# SpikeSynRL: Synaptic Level Reinforcement Learning-Derived Plasticity in Spiking Neural Networks

by

Aman Bhargava

B.A.Sc., Engineering Science, Machine Intelligence Option

Submitted to the University of Toronto Faculty of Engineering Science
in partial fulfillment of the requirements for the degree of

B.A.Sc., Engineering Science, Machine Intelligence Option

at the

UNIVERSITY OF TORONTO

May 2022



Author .....
University of Toronto Faculty of Engineering Science
April 7, 2022

Certified by .....
Dr. Milad Lankarany, Thesis Supervisor
Assistant Professor, Institute of Biomedical Engineering (BME)

# SpikeSynRL: Synaptic Level Reinforcement Learning-Derived Plasticity in Spiking Neural Networks

by

Aman Bhargava

Submitted to the University of Toronto Faculty of Engineering Science
on April 7, 2022, in partial fulfillment of the
requirements for the degree of
B.A.Sc., Engineering Science, Machine Intelligence Option

## Abstract

Biological neural information processing is an extremely efficient and effective mode of computation that remains poorly understood. An ongoing challenge in neural information processing is: how do neural networks learn? Specifically, how does an individual synapse infer its role in producing network-level behaviour and adjust itself accordingly to maximize some reward signal? While robust learning rules based on gradient descent have been used extensively in deep learning neural network models to great effect, they remain biologically implausible, costly, and non-applicable to neuron models that make use of spatio-temporal spikes to propagate and compute information.

Here we propose and validate a reinforcement learning methodology to infer local information-driven biologically plausible learning rules for spiking neural networks. In this formulation, synapses are modelled as a reinforcement learning agent executing the same policy (learning rule) and the action space for each synapse consists of increasing, decreasing, or taking null action on the synapse connectivity strength.

In this work, we present a series of theoretical arguments (computationally and biologically motivated) as well as full proposal and results for this phase of the investigation. Overall, we reveal strong results for the first phase of this research program in the tabular Q-learning case. We gain an understanding of the applicability of the original SynRL methodology, STDP features, and message passing, and accordingly reprioritize the plans for the remainder of the program.

Thesis Supervisor: Dr. Milad Lankarany
Title: Assistant Professor, Institute of Biomedical Engineering (BME)

THIS PAGE INTENTIONALLY LEFT BLANK

# Contents

# List of Figures

# List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 1

# Introduction

## 1.1 Motivation

Biological information processing is an incredibly efficient and effective mode of computation. Using only 20 Watts of power [16], the human brain is able to vastly outperform current machine intelligence systems in terms of data efficiency, power efficiency, adaptability, and performance in a variety of tasks [19, 46]. While current state-of-the-art deep learning algorithms are based, in part, on models of biological neural information processing [65, 64], it remains poorly understood how biological neural networks learn so effectively at scale [44, 63, 78, 3]. Understanding self organization and learning in biological systems is therefore of high utility in both neuroscience and machine intelligence as it may enable the creation similarly efficient and effective synthetic computational systems.

## 1.2 Project Overview

In this work, I apply and augment the reinforcement learning methodology for inferring learning rules in neural networks from my previous work [11]. Specifically, I apply the methodology to leaky integrate-and-fire (LIF) spiking neural networks (SNNs) [1], a more complex class of neural networks where robust biologically feasible training methods have yet to be created, particularly for large networks [59, 45, 80]. Given the

proof-of-concept nature of the results from my previous work [11], I also attempt to augment the method, increasing the reinforcement learning algorithms' state and action complexity to improve training efficiency. Overall, I aim to generate a robust, efficient, biologically-feasible learning rule for spiking neural networks using the reinforcement learning methodology from [11].

In this synaptic reinforcement learning methodology, each synapse is treated as a reinforcement learning agent executing the same policy to determine changes in connectivity. A more comprehensive overview of the work in [11] will be given in Section 3.3 along with the relevant reinforcement learning background in Section 3.2. At a high level, the method from [11] is as follows: Each synapse can take *actions* $a \in \mathcal{A}$ to either increase or decrease its connection strengths. It chooses actions based on its *state* $s \in \mathcal{S}$, which consists of locally available information such as the synapse's previous actions and network-level reward signals. The reinforcement learning algorithm aims to learn a function (policy) $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes some overall reward signal (i.e., the network-level reward signal). This mapping represents a synaptic-level learning rule. As a proof of concept, [11] produced a simple and reliable learning rule that was applicable to a wide variety of network sizes and problems.

We seek to augment this methodology by endowing the synapse policy with additional complexity to address the slow (albeit reliable) convergence of the work in [11]. The following additions are investigated in this work on feedforward LIF SNNs:

1. **Hebbian terms:** Synapse state will be augmented with a term representing the pre- and post-synaptic voltage spike correlation.

2. **Message passing:** The action space for each synapse will also include the ability to backpropagate a (biologically feasible) message to the previous layer.

3. **Memory:** Synapses will be given a readable/writable memory within their state space.

4. **Heterogeneity:** Multiple neuron "cell types" will be introduced, each with different reinforcement learning policies and behaviours.

Rigorous explanation and justification for each of these additions will be included in Section 3.4.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 2

# Background

In this chapter, relevant work to date is summarized on the topics of biological and artificial neural networks and learning rules. These works form the basis for the theoretical arguments for the proposed study and for the practical design decisions discussed in Chapter 3.

## 2.1 Biological Neural Networks

Biological neurons are electrochemically active cells with membrane properties that enable electrical impulse propagation and computation [60, 13]. Neurons are composed of a cell body, a dendritic tree, and an axon. Electrochemical stimulation is generally received at the dendritic tree where signals passively spread toward the cell body. Incoming signals are spatially and temporally summed at the cell body. If a threshold potential is reached, an action potential is initiated at the cell body and spreads along the axon through "active transmission".

The topology of biological neural networks is generally highly recurrent (i.e., there exist synaptic transmission pathways that return to the same neurons) [71]. Networks with no recurrence are referred to as "feed-forward" networks [71]. In general, there is a high degree of noise in biological neural networks, with large populations of neurons often employed representing information robustly [39, 6].

There are two primary regimes for information coding in neural networks: rate

coding and temporal coding [28]. In rate coding, information is primarily encoded as the *frequency of action potentials*. An example of this type of coding is found in muscular sensory innervation. As far back as 1926 [4], it was observed that the rate of firing of sensory nerves connected to a stretched muscles increased with the force stretching the muscle. Thus it was concluded that the rate of firing was operative for encoding muscular sensory signals.

Temporal coding, meanwhile, holds that the precise timing between action potentials carries information [24]. This model explains the ability for the brain to process very short time duration stimuli that have ill-defined rates of firing. In general, more information is transmissible via temporal coding than rate coding as many spike trains with the same overall rate can have vastly different inter-spike distances [29, 24].

Depending on the brain region and system in question, it appears that analysis through both coding perspectives may be useful [72].

## 2.2 Artificial Neural Networks

The dynamics of biological neural networks are highly nuanced. Ion fluxes and channel gating responsible for information propagation is governed by a number of competing factors in a highly nonlinear set of partial differential equations [60]. Moreover, heterogeneity in biological neural systems is widespread [34], making for wildly different behaviour in different brain regions. Simulating the behaviour of an individual neuron quickly becomes challenging as a result of the many voltage- and chemically-gated channels across the surface of the cell and the many ionic species that drive cell behaviour [60, 13]. Like in many biological systems, there is complexity to be found in nearly every part of neural systems, from the behaviour of synaptic integration to stimulation accommodation. As a result of this underlying complexity, there exist a variety of computational and mathematical models for neuron behaviour. Each model makes trade-offs by simplifying certain characteristics of neural network dynamics in exchange for improved computational efficiency.

