straightforward (although laborious) to explain all the transitions between states by the network topology and weights (Fig. 5). The bottom line is that only N2 can activate the Output, so the Output can spike only after N2 is activated by C provided N1 was previously quiescent (strong negative connections from N1 to N2), and only if this C follows B, which is necessary to bring N2 to a higher subthreshold state.

Note that the FSM model is only an aid to the analysis of the SNN activity; and we do not think it is very fruitful to use this model to understand imperfect organizers.

The other perfect recognizers have a larger number of states, and champion B3 (Table IV, Fig. 6,7,8) has the most (Table II). In its SNN (Fig. 8) the negative connection from N1 to Output is much stronger than the positive one from N2 to Output. This means Output can only spike if N2 spikes but N1 does not. N2 spikes (twice) always after receiving B (Table IV), and spikes from N2 are the sole possible positive contribution to N1 (Fig. 8). This is why N1 spikes if the penultimate symbol is B (Table IV). There is one exception—when this B is followed by C

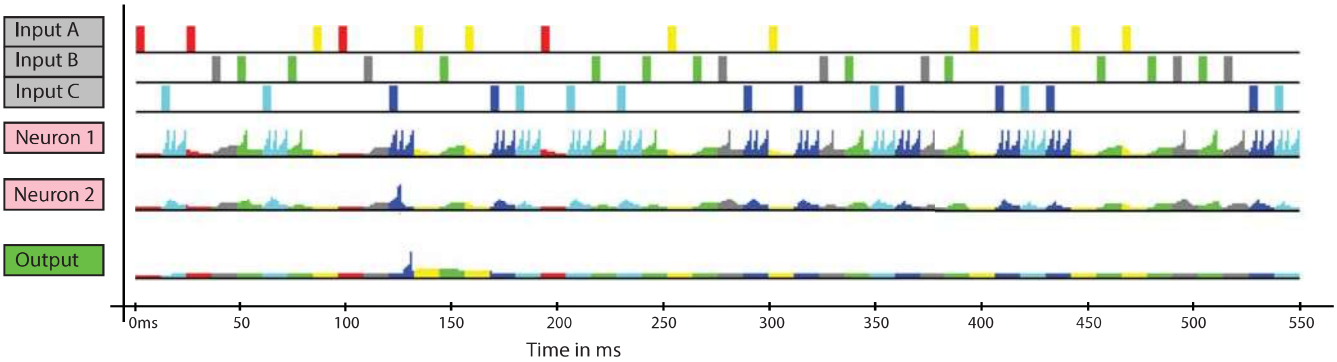


Fig. 3. The activity of the network for champion B0. The chunks of voltage traces are colored with the same color as the activity of the Input node that might have influenced it most directly, with two colors per symbol to distinguish the effect of 2 neighboring same symbols. The single spike of the Output neuron follows ABC that starts around 100 ms

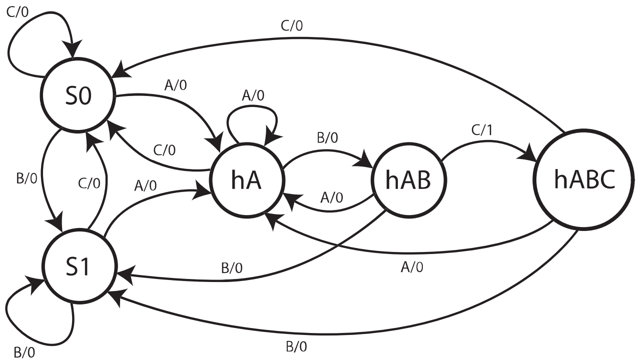


Fig. 4. The FSM whose states correspond to SNN states for champion B0

TABLE III: Analysis of the FSM States Corresponding to SNN States for Champion B0.

| | S0 | S1 | hA | hAB | hABC |
|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 3 | 1 | 0 | 0 | 3 |
| N2 spikes | 0 | 0 | 0 | 0 | 1 |
| Input ends | AC<br>BBC<br>CBC<br>CC | CB<br>BB | A | AB | ABC |

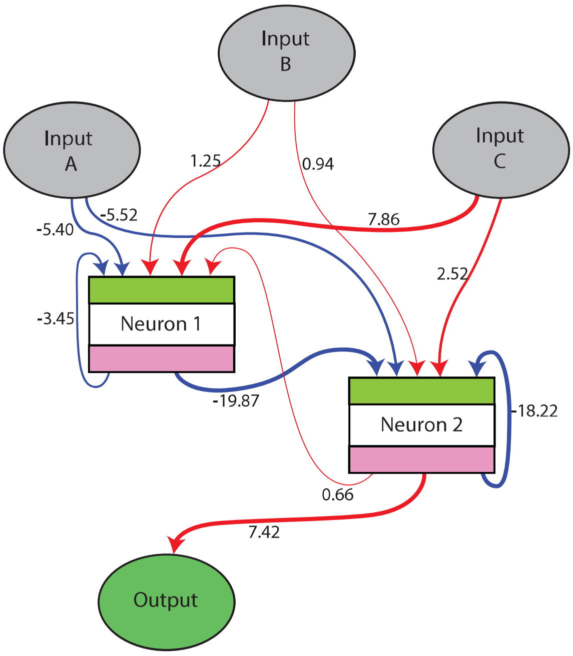


Fig. 5. The topology and weights of the network for champion B0.

in ABC, because of the strong negative connection from Input C to N1. But this is not the whole story, as N1 does spike after BBC and CBC (state S0). This can only be explained by going beyond the spiking patterns to analyze the variables of the neuron, and suggests that the network relies on quite precise values of the neuron's variables and precise timing of inputs. Consider, for example, an input sequence ending in ..AABC and one ending in ..ACBC. For ..AABC, the network will go (when AA is received) to hA2 (spiking pattern 000), then to hAB (002), and finally to hABC (100). For ..ACBC, the transitions will be S1 (input subsequence AC; spiking pattern 000) to S2 (CB; 002), and then S0 (BC; 010). Thus, the final C has a different effect on both the Output and N1, depending on the preceding symbols, even though the two network states that precede hABC have the same spiking patterns as the two network states that precede S0.

### *C. Robustness of the SNNs evolved for brief intervals to a changed input*

Our analysis of the networks indicates the networks do not maintain any states per se, the processing uses dynamic integration of the information in the input. Not surprisingly therefore, they are not very robust to small (1 ms) variations of the duration of signals and silences.

When the intervals were shortened to 7 ms, none of the top ten champions worked. For intervals extended to 9 ms, champion B0 continued to recognize ABC almost perfectly

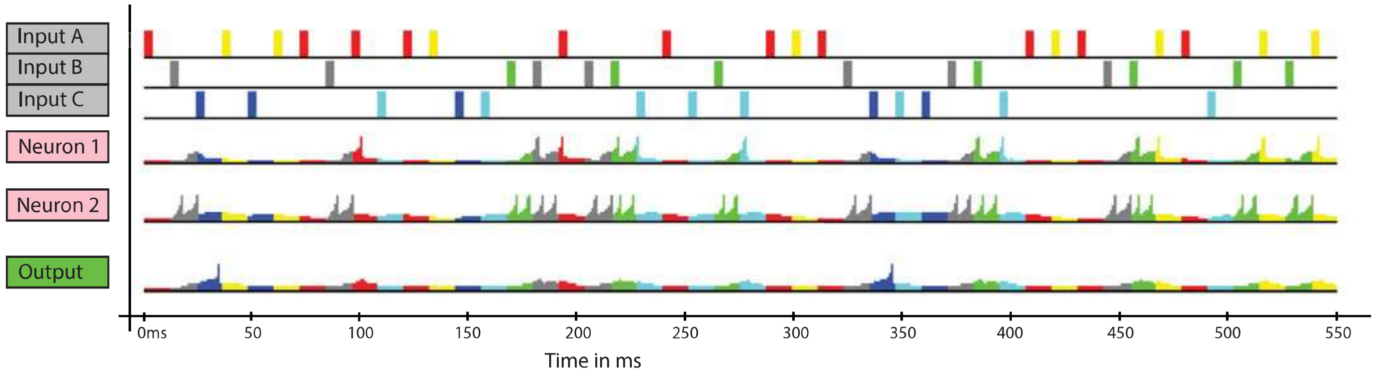


Fig. 6. The activity of the network for champion B3. The chunks of voltage traces are colored with the same color as the activity of the Input node that might have influenced it most directly, with two colors per symbol to distinguish the effect of two neighboring same symbols. Two spikes of the Output neuron follows ABC around 40 ms and 350 ms

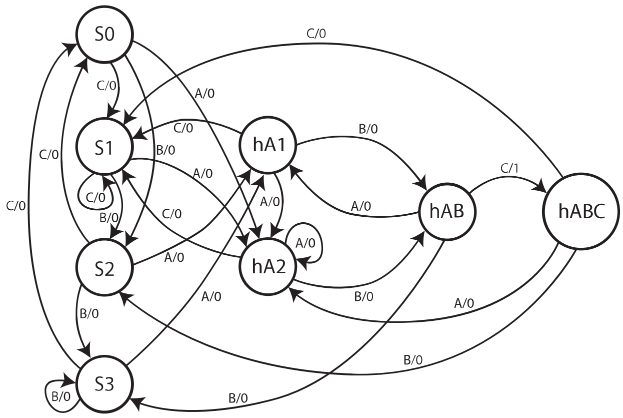


Fig. 7. The FSM whose states correspond to SNN states for champion B3

TABLE IV: Analysis of the FSM States Corresponding to SNN States for Champion B3

| | S0 | S1 | S2 | S3 | hA1 | hA2 | hAB | hABC |
|---|---|---|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| N2 spikes | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 |
| Input ends | BBC<br>CBC | AC<br>CC | CB | BB | BA | AA<br>CA | AB | ABC |

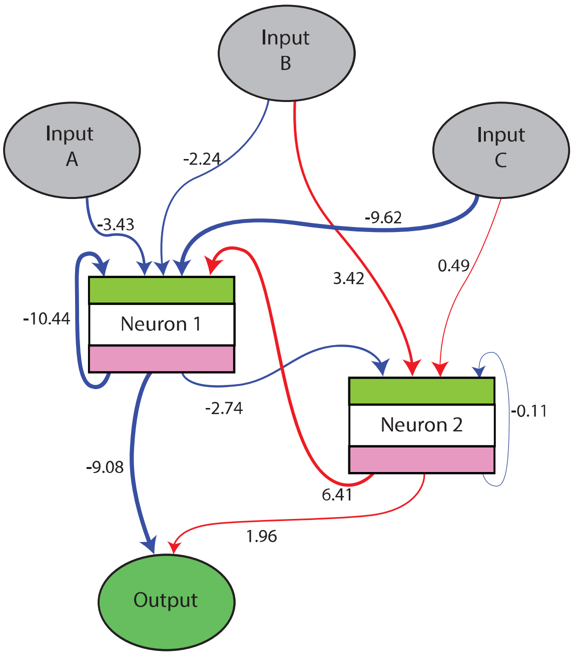


Fig. 8. The topology and weights of the network for champion B3

(reward 0.9995) but started to recognize BBC (penalty 0.0002). Champion B3 fared better (reward 0.9979, penalty 0.0000). Out of the remaining eight champions in the top ten, three showed some robustness: B4 (reward 0.9881, penalty 0.0000), B5 (reward 0.9993, penalty 0.0027), and B8 (reward 0.8470, penalty 0.0100).

We have then tested the robustness of ten champions evolved for brief intervals to extending the duration of the symbols by 1 ms, to 5 ms. All of the champions were fragile. However, when during such a 5 ms signal one random 1 ms was silent (we used five masks with random position of 0s: 01111, 10111,..., in a random order of masks for the input sequence), two champions showed some robustness (champion B0: reward 0.8938, penalty 0.0025; champion B5: reward 0.9246, penalty 0.0069). The Output neuron of both of these champions spiked less after symbols other then the last C in ABC when we used only three masks with flanking 1 ms (10111, 11011, 11101; champion B0: reward 0.8938, penalty 0.0025; champion B5: reward 0.8919, penalty 0.0050).

### *D. Evolution of SNNs for input with long intervals between symbols*

We then hypothesized that evolution of SNNs for input sequences with long intervals (100 ms) between symbols would result in higher robustness to the variation of the intervals,

TABLE V: The Recognition Efficiency for the Top SNNs Evolved for 100 ms Intervals Between Symbols, and the Number of Network States

| | | | | number of network states | | | | |
|---|---|---|---|---|---|---|---|---|
| Champion | Reward | Penalty | Problematic subsequences | S | hA | hAB | hABC | Total |
| L0 | 1.0000 | 0.0000 | none | 3 | 1 | 1 | 1 | 6 |
| L1 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| L2 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| L3 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| L4 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| L5 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| L6 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |
| L7 | 1.0000 | 0.0000 | none | 3 | 1 | 1 | 1 | 6 |
| L8 | 1.0000 | 0.0000 | none | 3 | 2 | 1 | 1 | 7 |
| L9 | 1.0000 | 0.0000 | none | 2 | 1 | 1 | 1 | 5 |

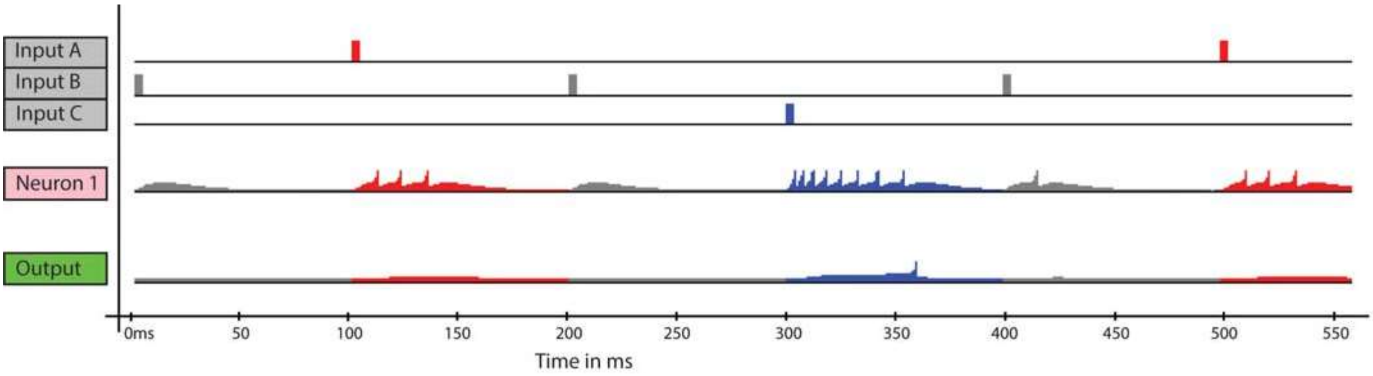


Fig. 9. The activity of the network for champion L5. The chunks of voltage traces are colored with the same color as the activity of the Input node that might have influenced it most directly, with two colors per symbol to distinguish the effect of two neighboring same symbols. The single spike of the Output neuron (after 350 ms) follows ABC that starts around 100 ms.

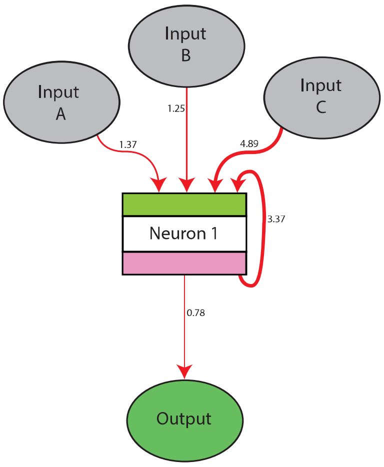


Fig. 10. The topology and weights of the network for champion L5.

TABLE VI: Analysis of the FSM States Corresponding to SNN States for Champion L5

| | S0 | S1 | hA | hAB | hABC |
|---|---|---|---|---|---|
| Output spikes | 0 | 0 | 0 | 0 | 1 |
| N1 spikes | 8 | 1 | 3 | 0 | 8 |
| Input ends | AC<br>BBC<br>CBC<br>CC | CB<br>BB | A | AB | ABC |

expecting that such longer intervals will promote maintenance of the network states, with more recurrent connections in the networks.

The yield in evolutionary runs with 100 ms intervals was much higher than for brief intervals—for 40 independent runs, we obtained ten perfect recognizers (Table V). Surprisingly, one of them (champion L5) had only one interneuron (Fig. 9, 10), with five states of the network (Table VI; with the FSM corresponding to L5 the same as the one corresponding to B0, Fig. 4; spikes were counted in 104 ms windows, not 12 ms windows as for the SNNs evolved for 8 ms intervals).

Contrary to our expectations, the SNNs evolved for long intervals were not very robust to shortening or lengthening of the interval between symbols. Only five sustained shortening the interval by 1 ms, to 99 ms (L0, L3, L5, L7, and L9 had reward above 0.99 and penalty lower than 0.0005), and only one, L9, showed robustness to shortening the interval to 90 ms (reward 1.0, penalty 0.0008; all others had reward 0.0, except L5, which recognized about half of ABC patterns in the sequence: reward 0.4493, penalty 0.0005). Longer intervals worked better—eight champions worked with 101 ms (L0, L5, L6, and L9 remained perfect, L1, L4, L3, and L6 recognized all ABCs correctly, but had non-zero penalty, all below 0.002), and even 110 ms (all eight champions had reward 1.0, and non-zero penalties below 0.002).

To sum up, our expectation that SNNs evolved for longer intervals would be able to maintain their state did not bear out. Their computation depended in a fragile way on the interplay between the state variables of the Output neuron, which balanced the Output's voltage precisely so that the Output neuron would spike only after the C in ABC.

For example, for the network with only one interneuron (L5), this interneuron produced 8 spikes both in the state S0 and hABC; in other words, after each C (Table VI). When this C followed AA, BB, CB, BA, and CA, the timing of these N1 spikes was exactly the same as when the C followed AB (in the correct pattern). However, the state variables of the Output neuron varied very slightly, so only for ABC these 8 spikes produced a spike in the Output. Not surprisingly therefore, even a very slight modification of AdEx parameters (0.1 ms smaller or larger $E_L$) caused all these networks to fail. In contrast, the SNNs evolved for 8 ms intervals could all sustain changing the $E_L$ from 70 mV to 69.5 and 70.5 mV.

Considering the fragility for the modification of the duration of silence intervals, it was surprising that the SNNs evolved for 100 ms intervals were more robust than those evolved for 8 ms to the modification of the signals. Champion L5 continued to recognize all ABC both when a 5 ms signal was masked with 5 random masks, and 3 random masks (with flanking 1s), but the Output neuron produced spikes in superfluous positions (after BBC, CBC; penalty 0.0012; the only other robust champion was L3: reward 0.9980, penalty 0.0013). Champion L5 worked also when the signal was extended to 5 ms without breaks (reward 1.0, penalty 0.0055; L4, L8 and L9 gave the same results as L5 here).

## IV. Conclusions and Future Work

We show that it is possible to evolve small networks recognizing simple temporal patterns in a continuous input. In future work we plan to extend the complexity of the patterns to longer ones, and regular expressions. We also plan to investigate if it is possible to evolve networks that maintain their states. Perhaps such network will evolve with noisy duration of the signals and intervals, or with a fitness function rewarding for state maintenance (for example, by rewarding for recurrent connections or for interneuron spikes throughout long intervals). Another possibility (and interesting in its own right) is the evolvability and robustness in the presence of noise on the neuron state variables, and when random changes to neuron parameters, and synaptic weights are introduced from one generation to the next. Our results obtained with a similar model—evolving artificial gene regulatory networks encoded in linear genomes—suggest that evolving networks in the presence of noise may promote robustness to other types of variability [24]; it will be interesting if a similar effect can be observed for evolving SNNs.

## References

[1] J. S. Isaacson, "Odor representations in mammalian cortical circuits," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 328–331, 2010.

[2] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature*, vol. 381, no. 6582, p. 520, 1996.

[3] P. Joris and T. C. Yin, "A matter of time: internal delays in binaural processing," *Trends in Neurosciences*, vol. 30, no. 2, pp. 70–78, 2007.

[4] W. Bialek, F. Rieke, R. de Ruyter van Steveninck, and D. Warland, "Reading a neural code," vol. 252, no. 5014, pp. 1854–1857, 1991.

[5] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, p. 76, 1996.

[6] G. Laurent, "Dynamical representation of odors by oscillating and evolving neural assemblies," *Trends in Neurosciences*, vol. 19, no. 11, pp. 489–496, 1996.

[7] F. Rieke, *Spikes: exploring the neural code*. MIT press, 1999.

[8] R. C. Decharms and A. Zador, "Neural representation and the cortical code," *Annual Review of Neuroscience*, vol. 23, no. 1, pp. 613–647, 2000.

