

Imperfect Synapses in Artificial Spiking Neural Networks

A thesis submitted in partial fulfilment of
the requirements for the Degree of
Master of Computer Science

by

Hayden Jackson



University of Canterbury

2016

To the future.

Abstract

The human brain is a complex organ containing about 100 billion neurons, connecting to each other by as many as 1000 trillion synaptic connections. In neuroscience and computer science, spiking neural network models are a conventional model that allow us to simulate particular regions of the brain. In this thesis we look at these spiking neural networks, and how we can benefit future works and develop new models.

We have two main focuses, our first focus is on the development of a modular framework for devising new models and unifying terminology, when describing artificial spiking neural networks. We investigate models and algorithms proposed in the literature and offer insight on designing spiking neural networks. We propose the Spiking State Machine as a standard experimental framework, which is a subset of Liquid State Machines. The design of the Spiking State Machine is to focus on the modularity of the liquid component in respect to spiking neural networks. We also develop an ontology for describing the liquid component to distinctly differentiate the terminology describing our simulated spiking neural networks from its anatomical counterpart. We then evaluate the framework looking for consequences of the design and restrictions of the modular liquid component. For this, we use nonlinear problems which have no biological relevance; that is an issue that is irrelevant for the brain to solve, as it is not likely to encounter it in real world circumstances. Our results show that a Spiking State Machine could be used to solve our simple nonlinear problems which only needed small networks. For more complex problems the liquid required a substantial growth in order to find a solution. Growing the network may be unfeasible for more challenging problems compared to manually constructed networks, which may provide better clarification of the situation and the mechanics involved.

Our second focus is the modelling of spontaneous neurotransmission in the brain, for which we propose a novel model called the *imperfect synaptic model*. Imperfect synapses allow for communication without spiking, by potential differences in the membrane potentials of neurons. We evaluated this method against the standard synapse model on the classical control system problem of balancing a pole on a cart. We found that networks with imperfect synapses responded quicker to external actions. Otherwise, there was no significant difference to networks of standard synapses. We discovered that by *doping* a network with imperfect synapses, the network learnt to balance the cart-pole system in half the time compared to networks with standard synapses and networks with only imperfect synapses. We suspect that doping imperfect synapses increase the separation potential of the network, by allowing some neurons to inhibit or excite other neurons without action potentials. This decreases the chance of repeated structures occurring within the network. Our proposed method does not cover all mechanisms of neurotransmission, and in general, neurons will require at least some prior stimulation. Computationally, the imperfect synaptic model requires more computation per propagation cycle.

Table of Contents

Abstract	iii
List of Figures	iv
Acknowledgments	vi
Chapter 1: Introduction	1
1.1 The Problem(s)	3
1.2 Research Objectives	4
1.2.1 Standard Modular Framework	4
1.2.2 Imperfect Synaptic Model	5
1.3 Contribution	5
1.4 Thesis Outline	6
Chapter 2: Background	7
2.1 Biological Neuron Systems	7
2.1.1 Action Potentials	9
2.1.2 Dale's Principle	11
2.1.3 Excitatory and Inhibitory Neurons	13
2.1.4 Hebbian Theory	13
2.2 Artificial Neural Networks	15
2.2.1 The Perceptron	15
2.2.2 Feed-forward Neural Network	17
2.2.3 Training and Learning	18
2.2.4 Recurrent Neural Networks	24

2.3	Spiking Neural Networks	25
2.3.1	Spiking Neurons Models	25
2.3.2	Spike Timing Dependent Plasticity	31
2.3.3	Back-propagation	34
Chapter 3: Framework and Design Philosophy		38
3.1	Experimental Observations	38
3.2	Modular Framework	40
3.2.1	Structure	40
3.3	Requirements and Properties	44
3.4	Construction and Training Procedure	46
3.5	Liquid Ontology	46
3.6	Liquid Implementation	48
Chapter 4: Nonlinear Problems and Framework Evaluation		50
4.1	Liquid State Machine Setup	53
4.2	Experiment: AND and XOR	55
4.3	Observations and Thoughts	57
4.4	Experiment: Non-linear Hypercube	58
4.5	Discussion and Summary	62
Chapter 5: Imperfect Spiking Neural Networks		63
5.1	Imperfect Synapses	64
5.2	Mathematical Model	65
5.3	Explanation	67
5.4	Doping Networks	68
5.5	Cart Pole	69
5.6	Experiment: Q-Learning	72
5.7	Experiment: Perceptrons	73

5.8	Experiment: Imperfect Synapse Comparison	75
5.9	Experiment: Doping Networks	78
5.10	Discussion and Summary	80
Chapter 6:	Summary	81
6.1	Thesis Summary	81
6.1.1	Thesis Contributions	83
6.1.2	Future Work	84
References		85

List of Figures

2.1	The structure of a multipolar neuron, showing the dendrites, soma, axon, and synapse terminals.	8
2.2	The structure of the lipid bilayer, the lipid tails hold the bilayer together by hydrophobic attraction. Public Domain image, by Mariana Ruiz Villarrea.	9
2.3	An approximation of a typical membrane response, were some stimulation is applied causing an action potential. Partial recreation from [20].	10
2.4	An action potential propagating along an axon via a feedback cyclic, towards a synaptic terminal. Modification of a Public Domain image, by Mariana Ruiz Villarrea.	12
2.5	To the left we see the perceptron splitting a linear problem. To the right we can see a nonlinear problem that cannot be split by a hyperplane from the perceptron, the orange line showing an attempted separation while the green show the nonlinear solution.	16
2.6	The standard layout of a feed-forward neural network.	17
2.7	The STDP function, discovered in [9]. Recreated from [97].	32
3.1	The Liquid State Machine and its main components.	41
3.2	Our proposed ontology, showing the generalised relationships.	47
4.1	A designed liquid that can solve the XOR problem.	53

4.2	The layout of the Liquid State Machine used in our experiments.	54
4.3	The diagram here describes a depicts liquid that can solve the 3 dimensional nonlinear hypercube.	59
5.1	Showing the propagation of information via Imperfect Synapses.	68
5.2	The cart-pole problem.	69
5.3	A plot of the fitness function of a Q-Learner solving the cart-pole problem via reinforcement learning.	72
5.4	A plot of the fitness function of a perceptron for the cart-pole problem in a student-teacher system.	73
5.5	The liquid of the Spiking State Machine here is a random graph of probabilistically connected neurons.	75
5.6	A comparison of the fitness function for agents with standard synapses and imperfect synapses, while trying to solve the cart-pole system. The last thirty trials are without a teacher.	76
5.7	A comparison of the fitness function for agents with standard synapses and imperfect synapses, while trying to solve the cart-pole system under external action. The last thirty trials are without a teacher.	77
5.8	The fitness function for an agent doped with imperfect synapses, for both the cart-pole problem with and without external forces.	79

Acknowledgments

I would like to thank Dr Kouros Neshatian (Supervisor) for all his support and guidance on completing this thesis. Thank you to my Friends and Family for their love and support. Finally, thank you to Susha Smith for her enduring love and support during the final months spent completing my thesis.

Chapter I

Introduction

The human brain is one of the most important and complex organs but the least understood tissue in our body, due to limitations that include being unable to study the brain in a living organism (in-vivo) [105]. The cerebral cortex is estimated to contain 15 billion to 35 billion neurons alone [79], where communication between neurons has been observed to consist via chemical or electrical signals transmitted by synapsing onto each other. We have used our understanding of the structure of the brain to develop artificial neural networks for machine learning tasks, which help further investigate the functionality of the brain and develop novel techniques and mechanisms based on imperfectness of synaptic connections.

The scientific community has seen recent advances in the creation of artificial brain tissue. We see in [61] that brain cells that have been grown artificially outside a living organism (in-vitro) form in discrete and interdependent brain regions which recapitulate features of the human cortical development. Furthermore, induced injury on bio-engineered brain tissue has shown to react with biochemical and electrophysiological outcomes that mimic observations in-vivo [105]. These advances allow us to better understand the development, disease, and injury of the brain. One can imagine that continued development of in-vitro brain tissue will result in artificial tissues that will be perfect

candidates for experimentation into brain-computer interfaces.

Artificial neural networks (simulations) in computer science have seen three major evolutions over the course of their academical lifetime. In 1943 a computational model called *the threshold logical unit* for artificial neural networks was discovered based on mathematics. 1975 saw the discovery of the back-propagation algorithm, which allows for quickly training multi-layer artificial neural networks based on the propagation of error throughout a network to update connecting synaptic weights. Back-propagation allowed the artificial networks to solve nonlinear classification problems such as the *exclusive or* problem. In 2009 and later recurrent neural networks and deep feed-forward neural networks were developed leading to powerful pattern recognition algorithms with wide domains of application. All these advances have a strong mathematical background but bare no resemblance to the neurons found in the human brain.

Spiking neural networks are considered as part of the third generation of artificial neural network models. These spiking neural networks are an artificial neural network model focused on realism and replication of neurobiological models found in the brain. This specific type of network sees the overlap of the neuroscience and the computer science fields, and there have been both software and hardware implementations. Due to their nature, spiking neural networks are directly analogous to bio-engineered brain matter, and therefore are useful for analysing how we could supposedly use these systems in preparation for future research on brain-computer interfaces.

One can imagine that one-day hardware may contain a chip with a bio-engineered brain to perform some computation and decision making. Initially, these chips would likely be used in time critical systems for pattern recognition problems, unless we find a better tool than our brain at pattern

recognition.

In this thesis, we propose a standard modular framework for analysing future bio-engineered neural networks, which would allow us to solve problems and devise new models of possible real-world features that might affect these future biological networks. We then apply this framework and propose a synaptic model that allows for imperfectness of connections between neurons, and through control problems, study its features and effects on solutions to temporal problems.

1.1 *The Problem(s)*

Artificial neural networks are a classification and regression algorithm in machine learning. The Spiking neural network variant focuses on realism and replicated neurobiological models found in brain tissue; these networks also offer native temporal functionality.

Normally, spiking neural network research focuses on optimisation based on mathematics, and only concentrates on information transferal via spike timing. These networks tend to use probabilistic models of connection and not biologically relevant layered approaches borrowed from feed-forward neural networks. This technique leads to models that do not include significant real world concerns, which is problematic if applied to real world situations and would be particularly relevant if we intend to use artificial brain tissue as part of physical hardware systems in future.

There is a need for:

- i) A standardised modular framework for devising and comparing models.
- ii) An understanding of what imperfect connections between neurons im-

plies and its usefulness.

Such that we can compare results with future physical networks and improve their use for problem-solving.

1.2 Research Objectives

The primary goal of this research is to investigate models inspired by biological brain tissue for spiking neural networks and provide observational evidence of the usefulness in problem solving. To do this, we need some framework so that we can systematically evaluate these models while allowing for ease of future reproduction. We also desire that our proposed solutions be simple but modular. We determine simplicity by the ability to apply problems to an agent without designing or iterating on the structure of the spiking neural network, while modularity in the framework should be such that application to future physical models is feasible and facilitates the discovery of new models. We will be focusing on a *Standard Modular Framework* and a biologically inspired synapse model called *Imperfect Synapses*, these are defined as follows.

1.2.1 Standard Modular Framework

We define this standard modular framework as a structure that exists in the experimental design that unifies the description of the interplay between the models contained in the spiking neural networks and the problem space.

1.2.2 Imperfect Synaptic Model

Imperfect synapses are synaptic connections that allow for communication via potential differences between neurons as well as current induced by action potentials.

1.3 Contribution

This thesis has three primary contributions:

- A Spiking State Machine as a standardised modular framework.
- The Imperfect Synaptic Model and observations from solutions to control problems.
- A technique called *doping* for spiking neural networks.

We analyse the *Spiking State Machine* as a standard modular framework for spiking neural network development, by applying it to nonlinear and nontemporal problems. We did this to determine the classes of problems to which it can be applied and to determine the consequences of its design. This allows us to address the sub-goal of *simplicity*, where a network does not need to be explicitly designed. The *imperfect synaptic model* offers new network dynamics. *Doping* is the technique of adding impurities to a network by replacing some synapses with imperfect synapses. We present empirical observations that doping a network with the imperfect synaptic model improves upon other network configurations by offering faster learning of the cart-pole problem.

This thesis provides a secondary contribution of a *Spiking Neural Network Ontology* which grew from our design process and the requirements of our framework. The ontology expands upon ways of describing communication

and relationships between neurons, and makes a clear separation between terminology describing biological and artificial networks while outlining a practical implementation strategy. This ontology also allows for the description of the communication mechanisms of imperfect synapses in spiking neural networks.

1.4 Thesis Outline

Chapter 2, lays out fundamental concepts and models used throughout this thesis. This chapter covers biological neurons and their descriptions, artificial neural networks and machine learning concepts, and spiking neural networks.

Chapter 3, describes the design, implementation and lessons learned in our preliminary investigation of the field. We propose a standard modular framework called the Spiking State Machine. We also offer an ontology of spiking neural networks making a clear separation between simulation and its biological counterpart.

Chapter 4, covers the experiments of nonlinear and nontemporal problems with the Spiking State Machines, addressing the simplicity sub-goal and observations for their use in the context of the standard modular framework.

Chapter 5, covers the Imperfect Synaptic Model and its inspirations, and application to the cart-pole problem. We also address the advantages and weaknesses we discovered.

Chapter 6, we offer a summary of the works within this thesis and propose some future directions.

Chapter II

Background

In this chapter, we cover fundamental concepts and literature associated with our research. The main topics are biological neurons and their descriptions, artificial neural networks and machine learning concepts, and spiking neural networks.

2.1 *Biological Neuron Systems*

A typical biological neuron has three distinct parts:

Dendrites These act as an input source into the neuron, collecting signals from other neurons.

Axon These are the output sink, which facilitates the transmission of the neurons response to other neurons.

Soma The central unit of the neuron which uses the collected signals from the dendrites and determines if some threshold is met to deliver a signal to the axon.

Synapses connect neurons allowing for the transfer of electrical signals by chemical release or direct ionic currents. An axon can synapse onto a dendrite of another neuron, another axon or axon terminal, the soma of another

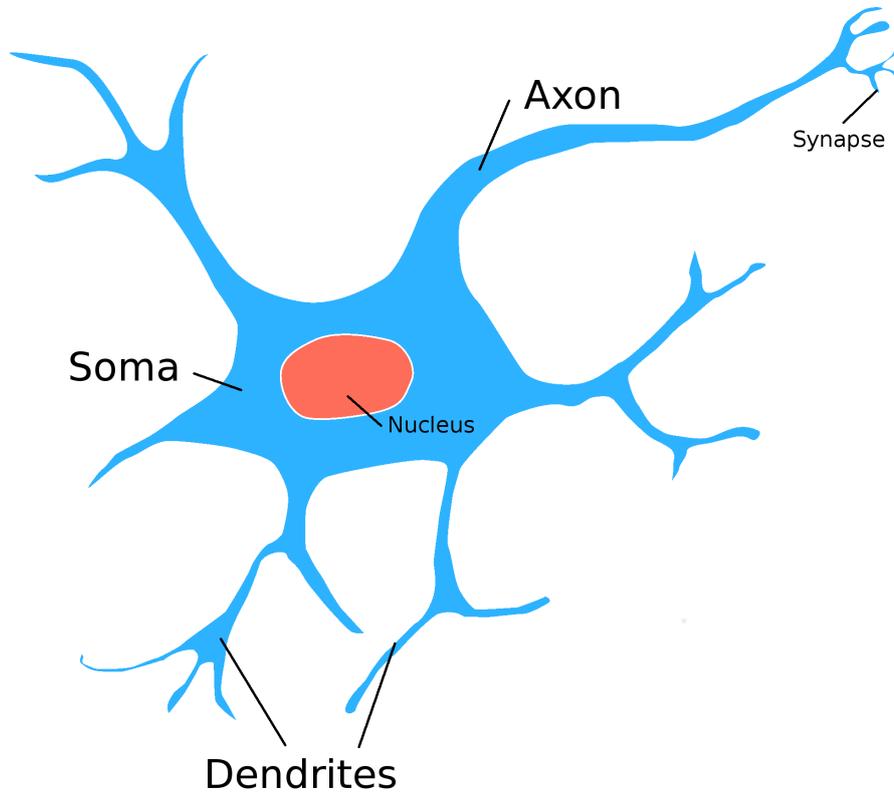


Figure 2.1: The structure of a multipolar neuron, showing the dendrites, soma, axon, and synapse terminals.

neuron, into the bloodstream, cells or cellular fluid.

Let us consider the communication of two neurons. We refer to the neuron which is transmitting a signal as the pre-synaptic neuron, and the receiver as the post-synaptic neuron. The communication of these neurons consists of short electrical pulses or commonly understood as spikes or action potentials. A neuron can ignore this communication if it is within a refractory period caused by recent stimulation. Spikes are considered not to carry any information but instead the timing, quantity and sending neuron of the spike that is important.

Bilayer sheet

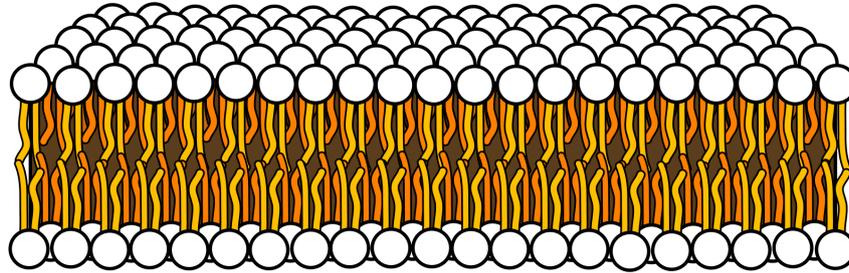


Figure 2.2: The structure of the lipid bilayer, the lipid tails hold the bilayer together by hydrophobic attraction. Public Domain image, by Mariana Ruiz Villarrea.

Lipid bilayer

The electrical capacity of a cell, including neurons, comes from the lipid bilayer which is a thin membrane that surrounds a cell, or subcellular structures. The lipid bilayer, seen in Figure 2.2, looks similar to the plates of the capacitor component of electronic systems. The bilayer is impermeable to ions allowing for ion concentrations inside and outside the cell to create an electric potential across the bilayer. Pore-forming proteins in the bilayer form ion channels which regulate ion concentrations to dissipate the ion gradient across the bilayer, the pores that have developed in the membrane only open under certain conditions. While ionic current flows through the membrane by ion pumps, transporting ions across the membrane walls against the ion gradient.