**Hodgkin-Huxley Model:** Hodgkin-Huxley–based neuron models are a gold standard for modelling the electrochemical properties of neuron membranes [60]. Under this model, voltage- and chemically-gated ion channels are modelled through population dynamics, and there is a high degree of influence between each ion type as they all effect the total membrane voltage. Synapse modelling in networks of such neurons is also nuanced, and the geometry of each neuron's axon and dendritic tree can be made arbitrarily complicated, particularly when using microscopy data to inform a simulation [13]. Integrating this set of partial differential equations, especially across complicated and networked cell geometries, is extremely computationally expensive.

**Integrate-and-Fire Models:** The variety of ionic species and interactions in Hodgekin-Huxley–based neuron models is the primary difficulty that integrate-and-fire models aim to simplify. The "perfect integrate-and-fire" neuron model, for instance, is one of the earliest neuron models where an input current causes a monotonic increase in neuron membrane voltage until a threshold is reached and an action potential (delta function spike) is initiated and the membrane voltage is reset [1]. This model is extremely simple and correspondingly computationally inexpensive to simulate at scale. However, it does not display refractory properties, accommodation, passive diffusion toward equilibrium potential, or adaptation to long-term stimulus. Accordingly, there are a number of integrate-and-fire descendant models that incorporate these properties in an effort to increase the realism of simulations while not substantially increasing the expensiveness of the simulation. For instance, leaky integrate-and-fire (LIF) models incorporate passive diffusion toward equilibrium potential, making them a popular investigative tool for understanding network-level properties of neural networks [26, 82]. Integrate-and-fire models can also notably exhibit both temporal and rate-based coding discussed in Section 2.1.

**Rate-Based Models:** While there are some computational tricks for more efficiently simulating certain network configurations of integrate-and-fire models (e.g., convolutional kernels for LIF feed forward networks in [80]), they remain expensive

and mathematically challenging to analyze and somewhat expensive to simulate on conventional (von Neumann) computer architectures. Rate-based models can be viewed as the next level of abstraction from the biological reality [14]. In this model, it is assumed that rate coding is the operative information propagation method. The firing rate of each neuron is represented by a scalar value, and downstream neurons take the weighted sum of each input neuron's firing rate. An activation function on this sum is applied to cap the maximum and minimum firing rates (firing rate cannot be negative and cannot exceed a maximum rate informed by the neuron's absolute refractory time between spikes) [27]. In feed-forward networks, the weighted sum can be expressed conveniently as a matrix product between the previous layers' neuron outputs and a matrix of the synaptic weights connecting to the subsequent layer of neurons. This forms the basis for most machine learning neural network models. As discussed in Section 2.3.2, the matrix multiplication enables the use of differential calculus and related optimization techniques on the network [64], as well as relatively efficient computer simulations.

**Stochastic Models:** Noise and stochasticity appears to be central to biological information processing [34]. While there are a number of stochastic neural network models that are designed to exemplify this property [74], they are not of high relevance to the work of this thesis. The incorporation of stochasticity into the proposed model is discussed in Section 3.4.

## 2.3 Theories of Learning in Neural Networks

Biological neural networks are composed of many interconnected neurons [13]. Connections between neurons are known as synapses, with the majority of synapses connecting the axon of a presynaptic cell to the dendrites or cell body of a postsynaptic cell [66, 42]. Depending on the brain region, connectivity can be quite dense [30], with each neuron having thousands of synaptic connections each influencing the activity of surrounding neurons. Biological neural networks also tend to be quite recurrent [71]

with a high degree of heterogeneity [34] in terms of cell type. It is widely believed that learning is a function of changes in connectivity between neurons in biological neural networks [13, 41, 20], and that network behaviour of a network is largely governed by this connectivity.

The notion of learning as synaptic weight updates is the prevailing in both biologically-informed and computationally-informed theories of learning in neural networks [20, 63, 44, 20, 78, 13, 47]. In this section, major ideas and approaches in both are surveyed. The line between biologically- and computationally-informed theories and models can be blurred at times, but it is useful to differentiate such theories when discussing the engineering of biologically-feasible explanations for learning.

### 2.3.1 Biologically-Informed Theories of Learning

As understanding learning is one of the central problems in the field of neuroscience, substantial effort has been made to experimentally derive properties of learning in neural networks. This section is focused on experimentally-derived findings from biological studies of learning and plasticity in the brain.

**Hebbian learning:** One of the earliest findings on learning in biological neural networks was made by Hebb in 1949 [31], who observed that synaptic connectivity increased when the presynaptic cell repeatedly stimulated the postsynaptic cell. In other words, "cells that fire together wire together" [43]. A more advanced version of this learning rule is spike time-dependent plasticity (STDP) [17], which is a temporally asymmetrical version of Hebbian learning. In STDP, the exact timing of pre- and post-synaptic spiking is of great importance in determining the increase or decrease in synaptic connectivity. If presynaptic spikes occurs very shortly before postsynaptic spikes, the connectivity increases dramatically. However, if the opposite occurs, the connectivity decreases. These learning rules on their own are notably unstable when implemented as a simulation, but offer effective explanations of observations of synaptic plasticity from experiments. Subsequent Hebbian learning rules such as the BCM learning rule [37] attempt to improve stability.

**Reward correlation learning:** An issue that arises with purely Hebbian learning rules is that they do not incorporate any sort of "reward signal". It is unclear how such learning rules would lead to the reward-optimized behaviour that is readily observed in animals and humans alike. Early theories on reward-driven synaptic plasticity suggested that each synapse would be optimized according to the combination of a Hebbian/STDP-like term and a network-level reward mediated term [75, 52]. This reward mediated term was widely hypothesized to be driven by correlation between changes in a given synapse's weight and the corresponding change in network-level reward signal. However, this approach does not appear to scale well to large network structures [51]. As the total number of synapses increases, the impact of a single synapse weight update has very relatively little traceable effect on the network-level behaviour. Thus, the purely reward correlation-driven methods for assigning "credit" to a given synapse weight update still do not appear to explain credit assignment in biological neural networks.

**Other Biological Observations:** There are a large number of other observations on synaptic plasticity in biological neural networks. A few notable findings include the following:

- **Long-term Potentiation (LTP):** When a presynaptic cell is intensely stimulated in a manner that causes post-synaptic activation, the connectivity between the cells is substantially increased for long time periods (hours-weeks) [21]. This is a Hebbian phenomena that is frequently observed in the hippocampus, a brain region associated with memory and learning. The molecular pathways for this phenomena are fairly well understood, and it is a notable phenomenon for its ability to span timescales: short term intense stimulation causes long-term changes in synaptic connectivity.

- **Eligibility traces:** The notion of a synaptic "eligibility trace" is ubiquitous in studies on synaptic plasticity [63, 25, 51]. Eligibility traces reference how certain synapses appear more "eligible" for changes in weight/plasticity at different times, depending on network events (e.g., synaptic activation, reward signal changes,

etc). This terminology also finds its way into description of computationally-derived learning rules.

- **Myriad neurotransmitter cell signalling pathways:** Presented in [69], there is strong evidence of dozens of distinct modulatory networks in mouse cortical neurons with independent molecular pathways in cortical neurons. This finding has major implications for how information is shared between neurons during learning, as previous studies had often either assumed perfectly symmetrical and specific information backpropagation between neurons, or assumed only a single modulatory dopaminergic reward signal for all neurons. The implications of this work are discussed at length in Section 2.3.2 and in [51].

- **Heterogeneity:** An important observation in biological networks is that there is a high degree of heterogeneity in cell types within networks [34]. This has been hypothesized to aid in learning and credit assignment [69, 51].