[9] E. Ahissar and A. Arieli, "Figuring space by time," *Neuron*, vol. 32, no. 2, pp. 185–201, 2001.

[10] J. Huxter, N. Burgess, and J. O'Keefe, "Independent rate and temporal coding in hippocampal pyramidal cells," *Nature*, vol. 425, no. 6960, pp. 828–832, 2003.

[11] J. E. Savage, *Models of Computation: Exploring the Power of Computing*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[12] T. Natschläger and W. Maass, "Spiking neurons and the induction of finite state machines," *Theoretical Computer Science*, vol. 287, no. 1, pp. 251–265, 2002.

[13] P. Tiňo and A. Mills, *Learning Beyond Finite Memory in Recurrent Networks of Spiking Neurons*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 666–675.

[14] U. Rutishauser and R. J. Douglas, "State-dependent computation using coupled recurrent networks," *Neural Computation*, vol. 21, no. 2, pp. 478–509, Feb. 2009.

[15] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[16] V. Steuber and D. J. Willshaw, "Adaptive leaky integrator models of cerebellar Purkinje cells can learn the clustering of temporal patterns," *Neurocomputing*, vol. 26, pp. 271–276, 1999.

[17] V. Steuber and E. De Schutter, "Rank order decoding of temporal parallel fibre input patterns in a complex Purkinje cell model," *Neurocomputing*, vol. 44, pp. 183–188, 2002.

[18] V. Steuber and D. Willshaw, "A biophysical model of synaptic delay learning and temporal pattern recognition in a cerebellar Purkinje cell," *Journal of Computational Neuroscience*, vol. 17, no. 2, pp. 149–164, 2004.

[19] V. Steuber, D. Willshaw, and A. Van Ooyen, "Generation of time delays: Simplified models of intracellular signalling in cerebellar Purkinje cells," *Network: Computation in Neural Systems*, vol. 17, no. 2, pp. 173–191, 2006.

[20] R. Maex and V. Steuber, "The first second: Models of short-term memory traces in the brain," *Neural Networks*, vol. 22, no. 8, pp. 1105–1112, 2009.

[21] B. Wróbel and M. Joachimczak, "Using the Genetic Regulatory evolving Artificial Networks (GReaNs) platform for signal processing, animat control, and artificial multicellular development," in *Growing Adaptive Machines*. Springer, 2014, pp. 187–200.

[22] B. Wróbel, A. Abdelmotaleb, and M. Joachimczak, "Evolving networks processing signals with a mixed paradigm, inspired by gene regulatory networks and spiking neurons," in *Bio-Inspired Models of Network, Information, and Computing Systems: 7th International ICST Conference, LNICST*, vol. 134. Springer, 2014, p. 135.

[23] R. Naud, N. Marcille, C. Clopath, and W. Gerstner, "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological Cybernetics*, vol. 99, no. 4-5, p. 335, 2008.

[24] M. Joachimczak and B. Wróbel, "Evolution of robustness to damage in artificial 3-dimensional development," *Biosystems*, vol. 109, no. 3, pp. 498–505, 2012.

## B.2 M. Yaqoob and B. Wróbel (2018) Robust Very Small Spiking Neural Networks Evolved with Noise to Recognize Temporal Patterns, Artificial Life, ALIFE 2018, Tokyo, Japan, July 23-27, 2018.

# Robust Very Small Spiking Neural Networks Evolved with Noise to Recognize Temporal Patterns

Muhammad Yaqoob[1], Borys Wróbel[1,2]

[1]Evolving Systems Laboratory, Adam Mickiewicz University in Poznan, Poland
[2]Systems Modeling Group, IOPAN, Sopot, Poland
yaqoob@evosys.org
wrobel@evosys.org

## Abstract

To understand how biological and bio-inspired complex computational networks can function in the presence of noise and damage, we have evolved very small spiking neural networks in the presence of noise on the membrane potential. The networks were built with adaptive exponential integrate and fire neurons. The simple but not trivial task we evolved the networks for consisted of recognizing a short temporal pattern in the activity of the network inputs. This task can be described in abstract terms as finding a specific subsequence of symbols (”ABC”) in a continuous sequence of symbols (”..ABCCCAAABCAC..”). We show that networks with three interneurons and one output neuron can solve this task in the presence of biologically plausible levels of noise. We describe how such a network works by mapping its activity onto the state of a finite state transducer—an abstract model of computation on continuous time series. We demonstrate that the networks evolved with noise are much more robust than networks evolved without noise to the modification of neuronal parameters and variation of the properties of the input. We also show that the networks evolved with noise are denser and have stronger connections than the networks evolved without noise. Finally, we demonstrate the emergence of memory in the evolved networks—sustained spiking of some neurons maintained thanks to the presence of self-excitatory loops.

## Introduction

Natural complex systems, including networks of biological neurons, maintain their functionality in the presence of noise and damage. Noise in natural neural networks originates from many sources, including thermal variations and small number of cellular components (for example, ion channels). These components, moreover, undergo constant turnover, and so do parts of the cell (such as dendrites), and the cells themselves. Including noise in models of biological systems helps in our understanding how reliable computation can be performed in the presence of noise, and in building more reliable artificial systems (Florian, 2003).

In biological neuronal networks and in artificial spiking neural networks, the timing of discrete events (spikes) represents the information received from the senses. Processing this information requires recognizing temporal patterns in neuronal activity by other neurons (Bialek et al., 1989; Gerstner et al., 1996; Laurent, 1996; Rieke, 1999; Decharms and Zador, 2000; Ahissar and Arieli, 2001; Huxter et al., 2003). Recognition of temporal patterns requires delays or maintaining the state of the network (Steuber and Willshaw, 1999; Steuber and De Schutter, 2002; Steuber and Willshaw, 2004; Steuber et al., 2006; Maex and Steuber, 2009). Intuitively, the necessity for precise synaptic delays seems a more fragile solution.

In this paper, we evolve very small spiking neural networks for simple pattern recognition in the presence of noise. We hope that analyzing the diverse solutions obtained using artificial evolution will allow us to identify the way robust pattern recognition can be accomplished. To represent the computation performed by the evolved spiking neural networks, we will use a formal computational model of the finite state transducer—a deterministic finite state automaton that receives a continuous sequence of symbols and produces a continuous output (Sipser, 1996). Our focus here is not the induction of a specific finite state automaton (as in Natschläger and Maass, 2002; Tiňo and Mills, 2005; Rutishauser and Douglas, 2009; where large recurrent multilayer spiking neural networks were used). Rather, we will use the formalism to illustrate how the evolved networks work. Finally, we will compare the functioning and structure of networks evolved in the presence of noise to networks evolved without it (which were the subject of our previous work, Yaqoob and Wróbel, 2017). Our preliminary analysis shows that even a relatively low level of noise during evolution results in much more robust networks, and that the networks evolved with noise are denser than the networks evolved without it.

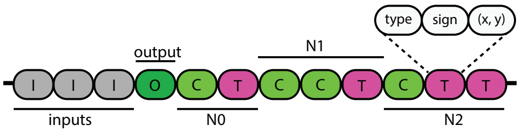


Figure 1: Encoding of a spiking neural network in a linear genome. See text for details.

## The Model

We use the model of evolution of spiking neural networks for temporal pattern recognition introduced previously in the artificial life software platform GReaNs (Wróbel et al., 2012; Wróbel, 2016; Yaqoob and Wróbel, 2017). The networks in this model are encoded in linear genomes built from genetic elements. Each element has a type ($input$, $output$, $cis$ and $trans$; the biological inspiration for such a representation and these terms has been discussed previously; Fig. 1), sign and two coordinates. The elements encode the nodes in the network: three input nodes (encoded by an $input$ element each), up to three interneurons (each encoded by a series of $cis$ elements followed by a series of $trans$ elements), and one output neuron (encoded by a single $output$ element). To determine the connectivity, every pair of $input$-$cis$, $trans$-$cis$ and $trans$-$output$ elements is considered. If the coordinates of two elements in a pair are such that the Euclidean distance is below a threshold (equal to 5), the presence of such a pair contributes to the weight of the connection between two nodes; the contribution is positive if the signs of the two elements coincide and negative otherwise (the contribution is $s_i s_j \frac{2(5-d_{i,j})}{10d_{i,j}+1}$, where $s_i, s_j$ are signs and $d_{i,j}$ is distance; the threshold prevents full connectivity). If the overall sum of such contributions for two neurons is positive, the link (synapse) is excitatory, otherwise it is inhibitory.

In our previous work (Yaqoob and Wróbel, 2017) we have shown that two interneurons are sufficient for the simple pattern recognition task considered here, with no noise. We have subsequently found out that artificial evolution could not find any efficient solution when noise was present and the network size was limited to two interneurons, but could do so with three interneurons. In the evolutionary runs described here we have thus limited the number of interneurons in the decoded network to up to three, by ignoring the rest of the genetic elements. This restriction was imposed both to limit the search space and to ease the analysis of the networks. Similarly, superfluous input or output elements, introduced for example by unequal crossing over (see below) were ignored.

As in our previous work (Yaqoob and Wróbel, 2017), in this paper we used the adaptive exponential integrate and fire neuronal model for each interneuron and output neuron. Each adaptive exponential neuron has four state variables (membrane potential $V$, adaptation $w$, excitatory conductance $g_E$, and inhibitory conductance, $g_I$):

$$\begin{aligned}\frac{dV}{dt} &= \frac{1}{C}(g_E(E_E - V) + g_I(E_I - V) - w) \\ &\quad + \frac{1}{\tau_m}(E_L - V + \Delta_T e^{(\frac{V-V_T}{\Delta_T})}) \qquad (1)\\ \tau_w \frac{dw}{dt} &= a(V - E_L) - w \qquad (2)\\ \frac{dg_E}{dt} &= \frac{-g_E}{\tau_E} \qquad (3)\\ \frac{dg_I}{dt} &= \frac{-g_I}{\tau_I} \qquad (4)\end{aligned}$$

The default values of the parameters we used here (Table 1) were the same as in our previous work (Yaqoob and Wróbel, 2017). They result in tonic spiking in response to constant input current (Naud et al., 2008).

When $V$ in the adaptive exponential neuron is high enough, $V$ quickly diverges to infinity because of the exponential term; this models a spike. For simulation purposes, the spike is cut at a finite value (here, 0 mV). After a spike occurs in a neuron, this neuron's $V$ is reset to $V_r$, and adaptation $w$ is incremented by $b$. In any neuron to which the neuron that spiked connects (in any postsynaptic neuron) with positive (negative) weight, the excitatory (inhibitory) conductance $g_E$ ($g_I$) is increased by synaptic gain ($gain_E$ or $gain_I$, respectively; here, 9 nS) multiplied by the absolute weight.

Modifying four parameters in the adaptive exponential neuron can bring qualitative change in neuronal behavior (qualitatively different responses to constant input current). These four bifurcation parameters (which are directly proportional to the four free parameters; Touboul and Brette, 2008; Naud et al., 2008) are: adaptation time constant $\tau_w$, adaptation conductance $a$, reset voltage $V_r$, and spike-triggered adaptation $b$. The remaining ones are scaling parameters: membrane capacitance $C$, threshold slope factor $\Delta_T$; three time constants, membrane ($\tau_m$), and excitatory/inhibitory ($\tau_E$/$\tau_I$); four potentials, effective rest ($E_L$), inhibitory ($E_I$) and excitatory ($E_E$) reverse, and effective threshold ($V_T$).

We used Euler integration with 1 ms step. When evolving with noise, a random value taken from a Normal distribution (mean 0, standard deviation 2 mV) was added to $V$ at every step. This level of noise is similar in magnitude to the level observed in biological neurons resulting from spontaneous or background synaptic activity (Paré et al., 1998; Destexhe and Paré, 1999; Anderson et al., 2000; Finn et al., 2007).

The task for the network was for the output neuron to spike at least once after the network received the activation from the input nodes in a specific order: first from input A, second from input B, third from input C. During both evolution and testing, input nodes were activated in a random order. Each such activation lasted for 6 ms. Only one input node could be active at a time. Each activation of an input node was followed by an interval of 16 ms during which no input node was active.

In abstract terms, this task corresponds to recognizing a pattern of symbols (”ABC”) in a continuous stream of symbols {A, B, C} received in a random order. In terms of modeling, when a network receives a symbol, it means that six spikes, each one 1 ms apart, are received by all the interneurons to which the input node corresponding to the symbol connects.

Table 1: Robustness to change of neuronal parameters and properties of the input sequences for 10 best champions evolved with noise and the most robust champion evolved without noise. The values in square brackets show the range of robustness, the values above them show relative robustness (see text for details).

| | | Top 10 individuals evolved with noise | | | | | | | | | | most robust |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parameter | default value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | without noise |
| $E_L$ | -70 mV | 0.77 | 1.00 | 0.48 | 0.61 | 0.68 | 0.77 | 0.58 | 0.29 | 0.48 | 0.32 | 0.04 |
| | | [-81, -57] | [-91, -60] | [-76, -61] | [-78, -59] | [-86, -65] | [-88, -64] | [-78, -60] | [-75, -66] | [-77, -62] | [-73, -63] | [-71, -70] |
| $V_r$ | -58 mV | 0.42 | 0.79 | 0.46 | 0.71 | 0.50 | 0.46 | 0.21 | 0.79 | 0.58 | 1.00 | 0.13 |
| | | [-63, -53] | [-63, -44] | [-63, -52] | [-62, -45] | [-60, -48] | [-60, -49] | [-60, -55] | [-64, -45] | [-61, -47] | [-65, -41] | [-60, -57] |
| $V_T$ | -50 mV | 0.40 | 1.00 | 0.40 | 0.80 | 0.80 | 0.80 | 0.80 | 0.40 | 0.60 | 0.20 | 0.00 |
| | | [-51, -49] | [-53, -48] | [-51, -49] | [-52, -48] | [-51, -47] | [-51, -47] | [-52, -48] | [-52, -50] | [-51, -48] | [-51, -50] | [-50, -50] |
| $\tau_m$ | 20 ms | 0.39 | 1.00 | 0.95 | 0.55 | 0.35 | 0.42 | 0.53 | 0.17 | 0.96 | 0.22 | 0.06 |
| | | [8, 44] | [6, 99] | [12, 100] | [8, 59] | [6, 39] | [7, 46] | [8, 57] | [10, 26] | [11, 100] | [9, 29] | [16, 21] |
| $\Delta_T$ | 2 mV | 0.28 | 0.89 | 1.00 | 0.83 | 0.83 | 0.83 | 0.83 | 0.44 | 0.89 | 0.28 | 0.13 |
| | | [1.7, 2.2] | [1.3, 2.9] | [1.5, 3.3] | [1.3, 2.8] | [1.6, 3.1] | [1.7, 3.2] | [1.5, 3.0] | [1.4, 2.2] | [1.6, 3.2] | [1.6, 2.1] | [2.0, 2.2] |
| $C$ | 0.2 nF | 0.40 | 1.00 | 0.50 | 0.50 | 0.50 | 0.60 | 0.30 | 0.40 | 0.60 | 0.20 | 0.00 |
| | | [0.17, 0.21] | [0.15, 0.25] | [0.17, 0.22] | [0.18, 0.23] | [0.17, 0.22] | [0.17, 0.23] | [0.19, 0.22] | [0.16, 0.20] | [0.17, 0.23] | [0.19, 0.21] | [0.2, 0.2] |
| $a$ | 2 nS | 0.60 | 0.87 | 0.37 | 0.87 | 1.00 | 0.87 | 0.90 | 0.27 | 0.40 | 0.73 | 0.03 |
| | | [-4, 14] | [-7, 19] | [-4, 7] | [-5, 21] | [-2, 28] | [-5, 21] | [-8, 19] | [-1, 7] | [-3, 9] | [-9, 13] | [2, 3] |
| $b$ | 0 pA | 0.60 | 0.74 | 0.28 | 0.79 | 1.00 | 0.68 | 0.82 | 0.28 | 0.47 | 0.39 | 0.013 |
| | | [0, 34] | [0, 42] | [0, 16] | [0, 45] | [0, 57] | [0, 39] | [0, 47] | [0, 16] | [0, 27] | [0, 22] | [0, 7] |
| $\tau_E$ | 5 ms | 0.73 | 0.91 | 0.64 | 1.00 | 0.64 | 0.73 | 0.91 | 0.45 | 0.55 | 0.64 | 0.18 |
| | | [4.7, 5.5] | [4.7, 5.7] | [4.6, 5.3] | [4.4, 5.5] | [4.5, 5.2] | [4.6, 5.4] | [4.4, 5.4] | [4.9, 5.4] | [4.7, 5.3] | [4.9, 5.6] | [4.9, 5.1] |
| $\tau_I$ | 5 ms | 0.43 | 0.57 | 1.00 | 0.95 | 1.00 | 0.76 | 0.43 | 0.29 | 0.71 | 0.43 | 1.00 |
| | | [4.5, 5.4] | [4.6, 5.8] | [4.5, 6.6] | [4.5, 6.5] | [4.8, 6.9] | [4.8, 6.4] | [4.7, 5.6] | [4.8, 5.4] | [4.6, 6.1] | [4.5, 5.4] | [4.1, 6.1] |
| $E_E$ | 0 mV | 0.50 | 0.83 | 0.75 | 0.92 | 1.00 | 0.83 | 0.75 | 0.58 | 0.67 | 0.50 | 0.08 |
| | | [-2, 4] | [-4, 6] | [-4, 5] | [-4, 7] | [-8, 4] | [-6, 4] | [-5, 4] | [-1, 6] | [-4, 4] | [0, 6] | [-1, 0] |
| $E_I$ | -70 mV | 0.33 | 0.56 | 0.89 | 0.78 | 0.67 | 0.56 | 0.33 | 0.33 | 0.44 | 0.22 | 1.00 |
| | | [-71, -68] | [-72, -67] | [-76, -68] | [-74, -67] | [-75, -69] | [-74, -69] | [-72, -69] | [-71, -68] | [-73, -69] | [-70, -68] | [-74, -65] |
| $gain_E$ | 9 nS | 0.50 | 1.00 | 0.75 | 0.95 | 0.95 | 0.85 | 0.75 | 0.65 | 0.80 | 0.50 | 0.05 |
| | | [8.6, 9.6] | [8.2, 10.2] | [8.2, 9.7] | [8.3, 10.2] | [7.7, 9.6] | [7.9, 9.6] | [8.1, 9.6] | [8.8, 10.1] | [8.2, 9.8] | [8.9, 9.9] | [8.9, 9.0] |
| $gain_I$ | 9 nS | 0.21 | 0.40 | 0.92 | 0.46 | 0.46 | 0.35 | 0.28 | 0.26 | 0.40 | 0.15 | 1.00 |
| | | [8.0, 9.8] | [7.2, 10.6] | [7.9, 15.7] | [7.1, 11] | [8.2, 12.1] | [8.2, 11.2] | [7.7, 10.1] | [7.7, 9.9] | [7.7, 11.1] | [8.0, 9.3] | [7.1, 15.6] |
| $noise$ | 2 mV | 0.85 | 1.00 | 0.93 | 0.85 | 0.78 | 0.78 | 0.89 | 0.74 | 0.93 | 0.74 | 0.04 |
| | | [0, 2.5] | [0, 2.9] | [0, 2.7] | [0, 2.5] | [0, 2.3] | [0, 2.3] | [0, 2.6] | [0, 2.2] | [0, 2.7] | [0, 2.2] | [0, 0.1] |
| $silence$ | 16 ms | 0.95 | 0.25 | 0.25 | 1.00 | 0.20 | 0.20 | 0.70 | 0.55 | 0.20 | 0.60 | 0.16 |
| | | [13, 32] | [14, 19] | [13, 18] | [10, 30] | [12, 16] | [13, 17] | [11, 25] | [12, 23] | [14, 18] | [10, 22] | [15, 18] |
| $signal$ | 6 ms | 0.50 | 0.50 | 0.50 | 1.00 | 1.00 | 0.50 | 0.50 | 0.00 | 1.00 | 0.00 | 1.00 |
| | | [6, 7] | [5, 6] | [6, 7] | [5, 7] | [4, 6] | [5, 6] | [5, 6] | [6, 6] | [6, 8] | [6, 6] | [6, 8] |
| Average relative robustness | | 0.52 | 0.78 | 0.65 | 0.80 | 0.73 | 0.65 | 0.62 | 0.41 | 0.63 | 0.42 | 0.30 |

The genomes in the initial population were created randomly as described previously (Yaqoob and Wróbel, 2017). Each evolutionary run had a constant population size (300), with size two tournaments, elitism (10 individuals) and crossover (30 individuals in each generation; there was no crossover in Yaqoob and Wróbel, 2017).