2.1.1 Action Potentials

The voltage potential that the lipid bilayer builds from the ionic gradient is known as the membrane potential of the cell. The ion channels in the

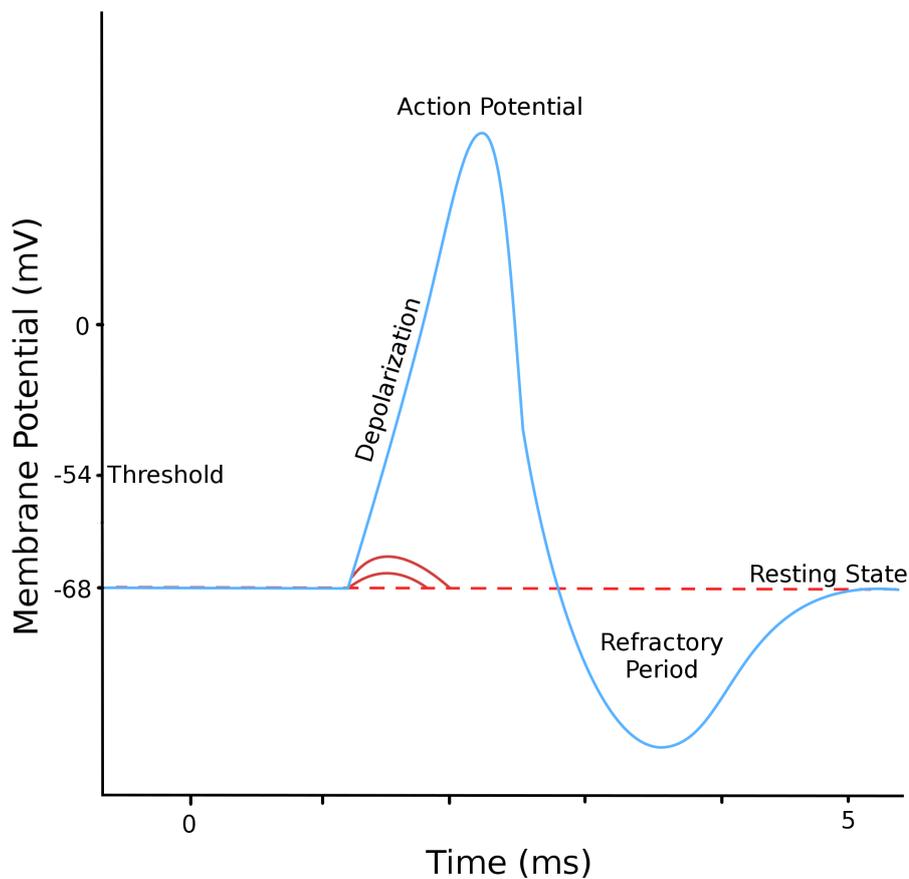


Figure 2.3: An approximation of a typical membrane response, were some stimulation is applied causing an action potential. Partial recreation from [20].

membrane vary across the cell body resulting in different electrical properties. Some parts of the membrane may be excitable such that they have two important levels of potential, the resting potential and the threshold potential, due to synaptic connections this region may depolarise or hyperpolarize. When the membrane potential passed the threshold potential from depolarization, an action potential is triggered. The action potential causes the membrane potential to rise rapidly before falling to resting potential again.

In Figure 2.3 we see an approximation of the typical membrane response, showing some of the phases that a cell goes through when an action potential, or spike, passes a section of the cell membrane. The peak is reached relatively quickly, approximately two milliseconds, after the membrane potential passed the membrane threshold. The membrane then returns to the resting potential after approximately five milliseconds has elapsed.

The voltage-gated ion channels in the membrane result in positive feedback loops as the membrane potential controls the state of the ion channels, while the ion channels control the membrane potential. The opening of the ion channels and their biophysical properties determine the trajectory which the action potential propagates. For sodium and potassium gated cells, the sodium ion is pumped outside the cell faster than the potassium ion, resulting in a state where there are more sodium ions outside the cell than potassium ions inside causing a negative charge across the membrane wall, as in Figure 2.4. When the membrane voltage is at resting potential, the sodium channels are closed allowing for the depolarization of the membrane. During an action potential, the sodium ions flow into the membrane while potassium ions dissipate out of the membrane, reversing the charge of the membrane wall and propagating the action potential along the membrane wall determined by the biophysical properties of the channels.

2.1.2 Dale's Principle

Dale's principle essentially states that a neuron performs the same chemical action on all of its synaptic connections. This process allows us to simplify the interactions between neurons making them easier to observe and model.

Sir Henry Dale alluded to this principle as follows:

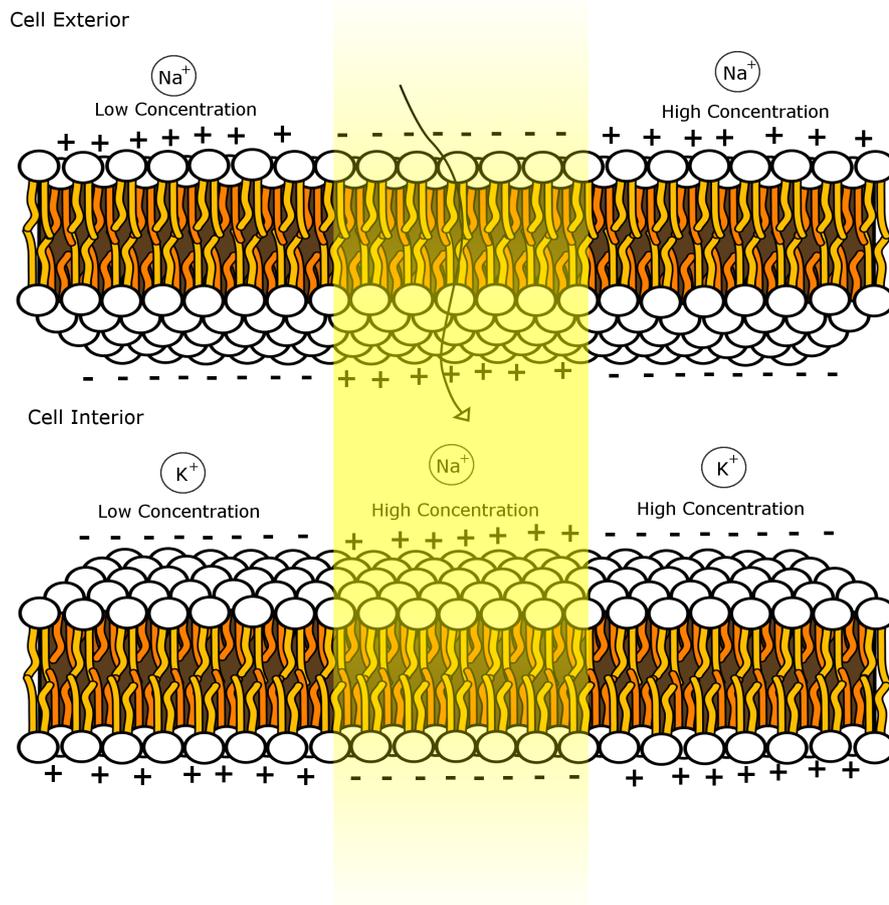


Figure 2.4: An action potential propagating along an axon via a feedback cyclic, towards a synaptic terminal. Modification of a Public Domain image, by Mariana Ruiz Villarrea.

“When we are dealing with two different endings of the same sensory neurone, the one peripheral and concerned with vasodilatation and the other at a central synapse, can we suppose that the discovery and identification of a chemical transmitter of axon-reflex vasodilatation would furnish a hint as to the nature of the transmission process at a central synapse?” [25]

While Sir Henry Dale did not propose Dale’s principle, later works by John Eccles defined and found relevance defining the principle as follows:

“I proposed that ‘*Dale’s Principle*’ be defined as stating that all the axonal branches of a neurone there was liberation of the same transmitter substance or substances.” [33]

2.1.3 *Excitatory and Inhibitory Neurons*

Biological evidence found in [47, 107] suggest the importance of excitatory and inhibitory neurons. The action potentials of excitatory neurons cause the post-synaptic neurons increase their membrane potential, while inhibitory neurons cause diffusion of the ions in the post-synaptic neuron reducing the membrane potential.

In the simulations performed in this paper excitatory neurons output a signal that encourages connected neurons to spike while inhibitory neurons output a signal that discourages spiking. This model allows one spiking neuron to counteract the input of another. All following neuron implementations allow for excitatory or inhibitory neurons by the modification of the input function.

2.1.4 *Hebbian Theory*

Hebbian Theory is concerned with the explanation of neurons during the learning process. Hebbian theory describes a mechanism for the adaptation of synaptic plasticity caused by the interaction of neurons within a network, where increased interaction suggests an increase synaptic strength between cells.

The theory states:

“Let us assume that the persistence or repetition of a reverberatory

activity tend to induce lasting cellular changes that add to its stability. The assumption can be precisely stated as follows: *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*" [101]

Long Term Potentiation

Long-term potentiation is the persistent increase in synaptic strength caused by frequent stimulation of synaptic terminals. Long-term potentiation was theorised in Hebbian's works and is now considered the major mechanisms underlying learning and memory in the brain.

Empirical evidence of long-term potentiation as a form of Hebbian learning has been recognised within the brain [3, 66, 10]. The experiment showed that via stimulation of a rabbit hippocampus, short bursting resulted in downstream neurons having an increased response from post-synaptic neurons implying some strengthening of synaptic strength. These increases in responses persisted for an extended period before returning to normal.

Long Term Depression

Unlike long-term potentiation, long-term depression was not theorised by Hebbian and works opposite to long-term potentiation. Long-term depression causes persistent synaptic weakening resulting from similar means as long-term potentiation.

Long-term depression was discovered by providing high-frequency stimulation to some neurons which again inducing long-term potentiation, though

other synapses saw a decrease in synaptic strength [30]. Other methods found that long-term depression could be induced by low-frequency stimulation of a pre-synaptic neuron; that is a neuron that has fired before stimulation [29].

2.2 Artificial Neural Networks

Artificial neural networks are a family of machine learning models inspired by biological neural networks and neuroscience. They are considered the first generation of neural networks, and the second generation with error-propagation algorithms for multilayer networks. Artificial neural networks can be regarded as a graph of simulated neurons that can be trained to solve linear and nonlinear classification problems and approximate functions. We separated these networks into two main classes which we define by the type of neuron model they use. Firstly artificial neurons, such as the perceptron, as covered in this section and the biologically modelled neurons which we separate off into section 2.3.1, due to their importance to our work. We briefly consider the neuron models, structures and learning of the artificial neurons in this section.

2.2.1 The Perceptron

A perceptron is a type of artificial neuron model that was designed to mimic biological neurons, though its function bears no resemblance. The perceptron is better known as a linear classifier and is commonly used as components of artificial neural networks as talked about in this section. A perceptron functions by partitioning the input space with a hyperplane and input vectors are classified based upon what side of the hyperplane they reside in, meaning

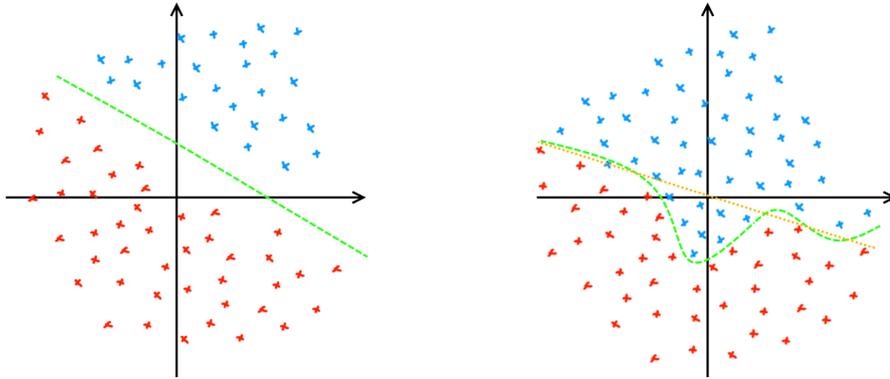


Figure 2.5: To the left we see the perceptron splitting a linear problem. To the right we can see a nonlinear problem that cannot be split by a hyperplane from the perceptron, the orange line showing an attempted separation while the green show the nonlinear solution.

that a single perceptron can solve all linear problems. A perceptron cannot solve nonlinear problems as we can see in Figure 2.5. We frequently use perceptrons in our works as they are known only to be linear classifiers, guaranteeing that nonlinear potential comes from other sources.

A perceptron can be defined as the piecewise function:

$$f(\mathbf{x}) = \begin{cases} 1, & \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

Where \mathbf{w} is a vector of weights on the inputs, \mathbf{x} is a vector of the inputs, and b is the bias of the perceptron. Perceptrons are able to solve linearly separable problems but incapable of solving nonlinearly separable problems such as the XOR function. Though perceptrons are not functionally similar to biological neurons the inspiration is evident such that they follow theories such as Dale's principle, since the output of the perceptron is passed to all predecessors without modification, regardless environmental situation.

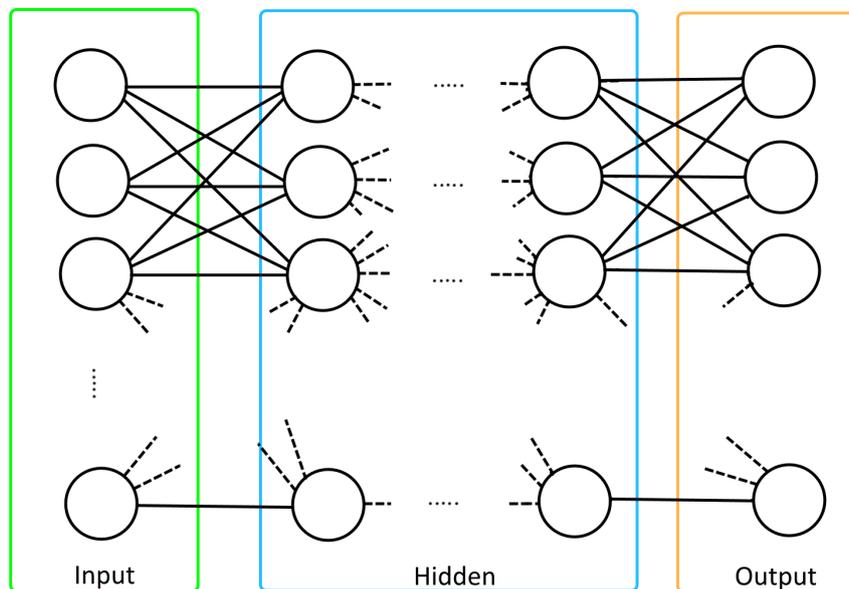


Figure 2.6: The standard layout of a feed-forward neural network.

2.2.2 Feed-forward Neural Network

A feed-forward neural network (multi-layer perceptron) is an improvement on single-layer perceptron that can solve nonlinearly separable problems by introducing a hidden layer of perceptrons. These neural networks are considered to have three layers. The first layer is the input layer which acts as the input stimuli of the network. The second layer is the hidden layer which processes the information received from the input layer. The third and final layer is the output layer which makes a decision based on the output of the hidden layer. The network structure can be seen in Figure 2.6. The feed-forward network is the general structure many artificial neural network models use for their topology. The topological structure ensures that there are no cycles within the network making it easy to propagate an error function into the network making learning efficient.

2.2.3 Training and Learning

We can consider Hebbian Theory when training these artificial neural networks. Learning tends to take the form of modifying the vector weights of inputs. Many methods cover the learning process for these artificial networks, but we will focus on the most popular method, back-propagation.

The back-propagation model was first introduced as a learning method for neural networks in 1988 [16, 89] and is considered as the start of the second generation of neural networks. Using the gradient descent optimisation method with back-propagation can be used to find local and global minima of the error function by propagating error back into the network to update the weights on the synapses by how much each neuron contributed to the error. While this method works well for feed-forward neural networks, it does have the limitation that convergence is slow and not guaranteed. The back-propagation with gradient descent is not guaranteed to find the global minimum without modifications to the learning method which would result in slower learning.

In general training methods can be classified as online or offline. The online learning approach sees the network accessing singular input vectors in a sequential fashion, where the system may not have access to a data set all at once. The online method is useful for adjusting agent behaviour during application. The offline learning model sees the system has an entire dataset available during the training process.

In general, for our neural networks, there are three main learning problems: supervised learning, reinforcement learning and unsupervised learning. These are more broadly problems from machine learning though the solutions are generally domain specific.

Supervised Learning

Supervised learning is the problem of organising the structure of a neural network into a state such that it approximates an unknown function. This unknown function is represented by a set of inputs vectors and an expected output result, this being the main distinction between supervised learning and other learning problems. The learnt structure of a neural network tends to be the modification of synaptic weights between neurons, while the topological structure remains unchanged.

The training method for supervised learning algorithms follows that for each input vector and output, find the error from the agent output to the expected and minimise future error. There are three main problems the calculation of an error function, how the network should be modified based upon the error function, and how accurate the modified network remains on unseen input vectors. The first of these problems requires that the output of the network be designed to match the domain of the ideal output, the modification of the activation function within the output layer such that a scalar value within the same domain is produced solves this problem for a perceptron. While the last of these problems is solved by splitting the data into a training set and a test set, where the test set is evaluated afterwards to determine the successfulness of the learnt agent. The second problem is non-trivial and has been addressed many times.

The method of back-propagation [16, 89] is a supervised learning algorithm that addresses the second problem of how the network structure should be modified based on the error function. It is a common method of learning for multi-layered perceptrons for data with known expected outputs. The algorithm is used in conjunction with the gradient descent of the error function to minimise future error and propagate changes back throughout the

network.

Considering the use of a back-propagation algorithm for a feed-forward network of perceptrons. The weight connecting perceptron i to perceptron j , or w_{ij} , is modified by the following:

$$\Delta w_{ij} = -\alpha \frac{\delta E}{\delta w_{ij}}, \quad (2.2)$$

Where α is the learning rate which is used to affect the speed and accuracy of learning, and E is the error from the output o the target t . For an output perceptron j or single perceptron this becomes:

$$\Delta w_{ij} = -\alpha o_i (o_j - t_j) o_j (1 - o_j), \quad (2.3)$$

Where o_i is the output of perceptron i and t_j is the output of perceptron j . Otherwise for an internal perceptron j where the error needs to be propagated up from the output layer:

$$\Delta w_{ij} = -\alpha o_i \left(\sum_{l \in L} \delta_l w_{jl} \right) o_j (1 - o_j) \quad (2.4)$$

The training method for the back-propagation algorithm follows, for each training input vector \vec{x} :

1. Feed the training input $u(\vec{x})$.
2. Compute error at output $y(\vec{x})$.
3. For each layer in the network starting from the immediate predecessor to the output layer:
 - (a) Compute Δw for all weights from layer $i - 1$ to layer i .