### 2.3.2 Computationally-Informed Theories of Learning

In this section, computationally-informed theories and methods of learning in both rate-based and spiking artificial neural networks are surveyed. These approaches are generally motivated by the reward-optimization behavior of neural networks [32, 64, 27]. Credit assignment and synaptic plasticity can be viewed as an optimization problem through this lens: neurons appear to assemble themselves in a way that, in part, maximizes a reward signal. While this does not appear to be the complete story from a neuroscientific point of view (i.e., neurons optimize for and are constrained by their own metabolic limitations), it has proven a useful investigative tool in understanding the dynamics of neural networks. Moreover, this computational/optimization-centric approach has given rise to modern neural networks in deep learning [47, 64].

**Supervised Learning:** Computational theories of learning are often framed as supervised learning problems. Notation for such problems is as follows (from [2, 27]): a dataset $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ holds $N$ training examples $(x, y)$, where $y^{(i)} \in \mathcal{Y}$ is the

desired output for input $x^{(i)} \in \mathcal{X}$. The goal of supervised learning is to infer a robust approximation function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y}$ are vectors in the case of rate-based neural networks and spike trains in spiking neural networks. To quantify the "goodness" of a given function $f$, a loss function $\mathcal{L}(\hat{y}, y)$ is used quantifies the distance between the desired $y$ and the predicted $\hat{y}$. The process of supervised learning is generally framed as an optimization of $\mathbb{E}\{\mathcal{L}(f(x), y)\}$ over the underlying distribution from which dataset $\mathcal{D}$ is drawn. The neural network is used as the function approximator $f$ in supervised neural network training and is parameterized by weights (synaptic connectivities) $\Omega = \{W^{(\ell)}\}_{\ell=0}^{L}$ where $L$ is the number of neural network layers and $W_{ij}^{(\ell)}$ is the synapse strength connecting neuron $j$ in layer $\ell - 1$ to neuron $i$ in layer $\ell$.

**Evolutionary Optimization:** Evolutionary optimization is a technique wherein random "mutations" are introduced to the subject(s) of an optimization, and the most optimal mutants are preserved for the next "generation" where the process repeats [68]. This methodology is largely inspired by Darwin's theory of natural selection, and includes asexual and sexual reproductive variants. In the context of neural network learning, mutations generally correspond to perturbations in a neural network's weights $W$, and the fitness of a given weight set $\Omega$ is given by the loss function $\mathbb{E}_{\mathcal{D}}\mathcal{L}(f_{\Omega}(x), y)$ [53].

Evolutionary optimization of neural networks is typically costly at scale due to the low specificity of positive and negative feedback signals during the optimization [23]. The optimization of many parameters at once in such a configuration is made difficult by this. The cost is compounded by the fact that the fitness function must be evaluated many times over the course of the optimization.

**Perceptron Learning Algorithm & Hopfield Networks:** In the early development of neural network theory, a number of mathematically derived learning rules were derived for particular network types and topologies: chiefly the perceptron learning algorithm and Hopfield networks. The perceptron learning algorithm [62] outlines a simple procedure for learning a classifier for **linearly separable data**.

The "perceptron" refers to an extremely simple rate-based single neuron model consisting of an affine transformation on the input vector followed by a signum function (i.e., $y_{pred} = \text{sign}(w^T x + w_0)$). The learning algorithm notably does not converge on non-linearly separable data.

Hopfield networks [33] are a type of recurrent neural network model that are able to effectively store "memories" (i.e., binary vectors). They are able to "recall" memories in the sense that they progressively de-noise corrupted binary vectors until the most similar memory vector is recovered. The learning rule for Hopfield networks consists of a set of matrix outer products between the "memory" vectors which can be viewed as a form of Hebbian or associative learning.

The work is notable for its direct use of statistical mechanics (specifically the Ising model for magnetic solids [50]) in understanding neural network dynamics. The ability for recurrent networks to store memories in a dynamically stable manner remains an important discovery in computational neuroscience. Hopfield networks are limited in that they do not generalize to multilayer network topologies nor networks with direct self-feedback, and are only capable of memory tasks involving storage and retrieval of memories.

**Gradient Descent & Backpropagation:** Gradient descent is the modern workhorse for the optimization of deep learning neural networks (i.e., rate-based models) [64, 46, 47, 27]. In this technique, the gradient of the network-level loss $\mathcal{L}$ is calculated with respect to each parameter (synapse weight) $w_{ij}$. Synapse weights $w_{ij}^{(\ell)} \in W^{(\ell)} \in \Omega$ are updated using the following equation:

$$w_{ij}^{(\ell)} \leftarrow w_{ij}^{(\ell)} - \epsilon \frac{\partial L}{\partial w_{ij}^{(\ell)}} \tag{2.1}$$

Where $\epsilon$ is some small positive constant, referred to as the learning rate. There are a number of gradient descent-based optimization methods for neural networks and beyond with various heuristic characteristics (e.g., momentum, learning rate adaptation, momentum adaptation, etc.) [22], but they are all fundamentally based

on the use of the loss gradient with respect to each parameter.

Gradient-based learning rules have been extremely successful in optimizing deep learning models/rate-based neural network models [47, 46]. Since the "learning signal" $\partial\mathcal{L}/\partial w_{ij}^{(\ell)}$ is very specific to a given weight $w_{ij}^{(\ell)}$, the assignment of "credit" for network-level behaviour is far more precise than in many of the other methods training neural networks. The conventional method for calculating the gradients $\partial\mathcal{L}/\partial w_{ij}^{(\ell)}$ is through either automatic differentiation [7] or backpropagation [64], both of which boil down to the repeated application of the chain rule (and dynamic programming used to save intermediate results). Full derivations of the backpropagation algorithm can be found in [64, 27]. When dynamic programming is used to save intermediate results in the application of the chain rule, intermediate "error signal" variables arise and appear to be "propagated" backward through the network. This information, sometimes referred to as the "gradient signal", is of the same dimensionality as each layer in the network and is backpropagated via multiplication with the synapse weights.

The reliance on multiplication by synapse weights in the backwards pass is referred to as "symmetric feedback connections". These symmetric connections are one of the primary objections to backpropagation as an explanation for learning in biological neural networks as they are not readily observed in the brain [78]. It is unclear how synapses would instantaneously measure the strength of downstream synapses, or propagate gradient information backwards across synapses. Furthermore, the method relies on differentiability of each network operation [80, 82]. The discontinuities introduced by action potentials in spiking neural networks/biological neural networks makes differentiation infeasible to perform directly. Due to the effectiveness of gradient-based optimization in deep learning neural networks, much work has been done to engineer and understand gradient approximation methods for spiking neural networks and under biologically-realistic information propagation constraints, discussed in the following sections.

**REINFORCE Learning Algorithms & Node Perturbation:** This family of learning algorithms offer a method for statistically inferring synaptic weight gradients

from changes in network loss resulting from noise injected into network activations [79, 51]. If the noise injected into each activation is relatively small and precisely known, a first-order approximation can be made to model the change in the loss. That is, for non-perturbed loss $\mathcal{L}$ and perturbed loss $\tilde{\mathcal{L}}$ caused by activation $h_i$ being injected with noise $\xi_i$, an estimate of $\partial\mathcal{L}/\partial h_i$ can be inferred via the first order approximation:

$$\tilde{\mathcal{L}} \approx \mathcal{L} + \frac{\partial \mathcal{L}}{\partial h_i}\xi_i \tag{2.2}$$

Such node perturbation methods are valuable to keep in mind in the investigation of subsequent learning rules. For instance, the work in [11] can be viewed as a reinforcement learning-inferred approximation of this algorithm. Limitations of this family of learning rules similarly revolve around the non-specificity of feedback signals. As the number of trainable parameters increase, it becomes difficult to determine the "eligibility" of a single parameter (synapse) within the first-order approximation. Attempts have been made to incorporate ideas of feedback alignment [49] as in [44, 80], and will be discussed in the subsequent sections.