Multi-point crossover was implemented in the following manner. First, two parents (A and B) are selected from population as winners of two independent size two tournaments. A cursor pointing to the genetic elements to be copied is initiated at the first element for both genomes. Then, one of the four schemes is chosen: copy an element from parent (i) A or (ii) B to offspring, advance cursor on both genomes, or copy from (iii) A or (iv) B, advance cursor only on the template copied from (each with probability 0.03). The probabilities of choosing actions (i) and (ii) were equal and 4 times larger than the (again, equal) probabilities of choosing (iii) or (iv). After an element is copied, the scheme stays the same as previously with probability 0.7, and otherwise a scheme is re-chosen (the same one can be chosen again), maintaining the ratio between the probabilities as above.

If an element was chosen for a point mutation (per element probability of 0.1), the coordinates were changed so that the associated point was moved in a random direction (drawn from a uniform distribution) by a distance drawn from a normal distribution (mean 0, standard deviation 1). Duplications (probability of 0.001 per genome) occurred twice as often as deletions (probability 0.0005). The starting element and the insertion site were chosen randomly (each element had the same chance of being chosen). The length of duplication/deletion was drawn from a geometric distribution with mean 11.

Each individual in the population was evaluated on six random sequences (different from each other, for each indidvidual, and in each generation), each with 500 symbols. Four sequences were generated with equiprobable occurrence of A, B, and C (and thus contained about 16 ABC subsequences each), and two other continuous sequences were constructed by concatenating, in random order, ABC with ABB, and ABA (two subsequences that are the most problematic to discriminate from ABC; Yaqoob and Wróbel, 2017).

The fitness function rewarded for spike(s) after the target subsequence and penalized for spikes elsewhere:

$$f_{fitness} = 1 - R + 4P \tag{5}$$

$R$ (for reward) is the fraction of time intervals when the input nodes are silent (each such silence is 16 ms long) after the last C in "ABC" and the output spikes at least once. In other words, it is the fraction of instances in which the output spikes correctly. $P$ (for penalty) is the fraction of instances in which the output spikes incorrectly. These instances can happen either in (i) 16 ms silence intervals not after ABCs, or in (ii) 6 ms time intervals in which one of the input nodes is active. Although $P$ in principle could reach 1, in practice it was always quite small, and the $f_{fitness}$ was below 1. We call an individual a perfect recognizer if $f_{fitness} = 0$. The number by which $P$ is multiplied, 4, was chosen after preliminary exploration to find a value that resulted in the highest evolvability (number of evolutionary runs that ended with perfect recognizers).

## Results and Discussion

Among 100 independent evolutionary runs without noise, 33 ended with perfect recognizers—champions with $f_{fitness} = 0$ when re-evaluated (tested) on 500 random input sequences (thus different than the sequences experienced during evolution) with equal probability of each symbol (in Yaqoob and Wróbel, 2017 we used different settings for artificial evolution—in particular, no cross over, different probabilities of duplications and deletions—and the yield was much lower; other difference were: time during which outputs were active were 4 ms, with 8 silences; the fitness function was more complex; excitatory/inhibitory gain was 5 nS; the output neuron had an offset current).

In the presence of noise, 1000 generations were needed to obtain 10 champions in 100 independent runs that were perfect recognizers when re-evaluated as above without noise—with noise they failed to produce a spike on output at most after 1 in 100 ABCs.

All perfect recognizers evolved without noise always produced only 1 spike in output after an ABC in the input sequence. The perfect recognizers evolved with noise belonged to two classes: when evaluated without noise, either (i) the output always spiked once, or (ii) always twice after each ABC. When evaluated with noise, the output neurons in both classes spiked once after some ABCs, and twice after the other ABCs in the same input sequence, but never more times.

To measure the robustness, we first analyzed what was the range of robustness for each parameter. In this preliminary analysis, only one parameter was changed at a time. The range of robustness was defined as the largest continuous set of parameter values around the default value for which a given network had the true positive rate of at least 99% and the false discovery rate of at most 5%. We define here the true positive rate and the false discovery rate as follows. The true positive rate is the average number of recognized ABCs (the number of 16 ms intervals after ABCs in which the output neuron spiked, correctly) divided by the actual number of ABCs in the input sequence. The false discovery rate is the average number of intervals (6 ms or 16 ms) in which the output spiked incorrectly (not in the interval of silence after an ABC), divided by the total number of intervals in which the output spiked. Since we are interested in temporal pattern recognition in a continuous input sequence, we actually evaluated the champions on 600-symbol input

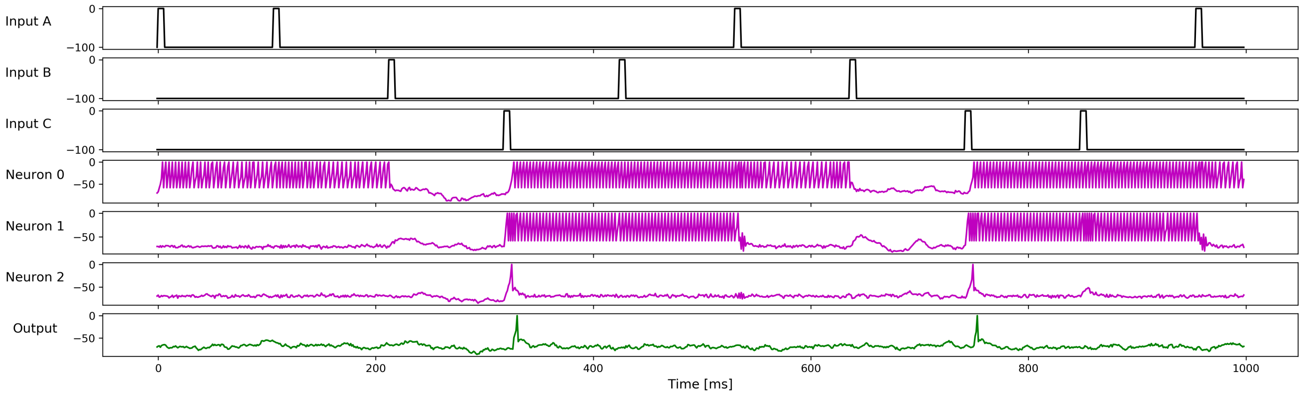


Figure 2: Network activity of the champion 3 evolved with noise on membrane potential. The individual is tested for signal length 6 ms, silence interval 100 ms (it evolved for 16 ms). The output neuron spikes after ABC around 330 ms and 700 ms

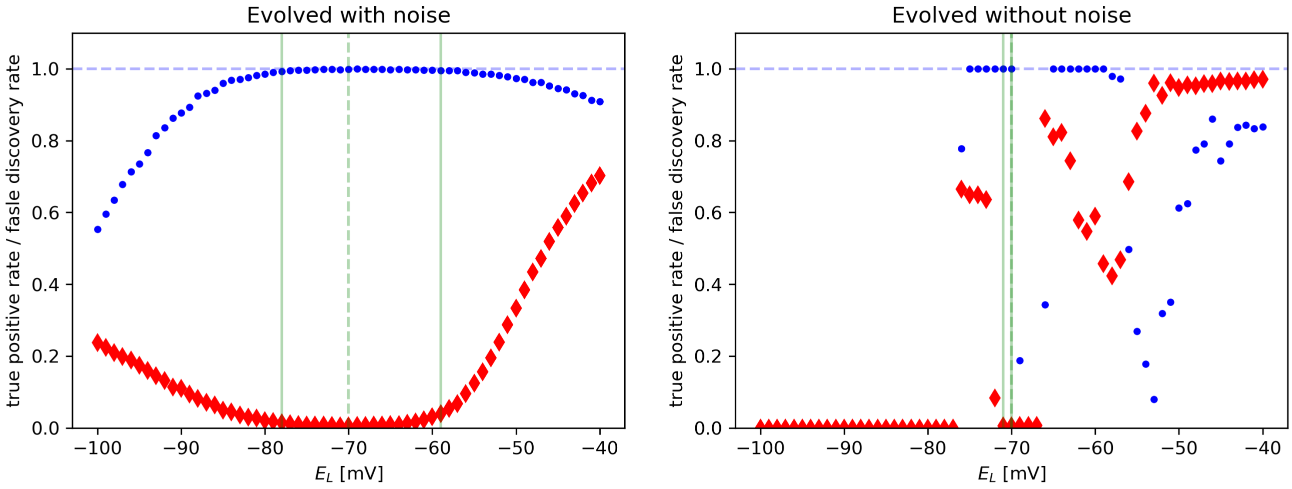


Figure 3: Example of robustness of network performance to change of a parameter (here, $E_L$). Blue circles show the true positive rate, red triangles show the false discovery rate (see text for more details). The range of robustness is showed by two continuous vertical lines, the dashed vertical line shows the default value, -70 mV. The network evolved (and tested) with noise (left; the network of champion 3) shows graceful degradation of network performance and a larger range of robustness than the network evolved (and tested) without noise (right; the network of the most robust champion evolved without noise).

sequences and considered the response to the last 500 symbols, over 500 such sequences. This is a very conservative approach as in practice discarding the response to the first symbol or a few at most would work equally well.

Second, we compared the ranges (differences between maximum and minimum value) for a given parameter across champions, defining relative robustness as the fraction of the maximum range for a given parameter among the champions (thus the champion with the largest range has the relative robustness 1.00, and the one with, say, half that range, has 0.50).

Third, we calculated the average relative robustness for each champion (Table 1; only the most robust champion evolved without noise is shown for simplicity). All the champions evolved with noise were robust to setting $\tau_w$ in the range from 1 to more than 1000 ms (the default value was 30 ms). Only the best champion evolved without noise was equally robust to changes of $\tau_w$, other champions evolved without noise had smaller ranges. When the integration step was changed from 1 ms to 0.5 ms, the champions evolved with noise displayed a drop in the true positive rate (from 0.99 to 0.98), while the false discovery rate remained unaffected. The networks evolved without noise were not robust to such a change.

The comparison between champions evolved and re-evaluated with noise to the champions evolved and re-evaluated without noise is conservative, as champions evolved with noise have much larger ranges of robustness when re-evaluated in the absence of noise. Moreover, the average relative robustness is a crude measure, as robustness to change of a particular parameter may be highly correlated to the robustness to change of other one(s); a more refined measure would give lower weights to relative robustness for parameters that belong to such a group.

We expected that noise will promote robustness to damage (here, change of neuronal parameters), as has been observed before in GReaNs for evolving gene regulatory networks, were damage affected an artificial developmental process (Joachimczak and Wróbel, 2012), and spiking neural networks (Wróbel, 2016).

In accordance with our expectations, the networks evolved with noise were much more robust to change in neuronal parameters than networks evolved without noise (Table 1; the second best, in terms of robustness, champion evolved without noise had average robustness 0.13, the rest had average robustness below 0.1).

Moreover, the networks evolved with noise showed graceful degradation beyond the range of robustness (for most of the parameters; $E_L$ is shown as an example in Fig. 3, taking champion 3 as the one evolved with noise and the most robust champion evolved without noise for comparison).

In contrast to gene regulatory networks regulating artificial development, networks evolved with noise functioned very well (in fact, better) when tested without noise. The most robust champion evolved with noise (champion 3) was, in particular, the most robust to lengthening the silences between the activity of the input nodes, and when this champion was tested without noise, these silences could be extended with no discernible limit, indicating that this network is able to maintain its states forever (Fig. 2).

Table 2: States of the finite state transducer corresponding to states of the network of the champion 3 evolved with noise

| | S | hA | hAB | hABC |
|---|---|---|---|---|
| Neuron 0 | 330 Hz | 333 Hz | 0 | 331 Hz |
| Neuron 1 | 333 Hz | 0 | 0 | 333 Hz |
| Neuron 2 | 0 | 0 | 0 | 1 spike |
| Output | 0 | 0 | 0 | 1 spike |

We can describe how this network functions by mapping the network activity to the states in a finite state transducer (Table 2, Fig. 4). Let us first assume that the network has already received some symbols (some input subsequence). If this subsequence ends with a C that did not follow AB or with a B that did not follow an A, the network is in the state S (starting state), in which (inter)neuron 0 and neuron 1 spike continuously, at high frequency. If an A is received, the network goes to a state hA ("had A"), in which only neuron 0 spikes in this fashion. If this A is followed by a B, all interneurons do not spike (state hAB, "had AB"). If this AB is followed by a C, neurons 0 and 1 again start spiking continuously, while neuron 2 and then the output neuron produce one spike each, immediately after the C (the state hABC, "had ABC").

Although this is not actually relevant for computing the output in response to a continuous input sequence, if the network receives a B with no previous history, it goes, essentially, to the state S. However, no previous history (and no activity in the network) is indistinguishable in this network from the state hAB. This is why the output neuron of this network will incorrectly spike after receiving a C with no previous history—in other words, if an input sequence starts with a C.

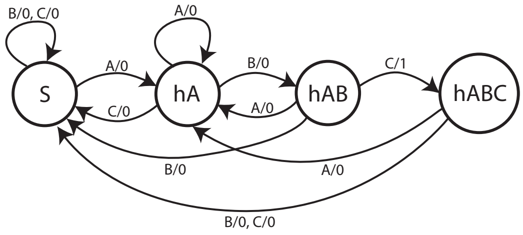


Figure 4: Minimal finite state transducer for recognizing ABC; the nodes represent states and edges represents transition from one state to another state on receiving an input symbol {A,B,C} and with producing an output {0: no spike(s), 1: spike(s) of the output neuron}.

The analysis of the network of the champion 3 evolved with noise (Fig. 5) shows that all interneurons connect to one another and each connects to itself (each has a self-loop). Two interneurons, neuron 0 and 1, have excitatory self-loops, which seem to be responsible for maintaining the continuous high-frequency spiking of these two neurons in states S, hA, hAB, and hABC. On the other hand, the recurrent inhibitory connections between interneurons (neuron 1 excites neuron 0, while 0 inhibits 1; neuron 2 excites 0, while 0 inhibits 2; neurons 1 and 2 inhibit one another) seem to, together with inhibitory connections from the input nodes to interneurons, bring the end of continuous spiking of neuron 0, or both neuron 0 and neuron 1, that corresponds to state transitions from S to hA, from hA to hAB, and from hABC to S. Finally, inhibitory connections from neurons 0 and 1 to the output neuron, together with an excitatory connection from neuron 2 to the output neuron, ensure that a spike in the output neuron is possible only after neuron 2 spikes (state hABC).

If we count the number of connections in the networks disregarding the associated weights (in other words, giving each one the same weight, 1), the 10 champion networks evolved with noise show significantly higher density (average 18.5, standard deviation 1.6) than the 10 champion

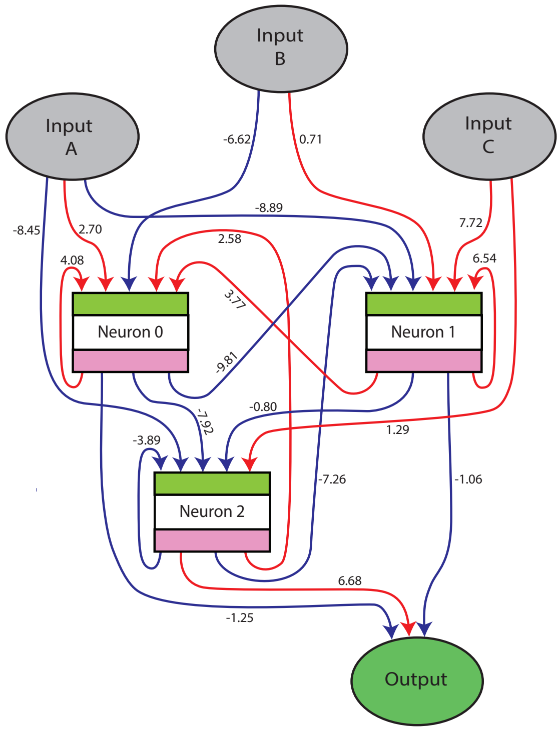


Figure 5: The topology of the network of the champion 3 evolved with noise. Blue lines are inhibitory connections, red lines are excitatory, numbers next to the lines show the weights.

networks evolved without noise (16.2, 2.1, respectively; $p = 0.01$, Mann-Whitney U test). Summing the absolute weight for all the edges gives an even more striking, more than 2-fold, difference (average for champions evolved with noise: 82.1, standard deviation: 13.8; without noise: 34.2, 11.8, respectively; $p = 0.0012$).

## Conclusions and future work

Our results show that evolving networks with noise leads to high robustness to modifying the neuronal parameters and variations of the input. In addition, the results show that networks evolved with noise are capable of maintaining their internal state infinitely. This memory seems to be kept thanks to excitatory self-loops, while switching state is possible thanks to inhibitory recurrent connections.

In future work, we plan to further investigate the robustness of the spiking neural networks evolved with noise to modifying parameters. The adaptive exponential model itself has in fact only four free parameters (Touboul and Brette, 2008), which are directly proportional to four bifurcation parameters in the model with the parametrization used here ($\tau_w$, $a$, $V_r$, and $b$; Naud et al., 2008). Similarly, we may expect that robustness to changes in synaptic gains ($gain_E$ or $gain_I$), $C$, $E_E$, and $E_I$ will be related, and networks robust to changing them will be expected to be robust to variation of synaptic weights. Therefore, it should be possible to change more than one parameter at a time, perhaps provided that certain relationships between them is maintained (for example, their quotient would have to stay within a certain range).

It will be also interesting if the networks are robust to changes of the parameters which actually result in the changed behavior of the neuron in terms of the response to constant input current. For example, we could evolve the networks with neurons showing tonic spiking, and then change parameters so that the behavior is tonic bursting or delayed spiking (Naud et al., 2008). Our preliminary results, incidentally, show that it is possible to evolve networks with neurons displaying all types of responses to the constant input current demonstrated for this model (Naud et al., 2008). It will be interesting to see if the networks evolved with noise in neurons showing other behavior than tonic spiking are equally robust to parameter changes.

We also plan to analyze robustness to introducing synaptic delays and absolute refractory periods. Such changes, similarly to changes in some neuronal parameters, could lead to lower firing rates in neurons whose continued spiking provides memory.

We would also like to know if the networks will be robust to changes in parameters affecting each neuron in the network independently (in this paper a particular change affected all neurons in the network in the same fashion). This last type of robustness is particularly interesting for transferring an evolved network able to perform a particular computation to analog neuromorphic hardware, in which setting particular (and the same) parameters for all the neurons in the network may be problematic.

Although the level of noise we have used here is biologically plausible, it is on a lower end of the spectrum observed for biological neurons (Paré et al., 1998; Destexhe and Paré, 1999; Anderson et al., 2000; Finn et al., 2007). It will be interesting to find out if higher, still biologically plausible, levels of noise (for example, noise as in this paper but with standard deviation of 4 mV) allow for evolvability, and if possibly result in more robustness to changes in neuronal parameters—and if lower levels of noise will result in lower robustness. Moreover, other models of noise (for example, an Ornstein-Uhlenbeck process; commonly used to model noise in neuroscience) on neuronal variables could be introduced—also, the presence of noise on input (such as variable signal or silence length). We wonder if different models of noise lead to similar levels of robustness to changes in neuronal parameters.

Another possible direction of future work could be evolution of networks for more complex tasks involving temporal pattern recognition (for example: more symbols, regular expressions, recognition of several patterns by the same network, with several output neurons). Our preliminary results indicate that our model scales up for more complex tasks. This direction of possible future research would allow us to find out what are the relationships between the complexity of the tasks and the size (and complexity) of the minimal-size spiking neural networks necessary to solve them.

## Acknowledgements

This work was supported by the Polish National Science Center (project EvoSN, UMO-2013/08/M/ST6/00922). MY acknowledges the support of the KNOW RNA Research Center in Poznan (No. 01/KNOW2/2014). We are grateful to Volker Steuber and Neil Davey for discussion and suggestions.

## References

Ahissar, E. and Arieli, A. (2001). Figuring space by time. *Neuron*, 32:185–201.

Anderson, J. S., Lampl, I., Gillespie, D. C., and Ferster, D. (2000). The contribution of noise to contrast invariance of orientation tuning in cat visual cortex. *Science*, 290:1968–1972.

Bialek, W., Rieke, F., van Steveninck, R. R. d. R., Warland, D., et al. (1989). Reading a neural code. In *Neural Information Processing Systems*, pages 36–43.

Decharms, R. C. and Zador, A. (2000). Neural representation and the cortical code. *Annual Review of Neuroscience*, 23:613–647.

Destexhe, A. and Paré, D. (1999). Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. *Journal of Neurophysiology*, 81:1531–1547.

Finn, I. M., Priebe, N. J., and Ferster, D. (2007). The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. *Neuron*, 54:137–152.