4. Update network weights.

There are also many other modifications and extensions of the back-propagation algorithm as well as other supervised learning algorithms developed. We discussed the back-propagation algorithm in this section because of its commonality in this field of research and that later we use it to address learning our experiments on classification.

Unsupervised Learning

The unsupervised learning problem is the problem of finding patterns, clusters or other features within data. Unlike supervised learning, unsupervised learning has no information about the classification of data. We do not make use of unsupervised learning techniques in this thesis, but this section remains for completeness on learning in artificial neural networks.

Since there is no error function in unsupervised learning, the approach favours inferring information that by underlying probability distributions, meaning that the given distribution can implicitly determine an error function by employing a cost function. The usage of unsupervised learning methods follows that given an input vector and cost function, we try to minimise the error of the cost of the output.

Reinforcement Learning

The reinforcement learning problem is similar to supervised learning, but instead of knowing the desired result and computing an error, the agent is positively rewarded for good output and negatively rewarded for bad output. The reinforcement learning problem is based upon the *law of effect* [109], which states that animals are more likely to perform actions that pro-

duce better rewards/effects than those that have negative rewards/effects, depending on the environmental situation. Many reinforcement learning algorithms focus on the act of learning a policy, a mapping of states to actions, such that performing such that those states maximise the total reward.

Reinforcement learning is best used for environments where the agent has direct control over the environment, as a result of this, the output of the agent determines the next input vector to the agent. This temporal element leads to self-referential cause and effect where the input vector at time t causes some action a_t from the agent that determines the input vector at time $t + 1$, and therefore in turn future actions of the agent. The environmental notion of time domain may either be continuous where the agent may be making decisions at some fixed time step or discrete where the agent takes action when an event occurs within the system. The agent receives some reward r_{t+1} after each action at time t which intern may change future actions of the agent.

The agent in reinforcement learning is trying to maximise their reward by choosing the actions which incur the maximum future reward. As a result of this the agent is attempting to maximise the quantity:

$$R = \sum_{t=0}^{N-1} r_{t+1}, \quad (2.5)$$

Here N determines the time at which the system incurs a terminal state or infinity if there is no terminal state. Since it is impossible to do the summation over infinite time, the return is discounted by some factor γ^t becoming:

$$R = \sum_{t=0}^{N-1} \gamma^t r_{t+1}, \quad (2.6)$$

Where the discount factor exists in the domain $0 \leq \gamma \leq 1$. This method is, however, problematic in online learning situations, as this makes initial actions more important where an agent is more likely to make mistakes. The agent can still find a solution, and that restarting time can help to negate the adverse effects of the initial mistakes. Therefore reinforcement learning can meet the criterion of optimality without being able to deduce an error upon input.

Reinforcement learning methods developed tend to fall into broadly; Markov Decision Processes, value function such as Monte Carlo methods, and temporal difference such as Q-learning [116]. The Q-Learning model consists of an agent in an environment with a finite number of states and a finite set of actions that can be performed per state. The algorithm contains a function or map of state-action combinations $Q(s, a)$, whereby Q initially returns an arbitrary value. For every action that the agent selects, Q is modified via value iteration:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_a [Q(s_{t+1}, a)] - Q(s_t, a_t) \right), \quad (2.7)$$

Where α is the learning rate, and γ is the discount factor. In [71] Q-learning is extended with linear function approximation that allows for guaranteed convergences when action values are estimated with a function approximator.

Implementations of reinforcement learning methods for artificial neural networks have met mixed success. A class of algorithms called residual algorithm that is a special case of Q-learning with back-propagation using linear

function approximators with guaranteed convergence, though this does not translate to nonlinear approximators such as feed-forward neural networks of perceptrons.

Recently more novel algorithms have been developed which address the problem of reinforcement learning with direct topological modification of an artificial neural network using evolutionary methods [104].

2.2.4 Recurrent Neural Networks

The recurrent neural networks is a specific class of neural networks, and they differ from the linear feed-forward neural network by having a reservoir of neurons that have no limitations on how they can be connected. Thereby these networks are defined by the presence of loops and cycles. The cyclic nature of these networks creates an internal state allowing the network to contain some form of temporal behaviour. This internal memory allows for previous input vectors to remain in the system for an arbitrary amount of time defined by the network structure. These networks are relevant to our work as we specifically use them for temporally defined problems.

Learning in recurrent networks needs to be adjusted for the cyclic connections. Some methods tend to favour gradient descent methods defined on the time axis for adjusting synaptic weights, while others favour genetic algorithms, such as [104], which evolve network structure including both synaptic weights and topological structure. A recurrent neural network may exist as the reservoir of the Echo State Machine where learning happens to an output layer external to the recurrent network. There exists a variant where the reservoir of Echo State Machine consists of spiking neurons, called the Liquid State Machine which was developed independently. We cover the Liquid State Machine in chapter 3 with reference to our work.

2.3 Spiking Neural Networks

The spiking neural network is a type of artificial neural networks which consists of biologically modelled neurons. They are considered part of the third generation of neural network models, which also includes other models such as deep learning.

Spiking neural networks operate on the theory that neurons within the network do not fire at each propagation cycle, such as perceptron based networks but rather release spikes at intervals where the membrane potential reaches a threshold value. Spiking neurons focus on the transmission of electrical charges, where received spikes cause an increase or decrease in the membrane potentials of downstream neurons.

2.3.1 Spiking Neurons Models

Many computational models mimic the behaviour of biological neurons in the brain. In this section, we discuss the most relevant models to our work; the Hodgkin-Huxley Model, the Izhikevich Model, and the Leaky Integrate and Fire model.

All of these models have in common that the neuron receives some input current from some source. Since we use excitatory and inhibitory models, see 2.1.3, our input function $I(t)$ for spiking sources:

$$I(t) = e - i \tag{2.8}$$

Where e is the excitatory inputs, and i is the inhibitory inputs. When a neuron spikes the excitatory or inhibitory input is adjusted by the weight of the connecting synapse. These inputs are functions of time and decay

according to:

$$\frac{de}{dt} = e/\tau_e, \quad (2.9)$$

$$\frac{di}{dt} = i/\tau_i, \quad (2.10)$$

Where τ_e and τ_i is the rate at which the excitatory and inhibitory inputs decay, accordingly.

Hodgkin-Huxley Model

The Hodgkin-Huxley model describes the ionic (chemical) mechanism which underlies the propagation of spikes in the axon of the giant squid from empirical evidence [49].

The standard Hodgkin-Huxley models two critical components of the biological cell, the capacitance of the lipid bilayer and the ionic channel. The current flowing through the lipid bilayer is represented by:

$$I_{lipid} = C_m \frac{dv_m}{dt}, \quad (2.11)$$

Where C_m is the capacitance of the lipid bilayer, v_m is the membrane potential. The current through ion channel c is:

$$I_c = g_c(v_m - v_c), \quad (2.12)$$

Where g_c is the electrical conductance factor of ion channel c , and v_c is the reversal potential of ion channel c . The reversal potential is the membrane potential threshold where there is no net flow on the given ion channel.

Therefore for a sodium Na and potassium K channelled cell the current

through the membrane follows:

$$I = C_m \frac{dv_m}{dt} + g_{Na}(v_m - v_{Na}) + g_K(v_m - v_K) + g_L(v_m - v_L), \quad (2.13)$$

Where the channel l represents an implicit channel of the leaky conductance of the cell. Obviously, this model can be extended to represent many different ion channels.

The properties of an excitatory neuron is modelled by four ordinary differential equations derived from equation 2.13 and a set dimensionless quantities k, n, a associated with the potassium, sodium and sodium inactivation channels. These equations are:

$$I = C_m \frac{dv_m}{dt} + \bar{g}_{Na} n^3 a (v_m - v_{Na}) + \bar{g}_K k^4 (v_m - v_K) + \bar{g}_L (v_m - v_L) \quad (2.14)$$

$$\frac{dp}{dt} = \alpha_p(v_m)(1 - p) - \beta_p(v_m)p \quad (2.15)$$

Where p is an element from the set k, n, a and the α_p and β_p functions are:

$$\alpha_p(v_m) = (p_{\text{inf}}(v_m)) / \tau_p \quad (2.16)$$

$$\beta_p(v_m) = (1 - p_{\text{inf}}(v_m)) / \tau_p \quad (2.17)$$

Where $p_{\text{inf}}(v_m)$ and $1 - p_{\text{inf}}(v_m)$ represent the steady state for activation and inactivation. These equations were originally proposed in [49] from the

empirical evidence from the giant squid as:

$$\begin{aligned}\alpha_k(v_m) &= \frac{0.01(v_m + 10)}{\exp\left(\frac{v_m+10}{10}\right) - 1}, & \beta_k(v_m) &= 0.125 \exp\left(\frac{v_m}{80}\right), \\ \alpha_n(v_m) &= \frac{0.1(v_m + 25)}{\exp\left(\frac{v_m+25}{10}\right) - 1}, & \beta_n(v_m) &= 4 \exp\left(\frac{v_m}{18}\right), \\ \alpha_a(v_m) &= 0.07 \exp\left(\frac{v_m}{20}\right), & \beta_a(v_m) &= \frac{1}{\exp\left(\frac{v_m+30}{10}\right) + 1}.\end{aligned}$$

The Hodgkin-Huxley neuron is an important biological neuron model that provides a rich array of biologically plausible features. This model is relatively expensive, and there have been various modifications to improve both; usability with reducing computational expense, and the biological model with different functions and model adjustments.

Izhikevich Model

The Izhikevich model [54] is a neuron model that combines the biological plausibility of Hodgkin-Huxley neurons with the computational efficiency. The Izhikevich provides a rich set of behaviours such as spiking, bursting, firing patterns, post-inhibitory spikes and bursts, continuous spiking with frequency adaptation, spike threshold variability, bi-stability in both resting and spiking states, sub-threshold oscillations and resonance. The model has shown empirical evidence of dynamics similar to the mammalian cortex on the rat's motor cortex.

The Izhikevich neuron model is defined by the membrane potential v_m , and the recovery variable u , which are modelled by the differential equa-

tions:

$$\frac{dv_m}{dt} = 0.04v_m^2 + 5v_m + 140 - u + I \quad (2.18)$$

$$\frac{du}{dt} = a(bv_m - u) \quad (2.19)$$

With the auxiliary for spike resetting when $v_m \leq v_{thres}$:

$$\text{if } v_m \leq r \text{ then } \begin{cases} v_m \leftarrow c \\ u \leftarrow u + d \end{cases} \quad (2.20)$$

Where a describes the timescale of the recovery variable, b describes the sensitivity of the recovery variable, c describes the reset potential after spiking, d describes the reset of the recovery variable after spiking, r is the spike firing potential I is an input current via synapse or injected.

Integrate and Fire Model

Leaky Integrate and Fire neurons follow a standard implementation from [110]. These neurons follow a set of mathematical equations to mimic the spiking nature of biological cortical neurons and are computationally efficient. The following differential equation models the simplest representation for this neuron model:

$$\frac{dv_m}{dt} = (v_{rest} - v_m + RI(t)) / \tau \quad (2.21)$$

Where v_m is the membrane potential, v_{rest} is the resting potential, v_{thres} is the membrane threshold potential, τ is the rate at which the membrane leaky charge, and v_{reset} is the membrane reset potential. A spike is generated when $v_m \leq v_{thres}$ and applies $v_m \leftarrow v_{reset}$. The leaky integrate and fire neurons has

been shown to model the neocortical pyramidal cell behaviours [83].

The range of neuron behaviours do not range as widely and the Izhikevich and Hodgkin-Huxley models. Therefore we can introduce an extra dynamic such as spiking with frequency adaptation, via hyperpolarization, by introducing an extension to the leaky integrate and fire model. This is the leaky integrate and fire with adaptation model[65], which is modelled by the ordinary differential equations:

$$\frac{dv_m}{dt} = (v_{rest} - v_m + RI(t) + g(d - v))/\tau, \quad (2.22)$$

$$\frac{dg}{dt} = -g/\tau_g, \quad (2.23)$$

Where g is a dimensionless adaptation variable and when a spike is generated the membrane voltage is reset and $g \leftarrow g + e$. The variable d is the adaptation switch threshold voltage and e is an arbitrary small increment value. We will only be using this model for our leaky integrate and fire neurons in our works, as we describe in chapter 3.

Poisson Model

Poisson neurons are useful as inputs to a network as they are easy to control but provide some random variation to excite interesting dynamics. Due to their spike timings being based on probability to get an accurate timing these neurons can be used in a group to average out the spike timing on a receiving neuron.

There are many ways to model Poisson neurons, the model used in this paper is a homogeneous Poisson process from [48] which approximates spike occurrence during a time interval $\delta t \leq 1$ millisecond given a firing rate r :

$$P(\text{Spike during } \delta t) \approx r\delta t. \quad (2.24)$$

To generate spike using the equations for every discrete time-step of δt a random real number, x , between 0 and 1 is generated and if $x < P(\text{Spike during } \delta t)$ then a spike is generated.

2.3.2 Spike Timing Dependent Plasticity

Spike timing dependent plasticity (or STDP) is an unsupervised learning mechanism that strongly relates to Hebbian Theory, such that high correlation between pre-synaptic and post-synaptic spike timing of a neuron has an effect on the synaptic connections. As discussed in section 2.1.4 the long term potentiation and long term depression of neurons caused by pre-synaptic and post-synaptic spike timing has strong biological evidence within the brain. This form of learning allows for anti-Hebbian learning as synaptic weakening is modelled alongside the Hebbian theory of synaptic strengthening. We see in Figure 2.7 the change in the strength of synaptic weights after 60 pre-post or post-pre spike pairings.

The weight change of a synapse connecting neurons i and j due to spike timing follows:

$$\Delta w_{ij} = \sum_{f=1}^N \sum_{n=1}^N W(t_i^n - t_j^f) \quad (2.25)$$

Where f counts the pre-synaptic spikes, n counts the post-synaptic spikes, and t^f is the spike arrival time, t^n is the spike firing time, and W is an STDP function, such as in Figure 2.7. A common choice for $W(x)$ is:

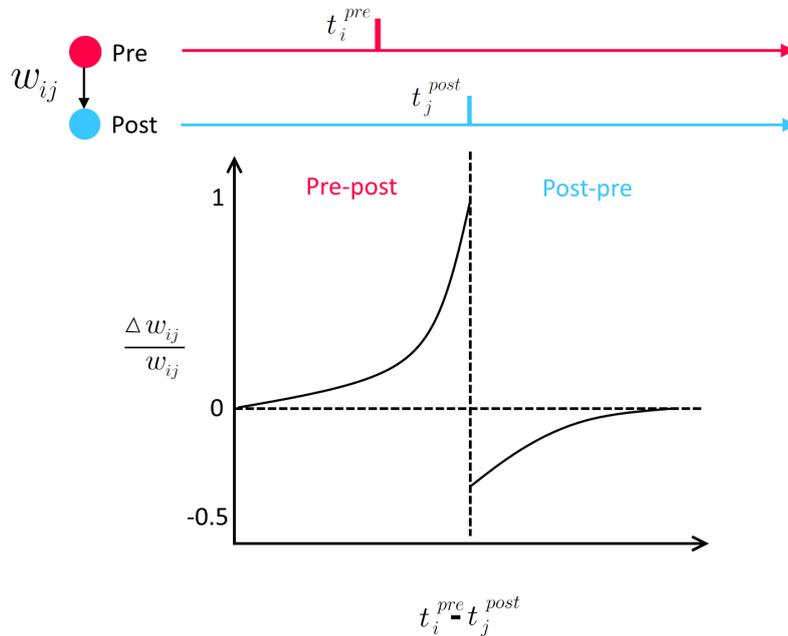


Figure 2.7: The STDP function, discovered in [9]. Recreated from [97].

$$W(x) = \begin{cases} A_P \exp \frac{-x}{\tau_P} & , \text{ for } x > 0 \\ -A_N \exp \frac{x}{\tau_N} & , \text{ for } x < 0 \\ 0 & , \text{ otherwise} \end{cases}$$

The behaviour of spike timing dependent plasticity allows for a biologically plausible features across a diverse range of neuron models [1]. While STDP is agreed to provide a convenient experimental model that provides important features of biologically relevant plasticity [97], it is unknown whether STDP, biologically, is a separate plasticity mechanism that exists within the brain alongside other mechanisms such as long-term potentiation and long-term depression. We notice that STDP does provide a frequency dependence compared to other mechanisms and potentiation and depression can only be

induced for a small window of frequencies [64, 98, 80].

Spiking timing dependent plasticity and spiking neural networks have been used together to develop a solution to the reinforcement learning problem.

MSTDP

In [37] the modulation of STDP by a reward function lead to reinforcement learning in spiking neural networks. The model was used to evolve networks that code solve both the rate encoded and temporally encoded exclusive-or (XOR) problem. The computational model changes the weight of a synapse from neuron i to j by:

$$\Delta w_{ij}(t + \delta t) = \alpha r(t + \delta t) \zeta_{ij}(t), \quad (2.26)$$

Where r is the reward function, and ζ_{ij} the eligibility trace of the synapse. Where ζ_{ij} was given by:

$$\begin{aligned} \zeta_{ij}(t) &= P_{ij}^+(t) f_i(t) + P_{ij}^-(t) f_j(t), \\ P_{ij}^+(t) &= P_{ij}^+(t - \delta t) \exp(-\delta t / \tau_+) + A_+ f_j(t), \\ P_{ij}^-(t) &= P_{ij}^-(t - \delta t) \exp(-\delta t / \tau_-) + A_- f_i(t). \end{aligned}$$

This model draws continuity between the unsupervised learning of STDP and reinforcement learning. This model is also biologically plausible where a neuromodulator in the brain could apply the reward signal to nearby neurons, such as found in [92] where the empirical investigation showed that dopamine carries a short-latency reward which indicates a difference between actual and predicted rewards. There has been speculation of the existence of STDP as

a biological mechanism [97], this work of modulation of STDP closely relates to the dopamine and acetylcholine modulation of long-term potentiation and depression of neurons in [93, 50, 106] even if STDP is not a biologically relevant mechanism.