**Surrogate Gradient Methods (SNNs):** Due to the non-differentiability of discontinuities caused by spiking behaviour in SNNs, gradient calculations are infeasible via direct application of backpropagation [59]. The temporal aspects of spike-based information propagation further complicates the calculation of loss gradients [67, 80, 51]. One method to address this, particularly in feed forward spiking neural networks, is via surrogate gradient computations [82, 67, 80, 56]. SNNs are rendered differentiable by approximating the discontinuities with differentiable replacement functions during backpropagation. Alternatively, stochastic spiking neural networks may model neuron behaviour as a probability distribution of spiking rather than directly as a function of membrane potential. The continuous probability distributions enable differentiation despite the apparent discontinuity caused by spikes, known as "expectation backpropagation" [70]. However, this method is computationally expensive and requires a high degree of stochasticity to generate strong gradient signals.

A more common method is to replace the non-differentiable term in the gradient calculation corresponding to the derivative of the spike train $S_i$ with respect to synapse weight $w_{ij}$ with an auxillary function on the membrane potential $U_i$. As in SuperSpike [80]:

$$\frac{\partial S_i}{\partial w_{ij}} \rightarrow \sigma'(U_i)\frac{\partial U_i}{\partial w_{ij}} \tag{2.3}$$

Where $\sigma()$ is a continuous function on the membrane potential that mimics properties of the spiking discontinuity. Importantly, $\sigma(U_i)$ must be near zero until $U_i$ approaches the spiking threshold, at which point it increases rapidly (but still continuously).

Surrogate gradient methods are a promising method for training spiking neural networks [82]. Notably, these methods still rely on symmetric feedback connections to compute the gradient. While some work has been done in an attempt to relax this requirement (e.g., in [80]) by leveraging principles of feedback alignment [49], symmetric feedback still appears necessary for strong network convergence in large ensembles of spiking neurons.

**Conversion of ANNs to SNNs:** A popular method for "training" spiking neural networks is to first train a rate-based/non-spiking artificial neural network (ANN) and then use the learned weights in a spiking neural network [59]. This technique takes advantage of the effectiveness of gradient descent-based optimization for rate-based neural networks and applies it to spiking neural networks. The approach usually aims to have the rate code of the ANN be reflected in the firing rate of the SNN units. While the approach has achieved success in training SNNs to perform a number of tasks, there are substantial drawbacks to the method. Computation on temporal codes is not easily achieved through this method since the conversion aims to still use the same rate codes as the ANN, just in a spiking neural network. The potential advantages of spike-based computation, therefore, are not fully actualized in terms of power efficiency, computational efficiency, temporally prioritized computation, etc.

**Feedback alignment:** Lillicrap's 2016 paper on feedback alignment [49] was highly influential on subsequent work on learning rules as it showed that **random feedback matrices** in backpropagation can still yield effective neural network training in certain situations. That is, in the calculation of loss gradients via backpropagation, the use of random feedback connections (random feedback matrices instead of the actual synaptic weight matrices) can still yield effective network convergence in rate-based neural networks.

While this finding applied best to small neural networks, it had strong implications for biologically feasible relaxations of the backpropagation algorithm. It has since heavily inspired work on biologically feasible neural network learning both in rate coding and spike coding neural networks.

**Cell-type–specific neuromodulation:** One of the key remaining problems with many learning rules that rely on "biologically feasible" backpropagation relaxations is that there is still high specificity of feedback signals. While the feedback signals may be calculated via surrogate gradients or using node perturbation (REINFORCE, [79]) or using non-symmetric feedback matrices (feedback alignment, [49]), they assume high specificity of information propagation. However, the biological literature suggests that there are relatively few **information propagation channels** at play for coordinating learning across biological neural networks [69] that are largely dependent on cell types. They are also highly diffuse signalling pathways, with all cells in a given vicinity receiving compounded signals from neighboring cells.

The paper titled "Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network" [51] makes use of this observation in an attempt to solve credit assignment in spiking neural networks. Specifically, Liu *et al* introduce 6 cell types. Feedback information from neurons of type $\alpha$ to neurons of type $\beta$ connected by synapses $w_{\alpha\beta}$ are considered to use the same molecular signalling pathway. That is, instead of symmetric feedback matrices consisting of all $w_{ij}$, the approximation $w_{\alpha\beta} = \langle w_{ij} \rangle_{i \in \alpha, j \in \beta}$ is used.

This formulation, referred to as a "multidigraph learning rule" (MDGL), approxi-

mates the loss derivative w.r.t. a given synapse using a Hebbian eligibility trace, a top-down reward signal, and cell-type–specific diffuse modulatory signals. The work is notable for its biological realism in the diffusion of information through the network and for its incorporation of multiple cell types, unlike most work in this domain.

## 2.4 Metalearning Approaches

A less popular approach to inferring learning rules for neural networks is metalearning: learning to learn [76]. In these approaches, some optimization ("learning") is performed to find effective learning rules for neural network training. Generally, such approaches have not succeeded in moving forward the state of the art in learning rules.

One of the earliest works in this domain was from Bengio *et al* [9, 8]. Here, a function $\Delta w(\cdot)$ is optimized for network learning. The function takes as arguments pre- and post-synaptic activity, the current synapse weights, and 2 neuromodulators. Depending on the task, gradient descent, simulated annealing, or genetic algorithms were applied to optimize $\Delta w(\cdot)$, where the function was represented as a graph connecting arguments to a set of operations, the result of which was summed to obtain $\Delta w(\cdot)$. Optimization thus occurred on the graph representing $\Delta w(\cdot)$. Due to the simplicity of the tasks upon which the method was tested, the authors indicated the work as primarily a proof-of-concept.

Similar work has been undertaken in recent years in [38] where Jordan *et al* used a similar computational graph optimization approach, this time with Cartesian evolutionary optimization, to infer spiking neural network learning rules. Terms corresponding to network-level reward signals, expected reward, and pre- and post-synaptic activity correlation are used as input for the plasticity rule in reward-driven learning experiments. Similar terms were used in correlation-driven plasticity rule and error-driven plasticity rule experiments. The experiments consisted of optimization on one or two neurons, so it is unclear how well the inferred learning rules generalize to networks of spiking neurons.

### 2.4.1 Reinforcement Learning Approaches

Reinforcement learning approaches to metalearning in this domain are fairly sparse [58, 18, 77, 57]. Within these studies, neurons are generally framed as RL agents in a partially observable Markov decision process (POMDP) with a reward signal based on network performance on a task. The works differ in terms of their formulation of the agents' state spaces, and some introduce competing reward schemas [58] to incentivize biological realism. Key issues were observed in the art that either place the "weight" of the learning task's challenge on existing non-biologically-feasible techniques and/or mis-formulate each neuron's action space. For example, [58] models each neuron as a deep Q-learning neural network optimized using gradient descent (policy gradient) methods. The network is then evaluated on OpenAI's Cartpole, a standard task for deep Q-learning neural networks trained using policy gradient methods [15, 73]. Both [58, 18] also frame the action space for each neuron as "fire" or "not fire" rather than as a set of actions to alter synaptic connectivity, an approach that is unsupported by the neuroscientific and modern machine learning literature. [58, 18, 77, 57] all train separate policies for each neuron/weight rather than training the same policy to be applied by all synapses, as suggested by the neuroscientific literature. In general, low success was achieved in network training with these approaches.

## 2.5 Synthesis: Current Limitations in Learning Rules

Overall, current learning rules fail to satisfactorily reproduce the effectiveness of learning and information processing in biological neural networks. Gradient descent-based learning rules dominate in terms of learning efficacy [27, 46, 47, 64], particularly in rate-based networks, but do not appear to be biologically feasible. Surrogate gradient methods and ANN conversion appear to be most effective for spiking neural network training [80, 67, 82, 56], but suffer from the same shortcoming of biological infeasibility and may not fully take advantage of the benefits of sparse spike-based computation. Relaxations on symmetric feedback and signal diffusion are still able to yield learning [49, 51], but they are not competitive at scale with non-relaxed

strategies. Historical methods such as node perturbation (REINFORCE) [79], and associative/Hopfield learning [33] are valuable perspectives, but also fail to explain reward-driven learning at scale [59].