Florian, R. V. (2003). Biologically inspired neural networks for the control of embodied agents. *Center for Cognitive and Neural Studies (Cluj-Napoca, Romania), Technical Report Coneural-03-03*.

Gerstner, W., Kempter, R., van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76.

Huxter, J., Burgess, N., and O'Keefe, J. (2003). Independent rate and temporal coding in hippocampal pyramidal cells. *Nature*, 425:828–832.

Joachimczak, M. and Wróbel, B. (2012). Evolution of robustness to damage in artificial 3-dimensional development. *Biosystems*, 109:498 – 505.

Laurent, G. (1996). Dynamical representation of odors by oscillating and evolving neural assemblies. *Trends in Neurosciences*, 19:489–496.

Maex, R. and Steuber, V. (2009). The first second: Models of short-term memory traces in the brain. *Neural Networks*, 22:1105–1112.

Natschläger, T. and Maass, W. (2002). Spiking neurons and the induction of finite state machines. *Theoretical Computer Science*, 287:251–265.

Naud, R., Marcille, N., Clopath, C., and Gerstner, W. (2008). Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99.

Paré, D., Shink, E., Gaudreau, H., Destexhe, A., and Lang, E. J. (1998). Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo. *Journal of Neurophysiology*, 79:1450–1460.

Rieke, F. (1999). *Spikes: exploring the neural code*. MIT press.

Rutishauser, U. and Douglas, R. J. (2009). State-dependent computation using coupled recurrent networks. *Neural Computation*, 21:478–509.

Sipser, M. (1996). *Introduction to the Theory of Computation*, page 87. International Thomson Publishing, 1st edition.

Steuber, V. and De Schutter, E. (2002). Rank order decoding of temporal parallel fibre input patterns in a complex Purkinje cell model. *Neurocomputing*, 44:183–188.

Steuber, V. and Willshaw, D. (2004). A biophysical model of synaptic delay learning and temporal pattern recognition in a cerebellar Purkinje cell. *Journal of Computational Neuroscience*, 17:149–164.

Steuber, V., Willshaw, D., and Van Ooyen, A. (2006). Generation of time delays: Simplified models of intracellular signalling in cerebellar Purkinje cells. *Network: Computation in Neural Systems*, 17:173–191.

Steuber, V. and Willshaw, D. J. (1999). Adaptive leaky integrator models of cerebellar Purkinje cells can learn the clustering of temporal patterns. *Neurocomputing*, 26:271–276.

Tiňo, P. and Mills, A. (2005). *Learning Beyond Finite Memory in Recurrent Networks of Spiking Neurons*, pages 666–675. Springer Berlin Heidelberg, Berlin, Heidelberg.

Touboul, J. and Brette, R. (2008). Dynamics and bifurcations of the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99:319.

Wróbel, B. (2016). Evolution of spiking neural networks robust to noise and damage for control of simple animats. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 686–696.

Wróbel, B., Abdelmotaleb, A., and Joachimczak, M. (2012). Evolving networks processing signals with a mixed paradigm, inspired by gene regulatory networks and spiking neurons. In *BIONETICS*, volume 134 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 135–149. Springer.

Yaqoob, M. and Wróbel, B. (2017). Very small spiking neural networks evolved to recognize a pattern in a continuous input stream. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 3496–3503.

## B.3 M. Yaqoob and B. Wróbel (2018) Very Small Spiking Neural Networks Evolved for Temporal Pattern Recognition and Robust to Perturbed Neuronal Parameters, International Conference on Artificial Neural Networks, ICANN 2018, Rhodes, Greece, October 5-7, 2018.

# Very Small Spiking Neural Networks Evolved for Temporal Pattern Recognition and Robust to Perturbed Neuronal Parameters

Muhammad Yaqoob[1] and Borys Wróbel[1,2]( )

[1] Evolving Systems Laboratory, Adam Mickiewicz University in Poznan,
Poznan, Poland
{yaqoob,wrobel}@evosys.org
[2] IOPAN, Sopot, Poland

**Abstract.** We evolve both topology and synaptic weights of recurrent very small spiking neural networks in the presence of noise on the membrane potential. The noise is at a level similar to the level observed in biological neurons. The task of the networks is to recognise three signals in a particular order (a pattern ABC) in a continuous input stream in which each signal occurs with the same probability. The networks consist of adaptive exponential integrate and fire neurons and are limited to either three or four interneurons and one output neuron, with recurrent and self-connections allowed only for interneurons. Our results show that spiking neural networks evolved in the presence of noise are robust to the change of neuronal parameters. We propose a procedure to approximate the range, specific for every neuronal parameter, from which the parameters can be sampled to preserve, at least for some networks, high true positive rate and low false discovery rate. After assigning the state of neurons to states of the network corresponding to states in a finite state transducer, we show that this simple but not trivial computational task of temporal pattern recognition can be accomplished in a variety of ways.

**Keywords:** Temporal pattern recognition · Spiking neural networks
Artificial evolution · Minimal cognition · Complex networks
Genetic algorithm · Finite state automaton · Finite state machine

## 1 Introduction

Information in biological neuronal systems is represented temporally by *precise* timing of voltage spikes [1,3,5,6,12,13,15]. Thus noise poses a fundamental problem for informational processing in biological systems [9] (and also artificial systems inspired by them). On the other hand, noise has been postulated to play a computational role [14]. For example, neuronal noise enables the phenomenon of stochastic resonance in neural networks—a process in which a weak signal

V. Kůrková et al. (Eds.): ICANN 2018, LNCS 11139, pp. 322–331, 2018.
https://doi.org/10.1007/978-3-030-01418-6_32

gets amplified to reach a threshold, or a strong signal is prevented from spiking [7,20,21]. Moreover, neural networks formed in the presence of background noisy synaptic activity can be expected to be robust to disturbances [11].

In this work, we analyse very small spiking neural networks (SNNs) evolved to perform a simple temporal pattern recognition task in the presence of noise. We will show that networks evolved with noise maintain functionality even when the parameters of the neuronal model are changed. In contrast to our previous work [23] in which just one neuronal parameter was varied at any given time (while all the other parameters were kept at the default value), here we investigate the robustness against varying *all* the parameters simultaneously. Although the model for evolving the topology and weights in the SNNs we use here does not in principle limit the number of neurons, we limited this number to either three or four interneurons and one output neuron.

It has been observed before that the same computational task can be accomplished by networks with different structures [16,19]. Our long-term goal is to understand how various solutions—obtained by evolving networks numerous times, independently—can accomplish simple, but not trivial computational tasks.

## 2 The Model

The networks in this work consist of adaptive exponential integrate and fire neurons [17] with the default values of the parameters that result in tonic spiking for constant input. The four state variables of each neuron, membrane potential $V$, adaptation $w$, excitatory and inhibitory conductance $g_E$ and $g_I$, are governed by the equations

$$\begin{aligned}\frac{dV}{dt} &= \frac{1}{C}(g_E(E_E - V) + g_I(E_I - V) - w) \\ &\quad + \frac{1}{\tau_m}(E_L - V + \Delta_T e^{(\frac{V - V_T}{\Delta_T})}) \qquad (1)\\ \tau_w \frac{dw}{dt} &= a(V - E_L) - w \qquad (2)\\ \frac{dg_E}{dt} &= \frac{-g_E}{\tau_E} \qquad (3)\\ \frac{dg_I}{dt} &= \frac{-g_I}{\tau_I} \qquad (4)\end{aligned}$$

with 13 parameters in total; the default values of parameters are presented in Table 1 [23,24]. We used Euler integration with 1 ms time step, and added a random value drawn from the normal distribution centered at 0 with standard deviation 2 mV to $V$ at each step; this level of noise is similar to that observed in biological neurons [2,8,10,18].

When $V$ of a neuron is above 0 mV, $V$ is reduced to $V_r$, while $w$ changes to $w + b$, and each neuron to which this neuron connects receives a spike. If

**Table 1.** The ranges of robustness for champions with 3 and 4 interneurons that were most robust (3/3 and 1/4, respectively) overall and for the most robust from the champions maintaining state (8/3 and 7/4).

| Parameter | Default value | 3/3 | 8/3 | 1/4 | 7/4 |
|---|---|---|---|---|---|
| $E_L$ | −70 mV | [−72, −67] | [−72, −66] | [−72, −67] | [−74, −68] |
| $V_r$ | −58 mV | [−59, −55] | [−60, −54] | [−60, −55] | [−59, −55] |
| $V_T$ | −50 mV | [−51, −48] | [−51, −48] | [−52, −49] | [−51, −48] |
| $\Delta_T$ | 2 mV | [1.6, 2.4] | [1.8, 2.3] | [1.8, 2.1] | [1.9, 2.2] |
| $C$ | 0.2 nF | [0.19, 0.22] | [0.17, 0.23] | [0.17, 0.21] | [0.17, 0.22] |
| $a$ | 2 nS | [−2, 4] | [1, 6] | [0, 3] | [1, 4] |
| $b$ | 0 pA | [0, 3] | [0, 4] | [0, 3] | [0, 2] |
| $\tau_m$ | 20 ms | [19, 22] | [18, 23] | [17, 21] | [17, 23] |
| $\tau_w$ | 30 ms | [29, 32] | [29, 33] | [27, 31] | [27, 31] |
| $\tau_E$ | 5 ms | [4.8, 5.2] | [4.9, 5.3] | [4.7, 5.1] | [4.9, 5.3] |
| $\tau_I$ | 5 ms | [4.9, 5.2] | [4.9, 5.3] | [4.6, 5.1] | [4.9, 5.3] |
| $E_E$ | 0 mV | [−2, 2] | [−2, 4] | [−3, 1] | [−1, 2] |
| $E_I$ | −70 mV | [−71, −67] | [−73, −68] | [−72, −67] | [−71, 68] |
| $gain_E$ | 7 nS | [6.9, 7.3] | [6.9, 7.3] | [6.8, 7.3] | [6.7, 7.2] |
| $gain_I$ | 7 nS | [6.8, 7.3] | [6.8, 7.4] | [6.8, 7.3] | [6.7, 7.2] |

the connection is excitatory (inhibitory), $g_E$ ($g_I$) in such a postsynaptic neuron is increased by the weight of the connection multiplied by the synaptic gain. Encoding of SNNs in our model has been described previously [22–24]. In order to recognise a subsequence of three signals in a random input stream, the network has three input nodes (one for each signal), either three or four interneurons, and a single output neuron. Dale rule [4] is not kept—a neuron can be both excitatory and inhibitory at the same time. Furthermore, input nodes cannot connect to the output neuron directly. Only interneurons can have self-loops. The settings for the artificial evolution in this work are as in our previous work [23], with three modifications: (i) the size of duplication of genetic elements was drawn from a geometric distribution with mean 6 (it was 11 previously), (ii) the elements coding for input and output were excluded both from duplications/deletions and crossover (they were allowed to undergo crossover in [23]), (iii) finally and most importantly, we modified slightly the way the fitness function is calculated, resulting in the procedure as follows.

During evolution, each individual was evaluated on six input streams with 500 signals, each signal 6 ms in duration and followed by 16 ms silence (each input stream thus lasted for 11 s). In four input streams, all signals (A, B and C) occurred with equal probability; two input streams were constructed by concatenating four triplets (with equal probability of occurrence): ABC and ABA, ABB, BBC (three triplets that our preliminary work showed the most problematic to distinguish from the pattern to be recognised, ABC). To calculate the fitness function, we calculated $R$ (for reward), the number of 22 ms intervals (signal plus silence) of the last C of each ABC in the input sequence during which the output neuron actually spiked, correctly, at least once, divided by the total number of intervals in the input stream for which it should spike. In other words, $R$ is the true positive rate (TPR) of the network. We also calculated $P$ (for penalty), the number of other 22 ms intervals (signal plus silence) with spikes on output (wrongly), divided by the total number of 22 ms intervals in the input stream in which spikes should not occur. In contrast, false discovery rate (FDR) of the network has the same numerator as $P$, but the denominator is all the 22 ms intervals in which the spikes of the output neuron were observed. The fitness function we used,

$$f_{fitness} = 1 - R + 4P \tag{5}$$

penalises strongly spikes that do not follow the target pattern. The constant 4 in the penalty term was chosen by the preliminary exploration of values with the objective to find a value that gave the highest yield of successful evolutionary runs. We define a successful run as one that ends with a champion that is a perfect recogniser. A perfect recogniser evolved without noise is a network that spikes only after the correct pattern. For networks evolved with noise, we consider an SNN a perfect recogniser if it has TPR $>$ 0.99 and FDR $<$ 0.01).

The slight modifications of the settings of the artificial evolution (from the ones used in [23]) had a quite pronounced effect on the yield of perfect recognisers when no noise was present (for three interneurons, 81% of runs versus 33% for the settings in [23]). However, the effect on the evolvability in the presence of noise was less pronounced.

For each champion, we obtained the ranges of parameters for which it was robust using the following algorithm. We repeatedly extended the ranges of all parameters around their default values, by a small value (specific for each parameter), at first in both directions. We then drew 100 random sets of parameters using such extended ranges, gave the same parameters to all neurons in the network, and checked if at least 90 among these 100 SNNs had TPR $>$ 0.90 and FDR $<$ 0.10 (each network was tested for one random, and thus different, input stream with 50000 signals, with equal probability of occurrence for A, B, and C). If so, the extended ranges were kept. If not, the ranges were shrunk back to the previous sizes and the problematic parameter was identified (by excluding one by one the parameters from extension, in one of the two directions, in the set of parameters for which the ranges can be extended, and checking if this allowed to extend the range keeping TPR $>$ 0.90 and FDF $<$ 0.10). The algorithm stopped when the set of parameters for which the range could be extended became empty.

The size of the ranges (maximum minus the minimum value) were compared for the networks evolved with the limit of three versus four interneurons using the James test implemented in the package Rfast of the R project (https://cran.r-project.org/). Proportions were compared using function prop.test in R.

## 3 Results and Discussion

In 100 independent runs for 3000 generations each, when we allowed for three interneurons, 13 runs ended with perfect recognisers. When we allowed for four interneurons, 19 runs out of 100 resulted in perfect recognisers. Our previous work [23] suggested that at least three interneurons are needed to obtain perfect recognisers in the presence of noise; here also we were unable to evolve with noise when less than three interneurons were allowed, and none of the runs when the limit was set to three resulted in a champion with less. In contrast, two champions out of 19 obtained when the limit was set to four interneurons ended up having three interneurons.

The size of the ranges of robustness for 13 networks evolved with the limit of three versus 17 networks with four interneurons was not significantly different. We then tested how robust were the networks when each neuron in the network was given a different set of parameters drawn from the obtained range (during the range expansion algorithm, all neurons always had the same parameters drawn from the range; in this test, as during expansion, we made 100 evaluations, each on a different random input stream with 50000 signals). None of the networks remained perfect recognisers, but some—noticeably champion 3 evolved with three interneurons (champion 3/3)—were quite robust to such a disruption (Table 2), and so were champions 8/3 and 5/3; and for the networks with 4 interneurons, champions 1/4 and 12/4.

We have previously proposed a way to map the network activity to the states of finite state transducers (FST) [23, 24]. Before we did such a mapping for the networks obtained here, we first analysed which networks could maintain their state for a very long time (in practice, noise may prevent a given network from maintaining the states infinitely). Nine out of 13 networks evolved with three interneurons sustained elongation of intervals between signals from 16 ms to at least 100 ms (Table 2; we assume that if the silence can be extended to 100 ms, the network maintains its state). Only four out of 17 with four interneurons did so (Table 2). Thus the fraction of perfect recognisers maintaining their state is

**Table 2.** Robustness of 13 networks evolved limiting the number of interneurons to three (top) and 19 networks evolved limiting the number of interneurons to four (bottom; champions with labels in bold evolved to have 3 interneurons), when sampling the neuronal parameters from the ranges of robustness specific for each champion, and their robustness to increased interval of silence between signals.

| | 0/3 | 1/3 | 2/3 | 3/3 | 4/3 | 5/3 | 6/3 | 7/3 | 8/3 | 9/3 | 10/3 | 11/3 | 12/3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TPR>0.99 & FDR<0.01 | 37 | 37 | 27 | 80 | 19 | 53 | 47 | 52 | 67 | 14 | 24 | 16 | 8 |
| TPR>0.95 & FDR<0.05 | 71 | 79 | 75 | 97 | 68 | 99 | 93 | 98 | 93 | 80 | 50 | 51 | 79 |
| TPR>0.90 & FDR<0.10 | 84 | 86 | 86 | 100 | 85 | 99 | 97 | 99 | 99 | 91 | 71 | 65 | 93 |
| Maximum interval of silence | ⩾100 | 35 | ⩾100 | 28 | ⩾100 | ⩾100 | 48 | ⩾100 | ⩾100 | ⩾100 | ⩾100 | ⩾100 | 19 |

| | 0/4 | 1/4 | 2/4 | 3/4 | 4/4 | 5/4 | **6/4** | 7/4 | **8/4** | 9/4 | 10/4 | 11/4 | 12/4 | 13/4 | 14/4 | 15/4 | 16/4 | 17/4 | 18/4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TPR>0.99 & FDR<0.01 | 5 | 66 | 1 | 39 | 39 | 18 | 58 | 39 | 11 | 34 | 17 | 38 | 64 | 17 | 5 | 24 | 29 | 7 | 48 |
| TPR>0.95 & FDR<0.05 | 64 | 88 | 70 | 69 | 75 | 75 | 90 | 86 | 58 | 89 | 67 | 74 | 92 | 76 | 47 | 65 | 70 | 72 | 96 |
| TPR>0.90 & FDR<0.10 | 90 | 94 | 92 | 85 | 86 | 91 | 93 | 98 | 83 | 95 | 86 | 83 | 98 | 93 | 73 | 81 | 82 | 90 | 98 |
| Maximum interval of silence | 17 | 20 | 24 | 21 | 36 | ⩾100 | ⩾100 | ⩾100 | 19 | 27 | 18 | 18 | 29 | ⩾100 | 23 | 50 | 18 | ⩾100 | 28 |

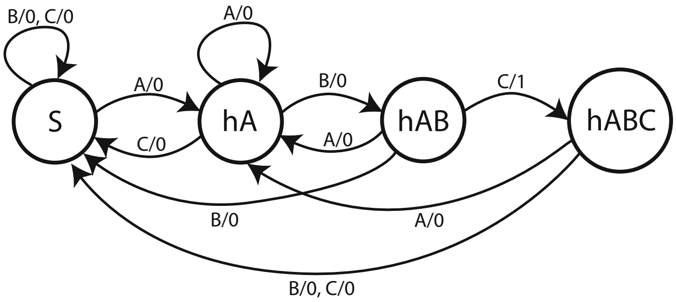


**Fig. 1.** Minimal FST for recognizing ABC. The nodes represent the states and edges represent the transitions from one state to another state on receiving an input symbol {A, B, C} and producing an output {0: no spike(s), 1: spike(s) of the output neuron}.

significantly larger for networks with three interneurons ($p = 0.017$; one-sided test). The reason for this might be that in networks with four interneurons the additional neuron acts as one more source of noise disrupting the memory maintained as self-sustained high-frequency spiking (see below).

We considered the network most robust if it had the highest number of sets of parameter values among 100 sets independently sampled from the robustness ranges (such as shown in Table 1) that gave TPR $>$ 0.99 and FDR $<$ 0.01. Interestingly, the most robust networks (3/3 and 1/4) failed to maintain their state. For mapping the network states on to the states of an FST, we have chosen therefore networks 8/3 and 7/4—the most robust of networks maintaining memory (Figs. 2 and 3).

There are four states in a minimal-size FST that recognises a pattern that consists of three different signals in a specific order in a stream of three signals (Fig. 1). In both networks (8/3 and 7/4) the state of the network after they receive ABC (state hABC, for *had ABC*) is reached after a transition from a state in which all interneurons have zero or zero/low activity (neural states Z or L, respectively; Tables 3 and 4). The same was the case for all the other perfect recognisers obtained in this work (not shown). This means that the output in each network will spike if the input stream consists of a single signal, C. Since we are interested here in recognition in a continuous stream of signals, we do not consider it a serious issue. Perhaps, however, introducing a strong penalty for output spikes after the initial C would allow us to obtain networks with different structure and activity; we plan to investigate this in our future work.

The interneurons of 8/3 are fully connected (Fig. 2), and all the interneurons have excitatory self-loops. However, it is not the case that full connectivity with self-loops for interneurons in networks evolved for three interneurons is a sufficient and necessary condition for state maintenance (for example, 6/3 and 12/3 have such a topology, but do not maintain the state, while 11/3 does so without full connectivity).