2.3.3 Back-propagation

While biological models lead to interesting dynamics, direct optimisation method inspired by back-propagation for spiking neural networks do exist. While these methods are not biologically relevant, they show how specific structures allow would allow biological neurons to solve difficult problems.

SpikeProp

The SpikeProp algorithm has been shown in [11] to solve many classification problems such as the Iris dataset [36], Wisconsin breast cancer dataset [72] and the Statlog Landsat dataset[63]. This algorithm uses the error in the spike timings to propagate error throughout the network to modify synapses.

For spiking neurons, the error function requires that the spike contribution function be some differential function with respect to time. Whereby the spike contribution of neuron i over the synaptic terminal k at time t is:

$$y_i^k(t) = \epsilon(t - t_i - d^k) \quad (2.27)$$

Where t_i is the firing time of neuron i and d^k is the synaptic delay of terminal k . Where $\epsilon(t)$ is the spike response function where for all $t < 0$ then $\epsilon(t) = 0$

otherwise:

$$\epsilon(t) = \frac{t}{\tau} \exp^{1-t/\tau} \quad (2.28)$$

Where τ is the membrane potential decay time constant.

To compute and propagate the error through the network we need to define some characteristics; Γ_j which is the set of immediate predecessors to neuron j and Γ^j is the set of immediate successors to neuron j , t_j^d is the desired spike timing and t_j^a is the actual spike timing, and w_{ij}^l is the weight of the synaptic terminal l from neuron i to neuron j .

For each neuron j in the output layer the error based on the input signal is:

$$\delta_j = \frac{(t_j^d - t_j^a)}{\sum_{i \in \Gamma_j} \sum_l w_{ij}^l (\partial y_i^l(t_j^a) / (\partial t_j^a))} \quad (2.29)$$

For each neuron j in the hidden layer the error based on the input signal is:

$$\delta_i = \frac{\sum_{j \in \Gamma^i} \delta_j \sum_k w_{ij}^k (\partial y_i^k(t_j^a) / (\partial t_j^a))}{\sum_{h \in \Gamma_i} \sum_l w_{hi}^l (\partial y_h^l(t_i^a) / (\partial t_i^a))} \quad (2.30)$$

For the synaptic terminal k connecting neuron i to neuron j the weight is changed by:

$$\Delta w_{ij}^k = -\alpha * y_i^k(t_j) \delta_j, \quad (2.31)$$

Where α is the learning rate. The algorithm follows:

1. Calculate δ_j for each neuron j in the output layer, via 2.29.
2. For each layer in the network starting from the immediate predecessor to the output layer:
 - (a) Calculate each δ_i for each neuron i in the layer, via 2.30.
3. For each neuron i connecting to neuron j in the output layer, change

the weights of all synapses via 2.31.

4. For each layer in the network starting from the immediate predecessor to the output layer:
 - (a) For each neuron i connecting to neuron j in the current layer, change the weights of all synapses via 2.31.

While this provides the superb supervised learning algorithm for spiking neural networks, the restrictions it imposes essentially defeats the purpose of using biologically inspired neurons. These restriction include but are not limited to: the network must be a feed-forward neural network, neurons are only allowed to fire once, inputs need to be constructed such that each layer causes the following to spike at some time, the excitatory and inhibitory neuron models are not accounted for, and the network requires a significant amount of synapses between each neuron with different delays.

SpikeProp Extensions

In [91] the SpikeProp algorithm was improved by introducing learning for; synaptic delay, synaptic time constants, and neuron thresholds. This improved algorithm is useful for reducing the complexity of the network used, though this still leaves restrictions such as the neurons only being able to spike once. Assuming that all neurons share the same spike contribution function these rules are as follows.

The delay learning rule:

$$\Delta d_{ij}^k = \alpha_d w_{ij}^k \frac{\partial y(t_j^a)}{\partial t} \delta_j, \quad (2.32)$$

Where α_d is the learning rate for delays.

The synaptic time constant learning rule:

$$\Delta\tau_{ij}^k = -\alpha_\tau w_{ij}^k y(t_j^a) \left[\frac{(t_j^a - t_i^a - d_k^{ij})}{(\tau_{ij}^k)^2} - \frac{1}{\tau_{ij}^k} \right] \delta_j, \quad (2.33)$$

Where α_τ is the learning rate for the synaptic time constants.

Finally, the learning rule for the membrane threshold of a neuron:

$$\Delta\vartheta_j = \alpha_\vartheta \delta_j \quad (2.34)$$

Where α_ϑ is the learning rate for neuron membrane thresholds.

Multi-SpikeProp

The Multi-SpikeProp algorithm from [40] does allow for multiple spikes from neurons within the hidden layer, though the output layer is still limited to only one spike. The algorithm solving the exclusive-or problem as well as epilepsy and seizure detection problems, offering better accuracy over standard SpikeProp with reduced synaptic terminals though requiring more learning iterations.

Chapter III

Framework and Design Philosophy

This chapter covers our decisions and discoveries from designing and implementing spiking neural networks. We propose the Spiking State Machine as a standard modular framework based on the observations we made, and we also propose an ontology describing our artificial spiking neural networks. This ontology clearly generalises components of these networks for the development of novel mechanisms that do not fit the canonical network description.

3.1 Experimental Observations

When implementing network and learning techniques to evaluate for use in our research, we made some important observations that were not well described in background literature.

During design planning we found that the representation and encoding of inputs are significant, this is commonly mentioned concerning rate-based encoding and temporal based encoding, although concerns that affect both encoding strategies are not discussed. The input representation needs to consider the fact that neurons can only propagate an input signal into the network if they receive frequent enough input stimulation to spike themselves. The inverse is also a problem where too much input can saturate a network where the membrane potential of every neuron quickly reaches the threshold

voltage even after a spike event has just occurred. As a result of these bounds emplaced on our neuron models we propose two solutions. The input stimulation must be designed to perfectly rest within a range of responses that do not over-saturate or under-provision the network. Another solution is to change the neuron models such that they adapt the membrane threshold when intense stimulation is frequently applied.

We also determined that all the parameters and hyperparameters that describe the network, or its components, are essential to the solution of the application. We attempted to reproduce the results of the literature to address the constraints of the models and learning methods but had difficulty in achieving the replication of results. We assume that poor replication was due to the lack of described parameters from network construction, synaptic weights, and neuron parameters such as refractory periods. The lack of implementation specifics on models also had an effect on the replication of results. Some works did not define the used refractory models which may or may not allow the build up of input, or the use of excitatory and inhibitory neuron models. This lack of cohesion leads to non-reproducible results, and we argue that this requires a solution to unify the environment and description.

We desire that our networks be designed to solve problems, such that we can devise and compare models. This desire means that learning is an important aspect which needs to be resolved. We have discussed in section 2.3.2 and section 2.3.3 solutions to the learning problem for spiking neural network. The algorithms provided are often over restrictive and impose a situation such that models and dynamics may not be studied carefully and compared. We see this with requirements that neurons may only spike once and that other spikes contribute nothing to the solution, or that synapses need to be adjusted

every propagation cycle, as well as requiring that features such as synaptic delays not be present within the network. These restrictions imposed by the learning algorithm ruin the potential for model comparisons, although these learning methods are not representative of unsupervised learning methods developed. We do not consider unsupervised methods as a possible solution, as they do not cover all problem spaces and are not designed to learn specific tasks as we discussed in section 2.2.3. Therefore we need to develop new universal learning methods for spiking neurons or work around the limitations that these methods impose until we further our understand of biological learning.

3.2 Modular Framework

We propose the Spiking State Machine as the standardised modular framework. The Spiking State Machine is a restrictive subclass of the more general Liquid State Machine, which focuses on the modularity of the liquid component. This framework bypasses issues we found in the previous section as we discuss later in the following subsections. In Spiking State Machines problem specific learning is separated from the spiking neural network. The framework eliminates the necessity of describing all aspects of the system, instead only requires that the network and its contained models are sufficiently defined.

3.2.1 Structure

A Liquid State Machine is a reservoir computing method, and the structure can be observed in Figure 3.1, consisting of three main features:

Input Transforms inputs into a continuous sequence of disturbances.

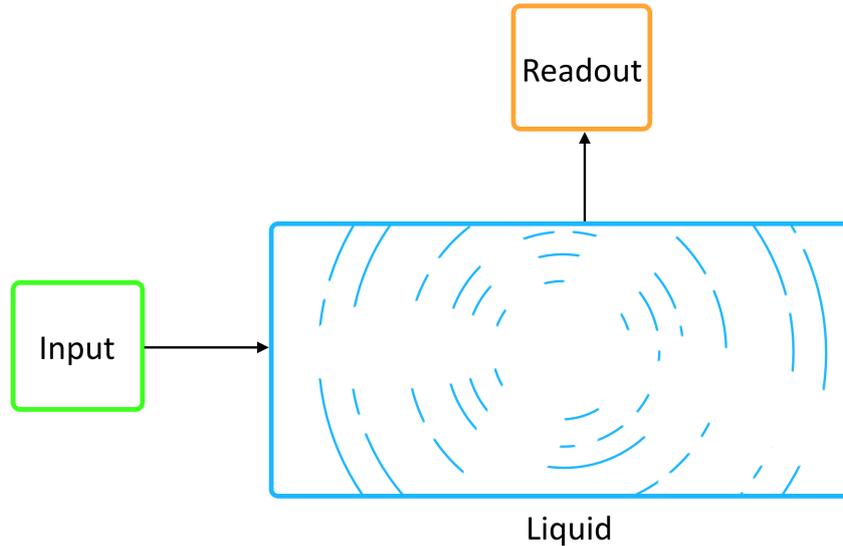


Figure 3.1: The Liquid State Machine and its main components.

Liquid A reservoir of computation units that holds some state based on current and previous inputs.

Readout A function or set of functions which transform the liquid state into the desired output signal.

In comparison to traditional recurrent neural network methods, Liquid State Machines offers two advantages; i) They are easier to train since only the readout layer needs to be adjusted, ii) The liquid can be used to solve more than one problem as the ideal liquid is universal. Therefore more readout functions can be attached for different problems. The Liquid State Machine bears a resemblance to kernel methods and support vector machines, as inputs are projected into a high-dimensional feature space by the network dynamics, the main difference only being the temporal dependence. The projecting into high-dimensional feature space makes it possible that classes are linearly separable.

The Spiking State Machine defines the input components as Poisson neuron groups and the readout layer as a single perceptron. We also define that the liquid component is required to be a form of spiking neural network for modularity.

For the input source, we attempted using both uniform spiking neurons as well as Poisson neuron models. We found through observation that the uniformly spiking model was not very useful at exciting dynamics even though spike timing is guaranteed. We noticed a Poisson model used in groups with a lower input voltage, allowed for the post-synaptic neurons to have higher average membrane potentials, requiring less variability to excite spiking. When used in a group the statistical noise in the firing rate of the Poisson neurons is averaged by the post-synaptic neuron. When evaluating the uniform model, we would see input voltage was completely decaying and not exciting post-synaptic neurons, or since the network is only excited at these uniform spike timings, network penetration may not be achieved due to the refractory period. We did not spend much time evaluating measures to make uniform models more useful but favoured Poisson neurons due to being used in most previous works and required no modifications for use.

When using Poisson neurons we found several considerations that need to be taken into account. The most notable are the firing rates of inputs and the amount of input they place into the network. The input has to have a significant enough effect to incite neurons further in the network to spike and propagate into the network. This can be mitigated by designing input or reducing the spiking threshold of the neuron model. The input also has not to be so large that it saturates the network, such that the state does not change as neurons are always reaching their firing potential whenever they can spike again. This is commonly caused by the neuron excitatory and

inhibitory inputs not decaying away quickly enough that they are constantly growing or at equilibrium with input sources.

The perceptron as the readout layer imposes that the liquid component produces the nonlinear potential required for the problems solution. The perceptron is also a well-studied classifier and has a wide range of possible learning techniques. This allows us to abstract learning based on the problem space away from the implementation of the spiking neural network, allowing for better model comparisons. We require that the perceptron records neuron spikes between updates, and weights spikes by their timing, this is temporal encoding and gives the perceptron a form of input memory. Theoretically, this perceptron could also be replaced with an arbitrary neuron model instead, although it is useful to receive input when the readout polled. It would also be possible to monitor the membrane potentials instead of spikes, but spikes are more deterministic in their presence, as a membrane potential may be at some value for a variety of reasons. Therefore the perceptron provides a learning mechanism separate from the spiking neural network but requires that the network state has separation potential over the input space.

In summary, the Spiking State Machine requires that the liquid be a spiking neural network that is robust under input noise and has separation potential over the input space. We also add to this that the refractory period of neurons allows for the build up of membrane potential if undefined, as to address a common issue with undefined refractory periods. This framework allows for devising new models within the liquid component, fulfilling the modular component of our design. In chapter 4 we evaluate the consequences of the design of the liquid component and determine whether this framework is useful.

3.3 Requirements and Properties

Due to the Spiking State Machine being a type of Liquid State Machine, there are some idealised conditions that need to be addressed [70]. In the ideal situation, the spiking neural network should satisfy the point-wise separation property, while the approximation property condition is already satisfied by the perceptron.

The point-wise separation property requires that the state of the spiking neural network be different for any two input functions into the network. Due to the spiking neural network being a modular component, this cannot be addressed by the framework. Therefore the modules that are used must satisfy the point-wise separation property. The point-wise separation property is also required of the network by the readout perceptron. The perceptron requires that the spiking neural network must separate the input space such that points within the space are linearly separable by their class.

The approximation property requires that the perceptron be able to approximate some function to arbitrary precision for some bounded input space. Therefore given the function being approximated is linear, the perceptron fulfills this property, as the perceptron approximates a hyperplane for a bounded input space. The perceptron being a linear classifier has an advantage over nonlinear classifiers, as the sum of squared errors has just a single local minimum which is also a global minimum, compared to nonlinear methods.

These properties are formally proven in [70]. The neuron models require other conditions such as time invariance and fading memory. The biological spiking neuron models we will use fit all these conditions. Our neuron models, charge potential from inputs, spike to remove potential, and leak potential

overtime, which follows the fading memory principle. The spiking neurons are also time invariant as they have deterministic behaviour based on the current membrane potential and applied inputs.

The spiking neural network (the liquid) of the Spiking State Machine is wholly responsible for the computational power of the system. For Liquid State Machines, W. Maass et. al. [70] uses a topological structure which is similar to a column. This column based liquid is inspired by the neocortical structure within the brain [112]. The liquid structures in literature favour layered networks of columns with a low connectivity. This restriction upon the liquid needs to be considered. W. Maass et al. poses that the addition of more neural components is a simple solution to increase the separation potential of the Liquid State Machine. Simply increasing the size of the neural component or adjusting connectivity may not be an easy task. That is if the Spiking State Machine is applied to physical brain tissue more neurons would be impossible to add. In artificially grown tissue, we would need to develop methods of deterministically growing neurons and connections.

The Liquid State Machine has universal power for computations with fading memory on functions of time, if any filter that is time invariant and has a fading memory can be approximated to an arbitrary precision [70, 69]. In [113] isolated word recognition was solved using a Liquid State Machine of spiking neurons, while [18] used a similar Liquid State Machine model for movement prediction of a rolling ball with reliable real-time results. These problems are natural peers for Liquid State Machines as they are temporal problems, and helps to show that it is excellent for model analysis for such problems. This also translates to our Spiking State Machines.

3.4 Construction and Training Procedure

The Spiking State Machine only requires the readout layer to be tuned to the problem, the typical usage under a supervised learning environment follows:

1. Define the liquid structure and initialize internals.
2. For each input vector $u(t)$:
 - (a) Encode $u(t)$ as a vector of frequencies $f(u(t))$.
 - (b) Feed $f(u(t))$ to the input groups and record internal state over a period T .
 - (c) Record the output $o(t+T)$ and use a supervised learning algorithm to adjust readout layer based on target $y(t)$.

3.5 Liquid Ontology

There does not appear to be an agreed upon standard ontology for artificial spiking neural networks, where the terminology is borrowed directly from the biological system. This leads to restrictions that require that explanations for classes and relationships need to remain true in both the biological and simulation domain or otherwise they cause unnecessary confusion. We, therefore, specify an ontology to rid ourselves of these restrictions, and where the biological classes are subclasses of the classes we define. We use this ontology to describe the liquid component of the Spiking State Machine.

We specify the main classes of our ontology as:

Neuron A neuron is the node of the directed graph, containing some computational model.

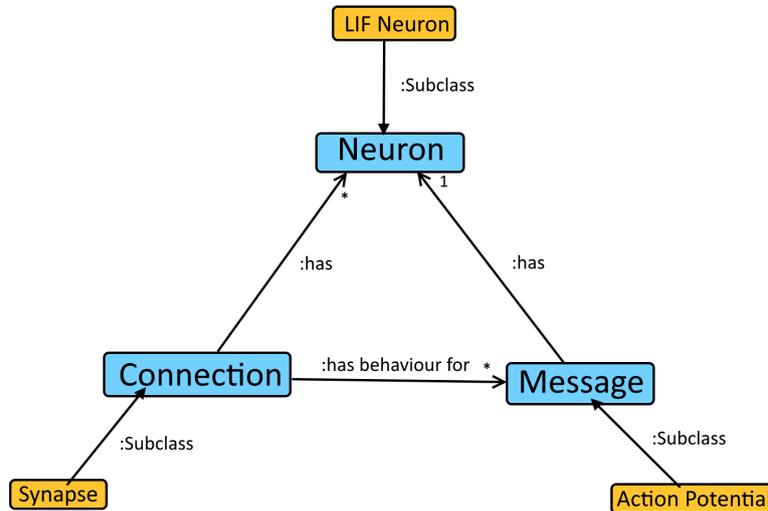


Figure 3.2: Our proposed ontology, showing the generalised relationships.

Connection A connection is the edges of the directed graph of the network, and allow for the transfer of information throughout the network.

Message A message is some arbitrary representation of information.

Here we note that a synapse is a subclass of the connection, we use the term connection as this is meant to represent any possible representation of connection whether abstract or physical. While an action potential or spike is a subclass of the message class, the same as with the synapse-connection relationship, this allows more representations from abstract to physically based.