Metalearning approaches for generating learning rules have had low success compared to manual gradient calculation or approximation approaches [9, 8, 23, 38]. Reinforcement learning approaches for training neural networks and generating learning rules have had even less success and frequently frame the problem away from learning rules in general [58, 18, 77, 57].

Modern learning rules tend to be derived through a manual gradient approximation process, sometimes followed by further computations to add biological realism (randomizing feedback matrices, feedback signal diffusion).

# Chapter 3

# Approach

In this section, the proposed mathematical framework and justification is outlined for the SpikeSynRL project, along with proposal-specific background on reinforcement learning.

## 3.1 Arguments for an Agent-Based approach to Learning Rules

As mentioned in Chapter 2, modern learning rules tend to be derived through a manual gradient approximation process, sometimes followed by further computations to add biological realism (randomizing feedback matrices, feedback signal diffusion). Nearly all modern learning rules attempt to infer some function $\Delta w(\cdot)$ that maps network information to changes in a given synapse strength. This process notably does not take advantage of several potential cellular computation paradigms that may drive biological learning. For example, synapse memory is likely used to inform subsequent "**decisions**" on synaptic weight modification [61, 81]. Memory and local information may also inform neuron and synapse "**decisions**" to release or not release certain neurotransmitters for inter-neuron signalling [36, 66]. The majority of work to date also do not consider heterogeneity in cell types [34], and when **message passing** (neuromodulation) between cells does exist, messages only contain signals that attempt to

approximate backpropagation feedback signals [56, 51, 82, 67, 80]. Studies introducing "biological realism" either add diffusion/uniformity to the backpropagation feedback signal transmission (amounting to averaging over feedback terms as in [51]) and/or introduce non-symmetric feedback matrices as in feedback alignment [49].

These approaches are reasonable since (1) gradient descent is by far the most effective artificial neural network training method that currently exists and (2) designing other **message passing** strategies to convey learning signals through a neural network is extremely difficult: There is a high degree of **self-dependence** in the learning rule itself: synapses must "**agree**" on what signals transmitted between neurons **encode**, how to **decoded** them, and must have a sensible way of applying the information contained in the signals to inform weight updates **and subsequent signalling**. Considering neurons with **memory** adds further complexity to designing learning rules: **cooperation** between synapses must be effective through time and **prioritize** current versus future potential **rewards**. Heterogeneity also adds to the difficulty as multiple different "**agent types**" must be considered in the formulation of an overall learning strategy.

Overall, approximating gradients by repeatedly applying the chain rule (i.e., backpropagation [64]) makes for substantially less headache.

Fortunately, solving for a self-dependent cooperative decision making policy that maximizes expected reward over time (in this case, learning-related rewards) is exactly the concern of reinforcement learning [73]. I hypothesize that this problem can be effectively framed as a multi-agent single-policy (or few policy) reinforcement learning problem in a partially observable Markov decision process (POMDP).

Framing the problem as such offers hope for addressing the aforementioned challenges with generating effective learning rules not based entirely on manual gradient approximation. Furthermore, biological feasibility can be incorporated as part of the design from the outset rather than introducing limitations after the core policy has been decided. It is my hope that this investigation can lead to effective local

information-driven learning rules that open new frontiers in neuromorphic computing and elucidate properties of interest for self-organization in biological neural networks.

## 3.2 Reinforcement Learning

As alluded to previously, reinforcement learning is concerned with inferring policies for agents to interact with an environment such that some cumulative reward function is maximized over time. This section provides an overview of the relevant mathematics for Markov decision processes, reinforcement learning, and partial observability.

### 3.2.1 Markov Decision Processes

Markov decision processes (MDPs) provide a framework for understanding and analyzing decision making. MDPs consist of a set of states $\mathcal{S}$, actions $\mathcal{A}$, reward function $R(s'|a,s)$, and transition model $P(s'|s,a)$ where $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ [73].

At discrete time steps $t$, agents in MDPs select action $a_t \in \mathcal{A}$. With probability $P(s'|s,a)$, the state at $s'$ is achieved given the selection of action $a$ in state $s$. Upon state transitions, agents collect a reward based on the reward function $R(s'|a,s)$. The goal for agents in an MDP is generally to maximize the total reward.

An implicit assumption in this framework is that the subsequent state probability $s'$ depends only on the current state $s$ and the action taken by the agent $a$ (i.e., $\Pr(\{s_{t+1} = s'\}) = P(s'|s_t, a_t)$). This assumption is commonly referred to as the "Markov property", and a process is considered "Markovian" if the property holds [12, 55].

A "fully observable" MDP is one where the state $s$ is fully visible. The probability distribution of future states is thus contained in the current state and the set of subsequent actions (or the policy that decides the subsequent actions). While a fully observable state may hold in some situations (e.g., board games, some video games, etc.), it frequently does not (e.g., predicting a robot's position based on noisy sensor readings, poker, natural language, etc.). MDPs where agents can only observe part of the state are called partially observable Markov decision processes (POMDPs) [5, 73].

POMDPs introduce a "sensor model" [40] for how state $s$ is indirectly observed by an agent. Agents observe "percepts" $e \in \Omega$ through sensor model $P(e|s)$ where $s$ is the "ground truth" state. Agents can attempt to reconstruct a "belief state" through a belief state model $P(s|e_1, e_2, \dots)$. This is also known as state estimation, particularly in robotics.

### 3.2.2 Solutions to Markov Decision Processes

A "solution" to an MDP is a **policy** $\pi : \mathcal{S} \to \mathcal{A}$ that maps from the current state to a good action [73]. An optimal policy $\pi^*$ would select actions that maximize for cumulative expected reward over time. To quantify this, intermediate variable $U^\pi(s)$ is introduced that represents expected value of a given state $s$ assuming that policy $\pi$ will subsequently be followed. That is,

$$\begin{aligned} U_\pi(s_t) &= \mathbb{E}_\pi[\sum_{t'=t}^{\infty} R(s_{t'})] \\ &= R(s_t) + \mathbb{E}[\sum_{t'=t+1}^{\infty} R(s_{t'})] \\ U_\pi(s_t) &= R(s_t) + \max_{a \in \mathcal{A}} P(s'|s_t, a) U_\pi(s') \end{aligned} \tag{3.1}$$

This Bellman optimality equation [73] is a core principal for reinforcement learning. The value of entering any state is dependent on the policy that will be followed afterward. $U_\pi(s)$ is defined recursively. The incorporation of expectation over future decisions and rewards is what enables reinforcement learning techniques to perform in environments where current actions affect future states and rewards.

Following the definition of $U_\pi$, we define the optimal policy $\pi^*$ to maximize

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P(s'|s, a) U_\pi(s') \tag{3.2}$$

### 3.2.3 Q-Learning

Q-learning [73] is a reinforcement learning method for learning policies $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from experience in an environment. While there are many reinforcement learning methods for MDPs, Q-learning is advantageous for its lack of reliance on an existing transition model, sensor model (in POMDPs), or state-reward mapping. By selecting actions and receiving rewards in an MDP, Q-learning seeks to infer a policy that maximizes for rewards over time.

The Q-function is learned through Q-learning. This function predicts the expected value of state-action pairs such that the following relationship holds:

$$Q(s, a) = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(s', a') \tag{3.3}$$

Where $\gamma \in [0, 1]$ is a discount factor for future reward. $gamma \approx 0$ leads to prioritization of more immediate reward while $\gamma \approx 1$ weighs expected immediate reward and far-future rewards equally. The Q-function can be applied as a policy $\pi$ by selecting the action that maximizes $Q$ at any given timestep. That is,

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a) \tag{3.4}$$

This formulation bears similarity to the Bellman optimality defined above, but instead of using a transition model $P(s'|s, a)$, the expected value across all possible outcomes is contained in the $Q$ function.