Going back to 8/3; both interneurons N1 and N2 self-excite themselves strongly—high-frequency spiking (H state) of N1 and N2 is observed in all states but hAB (which is maintained trivially—all neurons are inactive). When signal

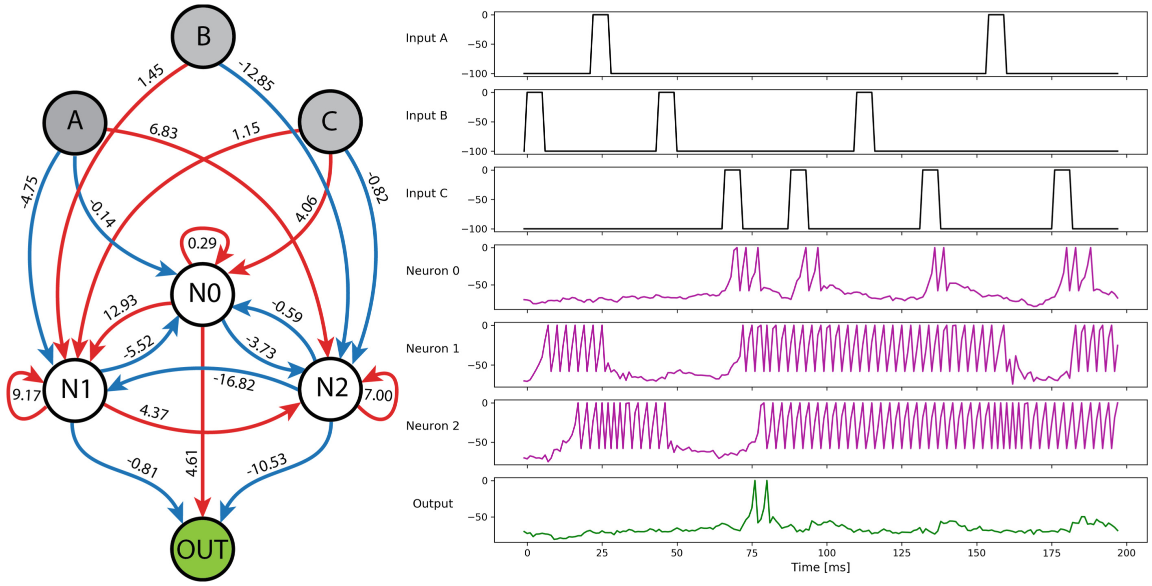


**Fig. 2.** The topology and activity of network 8/3.

**Table 3.** States of the neurons in network 8/3 in network states mapped on the states of the minimal FST. Z: zero, L: (zero or) low, H: high-spiking activity. See text for further details.

| | S | hA | hAB | hABC |
|---|---|---|---|---|
| Neuron 0 | L: 0, 2, 3 spikes | Z | Z | L: 3 spikes |
| Neuron 1 | H: 332 ± 1 Hz | L: 0, 1, 2 spikes | Z | H: 331 ± 1 Hz |
| Neuron 2 | H: 333 Hz | H: 334 ± 1 Hz | L: 1, 2 spikes | H: 329 Hz |
| Output | Z | Z | Z | L: 1, 2 spikes |

A is received, strong connection of input A to N2 puts N2 in the H state, and because of a strongly inhibitory connection both from input A and N2 to N1, N1 is in an L state in the network state hA. The activity of input B strongly inhibits N2; this is why the transition from network state hA to hAB corresponds to L or Z states of all interneurons. When a network in such a state receives a C, the excitatory connection from input C to N0 and N0's weak self-excitation combine to make N0 spike exactly three times, which is necessary for the output to spike once or twice (output can be excited only by N0); connections from N0 to N1 and from N1 to N2 are mainly responsible for putting both N1 and N2 in an H state. When, however, C is received in any other state, either N2 (state hA) or both N1 and N2 (states S and hABC) are in state H; their strong inhibitory connections to output prevent output from spiking (Fig. 2).

Limitations of space prohibit us from providing a similar analysis for 7/4. We do, however, provide the data (Fig. 3, Table 4) sufficient for making it.

Our preliminary analysis of the variability of the ways in which computation in this task is accomplished in networks that show state maintenance indicates that networks evolved with three interneurons belong to four distinct classes

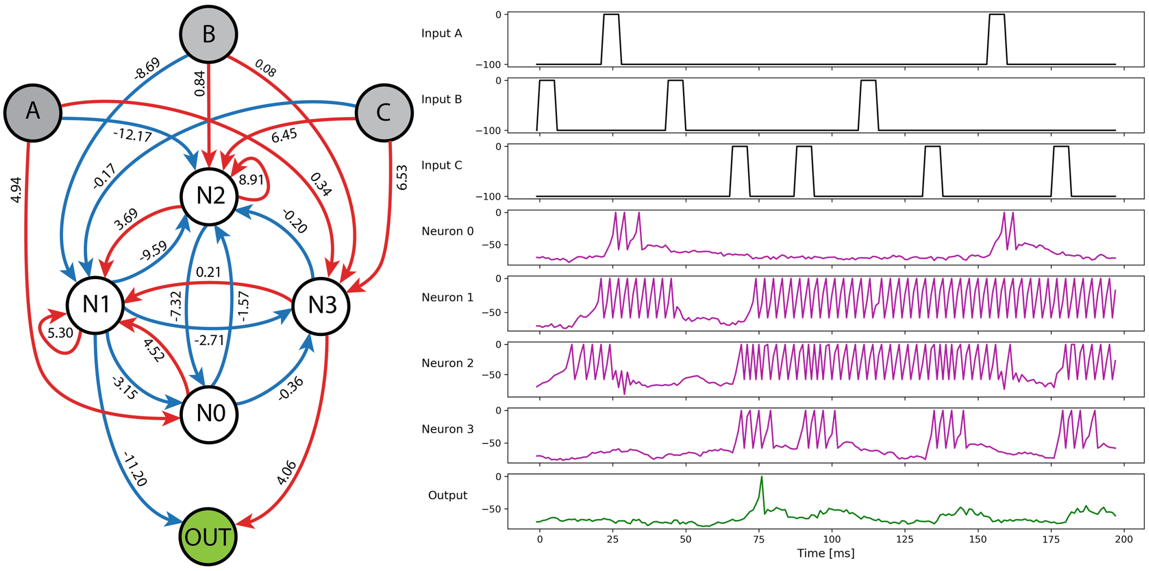


**Fig. 3.** The topology and activity of network 7/4.

**Table 4.** States of the neurons in network 7/4 in network states mapped on the states of the minimal FST. Z: zero, L: (zero or) low, H: high-spiking activity. See text for further details.

| | S | hA | hAB | hABC |
|---|---|---|---|---|
| Neuron 0 | Z | L: 2, 3 spikes | Z | Z |
| Neuron 1 | H: 330 ± 3 Hz | H: 280 ± 3 Hz | L: 1 spike | H: 330 Hz |
| Neuron 2 | H: 332 ± 2 Hz | L: 0, 1, 3 spikes | Z | H: 333 Hz |
| Neuron 3 | L: 0, 4 spikes | Z | Z | L: 4 spikes |
| Output | Z | Z | Z | L: 1, 2 spikes |

based on the assignment of neural states to network states. For network 8/3 we can encode this assignment as (S, hA, hAB, hABC) = (LHH, ZLH, ZZL, LHH), where Z means zero activity, L means zero or low activity (a few spikes at most), and H means high-frequency spiking. The order of symbols in each triplet assigned to a state follows the order of interneurons' labels (Table 3). Three other networks belong to this class, 0/3, 9/3, and 11/3 (such matching requires, of course, appropriate ordering of interneurons in each network). The other three possible classes are: (i) 4/3 and 7/3 have (ZHH, ZHL, ZLZ, LHH), (ii) 2/3 and 5/3 have (HHH, HLZ, LZZ, HHH), and (iii) 10/3 has (HHH, LHH, ZLL, HHH). The four networks that show state maintenance with four interneurons all belong to different classes based on such an assignment: whereas (i) 7/4 has (ZHHL, LHLZ, ZLZZ, ZHHL) (Table 4), (ii) 5/4 has (HZHH, HZHZ, LZHZ, HLHH), (iii) 13/4 has (HHHH, HLLH, LZZL, HHHH), and (iv) 17/4 has (HLLH, LZZH, ZZZL, HLLH). In our future work, we plan to further analyse the relationship between these classes and the network topologies, considering the signs and weights of the connections.

## 4 Conclusions and Future Work

We show that SNNs evolved to perform a simple but not trivial computational task in the presence of noise on neuronal membrane potential are robust to sampling all neuronal parameters from a certain range, and provide a procedure to approximate this range. Not surprisingly, we show that the range for varying all parameters is narrower than for varying a single parameter each time (as we did previously [23]). In future work, we plan to further fine tune this methodology—for example, by giving all neurons different parameters during this procedure, and considering the dependence relationships between parameters (we have observed, for example, that increasing the value of one parameter may allow increasing the value of another).

Setting a limit for the number of interneurons one higher than necessary to accomplish the tasks increased the yield of successful evolutionary runs (i.e., the evolvability), but resulted in a smaller fraction of networks that could maintain their state in the successful runs. Furthermore, there was no significant impact on the range of robustness to changes of parameters between slightly smaller and larger networks. In future work, we plan to investigate if larger networks will allow obtaining solutions in the presence of higher levels of noise. We would also like to see if other models of noise (such as an Ornstein-Uhlenbeck process, commonly used in computational neuroscience) impact evolvability and robustness. Another possible direction for future work is to investigate the evolution of recognition of longer patterns in the presence of noise.

In this work, we performed a preliminary analysis of how the networks accomplish the temporal pattern recognition with state maintenance by assigning neural states in network states corresponding to the state of an FST. We show that the solutions belong to different classes, and thus different topologies can allow solving this task. In future work, we will analyse in more detail the variety of solutions obtained in independent runs. We would also like to see if changing the spiking behavior of neurons during evolution (e.g., to bursting) or the model itself (e.g., to leaky integrate and fire) leads to other classes of solutions.

**Acknowledgements.** This work was supported by the Polish National Science Center (project EvoSN, UMO-2013/08/M/ST6/00922). MY acknowledges the support of the KNOW RNA Research Center in Poznan (No. 01/KNOW2/2014). We are grateful to Volker Steuber and Neil Davey for discussions and suggestions.

## References

1. Ahissar, E., Arieli, A.: Figuring space by time. Neuron **32**, 185–201 (2001)
2. Anderson, J.S., Lampl, I., Gillespie, D.C., Ferster, D.: The contribution of noise to contrast invariance of orientation tuning in cat visual cortex. Science **290**, 1968–1972 (2000)
3. Bialek, W., Rieke, F., de Ruyter van Steveninck, R.R., Warland, D., et al.: Reading a neural code. In: Neural Information Processing Systems, pp. 36–43 (1989)
4. Burnstock, G.: Autonomic neurotransmission: 60 years since sir henry dale. Ann. Rev. Pharmacol. Toxicol. **49**, 1–30 (2009)

5. Buzsáki, G., Chrobak, J.J.: Temporal structure in spatially organized neuronal ensembles: a role for interneuronal networks. Curr. Opin. Neurobiol. **5**, 504–510 (1995)
6. Decharms, R.C., Zador, A.: Neural representation and the cortical code. Ann. Rev. Neurosci. **23**, 613–647 (2000)
7. Destexhe, A., Rudolph, M., Fellous, J.M., Sejnowski, T.: Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. Neuroscience **107**, 13–24 (2001)
8. Destexhe, A., Paré, D.: Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. J. Neurophysiol. **81**, 1531–1547 (1999)
9. Faisal, A.A., Selen, L.P., Wolpert, D.M.: Noise in the nervous system. Nat. Rev. Neurosci. **9**, 292–303 (2008)
10. Finn, I.M., Priebe, N.J., Ferster, D.: The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. Neuron **54**, 137–152 (2007)
11. Florian, R.V.: Biologically inspired neural networks for the control of embodied agents. Center for Cognitive and Neural Studies (Cluj-Napoca, Romania), Technical report Coneural-03-03 (2003)
12. Gerstner, W., Kempter, R., van Hemmen, J.L., Wagner, H.: A neuronal learning rule for sub-millisecond temporal coding. Nature **383**, 76–78 (1996)
13. Huxter, J., Burgess, N., O'keefe, J.: Independent rate and temporal coding in hippocampal pyramidal cells. Nature **425**, 828–832 (2003)
14. Jacobson, G., et al.: Subthreshold voltage noise of rat neocortical pyramidal neurones. J. Physiol. **564**, 145–160 (2005)
15. Laurent, G.: Dynamical representation of odors by oscillating and evolving neural assemblies. Trends Neurosci. **19**, 489–496 (1996)
16. Marder, E.: Variability, compensation, and modulation in neurons and circuits. Proc. Natl. Acad. Sci. USA **108**(Suppl. 3), 15542–15548 (2011)
17. Naud, R., Marcille, N., Clopath, C., Gerstner, W.: Firing patterns in the adaptive exponential integrate-and-fire model. Biol. Cybern. **99**, 335–347 (2008)
18. Paré, D., Shink, E., Gaudreau, H., Destexhe, A., Lang, E.J.: Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo. J. Neurophysiol. **79**, 1450–1460 (1998)
19. Prinz, A.A., Bucher, D., Marder, E.: Similar network activity from disparate circuit parameters. Nat. Neurosci. **7**, 1345–1352 (2004)
20. Stacey, W., Durand, D.: Stochastic resonance improves signal detection in hippocampal neurons. J. Neurophysiol. **83**, 1394–402 (2000)
21. Wiesenfeld, K., Moss, F.: Stochastic resonance and the benefits of noise: from ice ages to crayfish and squids. Nature **373**, 33–36 (1995)
22. Wróbel, B., Abdelmotaleb, A., Joachimczak, M.: Evolving networks processing signals with a mixed paradigm, inspired by gene regulatory networks and spiking neurons. In: Di Caro, G.A., Theraulaz, G. (eds.) BIONETICS 2012. LNICST, vol. 134, pp. 135–149. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06944-9_10
23. Yaqoob, M., Wróbel, B.: Robust very small spiking neural networks evolved with noise to recognize temporal patterns. In: ALIFE 2018: Proceedings of the 2018 Conference on Artificial Life, pp. 665–672. MIT Press (2018)
24. Yaqoob, M., Wróbel, B.: Very small spiking neural networks evolved to recognize a pattern in a continuous input stream. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 3496–3503. IEEE (2017)

## B.4 M. Yaqoob, V. Steuber and B. Wróbel (2019) The Importance of Self-excitation in Spiking Neural Networks Evolved to Recognize Temporal Patterns, International Conference on Artificial Neural Networks, ICANN 2019, Munich, Germany, September 17-19, 2019.

# The Importance of Self-excitation in Spiking Neural Networks Evolved to Recognize Temporal Patterns

Muhammad Yaqoob[1], Volker Steuber[2], and Borys Wróbel[1]()

[1] Evolving Systems Laboratory, Adam Mickiewicz University in Poznan,
Poznan, Poland
{yaqoob,wrobel}@evosys.org
[2] University of Hertfordshire, Hatfield, UK
v.steuber@herts.ac.uk

**Abstract.** Biological and artificial spiking neural networks process information by changing their states in response to the temporal patterns of input and of the activity of the network itself. Here we analyse very small networks, evolved to recognize three signals in a specific pattern (ABC) in a continuous temporal stream of signals (...CABCACB...). This task can be accomplished by networks with just four neurons (three interneurons and one output). We show that evolving the networks in the presence of noise and variation of the intervals of silence between signals biases the solutions towards networks that can maintain their states (a form of memory), while the majority of networks evolved without variable intervals between signals cannot do so. We demonstrate that in most networks, the evolutionary process leads to the presence of superfluous connections that can be pruned without affecting the ability of the networks to perform the task and, if the unpruned network can maintain memory, so does the pruned network. We then analyse how these small networks can perform their tasks, using a paradigm of finite state transducers. This analysis shows that self-excitatory loops (autapses) in these networks are crucial for both the recognition of the pattern and for memory maintenance.

**Keywords:** Temporal pattern recognition · Spiking neural networks · Ex-loops · Self-loops · Artificial evolution · Minimal cognition · Complex networks · Genetic algorithm · Finite state transducer

## 1 Introduction

The current understanding of information processing in biological brains postulates that this processing is accomplished thanks to constant transitions of biological networks from one pattern of spiking activity to another [1,3,5,11,12, 15,23]. Temporal input patterns in all sensory modalities, including smell [25], sight [34], and hearing [14], influence these patterns of activity; and the patterns

I. V. Tetko et al. (Eds.): ICANN 2019, LNCS 11727, pp. 758–771, 2019.
https://doi.org/10.1007/978-3-030-30487-4_59

of neural activity determine the animal behaviour. One of the central problems in neuroscience is how biological neural circuits can accomplish such temporal processing. Answering this question may help in designing bio-inspired artificial cognitive systems. Of special interest is how this processing, which also involves the maintenance of the spiking activity (a form of memory), while depending on the precise timing of spikes, can be accomplished in the presence of noise [8,10]; and indeed may necessitate noise [6,13,29,37].

In this work, we analyse very small spiking neural networks (SNNs) evolved to perform a simple temporal pattern recognition task. We have shown previously that without noise, two interneurons are sufficient for this task, but such networks are fragile to even the slightest variation of the timing of inputs [39]. In contrast, networks with three interneurons can be evolved to recognize patterns consisting of three stimuli in the presence of noise, and they are robust to a change of neuronal parameters or duration of intervals of silence between the stimuli [38,40]. In this work, we use the same model of noise (on the membrane voltage) as previously; while its level is biologically realistic, and so including it adds to the biological plausibility of our model, our primary concern is to aid in the evolution of networks that can maintain their states (a form of memory) even as the intervals between stimuli are hugely increased when testing the evolved network. One of the original contributions of this paper is that evolving the networks both in the presence of noise and the variation of intervals between stimuli biases the networks towards those that can maintain their states.

We observe that a variety of network topologies resulting from an artificial evolutionary process can perform the same computational task [38–40]. This is also the case for biological networks [18,22]. By using artificial evolution, we are able to find the commonalities between the networks that can accomplish simple, but not trivial, computational tasks.

The recognition of temporal pattern requires temporal storage of the stimulus or delays [16,30–33]. Since our networks are very small, delays caused by synaptic delays are minimal. The main contribution of this paper is that the crucial connections that maintain the network state and memory in the presence of variable silent intervals are self-excitatory loops (autapses), which sheds new light on the importance of these connections that are commonly found in biological neural systems [26,36]. Furthermore, persistent spiking activity in response to short sensory input is common in all areas of brain [17] which perhaps is responsible for keeping short-term memory in accumulating tasks [27].

## 2 Methods

Each network in our model is encoded in a linear genome, and consists of three inputs, three interneurons, and one output neuron [38–40]. Inputs are not allowed to connect to the output neuron directly and only interneurons can have self-loops. A fully connected network with this structure can have up to 21 connections (up to nine connections from inputs to interneurons, six connections between the interneurons, three self-loops, and three connections from the interneurons to the output neuron).

Each input is dedicated to one signal (stimulus type), denoted as A, B and C. The interneurons and output neuron are modelled using Euler integration with 1 ms steps of the differential equations for adaptive exponential integrate and fire neurons [20]; we use the same parameter values as in [38–40]; these values result in tonic spiking in response to constant input current. Since this study focuses on the effect of network connectivity, the neuronal parameters are kept constant (allowing them to evolve would hugely increase the search space of the artificial evolutionary process). To simulate noisy synaptic background at a biologically realistic level [2,7,9,21], we add a random value taken from a normal distribution with standard deviation 2 mV and mean 0 to the membrane potential of each neuron at every 1 ms simulation step. When a neuron receives a spike the excitatory $g_E$ or inhibitory $g_I$ conductance is updated by the connection weight multiplied by the respective conductance gain. The value of the excitatory and inhibitory gain is 7 nS.

The task of the networks is to recognize three signals in a particular order (ABC) in a continuous random sequence (...BCACACC**ABC**ACBAC...), in which all signals appear with equal probability, and thus the correct patterns take up about 10% of time. To generate a variety of solutions, we use a genetic algorithm with a population of 300 individuals, with 100 independent runs for each of the two settings: in the first setting signals are followed by a constant interval of silence (16 ms), in the second setting the intervals vary, with a uniform distribution between 16 and 32 ms (in previous work, [38,40], we used noise on the membrane potential, but did not vary the interval of silences). In both settings the length of a each signal is 6 ms. We use the same genetic operators as in [40]; they can result in changes of weights, deletion and addition of edges (synapses) and the nodes (neurons) in the network (through deletion and duplication, respectively, of consecutive elements in the linear genomes; however, the maximum size of the network was limited as described above).