The relationships which are defined by the ontology includes. Messages have a source neuron. Connections have many neuron connections. Connections have behaviour for many types of messages. Finally, connections modify the state of neurons based on messages passing through them. The relationships

as described have zero to many cardinalities, where the subclass determines further restrictions. For example, a synapse will limit the number of neuron connections to two. The constraints that exist from the biological terminology do not hold, meaning that connections are not required to be directional, and messages can consist of any information, et cetera. Since we are still trying to simulate possible biological models, neurons remain the same. This ontology allows us to diversify our models, and clarify the difference between in-vitro and in-vivo systems from simulated systems.

3.6 Liquid Implementation

This ontological description of artificial spiking neural networks offers an almost direct translation into programmatic structure. Where neurons and connections, can be grouped by functionality, abstracting implementation for usability, suggesting object-orientated design. While functional programming is suited for messages and the state changes that connections apply to neurons.

Each neurons are a collection of parameters and functions. Parameters are stored in a object or by parallel arrays. Functions are either methods or kernels that updates the parameters by some delta in time according to the mathematical model. Therefore groups of neurons can be encapsulated by a single strategy that calls the update function for a group of neurons in an optimised manner. This strategy would also handle the messages that should be created and processed by the simulation.

Connections are also a collection of parameters, commonly by parallel matrices, where the synapse subclass requires that they have a weight, a source neuron, and a sink location. We keep the definition of the sink general,

as biological synaptic connections do not necessarily connect to other neurons but other cells in general. These connection external to the network are useful for the readout component of our Spiking State Machine. Since connections apply state updates to neurons, they should have attached appropriate anonymous functions that deal with certain message classes and neurons.

Messages are best represented by an event system, which may also cover other reactions within the simulation. The anonymous functions can be used to capture events, and these can be linked with a group of synapses. The most general application is capturing the spike event from the pre-synaptic neurons and applying some change to excitatory or inhibitory inputs of the post-synaptic neuron. The direction of the synapse defines the pre-synaptic and post-synaptic neurons.

Chapter IV

Nonlinear Problems and Framework Evaluation

The purpose of this chapter is to validate our simplicity sub-goal for the Spiking State Machine, and through experimentation, determine the consequences of the design.

In chapter 3 we discussed how we want to develop new models for spiking neural networks. This resulted in a standardised and modular experimental framework such that we can analyse and devise new models for spiking neural networks. We called this framework the *Spiking State Machine*. The Spiking State Machine is a partially defined Liquid State Machine that only requires implementation of the liquid component. The separation of inputs come from the network dynamics, and learning methods are not applied to the network but an external readout layer. We will experiment with nontemporal and nonlinear problems to see the capabilities of this framework outside its designated domain of temporal problems. Since the readout layer is a single perceptron which is a linear classifier, the nonlinear potential must be generated by the liquid.

A linear classification problem is a problem that can be solved by splitting classes with a hyperplane, that is in two-dimensions a straight line. A perceptron is a linear classifier meaning that it can be used to solve any linear

classification problem. A nonlinear classification problem cannot be solved by separating classes using a straight line. The most well known and simplest being the classical *exclusive or* (XOR) problem. The XOR problem is a specific problem of the nonlinear hypercube problems. A hypercube of n dimensions can define 2^{2^n} possible problems with two classes. For the two-dimensional hypercube, there are 16 possible problems, of which two are nonlinear, the XOR problem and the negation of the XOR problem. As the number of dimensions of the hypercube increases so does the number of nonlinear problems and the number of hyperplanes needed to separate the problem.

In [37] the XOR problem was solved with a biologically relevant method, using both rate encoding and temporal encoding. The method described was a modification of spike timing dependent plasticity, where the reward was modulated to change the synaptic weights of the network. Offering a biologically plausible solution to classification problems, as features of spike timing dependent plasticity is known to occur in the biological brain, see section 2.3.2. The consequences of this method include; requiring extensive per synapse modifications during training as synaptic weight changes occur frequently, and does not support models that include synaptic delay. This method while favourable does not satisfy the modular component of our desired solution and does not allow for the separation of learning from the network.

There are many nontemporal, linear and nonlinear problems. The logical and (AND) problem is a very basic linear classification problem that can easily be solved by a single perceptron. The AND problem is important to consider when experimenting because we need to ensure that we can linearly map features into the state of the network such that there is not a significant effect

on a perceptrons ability to solve that problem. A classical nonlinear problem is the *exclusive or* (XOR) problem. While this problem is considered simple and not representative of the diversity amongst nonlinear problems, it allows us to judge the feasibility of nonlinear potential before attempting more challenging problems. In comparison to the AND problem, the XOR problem is a nonlinear Boolean classification problem that is not solvable by a single perceptron, as it requires at least two hyperplanes to separate the classes. If we can solve the XOR problem using a perceptron by first transforming our inputs into a new feature space using a Spiking State Machine, we know that the non-linear separation power comes from the spiking neural network.

For spiking neural networks, the XOR problem is not a simple problem as discussed in [11, 91, 37], where it requires at least 5 hidden neurons and a significant amount of synapses, in some cases requiring 320 synapses. We can see in Figure 4.1 an optimally designed solution to the XOR problem which exposes the four possible input spaces to the readout layer of the Spiking State Machine, the weightings may vary based upon neuron parameters and implementation. The inhibitory neuron model is required to reduce the response of the neuron that exposes the 0, 0 input pair for any other input pair. We can deduce from this designed network that the non-linear power of a spiking neural network must then come from the dynamics of interacting excitatory and inhibitory neurons. Suggesting that the desired framework would require that all spiking neural networks must incorporate the excitatory and inhibitory neuron models, or find other models that create nonlinear behaviour.

We want to show that we can solve the nonlinear classification XOR problem with a spiking neural network, without per synapse modification by optimisation. The purpose of this experiment is to confirm our simplicity sub-goal

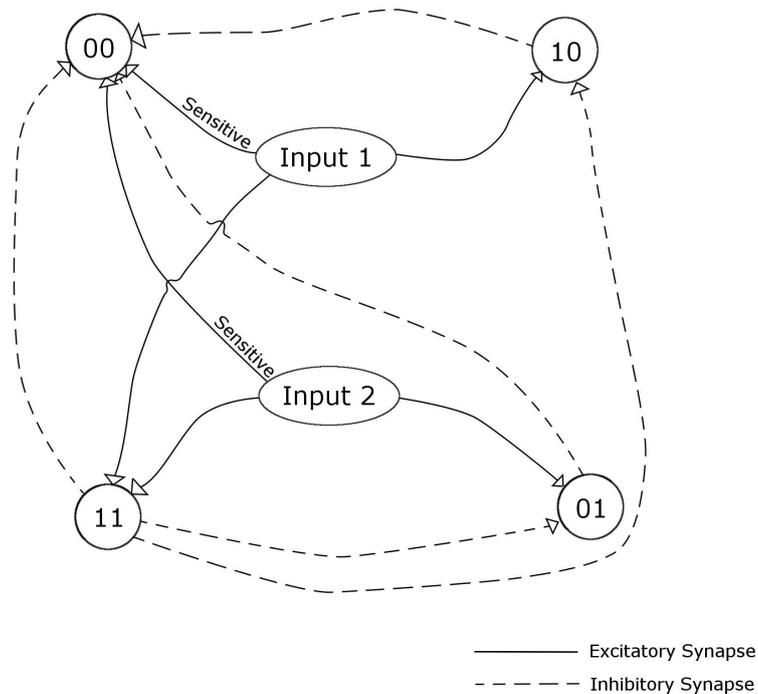


Figure 4.1: A designed liquid that can solve the XOR problem.

for the Spiking State Machine. The simplicity goal is in respect to being able to apply new problems arbitrarily, without a focus on manual design or reliance on iterative tweaking to obtain results.

4.1 *Liquid State Machine Setup*

The experimental setup used for both our AND and XOR experiments consists of a Spiking State Machine where the components are arranged as follows in Figure 4.2. The input layer connects to all neurons in the first layer of the

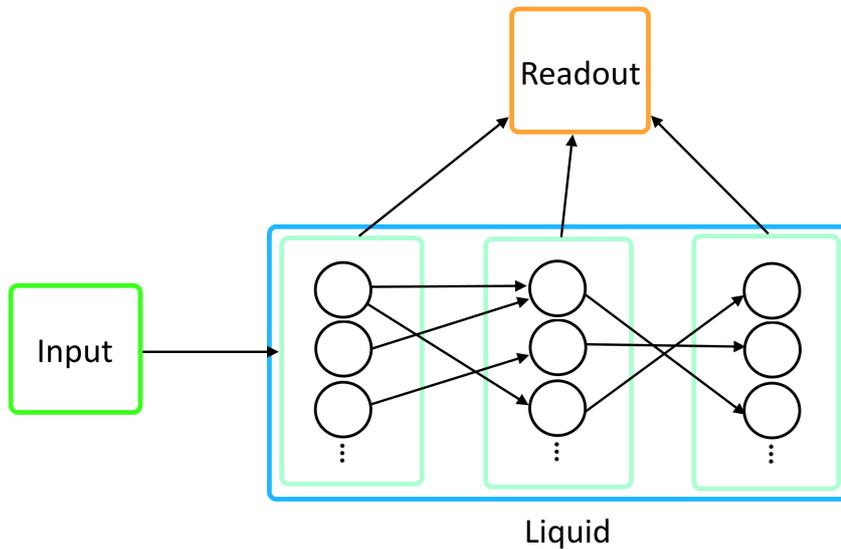


Figure 4.2: The layout of the Liquid State Machine used in our experiments.

liquid, each neuron in a layer is partially connected to the next layer, and the readout connects to every neuron in the liquid. The input component consists of a set of Poisson neurons, a spiking neural network as the liquid and a single perceptron as the readout as described by the Spiking State Machine. For each input feature, there is a group of 20 Poisson neurons that were used to average the firing rate to all post-synaptic neurons in the liquid. The liquid contains 60 leaky integrate and fire neurons, where there are 3 layers of 20 neurons each, 15 excitatory and 5 inhibitory. Each layer of the liquid connects to a neuron in the next layer, with a probability of 0.2 and a uniform random weight assigned between 0 and 1. The readout perceptron is connected to all neurons in the liquid, with small initial weights $w \ll 1$.

For the specifics of neuron communication; Poisson neuron spikes add 0.01 to the excitatory input of the post-synaptic neuron. Excitatory neuron spikes add the synapse weight to the excitatory input of the post-synaptic neuron,

inhibitory neurons spikes subtract the synapse weight to the inhibitory input of the post-synaptic neuron. Each set of inputs is fed to the system for 100 milliseconds before the readout layer reads the state of the network. The readout layer is a single perceptron connected to all neurons in the liquid, where the inputs are the weighted timings of spike inputs, this is a temporal encoding representation rather than the firing rate encoding that is used in previous works [113, 18].

The topological structure of the liquid model, or more specifically the topology of the spiking neural network, was a decision based off needing to isolate issues caused by poor parameter selection. Recursive elements were removed to reduce issues with isolating network saturation, and this was also beneficial to these classification problems, as they are not temporally relevant and therefore would not need recursive elements to keep temporally specific information in the network. The layered structure that we use is based on the column approach in [18, 70, 113] with simplification due to the spatial positioning of neurons not being relevant or necessary. This layered approach also makes it easy to visualise how the liquid might function, where more active input signals penetrate deeper into the network layers, while less active inputs only present on the first few layers. For fulfilment of our simplicity requirement, this is an arbitrary liquid structure that we brought through from our early design and experimentation phases.

4.2 Experiment: AND and XOR

We show the inherent linear and non-linear power of the spiking neural network by solving classical two input Boolean problems, these problems are not necessarily a biologically relevant task but allows us to guarantee that the non-linear power comes from the liquid and not the readout layer. We

have chosen to solve both the AND and XOR problems for our initial experiments.

The AND problem consists of mapping two binary inputs to one binary output: $\{0, 0\} \rightarrow 0$; $\{0, 1\} \rightarrow 0$; $\{1, 0\} \rightarrow 0$; $\{1, 1\} \rightarrow 1$.

The XOR problem consists of mapping two binary inputs to one binary output: $\{0, 0\} \rightarrow 0$; $\{0, 1\} \rightarrow 1$; $\{1, 0\} \rightarrow 1$; $\{1, 1\} \rightarrow 0$.

We transform the binary inputs into spiking equivalents by mapping 0 to a 1000Hz firing rate and 1 to a 2000Hz firing rate, for the associated input group of Poisson neurons. This setup is different from other approaches such as [37] where the lack of spiking represents the input of 0, so in our representation inputs must be separated as well as cancelled when both inputs are the same for both situations. The output of the perceptron was rounded to the closest integer and was then direct mapped to the expected output of the problem.

The training set was 120 of arbitrarily arranged examples, while our test set was 500 arbitrarily arranged examples. The arbitrary arrangement of many examples is necessary because of the temporal nature of the liquid, and we need to ensure that the ordering has a minimal effect on results. We ran 100 trials where each training iteration saw 20 training examples before verifying accuracy against the test set.

The AND problem was solved all 100 times with an accuracy $\geq 99\%$. To reach the accuracy threshold 42 sessions required seeing all 120 training examples. 25 sessions only need to see 100 training examples. 16 sessions only needed to see 80 training examples. 9 sessions only needed to see 60 training examples. 3 sessions only needed to see 40 training examples. 5 sessions only needed to see 20 training examples. This problem was not expected to be

difficult to solve as the perceptron from the readout layer could solve this without the liquid component. Any examples that were incorrectly classified was due to random statistical noise generated by the Poisson neurons incorrectly shifting the input in the transformation into a temporal feature space.

The XOR problem was solved 96 times with an accuracy $> 92\%$. To reach the accuracy threshold 51 sessions required seeing all 120 training examples. 9 sessions only need to see 100 training examples. 12 sessions only needed to see 80 training examples. 7 sessions only needed to see 60 training examples. 13 sessions only needed to see 40 training examples. 4 sessions only needed to see 20 training examples. The results that finished early naturally tended to have a high accuracy $\gg 92\%$, while those that did not meet the threshold had an accuracy $> 70\%$. This problem could not be solved by the readout alone and requires the liquid to separate classes.

4.3 Observations and Thoughts

From our observations of our simulations, the use of a sophisticated and computational expensive learning algorithm was not required to separate a classic nonlinearly separable problem. The perceptron which was used as the readout layer could not solve a nonlinearly separable problem itself by using back-propagation. The perceptron translated the state of the liquid or spiking neural network into the desired result. Our experiment had the liquid act as a form of input processor that allowed for the readout layer to solve the problem, where no learning was applied, and the only computationally expensive operations were state updates. From this result, it is possible to draw a comparison to the brain, where there may theoretically exist a liquid layer (or many layers). This layer may be some form of standard

network in the brain that transforms inputs and a single neuron can use this network to make an intelligent decision, or at least make a contribution towards one. This pattern could also be recursive in nature on these decision-making neurons and so forth.

For these simple classification problems, the Spiking State Machine meets our simple requirement for a standard environment. We argue that this is not sufficient evidence. We experimented on two-dimensional hypercube problems, and we want to extend this into three dimensions to determine the growth in complexity required by the Spiking State Machine.

4.4 Experiment: Non-linear Hypercube

The previous set of experiments are not representative of more complex non-linear problems but only showing how a Spiking State Machine can solve simple non-linear problems. To test the Spiking State Machine with more complex non-linear problems we scale the dimensions of the XOR problem. Since the XOR problem is a nonlinear hypercube of two dimensions, we scale to a non-linear problem mapped to the vertices of a three-dimensional hypercube.

If a hypercube has n dimensions and position of any vertex for each dimension could either be 0 or 1. Data points can be generated in clusters around each vertex sharing the same classification as that vertex. For nonlinearity, the sum of all components of the position vector of a vertex determines its classification. If the total is even, then the point belongs to the positive class. Otherwise, it belongs to the negative class. A vertex connects to other vertices by the edges of the hypercube, this guarantees no vertex neighbours a vertex of the same class.

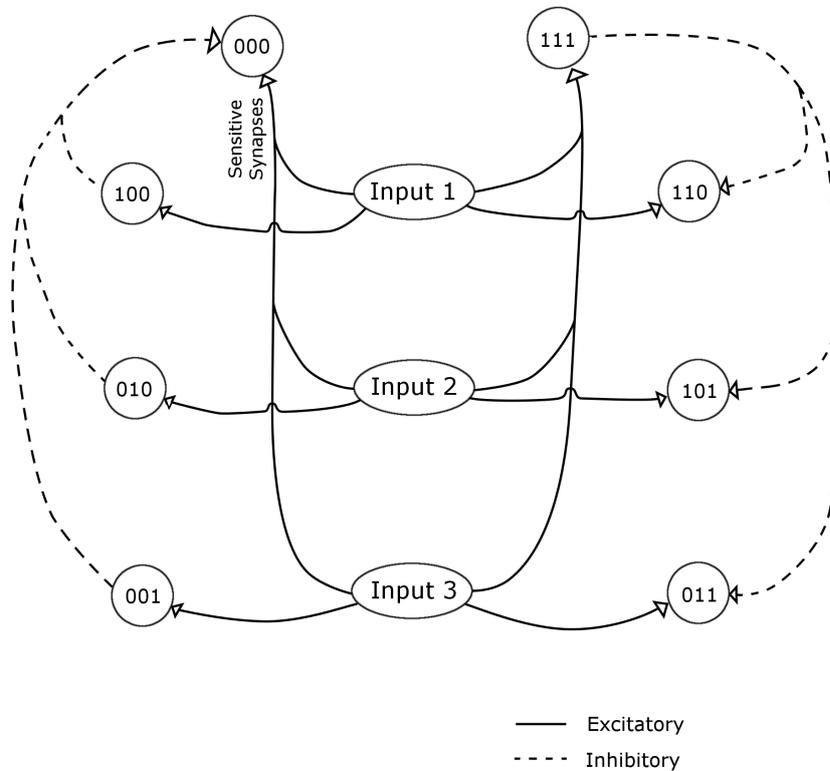


Figure 4.3: The diagram here describes a depicts liquid that can solve the 3 dimensional nonlinear hypercube.

For our three-dimensional hypercube we generate clouds around each vertex to make the problem more difficult. The cloud of each vertex had 50 points, totalling to 400 points. The points in each cloud were distributed by the gaussian distribution with a standard deviation of 0.1. This ensures that we are likely to have a slight buffer between each cloud at 0.5 on each axis.