Inferring the Q-function can be a challenging learning problem since the prediction target includes the Q-function itself. The Q-learning update equation updates the $Q$ function based on some new "experience" where action $a_t$ is taken that causes state transition from $s_t$ to $s_{t+1}$ and accompanying reward $R_{t+1}$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_{a' \in \mathcal{A}}(Q(s_{t+1}, a') - Q(s_t, a_t))] \tag{3.5}$$

Q-learning forms the basis for many reinforcement learning systems. Notably, the policy is memoryless: in order to reach optimality, the process and the state

observations must be assumed Markovian. The Q-function determines the value of a given state-action pair based only on the current state information. Optimally adapting such policies to POMDPs is therefore not straightforward from a theoretical perspective.

### 3.2.4 Reinforcement Learning in Partially Observable Processes

Per section 3.2.3, Q-learning generates memoryless policies by default. That is, $Q$ is only a function of the current state. In fact, most reinforcement agents learn memoryless policies [35]. This becomes problematic in model-free reinforcement learning applications where the agent acts in a POMDP without either a transition model or an observation model. For memoryless policies, theoretical guarantees about optimality can only be made when in fully observable Markov decision processes. Unfortunately, many reinforcement learning applications are not fully observable, least of all synaptic learning rules – an individual synapse does not observe the entire network state in full.

A common approach in reinforcement learning to address partial observability without an observation model is to train sequence models to approximate the Q-function [54]. Memory is taken care of by an RNN-LSTM, transformer model, or other sequence model that passes itself information through time steps to inform its state-action valuations. Alternatively, the agent's state can be augmented to include a large number of previous state observations. That is, $s_t = \{e_t, e_{t-1}, e_{t-2}, \dots\}$. While theoretical optimality is achievable through both options, they are expensive and not easily applicable to the synaptic learning problem at hand. While [11] incorporated 2 "history terms" representing previous observations, substantially increasing the number of history terms would quickly become extremely costly.

A recently revived approach is to endow agents with external writable memory [35]. As part of the action space, agents are able to chose actions that modify (write to) some memory. This memory is observed by the agent as part of the state. This approach

bears substantial similarity to training recurrent sequence models to approximate the Q-function. According to Icarte *et al* in [35], a sufficiently expressive memory (i.e., high dimensional) provably enables optimality in POMDPs.

In summary, the incorporation of memory of some sort is required for optimality in model-free reinforcement learning in POMDPs. Large-scale recurrent sequence models are a common method in deep reinforcement learning, along with inclusion of previous state observations as part of the model's state. The recently revived approach of incorporating editable memory units for reinforcement learning agents is of interest for this work thanks to its comparatively efficient implementation. The primary challenge with this approach, however, is training convergence: while theoretical guarantees exist that a policy using a sufficiently expressive memory module *can* express the optimal policy, it is unclear how easily it is found using existing reinforcement learning methods in this domain.

## 3.3 SynRL Methodology

In this section, I summarize the methodology from [11] as a basis for the novel work in this thesis. I refer to the general framework as "SynRL" (i.e., "synaptic-level reinforcement learning") and the current investigation as "SpikeSynRL" (i.e., "spiking synaptic-level reinforcement learning") for ease of reference.

I wrote SynRL in 2021 as a proof-of-concept for applying reinforcement learning to infer synaptic-level local information-driven learning rules in neural networks. In this general methodology, the state space $\mathcal{S}$ is comprised of the relevant local information for a given synapse, the action space for each synapse $\mathcal{A}$ is either a small increment, decrement, or null action on the synapse weight, and the reward function at each time step is positive if the network performance improved (e.g., decrease in loss $\mathcal{L}$) and negative if the network performance worsened. Importantly, the reinforcement learning policy was the same for all synapses. Every synapse applied the same policy, differing only in the locally available information in $\mathcal{S}$.

As a proof of concept, SynRL had an extremely simple state space and reinforcement

learning model. The state consisted solely of the previous two actions $a_{t-1}, a_{t-2} \in \mathcal{A}$ and the previous two network-level reward $R_{t-1}, R_{t-2}$. The reinforcement learning algorithm was a tabular Q-learning algorithm using an $\epsilon$-greedy exploration strategy during training. That is, with probability $1 - \epsilon$, the optimal action according to the Q-function was taken. With probability $\epsilon$, a random action $a_t \in \mathcal{A}$ was selected.

The primary result from SynRL was that an effective learning rule quickly emerged from training on very simple tasks. From training small neural networks to match each others' decision boundaries while simultaneously training the RL synaptic learning rule, a policy emerged that generalized well to a wide variety of problems. The policy was tested on optical character recognition tasks and was able to train a 20,000 parameter neural network to perform on part with an identical network trained using gradient descent. Perhaps due to the simplicity of the learning rule, it adapted well to different network shapes, tasks, and even activation functions.

This result bodes well for future investigations on the methodology, but it comes with significant caveats. Most importantly, the algorithm was extremely simple. It can be viewed as a non-local, discretized version of node perturbation (REINFORCE, see Section 2.3.2), or as an implementation of reward correlation learning (see Section 2.3.1). Like these methods, training speed is poor compared to gradient descent-based methods – by comparison, very little information is used in SynRL due to its simplicity. That said, the fact that the rule was automatically learned via the RL-based methodology was novel.

In light of these findings and limitations in SynRL, I aim to augment the method to understand its scalability to spiking neural networks with more information-rich state and action spaces, described next.

## 3.4 Proposed Methodology: SpikeSynRL

In this section, I outline and justify the proposed methodology for SpikeSynRL and the associated experimental design that will be used to triage and validate additions. As mentioned in 1.2, the primary additions include the addition of Hebbian terms in

the state space, message passing between synapses/neurons, incorporation of synapse memory, and heterogeneity of neuron/synapse types. The method will also be applied to a spiking neural network rather than a rate-based model as in the original SynRL paper.

Given the scope of this proposed methodology, a narrower set of tests to understand the applicability of various aspects of this proposal to training SNN's is detailed in Chapter 5.

### 3.4.1 Spiking Neural Network Model

This work focuses on feed-forward leaky integrate-and-fire spiking neural networks as the vehicle for training and validating the proposed method. As in [82], the dynamics for neuron $i$ on layer $\ell$ is based on the following update equations:

$$U_i^{(\ell)}[n+1] = (\beta_{\text{mem}} U_i^{(\ell)}[n] + (1-\beta_{\text{mem}}) I_i^{(\ell)}[n])(1 - S_i^{(\ell)}[n]) \tag{3.6}$$

Where $U_i^{(\ell)}[n]$ is the membrane potential at time step $n$ and $S_i^{(\ell)}[n] \equiv \Theta(U_i^{(\ell)}[n]-1)$ is the spike train of the neuron, defined directly through the heaviside step function $\Theta$ and the membrane potential $U_i^{(\ell)}$. As in [82], neuron dynamics are scaled so that the threshold potential is 1 and the resting potential is 0. Note that the rightmost term in Equation 3.6 accounts for an instantaneous membrane reset behaviour. $\beta_{\text{mem}} \equiv \exp(\frac{-\Delta t}{\tau_{\text{mem}}})$ stands in for the membrane's time constant $\tau_{\text{mem}}$ as a multiplicative factor scaled for a discrete time formulation at time resolution $\Delta t$. The remaining variable $I_i^{(\ell)}[n]$ is the total synaptic input current, given in terms of another update equation

$$I_i^{(\ell)}[n+1] = \beta_{\text{syn}} I_i^{\ell}[n] + \sum_j W_{ij}^{(\ell)} S_j^{(\ell-1)}[n] \tag{3.7}$$

Where $\beta_{\text{syn}} \equiv \exp(\frac{-\Delta t}{\tau_{syn}})$ is the membrane decay constant for the synapses and $W_{ij}^{(\ell)}$ is the synapse weight connecting neuron $j$ in layer $\ell-1$ to neuron $i$ in layer $\ell$. As in [82], $\tau_{\text{mem}} = 10\text{ms}$ and $\tau_{\text{syn}} = 5\text{ms}$.