Each individual in the population in each generation is evaluated on six sequences. Four out of these six sequences are generated randomly with equiprobable occurrence of three signals A, B and C; the remaining two sequences consist of four concatenated patterns in random order: ABC and three patterns that are hard to distinguish from this target (ABA, ABB, and BBC). The fitness function [38,40] rewards networks in which the output neuron spiked (at least once) in the correct intervals, and did not spike in the incorrect intervals: $f_{fitness} = 1 - R + 4P$, where $R$, reward ($P$, penalty) is the fraction of correct (incorrect) intervals in which output spiked. $P$ is multiplied by 4 in this formulation because its denominator is much larger than the numerator for the networks that have correct performance or are close to it (when the input sequence is random, 90% of intervals are incorrect). The correct intervals are those that start at the onset of the last signal (C) of the correct pattern (ABC) and end with the end of the silence that follows. Similarly, incorrect intervals start at the onset of each signal that is not C in ABC. Both correct and incorrect intervals last either 22 ms or, in the setting with variable silence intervals, 22–38 ms.

The false discovery rate (FDR) of the network is defined as the number of incorrect intervals in which output spiked divided by the sum of both incorrect and correct intervals in which the output spiked. The true positive rate (TPR) of the network is the same as $R$. We define a champion in a run as a perfect recognizer if its TPR is above 0.99 and FDR below 0.01 for the settings (constant or variable interval of silence between the signals) under which a champion was evolved.

In order to simplify the network analysis, we use pruning of superfluous edges in the network. Our pruning algorithm removes excessive connections in two steps in a loop: (i) a random connection is removed for testing; (ii) if $\text{TPR} < 0.95$ or $\text{FDR} > 0.05$, the connection is reinstated and labelled as vital; the loop is terminated when all the connections are labelled as vital.

## 3 Results and Discussion

Out of 100 independent runs in the presence of noise on membrane potential but with constant interval of silence between the signals, 15 ended with champions that were perfect recognizers; when in addition to noise the intervals of silence varied during evolution, the yield was 12%.

Even though our artificial evolutionary process allows for deletion of nodes (neurons) in the network, none of the perfect recognizers had less than three interneurons. In addition, even though pruning can result in a disconnection of a node, no network ended up with less than three interneurons after pruning. Perfect recognizers evolved only with noise had slightly more (19.20 on average; Table 1) edges than the perfect recognizers evolved also with variation of silences (18.83). This difference persisted after pruning (14.26 and 13.08 edges, respectively). None of these differences were statistically significant.

We tested both the evolved and pruned networks on a random sequence with 100,000 signals and 100 ms intervals of silence between signals. Our results (Table 1) show, firstly, that evolving the networks with both noise and variation of silences resulted in more perfect recognizers that can keep memory (11 out of 12) than for evolving only with noise (4 out of 15). Secondly, all these $11 + 4 = 15$ perfect recognizers kept memory also after pruning, demonstrating that the removed connections are unnecessary not only for recognizing the pattern but also for keeping memory.

Interestingly, while the perfect recognizers that kept memory had fewer self-excitatory loops after pruning (all had 2; Table 1) than the champions which did not keep memory (which on average had 2.42), the sum of weights of the self-excitatory loops was significantly higher in recognizers that kept memory (mean sum 14.6 vs. 12.3; $p = 0.002$, one-sided Wilcoxon test). This suggests that the memory is maintained in these networks through self-excitation; we will explore this issue further below by analysing the mechanisms by which some networks keep memory while other fail to do so.

Based on the TPR and FDR with 100 ms intervals of silence (Table 1), we can divide the 27 perfect recognizers into four groups: (i) 15 memory-keepers

**Table 1.** The number of edges and self-excitatory loops in perfect recognizers evolved with noise and constant (top, 15 champions) or variable (bottom, 12 champions) silences, and their robustness to the increase of silences to 100 ms

Champions evolved in the presence of constant (16 ms) silence intervals

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evolved edges | 19 | 21 | 20 | 19 | 18 | 19 | 19 | 18 | 21 | 16 | 20 | 20 | 19 | 20 | 19 |
| edges after pruning | 10 | 14 | 15 | 13 | 11 | 14 | 16 | 15 | 16 | 16 | 17 | 16 | 15 | 13 | 13 |
| self ex-loops | 1 | 3 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 2 | 2 | 2 | 2 |
| 100 ms TPR | 0.02 | 0.98 | 0.98 | 0.97 | 0.96 | 0.95 | 0.99 | 0.55 | 0.99 | 0.00 | 0.00 | 0.00 | 0.96 | 0.99 | 0.96 |
| 100 ms FDR | 0.99 | 0.37 | 0.01 | 0.12 | 0.56 | 0.01 | 0.03 | 0.90 | 0.20 | 1.00 | 1.00 | 0.99 | 0.58 | 0.01 | 0.56 |

Champions evolved in the presence of noise and variable (16-32 ms) intervals of silence

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| evolved edges | 19 | 21 | 20 | 19 | 20 | 18 | 16 | 20 | 18 | 19 | 18 | 18 |
| edges after pruning | 13 | 12 | 14 | 14 | 14 | 13 | 11 | 13 | 15 | 12 | 13 | 13 |
| self ex-loops | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 100ms TPR | 0.99 | 0.99 | 0.99 | 0.98 | 0.90 | 0.98 | 0.99 | 0.98 | 0.99 | 0.97 | 0.98 | 0.99 |
| 100ms FDR | 0.04 | 0.01 | 0.01 | 0.01 | 0.40 | 0.04 | 0.01 | 0.01 | 0.04 | 0.01 | 0.02 | 0.01 |

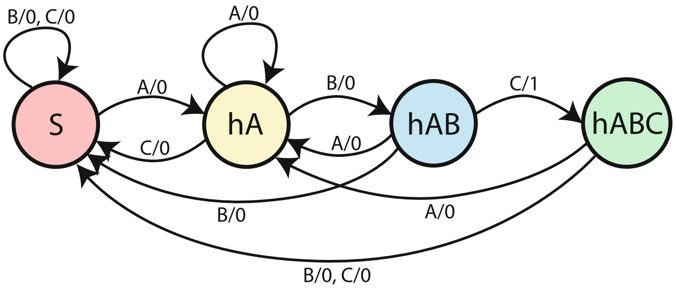


**Fig. 1.** Finite state transducer for recognizing ABC. The symbols above each arrow correspond to the input (A, B, C) and output (0: no spiking, 1: spiking of the output neuron).

(TPR remains high, and FDR low; evolved only with noise: number 2, 5, 6, 13; evolved with both noise and variation of silences: all except number 4), (ii) over-recognizers (TPR stays high, but FDR increases; evolved only with noise: 1, 3, 4, 8, 12, 14; evolved with both: number 4), (iii) wrong-recognizers (low TPR, high FDR; evolved only with noise: number 0, 7, 10, 11), (iv) mute networks (champion 9 evolved only with noise), for which long intervals of silence between signals result in only noise driven activity (negligible) of the output neuron. To illustrate what allows for both recognition and memory, we first analyse one memory-keeper evolved with both noise and variation of silences (champion 6; Fig. 2), one over-recognizer (champion 4; Fig. 3), one wrong-recognizer (champion 7; Fig. 4), and the mute network (champion 9; Fig. 5). The champions 4, 7 and 9 fail when silences are 50 ms long for the same reasons they fail with 100 ms silences; we use 50 ms in the figures to keep them compact. While the activities of the pruned networks that fail are slightly different from the activities of the evolved networks, they fail for the same reasons; we will present only the analysis of the pruned networks for simplicity.

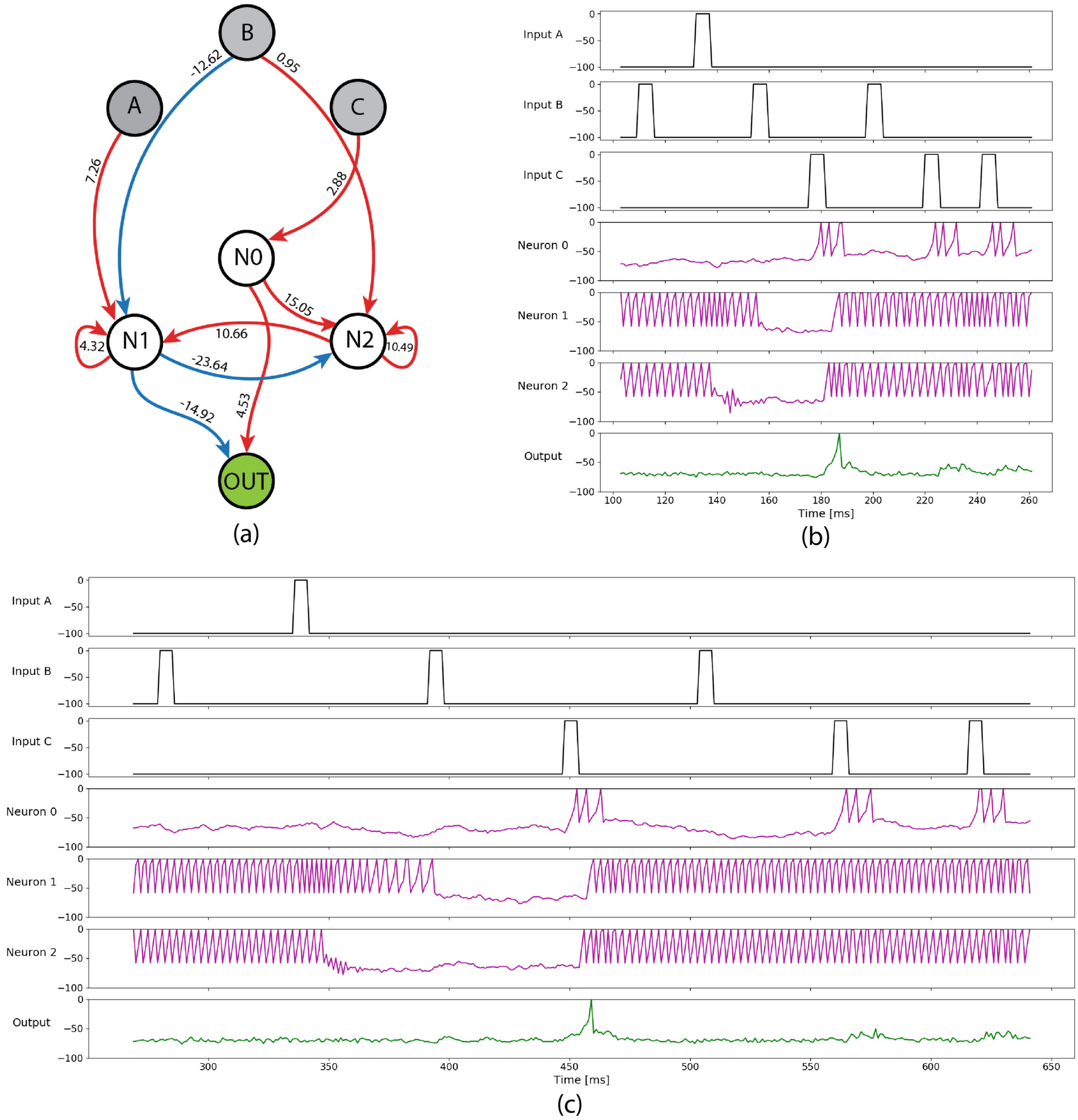


**Fig. 2.** Champion 6 evolved with both noise and variation of silences. The activity of the pruned network (a) is shown for short (16 ms; b), and long (50 ms; c) silences, which indicates that this champion is a memory-keeper.

In order to analyse the transitions of the network, we will use the paradigm we proposed previously, based on mapping the network states onto the states of a finite state transducer (FST) [38–40]. A FST, a formal computational model [28], is frequently used for analysing computations on time series performed in an online manner (that is, constantly producing an output for a continuous input). The minimal FST for recognizing a pattern with three symbols has four states (Fig. 1).

In the case of our memory-keeper (Fig. 2), the activity of input A, because of its excitatory connection to N1, makes N1 spike; this spiking continues thanks to the excitatory self-loop, while N0, N2, and output remain silent. Thus, we can denote the state hA (for 'had A', Fig. 1) as LHL, where L (H) means low (high) activity for interneurons in the order (N0, N1, N2). The state hAB (when the network receives B after A, for 'had AB') can be denoted as LLL, and the state hABC ('had ABC') as LHH. The only difference between the start state (S) and hABC is the intermittent activity of the output. Thus, the state of the network while waiting for the last signal in the pattern (state hAB) is maintained passively (all interneurons are silent), while the other states, hA and hABC/S, are maintained by the self-excitatory loops on N1 and N2. In hA, only N1 is continuously active; in hABC/S, both N0 and N1 are. The inhibitory connection from N1 to the output ensures that receiving a signal C will cause the output to spike (after N0 spikes) only when N1—and N2, which activates N1—are silent. The inhibitory connection from input B to N1 is necessary for N1 to cease its activity in hAB, but this does not happen when N2 is active (so when network is in the state hABC or S, it goes to S after receiving B). The inhibitory connection from N1 to N2 is necessary for the transition from hABC/S to hA (higher frequency of N1 shuts down N2). Finally, the weak excitatory connection from B to N2 is necessary to ensure that when N2 is silent (hAB), receiving a B would not silence both N1 and N2; indeed, when this connection is removed, the output wrongly spikes after receiving ABBC, ABBBC, etc. (the network recognizes the regular expression $AB^{+}C$, not just ABC).

While we only describe one memory-keeper here, in all such networks analysed so far, the state hAB is represented by LLL (and thus the networks will spike when they are initiated with no activity and receive just a C), and the networks maintain two states stably: hA and hABC/S (which differ only by the short-term activity of the output, triggered by the transition from hAB to hABC). Since the over-recognizer we have chosen for analysis (Fig. 3) shares its topology with the memory-keeper (Fig. 2), both recognize ABC correctly in the same fashion when the silence intervals are short—when the network receives A, it goes to the state LHL, when B follows, to LLL, and when C follows, to LHH. However, when the silences are long, the activity of N2 in LHH dies out (because the N2 self-excitatory loop is weak), and the network goes to LHL—the same as hA. When B is received in this state, all activity ceases (state LLL). If the next signal is C, the output neuron spikes, wrongly. This leads to a high FDR—the network recognizes the pattern BC when the intervals are long.

The analysis of the network activity of the wrong-recognizer (Fig. 4) reveals that the recognition when the intervals of silence are short depends on transitions from one unstable state to another. We can see that when the intervals are long, the S state (HHH) is stable, and maintained by the strong self-excitatory loop on N1. When A is received, N0 speeds up, and inhibits N1 (the networks goes to the state HLH). If the silence continues, the network goes to the state LLH (N0 does not have any self-loop), and then LLL (the self-loop of N2 is too weak to maintain its activity for long). If C is received at this point, N1 spikes, and without inhibition from N2, the output spikes, leading to the recognition of AC. When the intervals are short, the network transitions along the same trajectory when it recognizes ABC, but much quicker—when it is still in the state HLH after receiving A, the arrival of a B pushes it to LLH, which can relax to LLL

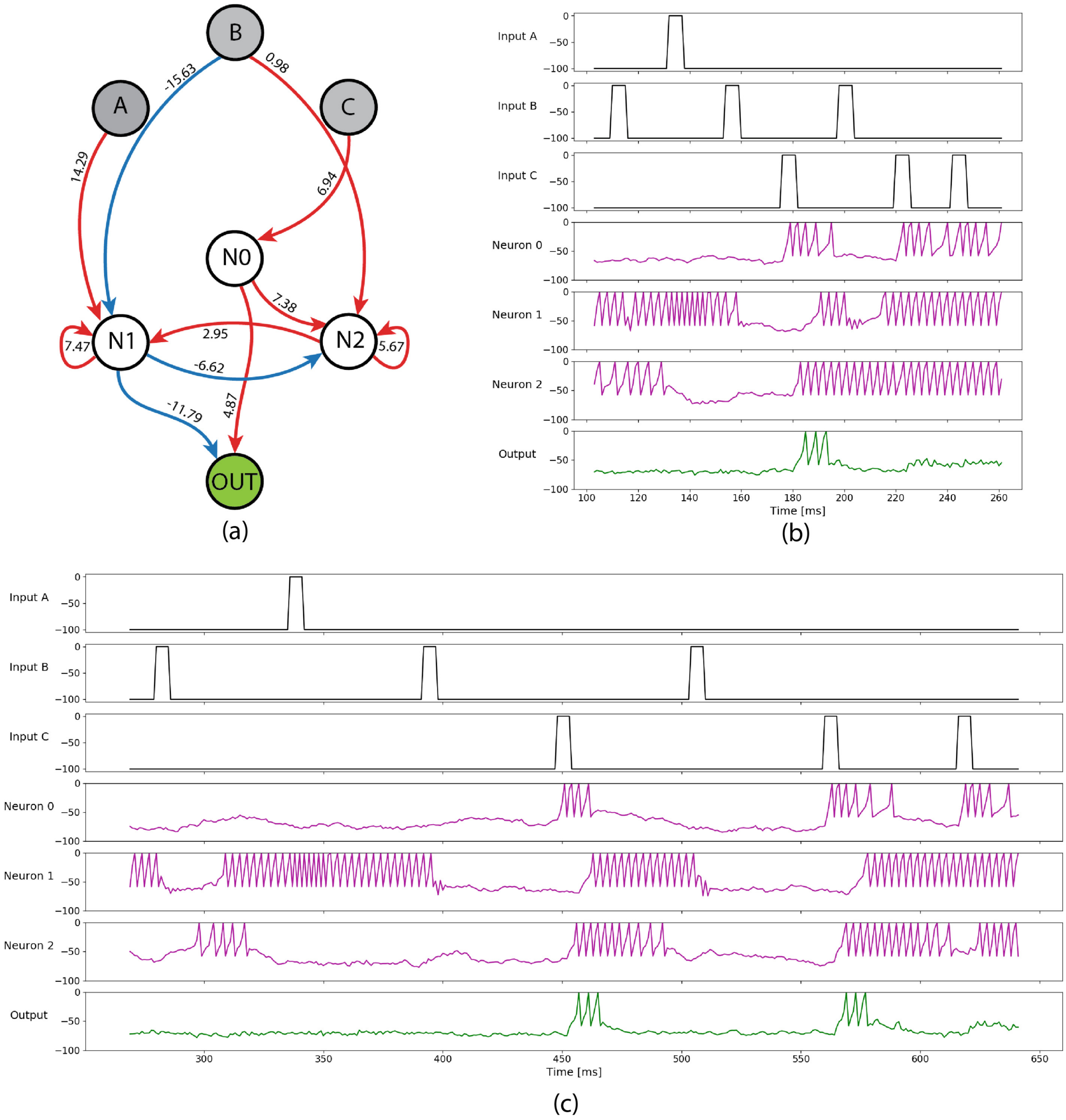


**Fig. 3.** Champion 4 evolved with noise and constant silences. The activity of the pruned network (a) is shown for short (16 ms; b), and long (50 ms; c) silences, for which this champion behaves as an over-recognizer.

in time to release the output from the inhibition from N2 when the spike of N1, induced by receiving a C, arrives. Any Bs that do not follow an A after a short interval of silence, and any Cs, cause the network to go the stable state HHH.