To solve this problem we firstly introduced an 8 layer liquid, similar to the previous experiments. Each layer of the liquid has 40 neurons, adding to a total of 320 neurons. We then ran a simulation on a three-dimensional

hypercube following the same procedure as in our previous experiments. This three-dimensional hypercube needs at least four hyperplanes for the solution. We found that this structure of the liquid could not solve this problem and otherwise learned to either always classify as the true (or false) class. This poor classification is because we are relying on the liquid to expose some useful information about the inputs. The liquid which is only of a certain size with randomly generated connections and weights may be unlikely to contain a subnetwork that can solve this problem.

A large liquid may be able to solve this problem; such that the probability of a subnetwork existing in the network that solves the problem is highly likely. The readout then would learn to use only that subnetwork to make the classification decision. One such subnetwork can be seen in Figure 4.3, this designed liquid consists of 8 neurons which detect one of the 8 different states of input, and inhibitory neurons are required to suppress contributions from some inputs. In experimentation, this liquid solved the 3 dimensional nonlinear hypercube problem 192 of the 200 trails with an accuracy of greater than or equal to 94%.

As the liquid connections and weightings are randomly generated, and we do not know how arbitrarily large we need to make the liquid. A liquid that clearly separates the input state space can be obtained by increasing the liquid size till a solution is found, in the form of a subnetwork. This counters our sub-goal of simplicity. To avoid growing our liquid towards this unknown point, we modified our readout to include 8 leaky integrate and fire neurons that connect to all neurons in the liquid and are attached to a single perceptron used for classification. These readout leaky integrate and fire neurons can be considered a special case of the liquid that allows us to search the liquid to generate neurons that react to one of the input

state spaces. We call these neurons *pseudo-liquid neurons* as they can be used in the liquid simulation updates but are considered part of the readout layer.

The three-dimensional Hypercube problem was repeated with the pseudo-liquid neurons model in the readout layer of a Spiking State Machine. The liquid layer consisted of 320 leaky integrate and fire neurons, where each neuron was randomly connected, with an average of two connections, to other neurons in the liquid. For each neuron group in the input, there was on average four connections to the liquid component. The problem was solved 82 times out of 100 with an accuracy greater than equal to 94%. The accuracy threshold used allows for noise from; the Poisson neurons causing incorrect input classification, and data points that have a significant distance from their parent vertex which results in inputs that are ambiguous to what class they belong. To reach the accuracy threshold 76 trials required seeing 40 training examples to learn the output. 5 trials needed to see 60 training examples to learn the output. 1 trials needed to see 60 training examples to learn the output.

We have learned from this that in an arbitrarily large liquid, conditions might arise that non-linear problems are solvable. Although it is cheaper to create pseudo-liquid neurons that can find unique input state spaces, such that the liquid component may be smaller. This then alludes toward the use of spike timing dependent plasticity or similar methods within the liquid (or even pseudo-liquid) such that these neurons may arise more naturally. We can conclude from this that while for simple problems the Spiking State Machine is a useful environment it does not uphold our sub-goal for simplicity when given more complex problems. It may be unfeasible to keep growing the size of the liquid indefinitely or optionally introducing new mechanisms to

achieve satisfactory results.

4.5 Discussion and Summary

For simple problems, this gives us a standardised experimental framework where we can switch out the liquid for our models and apply other learning algorithms. To solve more difficult problems we have offered three solutions; the liquid may need to be grown, the liquid can be designed per problem, or the readout layer may be adjusted to include pseudo-liquid neurons. Each of the given solutions may not be applicable, as it may be unfeasible to grow the liquid, designing the liquid may not be possible with physical networks, and the pseudo-layer requires a deeper understanding of the possible input state spaces. Therefore we cannot say that we have met our simplicity sub-goal for our standardised experimental framework, but we have learnt that the range of possible problems extends beyond temporal problems.

The Spiking State Machine offers a way to analyse simple network dynamics, which may be useful for future research on brain-computer interfaces but requires adaptation to each problem if investigating problem-solving abilities. We have also argued, and through observation show, that learning algorithms that require extensive network modifications are not required to solve non-linear classification problems. This claim requires that a network that solves the problem be a subnetwork of the liquid of a Spiking State Machine.

Chapter V

Imperfect Spiking Neural Networks

In this chapter, we propose the *imperfect synaptic model* and network *doping* technique. We apply these to the cart-pole problem through the use of a Spiking State Machine.

Our understanding of how neurons within the brain communicate to form intelligence is not complete, though we do know how external factors play an important contributing factor. We see in [82], that communication within the brain is affected by the neuroimmune system, while [24] discusses how the modulation of gut microbiota communicate with the central nervous system and its affect on brain function and behaviour. These discoveries should make us consider that intelligence in the brain may not be as simple as just the dynamics created by spiking neurons.

We want to introduce a novel technique to our liquid of our framework. This technique addresses the communication of information that is not represented by spikes. Biologically there is the presence of an event called spontaneous neurotransmission. This is communication that occurs in the synaptic terminals by chemical or electrical means without apparent causation [114]. We desire a model that provides a way for our networks to have an event analogous to this biological event.

5.1 *Imperfect Synapses*

We propose the *imperfect synaptic model*, that is based on the biological event of spontaneous neurotransmission. Our theoretical model addresses that some element in the biological system, such as a synapse or cerebral fluid, would allow for some minor current to flow between neighbouring neurons. The standard synaptic model assumes that a synapse is a perfect electrical insulator, meaning that information is only transferred by spikes. In biological systems the imperfection of synapses has been addressed with theoretical models [78] and spontaneous neurotransmission events [6, 114]. We use the control problem of a cart-pole system and compare our model to the standard synaptic model.

The ontology that we described in section 3.5 is important for describing these imperfect connections, as the addition of the generalised connection and message classes allow us to describe this imperfect communication alongside action potential based communication.

Our design for the imperfect synapse comes from high-level assumptions with a grounding in electronic systems. Our inspirations for imperfect current is an electronic component called a diode. Diodes are analogous to the synapse, and they block electrical current in a particular direction though not without some leakage.

To evaluate our model, we will perform simulations to assess the affect it has on problem-solving. We apply this to control systems as temporal problems are a natural problem for Spiking State Machines, or more specifically Liquid State Machines. A benefit of using control problems is there is immediate changes within the system from the actions taken. Therefore for different system states, we may see actions taken that result in states that other

agent models may not reach or solve. The cart-pole problem is a classical control problem that involves applying force to a cart to balance a pole. This is a problem that is relatable to the human experience where one might balance a pen, broom or some other pole-esque object when bored [99]. Our innate human intuition of physics [96] allows us to analyse and understand the physical reactions occurring within the cart-pole system.

5.2 Mathematical Model

Most synaptic models assume that synapses are perfectly insulated when there is not a presence of a spike. Our proposed model addresses that the potential difference between two neurons would allow some current to flow by some imperfectness factor. This imperfectness would act similar to the second law of thermodynamics that states that the entropy of any isolated system always increases. Therefore the system would reach an approximate equilibrium state should a constant input be applied, where the imperfect connections ensure this property. The imperfect synapse also suits the requirements of our Spiking State Machine, as inputs are encouraged by the equilibrium state to fade and thus only significant bits are transferred.

Our proposed model has the benefit of transferring information between neurons without waiting for spikes, but the signal does not have a significant effect on the state of the receiving neurons. Although there are small changes in current made by imperfect synapses, it can influence the spiking potential of neurons overtime. The imperfect connection also imposes an implicit self-referential relationship where the state of the neuron itself implies something about the received input.

Formally we can describe this imperfect effect by considering the neurons i

and j which are connected by a synapse ij , from i to j with a weight w_{ij} . The imperfect current between i and j is:

$$\delta_{ij} = \frac{v_i - v_j}{R_{total}} \phi_{imperfect}, \quad (5.1)$$

where v_j is the membrane potential of neuron j .

The total resistance R_{total} of the neurons and connection can be modelled as $1/w_{ij}$ for that connection, and $\phi_{imperfect}$ is the imperfectness factor, resulting in the equation:

$$\delta_{ij} = (v_i - v_j)w_{ij}\phi_{imperfect}. \quad (5.2)$$

This thesis considers networks without any spatial dimensionality. Therefore networks with spatially located neurons, the equations would have to be adapted to consider distances between neurons and the medium between them, introducing more parameters.

For each propagation cycle, the algorithm for imperfect synapse updates follows below:

1. For the current neuron j whose membrane potential v_j changed.
2. For each pre-synaptic synapse from neuron i :
 - (a) Find $\delta_{ij} = \phi_{imperfect}w_{ij}(v_i - v_j)$.
3. For each post-synaptic synapse to neuron k :
 - (a) Find $\delta_{jk} = \phi_{imperfect}w_{jk}(v_j - v_k)$.
4. Finally the total imperfect current to neuron j is $\sum_i \delta_{ij} - \sum_k \delta_{jk}$.

5.3 *Explanation*

The proposed imperfect synaptic model would biologically act like ions outside of the membrane being attracted to neighbouring neurons, inciting dynamics such as spontaneous neurotransmission due to the increased potential in the ion gradient. We see in Figure 5.1 the effect of imperfect synapse, where the blue lines show how current is propagated through the network from changes in membrane potential causing the third neuron to spike, and red lines showing the reaction on the standard model. The evolution of events in Figure 5.1 is as follows. At time t neuron A has a membrane potential just below the threshold potential due to a previous spike event, neuron B has a membrane potential less than A but greater than C , and neuron C is at resting potential. At time $t + 1$ neuron B receives some current causing an increase in membrane potential, while neuron C receives some spike from the network as well as imperfect current from neuron B . Finally, at time $t + 2$ neuron C reaches its threshold potential due to both past stimulation events and imperfect current, without this model it may have taken longer to reach the threshold potential.

This model is bidirectional meaning that in the situation that neuron B has a larger membrane potential than neuron A , current will flow from B to A . This model does not cover all types of spontaneous neurotransmission, as in general neurons still need to receive some form of stimulation to reach the threshold potential, due to this symmetry of the imperfect synaptic model. We would expect that this feature would in general make reaching the membrane potential more difficult, as membrane potential is now also lost due to this imperfect current.

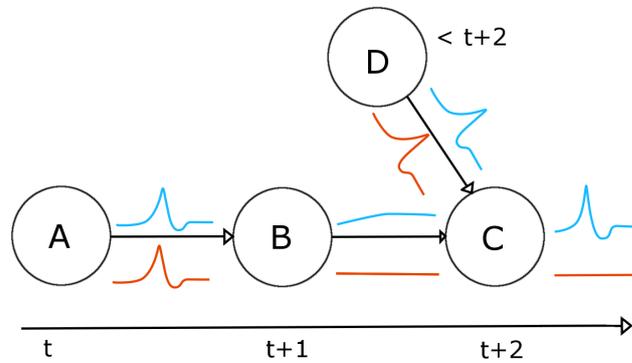


Figure 5.1: Showing the propagation of information via Imperfect Synapses.

5.4 Doping Networks

We also propose a method called *doping*, which replaces some synapses within a network with imperfect synapses with random imperfectness factors. In electrical systems doping is the act of purposely adding impurities to inert materials to produce electronic properties without causing electronic disorder. Doping in superconductors adds impurities to alter the charge of the system essentially adding excess electrons or removing electrons via ions. Here doping adds imperfect connections allowing for neurons within the network to inhibit or excite the membrane potential of neighbouring neurons without spiking.

The technique is applied after network creation, given some synapse threshold probability p and some bounds on the imperfectness factor, imp_{low} and imp_{high} the algorithm follows:

- For each synapse s in the network:

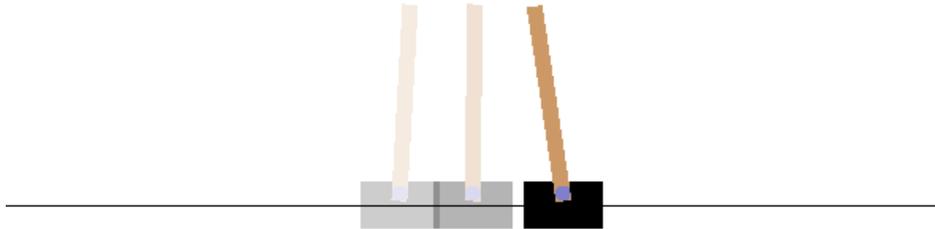


Figure 5.2: The cart-pole problem.

- Calculate the probability of being an imperfect synapse.
- If the probability is greater than p , then replace the synapse with an imperfect synapse with a random imperfectness factor between imp_{low} and imp_{high} .

5.5 *Cart Pole*

The cart-pole problem is a classical motor control task [4]. The problem consists of an agent trying to balance a vertical pole attached to a moving cart by applying a horizontal force to the cart. The cart is restricted to a small surface, the aim being to balance the pole on as close to the centre as possible.

The following equations represent the dynamics of the cart-pole system:

$$\ddot{x} = \frac{F - m_p l (\ddot{\theta} \cos \theta - \dot{\theta}^2 \sin \theta)}{m_c + m_p}, \quad (5.3)$$

$$\ddot{\theta} = \frac{g \sin \theta (m_c + m_p) - (F + m_s l \dot{\theta})^2 \sin \theta \cos \theta}{\frac{4}{3} l (m_c + m_p) - m_p l \cos^2 \theta}, \quad (5.4)$$

Where x is the cart position, θ is the pole angle from the perpendicular plane to the cart, F is the force applied to the cart, l is the half-length of the pole, m_c is the mass of the cart, m_p is the mass of the pole, g is gravity which is $9.81m/s^2$ (for Earth-like conditions).

In each tick of the simulation the Runge-Kutta method was used to approximate the ordinary differential equations of the system. Where each tick interval is $20ms$ and the forces applied to the cart could either be a left push or right push.

The cart-pole system can be seen as Markovian problem, with velocity information supplied to the agent, or non-Markovian problem, where only the cart position and angle of the pole is provided to the agent. The non-Markovian is considered the most difficult of the two, due to requiring the state of the system having to be inferred by the agent, specifically the velocity information. This requires that the agent has some memory which it can use to judge the velocity of the pole and cart from past inputs.

The problem can also be extended by adding additional poles of various lengths that are independent of each other. All poles need to be balanced and can both be Markovian or non-Markovian. This is considered a difficult problem and has been solved a few times, such as in [45, 104] which involve techniques not related to spiking neural networks.

The reward function we used, rewards the agent while the cart-pole system

is still within the system constraints:

$$r(t) = \begin{cases} 1 & , \text{ for } |\theta| < 12 \text{ degrees and } |x| < 2.4 \\ 0 & , \text{ otherwise} \end{cases} \quad (5.5)$$

For all cases of the cart-pole problem the fitness of an agent \mathcal{F} can be described by the equation derived from [45]:

$$\mathcal{F} = \frac{n}{N} \left(1 - \frac{\mu_{|x|}}{C_x}\right) \prod_{p=1}^P \left(1 - \frac{\mu_{|\theta_p|}}{C_\theta}\right), \quad (5.6)$$

where n is the number of ticks until a failure or success state, N is the total number of required ticks, P is the number of poles, $\mu_{|x|}$ is the mean position, $\mu_{|\theta_p|}$ is the mean angle for pole p , C_x is a scalar for successful positional values, and C_θ is a scalar for successful pole angles. An agent's fitness is greater for having the cart near the centre of the platform and the pole that is almost vertical for the entire simulation length. We modify this to better represent our reward function, where a balanced pole being the optimal condition the location of the cart being unimportant. The location of the cart is only important such that the cart does not come off the platform. The fitness function becomes:

$$\mathcal{F} = \frac{n}{N} \prod_{p=1}^P \left(1 - \frac{\mu_{|\theta_p|}}{C_\theta}\right). \quad (5.7)$$

In all the experiments in this chapter we consider the cart-pole problem being solved when an agent has an average total reward greater than 197 over 100 trails. Where the maximum total reward for a trial is 200 which is the same as the number of required ticks. These requirements are the same as the standard proposed by the OpenAI gym [15].

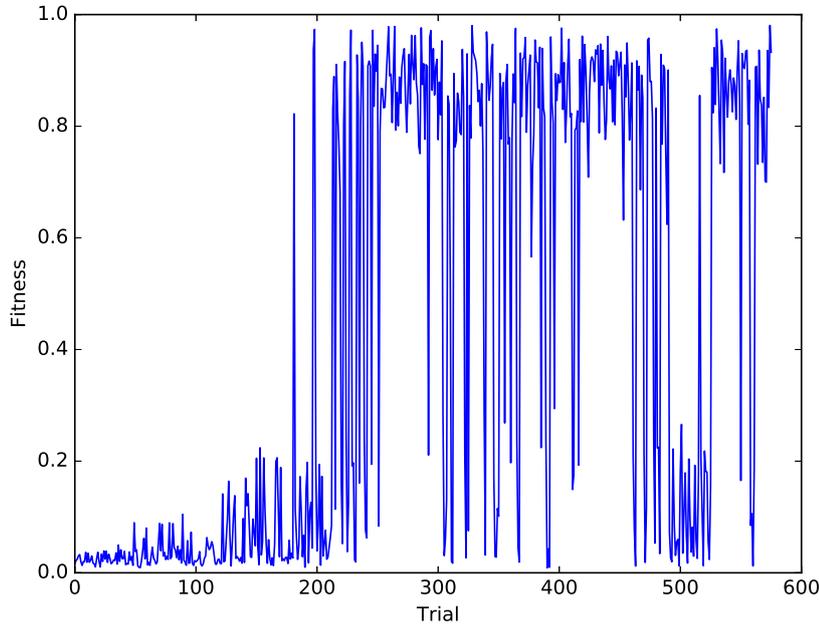


Figure 5.3: A plot of the fitness function of a Q-Learner solving the cart-pole problem via reinforcement learning.

5.6 Experiment: Q-Learning

Following previous implementations from literature [85] we used Q-learning with bins to solve the Markovian cart-pole problem, where equation 5.5 provides the reinforcement signal. On average the system took 528 trials to learn the solution to the cart-pole system. Where the Q-learner had three bins for the position, cart velocity, pole velocity, and six bins for pole angle. We see in Figure 5.3 the slow convergence rate of the Q-Learner.

We did not use Q-Learning for the non-Markovian equivalent, as Q-Learning can be used to find optimal action selection policy for any finite Markov decision process.