### 3.4.2 State and Action Space Augmentations

#### Hebbian and Spiking Terms

Correlation between pre- and post-synaptic activity $S_{\text{pre}}[n]$ and $S_{\text{post}}[n]$ (and/or $U_{\text{pre}}[n]$ and $U_{\text{post}}[n]$) is central to Hebbian learning. To provide Hebbian learning information for the synaptic reinforcement learning policy, a technique that emerges naturally from work in surrogate gradient learning in SNNs [80] is to convolve the spike trains with a **causal kernel** $\epsilon$ and form an inner product between the convolved input and output spike train signals to calculate a Hebbian correlation metric $H_{ij}^{\ell}$:

$$\begin{aligned} H_{ij}^{(\ell)}[n] &= \langle \epsilon_{\text{pre}} * S_j^{(\ell-1)}, \epsilon_{\text{post}} * S_i^{(\ell)} \rangle \\ &= \sum_0^n (\epsilon_{\text{pre}} * S_j^{(\ell-1)})[n] \cdot (\epsilon_{\text{post}} * S_i^{(\ell)})[n] \end{aligned} \tag{3.8}$$

We note that $\epsilon_{\text{pre}}$ and $\epsilon_{\text{post}}$ can different (e.g., translated, scaled) such that post-synaptic stimulation delays can be accounted for.

In addition to Hebbian terms, spike-timing-dependent plasticity (STDP) may be investigated by adjusting kernels $\epsilon$ and alternative products such that post-synaptic activity occurring immediately after a pre-synaptic spike results in a positive term while the opposite results in a negative term. BCM-like terms may also be investigated via the same method [37].

Metrics on absolute firing statistics for pre- and post-synaptic neurons (e.g., spike totals) may also be calculated and incorporated into the reinforcement learning algorithms' state space.

#### Message Passing and Cell Type Heterogeneity

Message passing is a primary interest for the proposed agent-based approach to neural network learning rule design. It is of great interest to see how error backpropagation signals may arise when the synapses learn message passing strategies under biological constraints.

Per [51, 69], genetic data shows that cell types are primary factors in the propagation

of neuromodulators as teaching signals in biological neural networks. Liu *et al* also emphasized the importance of local diffusion of neurotransmitter information – neuron/synapse-specific neuromodulator signalling was regarded as implausible. They introduced multiple cell types $C$ with each pair $\alpha, \beta \in C$ having some diffuse neuromodulator signalling mechanism.

In this work, I aim to preserve the diffusivity and locality of neuromodulation from [51]. By framing information backpropagation as a low-bandwidth information channel from layer to layer, each synapse is able to choose some action to output a neuromodulator (or not). Depending on the cell type, the neuromodulator release will be averaged with the signals outputted by the rest of the cells of the same type. A vector containing the averaged neuromodulator release from each cell type is then propagated to all units in the previous layer, with the averaging operation representing diffusion.

This mechanism can be made more nuanced through the following changes if more backpropagation information is desired:

1. Increase the number of cell types.

2. Implement **local** diffusivity of neuromodulator release. Define a 3 dimensional spatial embedding for each neuron (potentially based on the synapse weights) and use a distance metric between each unit to form weighting for the locally diffused neuromodulator feedback vectors.

3. Provide a higher-dimensionality action for message passing (i.e., multiple neuromodulators per synapse).

4. Implement a somatic mechanism for aggregating feedback from each synapse, processing it, then transmitting it to the previous layer in a biologially plausible fashion.

#### Synapse Memory

Synapse memory is a challenging feature to implement in an efficient manner conducive to a synaptic-level learning rule. One option is to augment the action space of each synapse with the ability to write to a memory unit. This memory unit's value would then be visible to the reinforcement learning policy as part of its state space. This method, while elegant and theoretically able to express optimal policies in POMDPs, may be difficult to train.

Another option for endowing the synaptic reinforcement learning algorithm with memory is to use a simple sequence model (e.g., a small-scale RNN) as the Q-function approximator. Unlike a writable memory unit, convergence for small RNNs is better understood for a wide variety of applications in and outside of reinforcement learning. Assuming that the hidden state is small enough, it should also have minimal overhead. Unfortunately, the interpretability of the trained policy RNN would be poor. At the very least, additional work would be required to analyze the resulting RNNs input-output characteristics.

## 3.5 Experimental Design

In order to understand the effectiveness (or lack thereof) of the proposed method, I plan to implement progressively more complicated experimental configurations. Starting with a recapitulation of SynRL on spiking neural networks, I plan to incorporate the aforementioned augmentations in the following order:

1. Original SynRL methodology implemented for SNN's.
2. STDP/Hebbian state space terms.
3. Quiescence terms (did the pre/post-synaptic neuron fire whatsoever?)
4. Discrete message passing policy.
5. Convert from discrete to continuous state space with linear $Q$ function.

6. Cell type heterogeneity and diffuse message passing (start: 1D binary).
7. Writable memory unit (start: 1D binary).
8. Convert from memoryless to sequence model (shallow RNN).
9. Remove discretization of input variables.
10. Experiment with increasing overall complexity (dimensionality of transmission, Q-function, etc.)

Comparisons will be drawn with previous SpikeSynRL experiments as well as with popular existing approaches from the art such as surrogate gradient methods with varying degrees of symmetry in feedback weights. As in SynRL, additional features and complexities in experimental design will be incorporated incrementally so that failure modes in the system can be located and addressed with precision.

Due to the expansive scale of the proposed research program, this thesis paper is scoped to the theoretical arguments motivated from Chapters 2-3 and experimental design elements 1-4 from above. Future work will leverage the software architecture (Chapter 4) and the results (Chapter 5) to inform further research on steps 4-10.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 4

# Software Architecture

## 4.1 Overview

Software for this project was created in the Julia programming language [10]. It is composed of the following components:

- `LIFIterable.jl`: A module for solving the leaky integrate-and-fire spiking neural network model used in this study from [82]. Given a set of weight matrices, input spike train, and simulation constants, it produces voltage traces, current traces, and spike trains for all neurons for the duration of the input spike train.

- `SpikeSynRLv*.jl`: Modules implementing various versions (`v*`) of the SpikeSynRL method, including network state management, Q-learning, weight update, and structs.

- `SSRLv*_experiment.jl`: A script for running a single supervised learning experiment with the SpikeSynRL method.

The system was built to be run in a multithreaded fashion on `*nix` platforms. Emphasis was placed on replicability and logging for the system such that various policies, spike train inputs, and weight matrices could be re-used from before, during, and after an experiment. Code for the project can be found in the repository at

https://github.com/amanb2000/SpikeSynRL The following sections detail the key design decisions for each component. The following sections detail the key design decisions for each of the aforementioned primary components.

## 4.2 Leaky Integrate-and-Fire Solver

In order to have full controllability over features extracted while the differential equations governing the network are integrated, the solver was written from scratch using the highly efficient Julia arrays and linear algebra modules. The difference equations governing the discrete time SNN model can be found in Section 3.4.

While the the transformation from incoming spike trains $S_j^{(\ell-1)}$ to total synaptic input current $I_i^{(\ell)}$ in Equation 3.7 could be performed as a convolution with a causal exponential kernel, we found that this yielded negligible performance improvements at the scale of the proposed experiments in Chapter 5. Therefore, an iterative approach was used to perform the operation in Equation 3.7. Similarly, since the transformation from total synaptic input current $I_i^{(\ell)}$ to membrane potential $U_i^{(\ell)}$ and spike train $S_i^{(\ell)}$ is highly non-linear, the same iterative approach was used to perform the operation.