Finally, in champion 9 evolved with constant short intervals of silence, the recognition of ABC crucially depends on the network being in a particular unstable state when the network receives a C after having received the pattern AB—to activate the output, N2 (active after C is received) needs to spike fast; and for N2 to spike fast, N2 needs to receive also the activation of N1. However, N1 cannot spike too frequently because it inhibits the output. In addition, N0 (which also inhibits the output) needs to be inactive. This particular state can only be

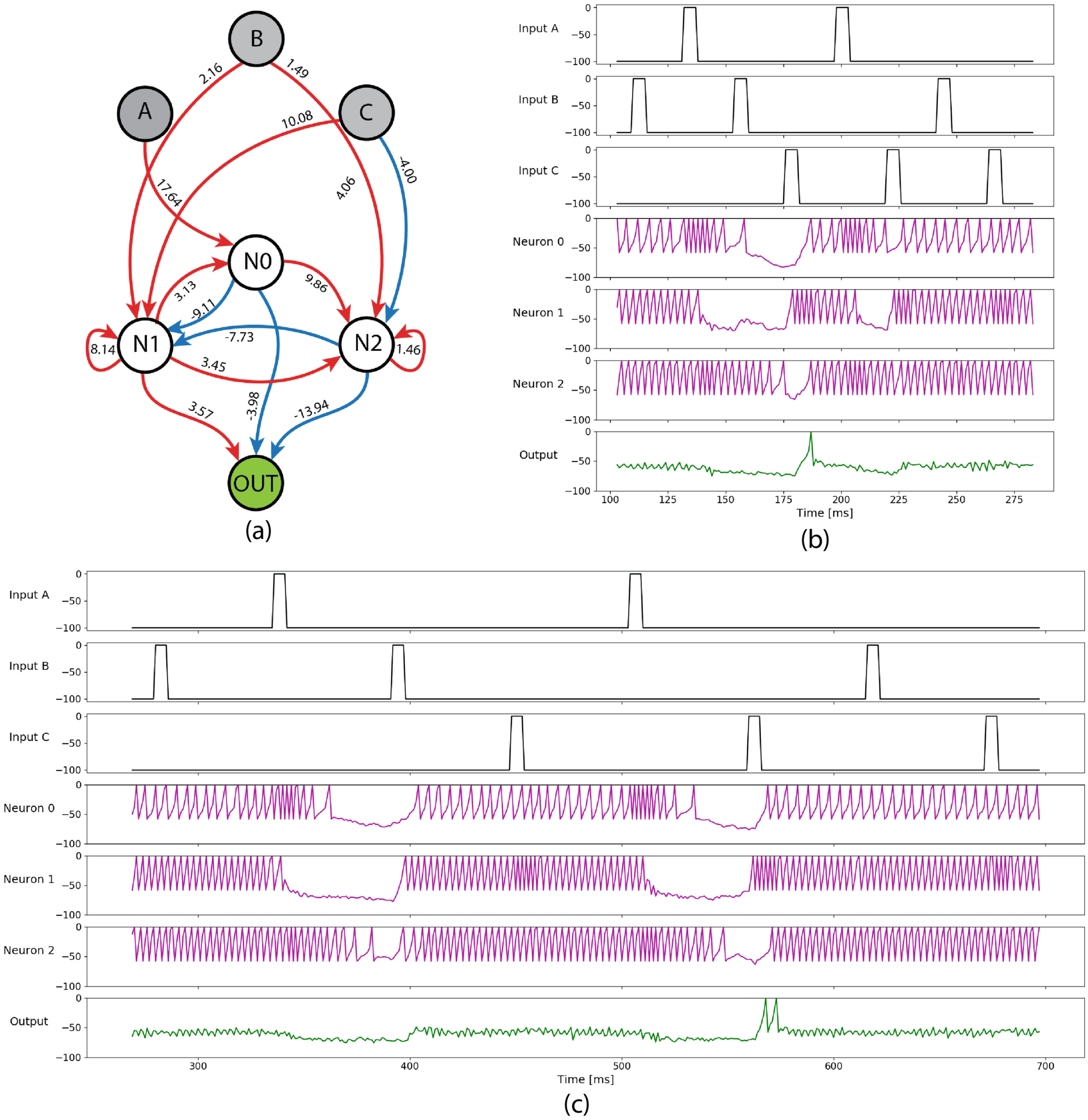
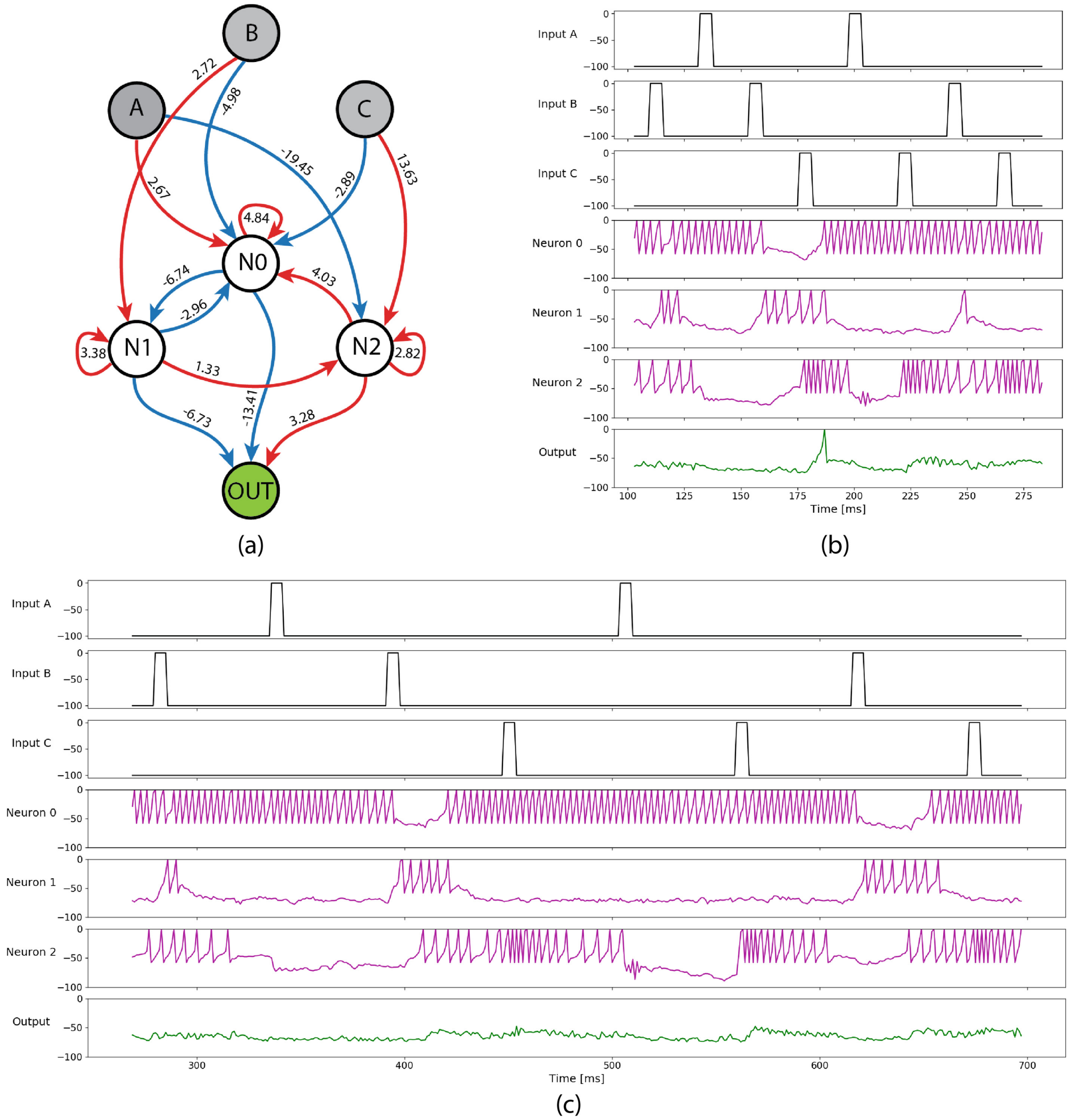


**Fig. 4.** Champion 7 evolved with noise and constant silences. The activity of the pruned network (a) is shown for short (16 ms; b), and long (50 ms; c) silences, for which this champion behaves as a wrong-recognizer.

achieved if the network first reaches the state HLL (which is stable thanks to the self-excitatory loop on N0). With short silences, this state is reached after receiving an A. When the silence is long, the right conditions for the output to spike never occur—even though there are times when N1 spikes slowly with N0 inactive, N2 never spikes frequently enough at that time to drive the output to spike. Moreover, even though N1 also has a self-excitatory loop, its activity cannot be sustained for long when N0 (which inhibits N1) is active, and because N1 activates N2, which in turn activates N0, N1 can only spike slowly.

Our experimental setup did not impose the Dale's rule [4] on the evolved networks. This is because our preliminary experiments showed that imposing the rule would require permitting more (roughly double) interneurons during



(a)

(b)

(c)

**Fig. 5.** Champion 9 evolved with noise and constant silences. The activity of the pruned network (a) is shown for short (16 ms; b), and long (50 ms; c) silences, for which this champion behaves as a mute network.

evolution, increasing the search space. All of the networks we analysed had at least one interneuron which had both excitatory and inhibitory connections to other neurons in the network, even after pruning the superfluous connections. However, once a network is evolved, it is straightforward to transform it to a network that conforms to the Dale's rule. This can be done by splitting a neuron that violates the rule into two new neurons, one excitatory and one inhibitory (Fig. 6). Pruning superfluous connections from the network leads to fewer neurons for which such splitting is necessary. Both new neurons receive the same inputs (with the same weights) as the original neuron. The weights of the outgoing connections are also maintained. If an excitatory self-loop is present, it is maintained with the same weight for the new excitatory neuron, and a new excitatory connection with the same weight is created from the new excitatory to the new inhibitory neuron (an analogous operation can be made for an inhibitory self-loop, should one exist). The formation of this new connection could, in principle, affect the functionality of the network, as it creates an additional synaptic delay. However, the networks evolved with noise can be expected to be robust to such a perturbation. On the other hand, creating two noisy neurons instead of one creates, in principle, a more noisy network. The performance of the networks analysed in this paper was not affected by the transformation detailed here. This applies, in particular, to champion 6, the memory-keeper, who has only one neuron (N1) that violates the rule (Figs. 2 and 6).

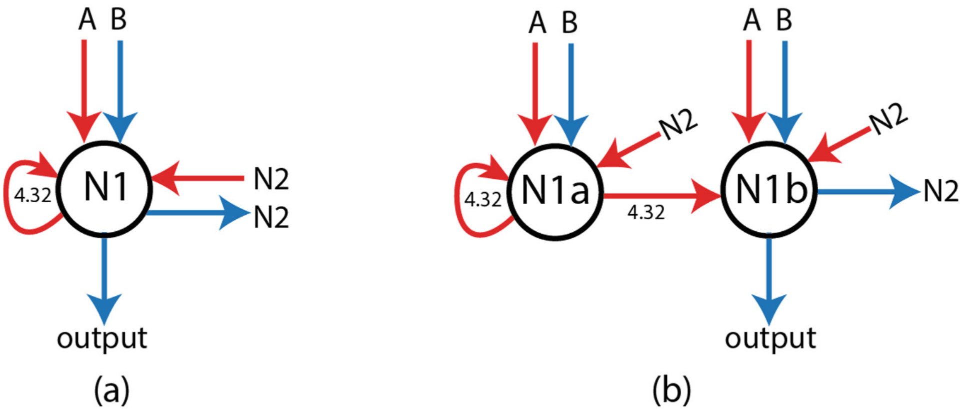


**Fig. 6.** Splitting a neuron that violates the Dale's rule in order to create the network that conforms to the rule. The single interneuron, N1, that violates the rule in champion 6 (Fig. 2), with two inhibitory outputs and one excitatory self-loop (a), can be split into two new neurons, one excitatory and one inhibitory (b).

## 4 Conclusions and Future Work

Our analysis of very small networks evolved to recognize simple temporal patters reveals that many connections in these networks can be removed without impairing the performance of the network. Such pruning allows for a much easier understanding of the mechanisms in which the networks accomplish this minimally cognitive task.

For the networks analysed in this paper, these mechanisms depend crucially on the presence of strong self-excitatory loops, necessary for both the pattern recognition and maintenance of network state—a form of memory. Our results indicate that to recognize a pattern consisting of three symbols with state maintenance, the networks need to consist of at least three interneurons (with one output), and need to have two self-excitatory loops with the weights sufficient to maintain the network states. Our analysis of the activity of the networks that keep memory, by mapping the network states on the states of an FST, shows that all the perfect recognizers that maintain the states of the network represent the state before the arrival of the final symbol by inactivity of the network (a state that does not need to be maintained actively). In all these networks, the accepting state differs from the start state only by the intermittent activity of the output (triggered on the transition to this state from the state of inactivity). This state is maintained actively, and so is the state reached after receiving the first symbol in the pattern.

Our analysis of the networks that fail to maintain the memory correctly reveals the following preliminary insights. With long intervals of silence between the signals, over-recognition happens when the network does not maintain the start state or the state after the correct pattern is recognized (which may be the same), but instead over the long interval of silence transitions to the same state as the one reached after receiving the first symbol in the pattern. With long silences, such networks continue to recognize the correct pattern but start to recognize wrong patterns. On the other hand, the networks that cannot maintain the other states with long silences can either stop recognizing the correct pattern while recognizing wrong ones, or become completely mute.

We show that the perfect recognizers that keep memory function essentially follow the paradigm of an FST. Previous work on creating finite state machines based on recurrent spiking neural networks [19,24,35] considered large multilayer networks. Here we show, essentially, a method to obtain very small networks that work a finite automata using artificial evolution. In future work, we plan to investigate the limits of the length of the temporal patterns for which prefect recognizers that keep memory can be evolved, and how many self-excitatory loops are necessary in such recognizers, for both shorter and longer patterns.

In the work reported here and previously [38–40], we have allowed only for the topology to change. In principle, the neuronal parameters could also be evolved, but this would hugely increase the search space. However, other spiking behaviours (for example, bursting) of the neurons in the network could perhaps lead to different classes of solutions. We plan to explore this issue in our future work using two approaches: (i) allowing a discrete change of the behaviour of each neuron in the network during evolution (for example, from tonic spiking to bursting, a change of the values of several parameters in one step), (ii) by exploring if the solutions change when all the neurons in the network have the same behaviour (different than used here). We could also modify our model of artificial evolution to allow for a more efficient search for the solutions; a different evolutionary model might possibly also lead to different classes of solutions.

Furthermore, we plan to revisit the question of robustness of the evolved networks to changes of parameters and synaptic weights. We also plan to investigate if other models of noise (such as an Ornstein-Uhlenbeck process, commonly used in computational neuroscience), variation of silences or neuronal parameters during evolution will influence the types of solutions, their evolvability and robustness.

**Acknowledgements.** This work was supported by the Polish National Science Center (project EvoSN, UMO-2013/08/M/ST6/00922). MY acknowledges the support of the KNOW RNA Research Center in Poznan (No. 01/KNOW2/2014) and POWR.03.02.00-00-I006/17.

## References

1. Ahissar, E., Arieli, A.: Figuring space by time. Neuron **32**, 185–201 (2001)
2. Anderson, J.S., Lampl, I., Gillespie, D.C., Ferster, D.: The contribution of noise to contrast invariance of orientation tuning in cat visual cortex. Science **290**, 1968–1972 (2000)
3. Bialek, W., Rieke, F., van Steveninck, R.R.d.R., Warland, D., et al.: Reading a neural code. In: Neural Information Processing Systems, pp. 36–43 (1989)
4. Burnstock, G.: Autonomic neurotransmission: 60 years since sir Henry Dale. Annu. Rev. Pharmacol. Toxicol. **49**, 1–30 (2009)
5. Decharms, R.C., Zador, A.: Neural representation and the cortical code. Annu. Rev. Neurosci. **23**, 613–647 (2000)
6. Destexhe, A., Rudolph, M., Fellous, J.M., Sejnowski, T.: Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. Neuroscience **107**, 13–24 (2001)
7. Destexhe, A., Paré, D.: Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. J. Neurophysiol. **81**, 1531–1547 (1999)
8. Faisal, A.A., Selen, L.P., Wolpert, D.M.: Noise in the nervous system. Nat. Rev. Neurosci. **9**, 292–303 (2008)
9. Finn, I.M., Priebe, N.J., Ferster, D.: The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. Neuron **54**, 137–152 (2007)
10. Florian, R.V.: Biologically inspired neural networks for the control of embodied agents. Center for Cognitive and Neural Studies (Cluj-Napoca, Romania), Tech. rep. Coneural-03-03 (2003)
11. Gerstner, W., Kempter, R., van Hemmen, J.L., Wagner, H.: A neuronal learning rule for sub-millisecond temporal coding. Nature **383**, 76–78 (1996)
12. Huxter, J., Burgess, N., Okeefe, J.: Independent rate and temporal coding in hippocampal pyramidal cells. Nature **425**, 828–832 (2003)
13. Jacobson, G., et al.: Subthreshold voltage noise of rat neocortical pyramidal neurones. J. Physiol. **564**, 145–60 (2005)
14. Joris, P., Yin, T.: A matter of time: internal delays in binaural processing. Trends Neurosci. **30**, 70–78 (2007)
15. Laurent, G.: Dynamical representation of odors by oscillating and evolving neural assemblies. Trends Neurosci. **19**, 489–496 (1996)
16. Maex, R., Steuber, V.: The first second: models of short-term memory traces in the brain. Neural Netw. **22**, 1105–1112 (2009)
17. Major, G., Tank, D.: Persistent neural activity: prevalence and mechanisms. Curr. Opin. Neurobiol. **14**, 675–684 (2004)
18. Marder, E.: Variability, compensation, and modulation in neurons and circuits. Proc. Nat. Acad. Sci. U.S.A. **108**, 15542–15548 (2011)
19. Natschläger, T., Maass, W.: Spiking neurons and the induction of finite state machines. Theoret. Comput. Sci. **287**, 251–265 (2002)
20. Naud, R., Marcille, N., Clopath, C., Gerstner, W.: Firing patterns in the adaptive exponential integrate-and-fire model. Biol. Cybern. **99**, 335–347 (2008)
21. Paré, D., Shink, E., Gaudreau, H., Destexhe, A., Lang, E.J.: Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo. J. Neurophysiol. **79**, 1450–1460 (1998)
22. Prinz, A.A., Bucher, D., Marder, E.: Similar network activity from disparate circuit parameters. Nat. Neurosci. **7**, 1345–1352 (2004)

23. Rieke, F., Warland, D., de Ruyter van Steveninck, R., Bialek, W.: Spikes: Exploring the Neural Code. MIT Press, Cambridge (1999)
24. Rutishauser, U., Douglas, R.J.: State-dependent computation using coupled recurrent networks. Neural Comput. **21**, 478–509 (2009)
25. Isaacson, J.S.: Odor representations in mammalian cortical circuits. Curr. Opin. Neurobiol. **20**, 328–31 (2010)
26. Saada, R., Miller, N., Hurwitz, I., Susswein, A.J.: Autaptic excitation elicits persistent activity and a plateau potential in a neuron of known behavioral function. Curr. Biol. **19**, 479–84 (2009)
27. Seung, H.S., Lee, D.D., Reis, B.Y., Tank, D.W.: The autapse: a simple illustration of short-term analog memory storage by tuned synaptic feedback. J. Comput. Neurosci. **9**, 171–185 (2000)
28. Sipser, M.: Introduction to the Theory of Computation. International Thomson Publishing, Stamford (1996)
29. Stacey, W., Durand, D.: Stochastic resonance improves signal detection in hippocampal neurons. J. Neurophysiol. **83**, 1394–1402 (2000)
30. Steuber, V., De Schutter, E.: Rank order decoding of temporal parallel fibre input patterns in a complex Purkinje cell model. Neurocomputing **44–46**, 183–188 (2002)
31. Steuber, V., Willshaw, D.J.: Adaptive leaky integrator models of cerebellar Purkinje cells can learn the clustering of temporal patterns. Neurocomputing **26–27**, 271–276 (1999)
32. Steuber, V., Willshaw, D.: A biophysical model of synaptic delay learning and temporal pattern recognition in a cerebellar Purkinje cell. J. Comput. Neurosci. **17**, 149–164 (2004)
33. Steuber, V., Willshaw, D., Ooyen, A.V.: Generation of time delays: simplified models of intracellular signalling in cerebellar Purkinje cells. Netw. Comput. Neural Syst. **17**, 173–191 (2006)
34. Thorpe, S., Fize, D., Marlot, C.: Speed of processing in the human visual system. Nature **381**, 520–522 (1996)
35. Tino, P., Mills, A.J.S.: Learning beyond finite memory in recurrent networks of spiking neurons. Neural Comput. **18**, 591–613 (2005)
36. Wang, C., et al.: Formation of autapse connected to neuron and its biological function. Complexity **2017**, 1–9 (2017)
37. Wiesenfeld, K., Moss, F.: Stochastic resonance and the benefits of noise: from ice ages to crayfish and squids. Nature **373**, 33–36 (1995)
38. Yaqoob, M., Wróbel, B.: Robust very small spiking neural networks evolved with noise to recognize temporal patterns. In: ALIFE 2018: Proceedings of the 2018 Conference on Artificial Life - MIT Press, pp. 665–672 (2018)
39. Yaqoob, M., Wróbel, B.: Very small spiking neural networks evolved to recognize a pattern in a continuous input stream. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI) - IEEE, pp. 3496–3503 (2017)
40. Yaqoob, M., Wróbel, B.: Very small spiking neural networks evolved for temporal pattern recognition and robust to perturbed neuronal parameters. In: Artificial Neural Networks and Machine Learning - ICANN, pp. 322–331 (2018)

# Bibliography

[1] Ahmed Abdelmotaleb, Neil Davey, Maria Schilstra, Volker Steuber, and Borys Wróbel. Evolving spiking neural networks for temporal pattern recognition in the presence of noise. *Artificial Life 2014*, 2014.

[2] EC Adrian. The basis of sensation, new york, w. w, 1928.

[3] Ehud Ahissar and Amos Arieli. Figuring space by time. *Neuron*, 32:185–201, 2001.