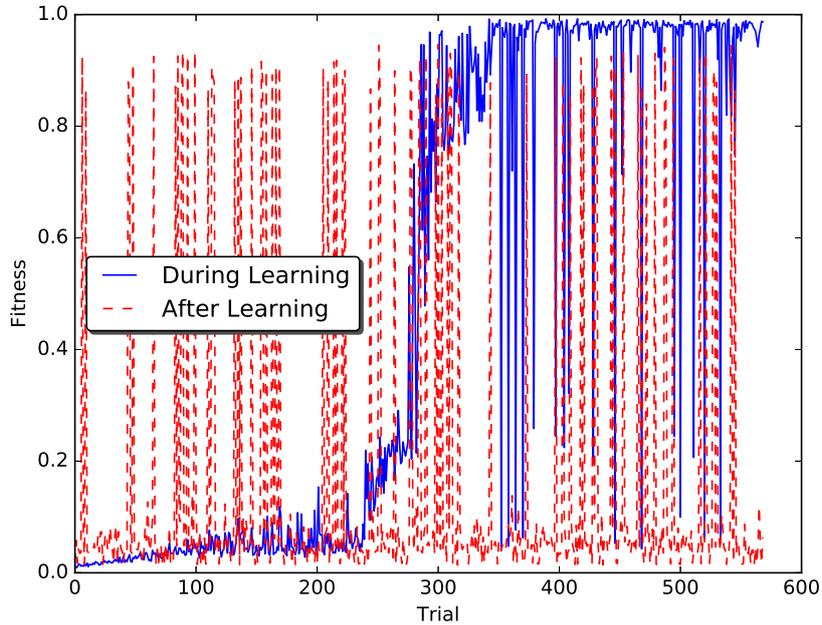


Figure 5.4: A plot of the fitness function of a perceptron for the cart-pole problem in a student-teacher system.

5.7 Experiment: Perceptrons

We use a perceptron to evaluate whether the readout of the Spiking State Machine can solve the cart-pole problem. We want to be able to analyse that the liquid found the solution, rather than the perceptron innately providing the solution.

Using a single perceptron and Q-Learning and a technique we call a student-teacher system. Where the student makes the action decision based upon the environment, and the teacher reinforces the student by providing an approximate error that leads the student towards the desired model. This is similar to actor-critic learning, but rather than reinforcement from the critic which suggests to the actor to repeat actions, the teacher provides more

information to the student in the form of error. The downside to this way of learning is that it requires a teacher that models the desired solution, which may not be possible. For our experiments, the perceptron used gradient descent on the error generated from the teacher for learning.

We attempted to solve the Markovian cart-pole problem, meaning the perceptron had access to the velocity information of the system. We found that the perceptron could balance the cart-pole system, while there was some error shown by the teacher. When the student-teacher system met our requirements for a solution to the cart-pole problem we removed the teacher and repeated the experiment with the learnt weights. We found that the perceptron could occasionally balance the system under certain initial conditions. Therefore in our following experiments we want to detach the teacher after a possible solution is found, and try to solve the cart-pole problem for a longer period of time. This will ensure that the dynamics are those of the liquid, if the agent can solve the problem without a teacher.

We then attempted the non-Markovian cart-pole problem, meaning velocity information was not provided to the agent. The perceptron averaged 9 ticks or 0.18 seconds of simulated time. For both problems, due to the symmetry, all inputs also had a negative input partner, this increased the amount of time a perceptron could maintain a balanced pole. The perceptron also had a connection to itself thereby always knowing what was the last action taken. When providing more knowledge of previous actions, there was a degradation in performance.

In our experiments, we have seen that under current conditions that a single perceptron cannot solve the Markovian and non-Markovian cart-pole problem. We see in Figure 5.4 a comparison of the fitness of a perceptron with a teacher (blue) and without a teacher (red and dotted) for the Markovian

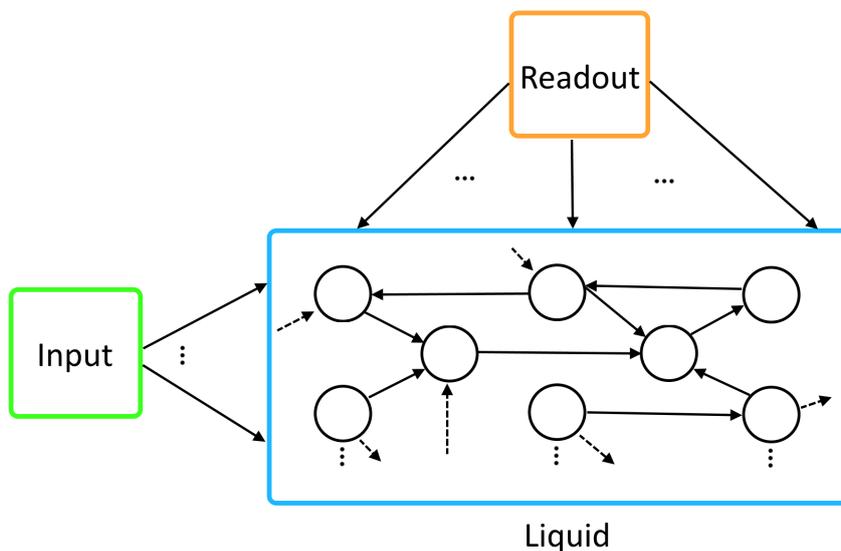


Figure 5.5: The liquid of the Spiking State Machine here is a random graph of probabilistically connected neurons.

cart-pole problem. We also found that for situations where the perceptron may learn a range of actions for the non-Markovian problem, the agent discovered that no action caused the pole to fall the slowest, thereby gaining the largest possible reward.

5.8 Experiment: Imperfect Synapse Comparison

We use a Spiking State Machine as the student of our student-teacher learning experiment to test our proposed model. Our liquid for our experiments is a randomly connected graph of 120 neurons, where each neuron has an average of four connections. This random structure which includes cycles allows for the state of the system to remain within the liquid. The inputs are the same as in section 5.7 except they are transformed using into spike trains using Poisson neurons. The readout is the same perceptron model and learning

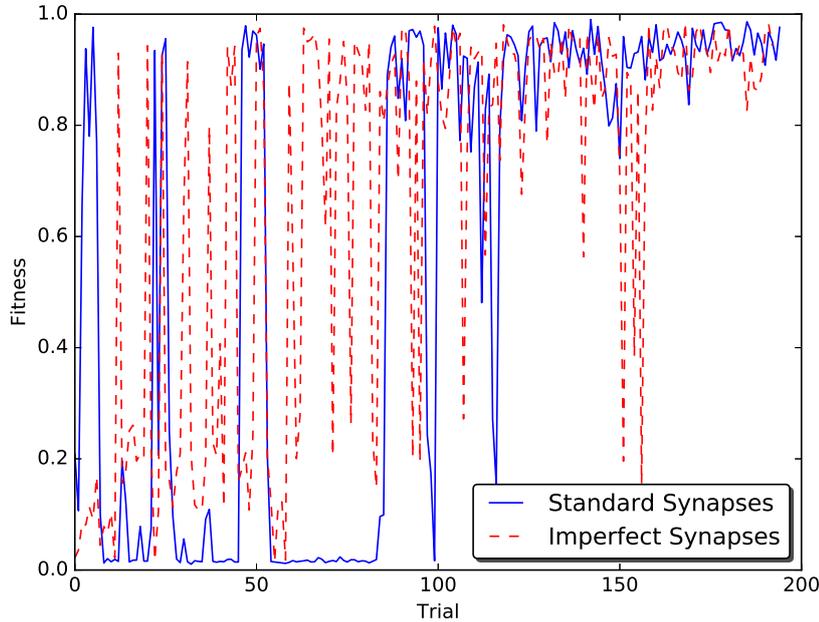


Figure 5.6: A comparison of the fitness function for agents with standard synapses and imperfect synapses, while trying to solve the cart-pole system. The last thirty trials are without a teacher.

from section 5.7. We see this structure more clearly in Figure 5.5.

We see in Figure 5.6 a comparison of the fitness of agents with and without imperfect synapses for the Markovian cart-pole problem. In the last 30 trials, the agent does not receive error correction from the teacher, and requires the agent to balance the cart-pole system for at least 400 ticks of simulated time. The agent with imperfect synapses had an imperfectness factor of 0.05. We saw there is not much difference in the fitness in either model, and they both solved the problem and balanced the pole with and without the teacher, for all 100 repeats of the experiment. The imperfect synapses would occasionally score low on the fitness function though still successfully balanced the pole, meaning that the pole was balanced at large oscillatory angles, failing to address the growth in the acceleration of the system. The agent with the

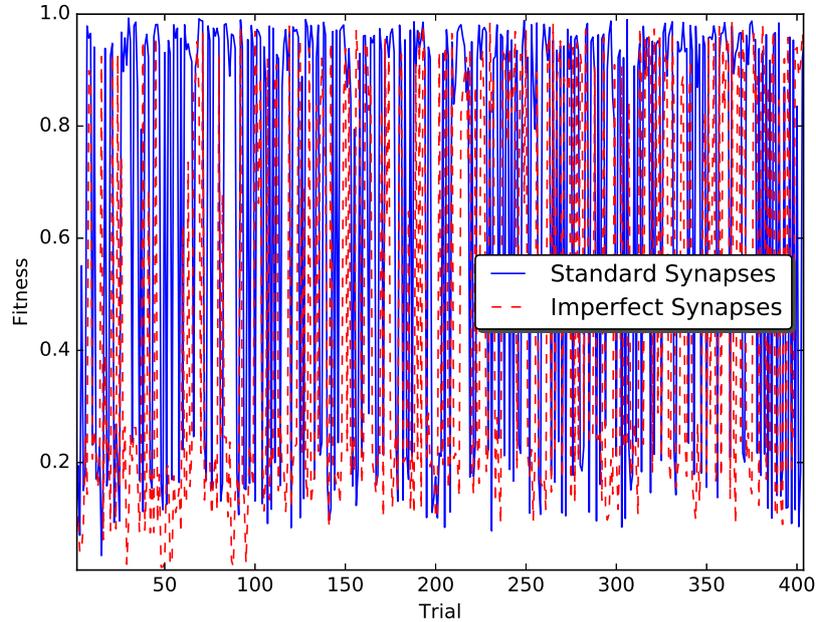


Figure 5.7: A comparison of the fitness function for agents with standard synapses and imperfect synapses, while trying to solve the cart-pole system under external action. The last thirty trials are without a teacher.

imperfect synapses had a fitness function that was on average lower than the average fitness of the agent with the standard synaptic model.

In the Non-Markovian experiment, the design of our current liquid did not introduce enough memory for either agent to sufficiently solve the cart-pole problem. There was no significant difference between the balancing time of the two agents, both having an average time balanced of roughly 98 ticks of simulated time. To solve this problem, another liquid may be required. We also conclude that because the action decision is made by the perceptron, the liquid may not necessarily know what actions were performed.

We then tested both agents by applying a strong action on the cart-pole system while they were balancing the pole, making the problem considerably

more difficult as the agent has to balance the pole and stop the cart falling off the edge of the platform.

This force was applied for 5 ticks at a random interval between 50 ticks to 150 ticks of elapsed simulation. We performed this experiment to evaluate if there was any benefit for the increased information transferal of the imperfect synapses. We hypothesised that the imperfect synapses would recover from the external action more quickly without failure, or at least with less failure.

We can see that in Figure 5.7 that this problem was considerably harder to balance. We found that, on average, the agent with standard synapses could balance the cart-pole system for 175 ticks of simulated time. While the agent with imperfect synapses could only balance for 162 ticks of simulated time, on average. We found that the agent with the *imperfect synaptic model* did respond more quickly to the action, although as we saw in the previous experiment, the favour of large oscillatory actions resulted in lesser performance and ultimately failure. The agent with standard synapses did not respond as quickly to external forces but was able to balance the system for longer. After 1000 trials, for all 20 repeats of the experiment, the agents did not meet our requirement for a solution to the cart-pole problem.

5.9 Experiment: Doping Networks

While the *imperfect synaptic model* did solve the cart-pole problem, its fitness was poor in comparison to the agent with standard synapses. Although the imperfect synaptic model had an increase in response time due to external forces when compared to the network with standard synapses. We posed that it is possible to have the benefits of the imperfect synapses without the

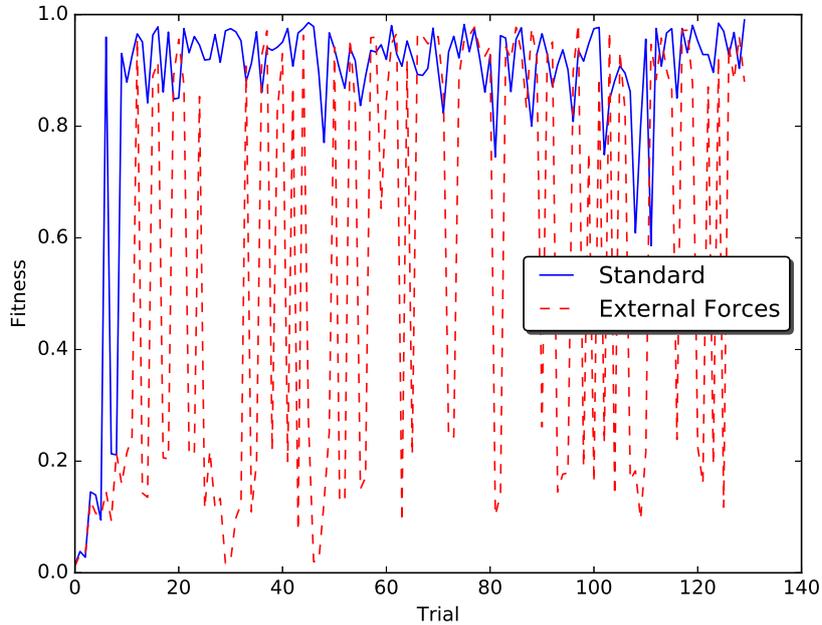


Figure 5.8: The fitness function for an agent doped with imperfect synapses, for both the cart-pole problem with and without external forces.

reduced fitness function. We propose a method called *doping*, which replaces some synapses within a network with imperfect synapses with random imperfectness factors. We repeated the above experiments where a synapse has a 50% probability of being an imperfect synapse, and where the imperfectness factor was a uniform random number between 0 and 0.1.

We can see in Figure 5.8 that the doped agent was able to solve the standard cart-pole problem, where the average fitness was closer but not the same as the agent with standard synapses. We discovered that, on average, the agent learnt to solve the cart-pole problem in half the time compared to the other agents. We suspect this is due to imperfect synapses reducing the spiking potential of some neurons closer to the input layer while causing neurons further in the liquid to spike more often, increasing the separation potential.

We also saw that the doped agent was not able to successfully balance the cart-pole problem under external forces, having an average time of 178 ticks before failure.

5.10 Discussion and Summary

We proposed the *imperfect synaptic model* that represents the flow of ions based on differences in membrane potential allowing for spontaneous neurotransmission. We see that the imperfect synapses have a lower fitness function as compared to the standard model in the control problem of balancing a pole on a cart, and on its own did not offer any benefit over standard synaptic models. We discovered that by *doping* a network with imperfect synapses the cart-pole system learnt to solve the problem significantly faster. We suspect this is due to the imperfect synapses increasing the separation potential. This is likely by reducing the spiking potential of some neurons closer to the input and increasing the potential of neurons deeper within the network. This decreases the chance of repeated structures occurring within the network. Computationally, the imperfect synaptic model requires more computation per propagation cycle, and at an implementation level a redesign is necessary to allow for the different events a neuron exhibits than just activation potential.

Chapter VI

Summary

In this chapter we summarise the works in this thesis, conclude what has been learnt and state new questions that arose.

6.1 Thesis Summary

We proposed an ontology of three main classes, which were neurons, connections and messages. This ontology offers a distinct separation of the simulated networks from its biological counterpart, and generalises classes to encompass a broader range of functionality. Where neurons are still computational units, connections represent any form of relationship of informational connection between neurons where the standard synapse is a subclass, and messages encapsulate communications such as action potentials and our proposed imperfect current.

We proposed a standardised modular framework consisting of a Spiking State Machine. We evaluated the consequences of the design against the nonlinear *exclusive or* problem and a nonlinear three-dimensional hypercube due to their unimportance in biological systems. We determined that for problems that do not require a complex liquid component, the Spiking State Machine is useful for the comparison of spiking neural network models. We found that this liquid component does pose some restrictive consequences on more challenging problems, as there is a reliance on some subnetwork(s) appearing

within the liquid, to find a solution. To obtain this subnetwork we proposed three options: grow the size of the liquid. Design the structure of the liquid. Finally, adjust the readout layer to include pseudo-liquid neurons. All the options given have consequences such that; it may be unfeasible to grow the liquid, designing the liquid may not be possible with physical networks, and the pseudo-liquid layer requires a deeper understanding of the possible input state spaces. Designing the liquid offers a better explanation of dynamics and features of the studied models, but requires a deeper understanding of how they interact in a network. Therefore we cannot say that we have met our simplicity sub-goal for our standardised experimental framework, but we have learnt that the range of possible problems extends beyond temporal problems.

Using the modular framework, we proposed a novel synaptic model that was designed upon spontaneous neurotransmission in the brain, called imperfect synapses. This model was designed to create a flow of current between neurons induced by differences in membrane potential, due to the ion distributions around them. Our proposed model in most situations only modelled spontaneous neurotransmission events that rely upon previous stimulation, rather than all forms of spontaneous neurotransmission. We found, by comparing this method to the standard synaptic model, that there was no significant difference when solving the Markovian cart-pole problem. We then repeated the experiment and applied some external force to the system during simulation such that the agent had to adjust for the force. We found that the imperfect synapses responded to the external force quicker but were unable to solve the problem. We then proposed a novel technique called doping that places impurities in the form of imperfect synapses into the network. We discovered that the doped network learnt to solve the cart-pole problem in half the number of trials, when compared to the networks with the standard

synaptic model and networks with only imperfect synapses.

6.1.1 Thesis Contributions

In summary our main contributions in this thesis are:

- A standardised modular framework called the Spiking State Machine.
- The Imperfect Synaptic Model.
- A technique called *doping* for spiking neural networks.

We analysed the *Spiking State Machine* as a standard modular framework for spiking neural network development, by applying it to nonlinear and non-temporal problems. The sub-goal of simplicity of use, where a network does not have to be explicitly designed was not met. We found with a three-dimensional nonlinear hypercube, the design of the liquid component needed consideration, where we offered three solutions. The *imperfect synaptic model* offered new network dynamics, which alone did not seem significant. However, the technique of *doping* a network, which replaces some synapses with imperfect synapses, did offer significant results. We present empirical observations that doping a network improves upon other network configurations by offering faster learning of the cart-pole problem.

This thesis provides a secondary contribution of a *Spiking Neural Network Ontology* which grew from our design process and the requirements of our framework. The ontology expands upon ways of describing communication and relationships between neurons, and makes a clear separation between terminology describing biological and artificial networks, while outlining a practical implementation strategy. This ontology allows for the description of the communication mechanisms of imperfect synapses in spiking neural networks.

6.1.2 *Future Work*

Some future questions to address are:

- Is it possible to grow artificial brain tissue, with deterministic size and connectivity? Making Spiking State Machine more feasible for these liquid models.
- Does spiking timing dependent plasticity improve solutions to the non-linear hypercube problems, and do we see an effect similar to the pseudo-liquid neurons? Nullifying the need for other learning methods that defeat the purpose of the Spiking State Machine.
- Is there a way to judge the size of the liquid component based upon problem complexity, and what types of complexity contribute to liquid growth?
- What liquid would allow the Spiking State Machine to solve the non-Markovian cart-pole problem?
- What is the optimal imperfection factor, and does this change with doping rates? What is the optimal doping strategy, and does the improved learning rate translate to other problems?
- How do imperfect synapses perform in spatially defined networks, and how does distance impact this? How do imperfect connections without synapses affect results?

References

- [1] Larry F Abbott and Sacha B Nelson. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3:1178–1183, 2000.
- [2] Christian Albers, Maren Westkott, and Klaus Pawelzik. Perfect associative learning with spike-timing-dependent plasticity. In *Advances in neural information processing systems*, pages 1709–1717, 2013.
- [3] Per Andersen. A prelude to long-term potentiation. *Philosophical Transactions-Royal Society of London Series B Biological Sciences*, 358:613–616, 2003.
- [4] Charles W Anderson. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9(3):31–37, 1989.
- [5] P Arena, S De Fiore, L Patané, M Pollino, and C Ventura. Stdp-based behavior learning on the tribot robot. In *SPIE Europe Microtechnologies for the New Millennium*, pages 736506–736506. International Society for Optics and Photonics, 2009.
- [6] Deniz Atasoy, Mert Ertunc, Krista L Moulder, Justin Blackwell, Chi-Hye Chung, Jianzhong Su, and Ege T Kavalali. Spontaneous and evoked glutamate release activates two populations of nmda receptors with limited overlap. *The Journal of Neuroscience*, 28(40):10151–10166, 2008.
- [7] Evyatar Av-Ron, John H Byrne, and Douglas A Baxter. Teaching

- basic principles of neuroscience with computer simulations. *Journal of Undergraduate Neuroscience Education*, 4(2):A40–A52, 2006.
- [8] Leemon Baird et al. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pages 30–37, 1995.
- [9] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience*, 18(24):10464–10472, 1998.
- [10] Tim VP Bliss and Terje Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356, 1973.
- [11] Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1):17–37, 2002.
- [12] Christoph Börgers and Nancy Kopell. Synchronization in networks of excitatory and inhibitory neurons with sparse, random connectivity. *Neural computation*, 15(3):509–538, 2003.
- [13] Johan Borglin. Classification of hand movements using multi-channel emg. 2012.
- [14] Stephen Boyd and Leon Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions on circuits and systems*, 32(11):1150–1161, 1985.

- [15] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [16] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.
- [17] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.
- [18] Harald Burgsteiner, Mark Kröll, Alexander Leopold, and Gerald Steinbauer. Movement prediction from real-world images using a liquid state machine. *Applied Intelligence*, 26(2):99–109, 2007.
- [19] Kyriakos C Chatzidimitriou and Pericles A Mitkas. A neat way for evolving echo state networks. In *ECAI*, pages 909–914, 2010.
- [20] Chris73. Action potential. https://commons.wikimedia.org/wiki/File:Action_potential.svg, 2007. [Online; accessed 17-September-2016].
- [21] Rui P Costa and P Jesper Sjöström. One cell to rule them all, and in the dendrites bind them. *Frontiers in synaptic neuroscience*, 3, 2011.
- [22] BC Coutinho, Sungryong Hong, Kim Albrecht, Arjun Dey, Albert-László Barabási, Paul Torrey, Mark Vogelsberger, and Lars Hernquist. The network behind the cosmic web. *arXiv preprint arXiv:1604.03236*, 2016.
- [23] Alessandro Cristini, Mario Salerno, and Gianluca Susi. A continuous-time spiking neural network paradigm. In *Advances in Neural Net-*

- works: Computational and Theoretical Issues*, pages 49–60. Springer, 2015.
- [24] John F Cryan and Timothy G Dinan. Mind-altering microorganisms: the impact of the gut microbiota on brain and behaviour. *Nature reviews neuroscience*, 13(10):701–712, 2012.
- [25] Henry Dale. Pharmacology and nerve-endings. *Journal of the Royal Society of Medicine*, 28(3):319–332, 1935.
- [26] Alexandre Devert, Nicolas Bredeche, and Marc Schoenauer. Unsupervised learning of echo state networks: A case study in artificial embryogeny. In *Artificial Evolution*, pages 278–290. Springer, 2008.
- [27] Kenji Doya. *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.
- [28] Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh Rao. Bifurcations in the learning of recurrent neural networks 3. *learning (RTRL)*, 3:17, 1992.
- [29] Serena M Dudek and Mark F Bear. Homosynaptic long-term depression in area ca1 of hippocampus and effects of n-methyl-d-aspartate receptor blockade. *Proceedings of the National Academy of Sciences*, 89(10):4363–4367, 1992.
- [30] Thomas Dunwiddie and Gary Lynch. Long-term potentiation and depression of synaptic responses in the rat hippocampus: localization and frequency dependency. *The Journal of Physiology*, 276(1):353–367, 1978.
- [31] Sergei Dytckov, Masoud Daneshtalab, Mojtaba Ebrahimi, Hafeez

- Anwar, Juha Plosila, and Hannu Tenhunen. Efficient stdp micro-architecture for silicon spiking neural networks. In *Digital System Design (DSD), 2014 17th Euromicro Conference on*, pages 496–503. IEEE, 2014.
- [32] Idongesit Ebong, Durgesh Deshpande, Yalcin Yilmaz, and Pinaki Mazumder. Multi-purpose neuro-architecture with memristors. In *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, pages 431–435. IEEE, 2011.
- [33] John Eccles. From electrical to chemical transmission in the central nervous system. *Notes and records of the Royal Society of London*, pages 219–230, 1976.
- [34] Andreas K Fidjeland, Etienne B Roesch, Murray P Shanahan, and Wayne Luk. Nemo: a platform for neural modelling of spiking neurons using gpus. In *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on*, pages 137–144. IEEE, 2009.
- [35] Andreas K Fidjeland and Murray P Shanahan. Accelerated simulation of spiking neural networks using gpus. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.
- [36] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
- [37] Răzvan V Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.
- [38] Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Rein-

- forcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Comput Biol*, 9(4):e1003024, 2013.
- [39] David Gamez, Zafeirios Fountas, and Andreas K Fidjeland. A neurally controlled computer game avatar with humanlike behavior. *Computational Intelligence and AI in Games, IEEE Transactions on*, 5(1):1–14, 2013.
- [40] Samanwoy Ghosh-Dastidar and Hojjat Adeli. A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. *Neural Networks*, 22(10):1419–1431, 2009.
- [41] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.
- [42] Adrian W Gilmore, Steven M Nelson, and Kathleen B McDermott. A parietal memory network revealed by multiple mri methods. *Trends in cognitive sciences*, 2015.
- [43] Cornelius Glackin, Liam Maguire, Liam McDaid, and Heather Sayers. Receptive field optimisation and supervision of a fuzzy spiking neural network. *Neural Networks*, 24(3):247–256, 2011.
- [44] Dan FM Goodman and Romain Brette. The brian simulator. *Frontiers in neuroscience*, 3:26, 2008.
- [45] Frédéric Gruau, Darrell Whitley, and Larry Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the 1st annual conference on genetic programming*, pages 81–89. MIT Press, 1996.

- [46] Beata J Grzyb, Eris Chinellato, Grzegorz M Wojcik, and Wieslaw A Kaminski. Which model to use for the liquid state machine? In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 1018–1024. IEEE, 2009.
- [47] Anirudh Gupta, Yun Wang, and Henry Markram. Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex. *Science*, 287(5451):273–278, 2000.
- [48] David Heeger. Poisson model of spike generation. *Handout, University of Stanford*, 5, 2000.
- [49] 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.
- [50] Yan-You Huang, Eleanor Simpson, Christoph Kellendonk, and Eric R Kandel. Genetic evidence for the bidirectional modulation of synaptic plasticity in the prefrontal cortex by d1 receptors. *Proceedings of the national academy of sciences of the United States of America*, 101(9):3236–3241, 2004.
- [51] Yanping Huang and Rajesh P Rao. Neurons as monte carlo samplers: Bayesian inference and learning in spiking networks. In *Advances in neural information processing systems*, pages 1943–1951, 2014.
- [52] Richard Ivey, Daniel Bullock, and Stephen Grossberg. A neuromorphic model of spatial lookahead planning. *Neural Networks*, 24(3):257–266, 2011.
- [53] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

- [54] Eugene M Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.
- [55] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.
- [56] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002.
- [57] Herbert Jaeger. Controlling recurrent neural networks by conceptors. 2014.
- [58] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.
- [59] Nikola K Kasabov. Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*, 52:62–76, 2014.
- [60] Taras Kowaliw, Nicolas Bredeche, Sylvain Chevallier, and René Dourzat. Artificial neurogenesis: An introduction and selective review. In *Growing Adaptive Machines*, pages 1–60. Springer, 2014.
- [61] Madeline A Lancaster, Magdalena Renner, Carol-Anne Martin, Daniel Wenzel, Louise S Bicknell, Matthew E Hurles, Tessa Homfray, Josef M Penninger, Andrew P Jackson, and Juergen A Knoblich. Cerebral organoids model human brain development and microcephaly. *Nature*,

- 501(7467):373–379, 2013.
- [62] Anna Levina, J Michael Herrmann, and Theo Geisel. Dynamical synapses causing self-organized criticality in neural networks. *Nature physics*, 3(12):857–860, 2007.
- [63] M. Lichman. UCI machine learning repository, 2013.
- [64] John Lisman and Nelson Spruston. Postsynaptic depolarization requirements for ltp and ltd: a critique of spike timing-dependent plasticity. *Nature neuroscience*, 8(7):839–841, 2005.
- [65] Ying-Hui Liu and Xiao-Jing Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10(1):25–45, 2001.
- [66] Terje Lømo. The discovery of long-term potentiation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1432):617–620, 2003.
- [67] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.
- [68] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.
- [69] Wolfgang Maass and Henry Markram. On the computational power of circuits of spiking neurons. *Journal of computer and system sciences*, 69(4):593–616, 2004.
- [70] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural

- computation based on perturbations. *Neural Comput.*, 14(11):2531–2560, November 2002.
- [71] Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 719–726, 2010.
- [72] Olvi L Mangasarian, W Nick Street, and William H Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.
- [73] Henry Markram and Misha Tsodyks. Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature*, 382:807–810, 1996.
- [74] Timothée Masquelier, Rudy Guyonneau, and Simon J Thorpe. Competitive stdp-based spike pattern learning. *Neural computation*, 21(5):1259–1276, 2009.
- [75] Ammar Moheemmed, Stefan Schliebs, Satoshi Matsuda, and Nikola Kasabov. Span: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, 22(04):1250012, 2012.
- [76] Fearghal Morgan, Seamus Cawley, B McGinley, Sandeep Pande, Liam J McDaid, Brendan Glackin, John Maher, and Jim Harkin. Exploring the evolution of noc-based spiking neural networks on fpgas. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 300–303. IEEE, 2009.
- [77] Takashi Nakano, Makoto Otsuka, Junichiro Yoshimoto, and Kenji

- Doya. A spiking neural network model of model-free reinforcement learning with high-dimensional sensory input and perceptual ambiguity. *PloS one*, 10(3):e0115620, 2015.
- [78] Roger A Nicoll and Robert C Malenka. Leaky synapses. *Neuron*, 23(2):197–198, 1999.
- [79] DP Pelvig, H Pakkenberg, AK Stark, and B Pakkenberg. Neocortical glial cell numbers in human brains. *Neurobiology of aging*, 29(11):1754–1762, 2008.
- [80] Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *The Journal of neuroscience*, 26(38):9673–9682, 2006.
- [81] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation*, 18(6):1318–1348, 2006.
- [82] Ning Quan and William A Banks. Brain-immune communication pathways. *Brain, behavior, and immunity*, 21(6):727–735, 2007.
- [83] Alexander Rauch, Giancarlo La Camera, Hans-Rudolf Lüscher, Walter Senn, and Stefano Fusi. Neocortical pyramidal cells respond as integrate-and-fire neurons to in vivo-like input currents. *Journal of neurophysiology*, 90(3):1598–1612, 2003.
- [84] Banafsheh Rekabdar, Monica Nicolescu, and Mircea Nicolescu. An unsupervised learning approach for classifying sequence data for human robotic interaction using spiking neural network. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot*

- Interaction Extended Abstracts*, pages 213–214. ACM, 2015.
- [85] Martin Riedmiller. Neural fitted q iteration—first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.
- [86] Martin Riedmiller, Jan Peters, and Stefan Schaal. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 254–261. IEEE, 2007.
- [87] Sebastian Risi and Kenneth O Stanley. Indirectly encoding neural plasticity as a pattern of local rules. In *From Animals to Animats 11*, pages 533–543. Springer, 2010.
- [88] Cyrille Rossant, Dan FM Goodman, Bertrand Fontaine, Jonathan Platkiewicz, Anna K Magnusson, and Romain Brette. Fitting neuron models to spike trains. *Frontiers in neuroscience*, 5:9, 2011.
- [89] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [90] Tod Russell. Spiking neurons models for control and evolutionary robotics. 2005.
- [91] Benjamin Schrauwen and Jan Van Campenhout. Extending spikeprop. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1. IEEE, 2004.
- [92] Wolfram Schultz. Getting formal with dopamine and reward. *Neuron*, 36(2):241–263, 2002.

- [93] Jeremy K Seamans and Charles R Yang. The principal features and mechanisms of dopamine modulation in the prefrontal cortex. *Progress in neurobiology*, 74(1):1–58, 2004.
- [94] H Sebastian Seung. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073, 2003.
- [95] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.
- [96] Bruce Sherin. Common sense clarified: The role of intuitive knowledge in physics problem solving. *Journal of research in science teaching*, 43(6):535–555, 2006.
- [97] Jesper Sjöström and Wulfram Gerstner. Spike-timing dependent plasticity. *Spike-timing dependent plasticity*, page 35, 2010.
- [98] Per Jesper Sjöström, Gina G Turrigiano, and Sacha B Nelson. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6):1149–1164, 2001.
- [99] Laura J Solomon and Esther D Rothblum. Academic procrastination: Frequency and cognitive-behavioral correlates. *Journal of Counseling psychology*, 31(4):503, 1984.
- [100] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.
- [101] Roger W Sperry. Cerebral organization and behavior. *Science*,

- 133(3466):1749–1757, 1961.
- [102] Ioana Sporea and André Grüning. Supervised learning in multilayer spiking neural networks. *Neural computation*, 25(2):473–509, 2013.
- [103] Olaf Sporns, Dante R Chialvo, Marcus Kaiser, and Claus C Hilgetag. Organization, development and function of complex brain networks. *Trends in cognitive sciences*, 8(9):418–425, 2004.
- [104] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.
- [105] Min D Tang-Schomer, James D White, Lee W Tien, L Ian Schmitt, Thomas M Valentin, Daniel J Graziano, Amy M Hopkins, Fiorenzo G Omenetto, Philip G Haydon, and David L Kaplan. Bioengineered functional brain-like cortical tissue. *Proceedings of the National Academy of Sciences*, 111(38):13811–13816, 2014.
- [106] Christiane M Thiel, Karl J Friston, and Raymond J Dolan. Cholinergic modulation of experience-dependent plasticity in human auditory cortex. *Neuron*, 35(3):567–574, 2002.
- [107] Alex M Thomson, David C West, Yun Wang, and A Peter Bannister. Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2–5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral cortex*, 12(9):936–953, 2002.
- [108] M. Thon and H. Jaeger. Links between multiplicity automata, observable operator models and predictive state representations - a unified learning framework. *Journal of Machine Learning Research*, 16:103–

147, 2015.

- [109] Edward L Thorndike. Animal intelligence: an experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i, 1898.
- [110] Todd W Troyer and Kenneth D Miller. Physiological gain leads to high isi variability in a simple model of a cortical regular spiking cell. *Neural Computation*, 9(5):971–983, 1997.
- [111] Misha Tsodyks, Klaus Pawelzik, and Henry Markram. Neural networks with dynamic synapses. *Neural computation*, 10(4):821–835, 1998.
- [112] Misha Tsodyks, Asher Uziel, Henry Markram, et al. Synchrony generation in recurrent networks with frequency-dependent synapses. *J Neurosci*, 20(1):825–835, 2000.
- [113] David Verstraeten, Benjamin Schrauwen, Dirk Stroobandt, and Jan Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.
- [114] Catherine R Wasser and Ege T Kavalali. Leaky synapses: regulation of spontaneous neurotransmission in central synapses. *Neuroscience*, 158(1):177–188, 2009.
- [115] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [116] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.
- [117] Grzegorz M. Wojcik and Wieslaw A. Kaminski. Liquid state machine

- built of hodgkinhuxley neurons and pattern recognition. *Neurocomputing*, 5860:245 – 251, 2004. Computational Neuroscience: Trends in Research 2004.
- [118] Grzegorz M Wojcik and Wieslaw A Kaminski. Liquid state machine and its separation ability as function of electrical parameters of cell. *Neurocomputing*, 70(13):2593–2597, 2007.
- [119] Xiaohui Xie and H Sebastian Seung. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69(4):041909, 2004.
- [120] Yan Xu, Xiaoqin Zeng, Lixin Han, and Jing Yang. A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks*, 43:99–113, 2013.
- [121] Tadashi Yamazaki and Shigeru Tanaka. The cerebellum as a liquid state machine. *Neural Networks*, 20(3):290–297, 2007.
- [122] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.
- [123] Jian-Hua Zhang, Valarie A Barr, Yinyuan Mo, Alexandra M Rojkova, Shaohua Liu, and William F Simonds. Nuclear localization of g protein $\beta 5$ and regulator of g protein signaling 7 in neurons and brain. *Journal of Biological Chemistry*, 276(13):10284–10289, 2001.