[4] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[5] Jeffrey S. Anderson, Ilan Lampl, Deda C. Gillespie, and David Ferster. The contribution of noise to contrast invariance of orientation tuning in cat visual cortex. *Science*, 290:1968–1972, 2000.

[6] A. Bacci and J. Huguenard. Enhancement of spike-timing precision by autaptic transmission in neocortical inhibitory interneurons. *Neuron*, 49:119–130, 2006.

[7] Karim Benchenane, Adrien Peyrache, Mehdi Khamassi, Patrick L Tierney, Yves Gioanni, Francesco P Battaglia, and Sidney I Wiener. Coherent theta oscillations and reorganization of spike timing in the hippocampal-prefrontal network upon learning. *Neuron*, 66(6):921–936, 2010.

[8] Nils Bertschinger and Thomas Natschlger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, 16:1413–36, 08 2004.

[9] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[10] William Bialek, Fred Rieke, Robert R de Ruyter van Steveninck, David Warland, et al. Reading a neural code. In *NIPS*, pages 36–43, 1989.

[11] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 419–424. Bruges, 2000.

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

[13] Nicolas Brunel and Mark CW Van Rossum. Lapicques 1907 paper: from frogs to integrate-and-fire. *Biological cybernetics*, 97(5):337–339, 2007.

[14] Daniel Bullock, John C. Fiala, and Stephen Grossberg. A neural model of timed response learning in the cerebellum. *Neural Networks*, 7(6):1101–1114, 1994. Models of Neurodynamics and Behavior.

[15] Geoffrey Burnstock. Autonomic neurotransmission: 60 years since sir Henry Dale. *Annual Review of Pharmacology and Toxicology*, 49:1–30, 2009.

[16] Gemma A Calvert. Crossmodal processing in the human brain: insights from functional neuroimaging studies. *Cerebral cortex*, 11(12):1110–1123, 2001.

[17] Peter Cariani. Temporal coding of periodicity pitch in the auditory system: an overview. *Neural plasticity*, 6(4):147–172, 1999.

[18] Tung-Bo Chen and Von-Wun Soo. A comparative study of recurrent neural network architectures on learning temporal sequences. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 4, pages 1945–1950. IEEE, 1996.

[19] D. Chialvo. Emergent complex neural dynamics. *Nature Physics*, 6:744–750, 2010.

[20] Sayeed Shafayet Chowdhury, Chankyu Lee, and Kaushik Roy. Towards understanding the effect of leak in spiking neural networks. *Neurocomputing*, 464:83–94, 2021.

[21] Gourav Datta, Haoqin Deng, Robert Aviles, and Peter A Beerel. Towards energy-efficient, low-latency and accurate spiking lstms. *arXiv preprint arXiv:2210.12613*, 2022.

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

[23] Gilberto de Paiva. Pattern recognition theory of mind. *arXiv preprint arXiv:0907.4509*, 2009.

[24] Rob R de Ruyter van Steveninck, Geoffrey D Lewen, Steven P Strong, Roland Koberle, and William Bialek. Reproducibility and variability in neural spike trains. *Science*, 275(5307):1805–1808, 1997.

[25] R Christopher Decharms and Anthony Zador. Neural representation and the cortical code. *Annual Review of Neuroscience*, 23:613–647, 2000.

[26] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural networks*, 121:294–307, 2020.

[27] Alain Destexhe and Denis Paré. Impact of network activity on the integrative properties of neocortical pyramidal neurons in vivo. *Journal of Neurophysiology*, 81:1531–1547, 1999.

[28] Alain Destexhe, Michael Rudolph, and Denis Paré. The high-conductance state of neocortical neurons in vivo. *Nature reviews neuroscience*, 4(9):739–751, 2003.

[29] Kshitij Dhoble. *Spatio-/spectro-temporal pattern recognition using evolving probabilistic spiking neural networks*. PhD thesis, Auckland University of Technology, 2013.

[30] Bertrand du Castel. Pattern activation/recognition theory of mind. *Frontiers in computational neuroscience*, 9:90, 2015.

[31] Hadyn D Ellis, Dylan M Jones, and Nick Mosdell. Intra-and inter-modal repetition priming of familiar faces and voices. *British Journal of Psychology*, 88(1):143–156, 1997.

[32] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[33] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature Reviews Neuroscience*, 9:292–303, 2008.

[34] David Ferster and Nelson Spruston. Cracking the neuronal code. *Science*, 270(5237):756–757, 1995.

[35] Ian M. Finn, Nicholas J. Priebe, and David Ferster. The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. *Neuron*, 54:137–152, 2007.

[36] Răzvan V Florian. The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLoS ONE*, 2012.

[37] Nicolas Fourcaud-Trocmé, David Hansel, Carl Van Vreeswijk, and Nicolas Brunel. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of neuroscience*, 23(37):11628–11640, 2003.

[38] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[39] Wulfram Gerstner, Richard Kempter, J Leo van Hemmen, and Hermann Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78, 1996.

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

[41] Asif A Ghazanfar, Joost X Maier, Kari L Hoffman, and Nikos K Logothetis. Multisensory integration of dynamic faces and voices in rhesus monkey auditory cortex. *Journal of Neuroscience*, 25(20):5004–5012, 2005.

[42] Tim Gollisch and Markus Meister. Rapid neural coding in the retina with relative spike latencies. *science*, 319(5866):1108–1111, 2008.

[43] Peiliang Gong, Pengpai Wang, Yueying Zhou, and Daoqiang Zhang. A spiking neural network with adaptive graph convolution and lstm for eeg-based brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2023.

[44] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[45] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428, 2006.

[46] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428, 2006.

[47] P. Heil. Auditory cortical onset responses revisited. i. first-spike timing. *Journal of neurophysiology*, 77 5:2616–41, 1997.

[48] Walter Heiligenberg. *Neural Nets in Electric Fish (Computational Neuroscience)*. MIT press, 1991.

[49] Geoffrey E Hinton, Terrence J Sejnowski, et al. Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[52] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[53] John J Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, 1995.

[54] John J Hopfield and David W Tank. neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[55] John Huxter, Neil Burgess, and John O'keefe. Independent rate and temporal coding in hippocampal pyramidal cells. *Nature*, 425:828–832, 2003.

[56] Jeffry S Isaacson. Odor representations in mammalian cortical circuits. *Current Opinion in Neurobiology*, 20(3):328–331, 2010.

[57] Masao Ito. Long-term depression. *Annual review of neuroscience*, 12(1):85–102, 1989.

[58] E. M. Izhikevich. Simple model of spiking neurons. *Trans. Neur. Netw.*, 14:1569–1572, 2003.

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

[60] Michal Joachimczak and Borys Wróbel. Evolving gene regulatory networks for real time control of foraging behaviours. In *ALIFE*, 2010.

[61] Michal Joachimczak and Borys Wróbel. Processing signals with evolving artificial gene regulatory networks. In *ALIFE*, 2010.

[62] Michał Joachimczak and Borys Wróbel. Open ended evolution of 3d multicellular development controlled by gene regulatory networks. In *Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems*, pages 67–74, Cambridge, MA, 2012. MIT Press.

[63] Roland Johansson and Ingvars Birznieks. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature neuroscience*, 7:170–7, 03 2004.

[64] Roland Johansson and Ingvars Birznieks. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature neuroscience*, 7:170–7, 03 2004.

[65] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[66] Philip Joris and Tom Yin. A matter of time: internal delays in binaural processing. *Trends in Neuroscience*, 30:70–78, 2007.

[67] Philip Joris and Tom CT Yin. A matter of time: internal delays in binaural processing. *Trends in Neurosciences*, 30(2):70–78, 2007.

[68] N Kasabov, K Dhoble, N Nuntalid, and G Indiveri. On-line spatio-and spectrotemporal pattern recognition with evolving spiking neural networks utilising integrated rank oder-and spiketime learning. *Neural Networks*, 2011.

[69] Nikola K Kasabov. *Evolving connectionist systems: the knowledge engineering approach*. Springer Science & Business Media, 2007.

[70] Christoph Kayser, Marcelo A Montemurro, Nikos K Logothetis, and Stefano Panzeri. Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns. *Neuron*, 61(4):597–608, 2009.

[71] Steven W Keele and Richard Ivry. Does the cerebellum provide a common computation for diverse tasks? a timing hypothesis a. *Annals of the New York Academy of Sciences*, 608(1):179–211, 1990.

[72] Richard Kempter, Wulfram Gerstner, J Van Hemmen, and Hermann Wagner. Temporal coding in the sub-millisecond range: Model of barn owl auditory pathway. *Advances in neural information processing systems*, 8, 1995.

[73] Muhammad Aamir Khan, Volker Steuber, Neil Davey, and Borys Wróbel. Spiking neural networks evolved to perform multiplicative operations. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 314–321, Cham, 2018. Springer International Publishing.

[74] Robert Kim and Terrence Sejnowski. Strong inhibitory signaling underlies stable temporal dynamics and working memory in spiking neural networks. *Nature Neuroscience*, 24:1–11, 01 2021.

[75] Bruce W Knight. Dynamics of encoding in a population of neurons. *The Journal of general physiology*, 59(6):734–766, 1972.

[76] Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in neuroscience*, 14:439, 2020.

[77] Louis Lapicque. Tableau général des poids somatique et encéphalique dans les espèces animales. *Bulletins et Mémoires de la Société d'Anthropologie de Paris*, 8(1):248–270, 1907.

[78] Gilles Laurent. Dynamical representation of odors by oscillating and evolving neural assemblies. *Trends in Neurosciences*, 19:489–496, 1996.

[79] Qian Liu, Lifan Long, Qian Yang, Hong Peng, Jun Wang, and Xiaohui Luo. Lstm-snp: A long short-term memory model inspired from spiking neural p systems. *Knowledge-Based Systems*, 235:107656, 2022.

[80] Ali Lotfi Rezaabad and Sriram Vishwanath. Long short-term memory spiking networks and their applications. In *International Conference on Neuromorphic Systems 2020*, pages 1–9, 2020.

[81] Brian Nils Lundstrom and Adrienne L Fairhall. Decoding stimulus variance from a distributional neural code of interspike intervals. *Journal of Neuroscience*, 26(35):9030–9037, 2006.

[82] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[83] Mark D McDonnell and Lawrence M Ward. The benefits of noise in neural systems: bridging theory and experiment. *Nature Reviews Neuroscience*, 12(7):415–425, 2011.

[84] MR Mehta, AK Lee, and MA Wilson. Role of experience and oscillations in transforming a rate code into a temporal code. *Nature*, 417(6890):741–746, 2002.

[85] Markus Meister and Michael J. Berry. The neural code of the retina. *Neuron*, 22(3):435–450, 1999.

[86] Raoul-Martin Memmesheimer, Ran Rubin, Bence P Ölveczky, and Haim Sompolinsky. Learning precisely timed spikes. *Neuron*, 82(4):925–938, 2014.

[87] Ammar Mohemmed, Stefan Schliebs, Satoshi Matsuda, and Nikola Kasbov. Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, 22(04):1250012, 2012.

[88] Richard Naud, Nicolas Marcille, Claudia Clopath, and Wulfram Gerstner. Firing patterns in the adaptive exponential integrate-and-fire model. *Biological Cybernetics*, 99:335–347, 2008.

[89] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[90] Andrzej Nski and Filip Ponulak. F.: Comparison of supervised learning methods for spike time coding in spiking neural networks. *Int. J. Appl. Math. Comput. Sci*, 16:101–113, 01 2006.

[91] John O'Keefe and Michael L Recce. Phase relationship between hippocampal place units and the eeg theta rhythm. *Hippocampus*, 3(3):317–330, 1993.

[92] Thomas S Otis and WF Gilly. Jet-propelled escape in the squid loligo opalescens: concerted control by giant and non-giant motor axon pathways. *Proceedings of the National Academy of Sciences*, 87(8):2911–2915, 1990.

[93] Denis Paré, Eric Shink, Hlne Gaudreau, Alain Destexhe, and Eric J. Lang. Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo. *Journal of Neurophysiology*, 79:1450–1460, 1998.

[94] J. Perez-Orive, Ofer Mazor, Glenn C Turner, S. Cassenaer, Rachel I. Wilson, and G. Laurent. Oscillations and sparsening of odor representations in the mushroom body. *Science*, 297:359 – 365, 2002.

[95] Stephen P Perrett, Blenda P Ruiz, and Michael D Mauk. Cerebellar cortex lesions disrupt learning-dependent timing of conditioned eyelid responses. *Journal of Neuroscience*, 13(4):1708–1718, 1993.

[96] Gualtiero Piccinini and Andrea Scarantino. Information processing, computation, and cognition. *Journal of biological physics*, 37(1):1–38, 2011.

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

[98] Filip Ponulak and Andrzej Kasiski. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, 71:409–33, 01 2011.

[99] Pasko Rakić, J. P. le Bourgeois, Maryellen Fazen Eckenhoff, Nada Zećević, and Patricia S. Goldman-Rakic. Concurrent overproduction of synapses in diverse regions of the primate cerebral cortex. *Science*, 232 4747:232–235, 1986.

[100] Werner Reichardt. Autokorrelations-auswertung als funktionsprinzip des zentralnervensystems. *Zeitschrift für Naturforschung B*, 12(7):448–457, 1957.

[101] Fred Rieke. *Spikes: exploring the neural code.* MIT press, 1999.

[102] AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*, volume 1. University of Cambridge Department of Engineering Cambridge, 1987.

[103] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[104] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient

event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[105] Rufin Van Rullen and Simon J. Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13(6):1255–1283, 2001.

[106] Jeffry S Isaacson. Odor representations in mammalian cortical circuits. *Current Opinion in Neurobiology*, 20:328–31, 2010.

[107] Hannes P. Saal, Sethu Vijayakumar, and Roland S. Johansson. Information about complex fingertip parameters in individual human tactile afferent neurons. *Journal of Neuroscience*, 29(25):8022–8031, 2009.

[108] Stefan Schliebs and Nikola Kasabov. Evolving spiking neural networka survey. *Evolving Systems*, 4(2):87–98, 2013.

[109] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[110] Yoonsik Shim, Andrew Philippides, Kevin Staras, and Phil Husbands. Unsupervised learning in an ensemble of spiking neural networks mediated by itdp. *PLoS computational biology*, 12(10):e1005137, 2016.

[111] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[112] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE, 2019.

[113] Michael Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1996.

[114] William R Softky and Christof Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *Journal of neuroscience*, 13(1):334–350, 1993.

[115] Jeong-Woo Sohn, Byoung-Tak Zhang, and Bong-Kiun Kaang. Temporal pattern recognition using a spiking neural network with delays. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 4, pages 2590–2593. IEEE, 1999.

[116] Barry E Stein and M Alex Meredith. *The merging of the senses.* The MIT press, 1993.

[117] RB Stein and Alan Lloyd Hodgkin. The frequency of nerve action potentials generated by applied currents. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 167(1006):64–86, 1967.

[118] Branko Šter. Selective recurrent neural network. *Neural processing letters*, 38:1–15, 2013.

[119] Volker Steuber and David Willshaw. A biophysical model of synaptic delay learning and temporal pattern recognition in a cerebellar Purkinje cell. *Journal of Computational Neuroscience*, 17:149–164, 2004.

[120] R Steveninck, Geoffrey Lewen, S Strong, Roland Koberle, and William Bialek. Reproducibility and variability in neural spike trains. *Science (New York, N.Y.)*, 275:1805–8, 04 1997.

[121] Charles F Stevens and Yanyan Wang. Facilitation and depression at single central synapses. *Neuron*, 14(4):795–802, 1995.

[122] H. Swadlow and A. G. Gusev. Receptive-field construction in cortical inhibitory interneurons. *Nature Neuroscience*, 5:403–404, 2002.

[123] David W Tank and JJ Hopfield. Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences*, 84(7):1896–1900, 1987.

[124] Timothy J Teyler and P DiScenna. Long-term potentiation. *Annual review of neuroscience*, 10(1):131–161, 1987.

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

[126] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[127] Simon Thorpe, Denise Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520, 1996.

[128] Simon J. Thorpe, Arnaud Delorme, and Rufin van Rullen. Spike-based strategies for rapid processing. *Neural networks : the official journal of the International Neural Network Society*, 14 6-7:715–25, 2001.

[129] KP Unnikrishnan, John J Hopfield, and David W Tank. Connected-digit speaker-dependent speech recognition using a neural network. *IEEE Transactions on Signal Processing*, 39(3), 1991.

[130] Hendrik Van Der Loos and Edmund M. Glaser. Autapses in neocortex cerebri: synapses between a pyramidal cell's axon and its own dendrites. *Brain Research*, 48:355 – 360, 1972.

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

[132] Craig M Vineyard, Sam Green, William M Severa, and Çetin Kaya Koç. Benchmarking event-driven neuromorphic architectures. In *Proceedings of the International Conference on Neuromorphic Systems*, pages 1–5, 2019.

[133] Katharina Von Kriegstein and Anne-Lise Giraud. Implicit multisensory associations influence voice recognition. *PLoS biology*, 4(10):e326, 2006.

[134] Jilles Vreeken. Spiking neural networks, an introduction, 2003.

[135] Jilles Vreeken et al. *Spiking neural networks, an introduction.* Utrecht University: Information and Computing Sciences, 2003.

[136] PD Wall. Pain: A spike-interval coded message in the brain. *Journal of Neurology, Neurosurgery, and Psychiatry*, 45(6):573, 1982.

[137] Xiaoqin Wang, T. Lu, and L. Liang. *Temporal and Rate Representations of Time-Varying Signals in Auditory Cortex.* Nature Neuroscience, 2005.

[138] Udo Wehmeier, Dawei Dong, Christof Koch, and David Van Essen. Modeling the mammalian visual system. In *Methods in neuronal modeling: From synapses to networks*, pages 335–359. MIT Press, 1989.

[139] Hua wei Fan, Yafeng Wang, Hengtong Wang, Ying-Cheng Lai, and X. Wang. Autapses promote synchronization in neuronal networks. *Scientific Reports*, 8, 2017.

[140] Laura Wiles, Shi Gu, Fabio Pasqualetti, Danielle Bassett, and David Meaney. Autaptic connections shift network excitability and bursting. *Scientific Reports*, 7, 08 2016.

[141] Ronald J Williams and David Zipser. Gradient-based learning algorithms for recurrent. *Backpropagation: Theory, architectures, and applications*, 433:17, 1995.

[142] Borys Wróbel. Evolution of spiking neural networks robust to noise and damage for control of simple animats. In *Parallel Problem Solving from Nature – PPSN XIV*, pages 686–696, 2016.

[143] Borys Wróbel, Ahmed Abdelmotaleb, and Michał Joachimczak. Evolving networks processing signals with a mixed paradigm, inspired by gene regulatory networks and spiking neurons. In *International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, pages 135–149. Springer, 2012.

[144] Borys Wróbel, Ahmed Abdelmotaleb, Michał Joachimczak, et al. Evolving spiking neural networks in the greans (gene regulatory evolving artificial networks) platform. In *EvoNet2012: Evolving Networks, from Systems/Synthetic Biology to Computational Neuroscience Workshop at Artificial Life XIII*, pages 19–22, 2012.

[145] Borys Wróbel and Michał Joachimczak. Using the genetic regulatory evolving artificial networks (greans) platform for signal processing, animat control, and artificial multicellular development. In *Growing Adaptive Machines*, pages 187–200. Springer, 2014.

[146] Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, 23(7):819–835, 2010.

[147] Muhammad Yaqoob, Volker Steuber, and Borys Wróbel. The importance of self-excitation in spiking neural networks evolved to recognize temporal patterns. In *Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation*, pages 758–771, 2019.

[148] Muhammad Yaqoob and Borys Wróbel. Very small spiking neural networks evolved to recognize a pattern in a continuous input stream. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI) – IEEE*, pages 3496–3503, 2017.

[149] Muhammad Yaqoob and Borys Wróbel. Robust very small spiking neural networks evolved with noise to recognize temporal patterns. In *ALIFE 2018: Proceedings of the 2018 Conference on Artificial Life – MIT Press*, pages 665–672, 2018.

[150] Muhammad Yaqoob and Borys Wróbel. Very small spiking neural networks evolved for temporal pattern recognition and robust to perturbed neuronal parameters. In *Artificial Neural Networks and Machine Learning – ICANN*, pages 322–331, 2018.

[151] Ergin Yilmaz, M. Ozer, Veli Baysal, and M. Perc. Autapse-induced multiple coherence resonance in single neurons and neuronal networks. *Scientific Reports*, 6, 2016.

[152] John Zachary Young. Fused neurons and synaptic contacts in the giant nerve fibres of cephalopods. *Philosophical Transactions of the Royal Society of London. Series B, Biological sciences*, 229(564):465–503, 1939.

[153] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.