Overall, this resulted in a layer-by-layer iterative solver for the SNN model. Parallelism for each layer's integration is handled via the Julia array module's multithreading for matrix-vector slicing and modification, which is in turn handled by computer system's BLAS (the basic linear algebra subroutines).

Further optimizations may include the use of sparse matrix modules for spike trains as well as alternative solving methods based on the spike rate regime (rate versus frequency coding regime).

## 4.3 SpikeSynRL Modules

The SpikeSynRL modules contain the novel code for implementing the system. Per the scope of this experimental work, they implement tabular reinforcement learning models to discover plasticity rules in spiking neural networks. Various versions of this

module exist to test and compare various state space formulations, discussed in detail in Chapter 5. Overall, the following functionalities are implemented in the modules:

1. **Structs** specifying data organization for network and synapse states, primarily in service of generating reinforcement learning state space elements from Section 3.4.

2. **State management** functions for computing various features, state vectors, and initializing state matrices for each synapse.

3. **Training functions** for performing network and Q-learning training operations, including the main training loop for fitting an SNN to a particular input-output relationship.

Memory efficiency and parallelizability were key objectives for these systems. That said, for ease of implementation and experimentation, features such as STDP and quiescence were computed based on cached spike train data rather than being calculated in real time in the LIF model integrator. This enabled more efficient exploration of potential features extracted from neuron trace(s) value(s).

### 4.3.1 Structs

There are two broad categories of data that must be tracked throughout training: network-level data and synapse-level data. Network-level data includes elements such as the global reward function, network loss, and network topology. Synapse-specific data includes the previous actions for the synapse, voltage/current traces, STDP, quiescence, syapse messages (i.e., neurotransmitters).

The `NetworkState` struct holds network- and synapse-level data that completely describes the network's state at a given training iteration. Scalar synapse-specific information is held in a collection of matrices at the to level of `NetworkState`. Non-scalar synapse-specific information is held in a `SynapseState` struct. In this implementation, this holds only the set of previous actions and a cached version of the most recently constructed synapse state vector, used for Q-learning. A set of matrices of `SynapseState` structs is held within `NetworkState`. This formulation enables arbitrary data to be

stored about the network and each synapse, with scalar information about synapses stored in matrices to enable fast matrix operations where possible.

The following information is stored at the top level of `NetworkState`:

- Weight matrices $\Omega$.
- Recent network-level rewards $\vec{R}$.
- Loss record $\{\mathcal{L}_i\}_{i=1}^{N}$.
- Network topology.
- STDP value matrix.
- Quiescence value matrix.
- Outgoing synapse messages.
- Incoming synapse messages.
- Outgoing somatic messages.
- Matrices of `SynapseState` structs.

The following is contained in the `SynapseState` struct:

- List of previous actions $\{a_t\}_{t\in[M]}$.
- The most recent synapse state vector (used in Q-learning updates).

The meaning and method of computation for each of these elements is discussed in the following section.

### 4.3.2 State Management

In order to train network (and potentially the Q-function), the state of each synapse must be computed on each training iteration. In a perfect world, the synapse state would be updated causally and in real time as the LIF equations are integrated.

However, to quickly extend and investigate new features for the state vectors, the features are computed offline based on a copy of the LIF integration outputs (i.e., current/voltage traces and spike trains for all synapses) and the current `NetworkState`.

As mentioned in Section 3.4, the reward signal is calculated based on whether the network-level loss improved or deteriorated between the most recent two iterations. Quiescence refers to whether or not a particular neuron fired whatsoever in the most recent iteration. The outgoing synapse messages are returned by the message policy $Q_m : \mathcal{S} \to \{1, 2, \ldots, \text{message dim}\}$. The outgoing layer message is an average of the outgoing synapse messages in the layer, and the outgoing somatic message is the average outgoing synapse message for a given neuron. STDP is calculated using the following equation:

$$\text{STDP}_{ij} = \sum_{t_1 \in S^{post}} \sum_{t_2 \in S^{post}} W(t_1 - t_2) \tag{4.1}$$

Where $W(\cdot)$ is defined as

$$W(\Delta t) = \begin{cases} A_+ e^{-\Delta t/\tau_+} & \text{if } \Delta t > 0 \\ A_- e^{\Delta t/\tau_-} & \text{otherwise} \end{cases} \tag{4.2}$$

Where $A_+, A_-, \tau_+, \tau_-$ are parameters governing the STDP curve per [17, 48].

Upon each training iteration, these attributes are updated within `NetworkState` based on the most recent voltage/current traces, spike trains, network loss, and the current `NetworkState`. A function to construct the state vector for a given synapse at layer $\ell$ on row $i$ and column $j$ of the weight matrix returns the Q-learning state vector for the synapse. This is the value cached in the `SynapseState` struct on each iteration for Q-learning purposes.

### 4.3.3 Training Function

As in [11], training is two fold: a neural network's weights $\Omega$ are updated based on the output of the weight update Q-function policy $Q$. If Q-learning is enabled, the $Q$ function is also updated. In this new version, an message passing policy $Q_m$ is introduced

that accepts identical state vectors as $Q$ and outputs one of message dim potential messages (one-hot encoded). Learning for $Q, Q_m$ occurs identically. Pseudocode for the training process is in Algorithm 1.

**Algorithm 1:** Spike Synaptic RL Training Algorithm.

**Input :** NetworkState $NET$, $Q$ function, $Q_m$ message passing Q-function, dataset $(S_{in}, S_{out})$, *iterations*, $\epsilon, \alpha_s, \alpha_q, \gamma$, boolean *Train-Policy*

**Output :** $Q$ function, updated $NET$

old-loss $\leftarrow \mathcal{L}(S_{out}, NET(S_{in}))$;
**for** $t = 1 \ldots$ *iterations* **do**
  **for** *each synapse* $w$ *in NET starting on the last layer* **do**
    $w.a_t \leftarrow \pi(w.s_t)$;
    $w \leftarrow w + w.a_t$;
    $w.messageout \leftarrow \pi_m(w.s_t)$;
    **if** $w$ *is the last synapse in the layer* **then**
      Compute outgoing layer message;
      Compute outgoing somatic messages;
    **end**
  **end**
  new-loss $\leftarrow \mathcal{L}(S_{out}, NET(S_{in}))$;
  $R_t \leftarrow$ sign(old-loss − new-loss);
  Compute the new network state variables based on
  **if** *Train-Policy* **then**
    **for** *each synapse* $w$ *in NET* **do**
      Apply temporal difference update equation (7) to update $Q, Q_m$ using $w.s_t$, $w.a_t$, $w.s_{t-1}$, $w.a_{t-1}$, $w.messageout$ and $R_t$.
    **end**
  **end**
**end**
**return** $Q, Q_m, NET$

Note that the aforementioned state variables are updated during the loss calculation

and that the state vectors for each synapse are recomputed and cached upon access to $w.s_t$.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 5

# Results

In this section we describe the experiments and associated results used to understand parts 1-4 of the research program described in Section 3.5.

In general, the experiments involve generating a random spike train input/output relationship by generating some random Poisson spike trains as input and obtaining the output from a randomly instantiated SNN. A new randomly established SNN is then trained using the proposed methodology attempting to match the input-output relationship of the other network.

## 5.1 SynRL on Spiking Neural Networks

The first experiments concern the direct application of the SynRL methodology outlined in [11] and summarized in Section 3.3 to spiking neural networks. The results of this experiment aid in estimating the applicability of the reinforcement learning and discrete/tabular Q-function form used in [11], informing further experimentation.

### 5.1.1 Single Synapse Neuron

First we measure the efficacy of the original SynRL on a single input/single output network matching task. In this experiment, the network to train begins very close to perfect performance with a van Rossum distance of only 10.24 (Figure 5-1).

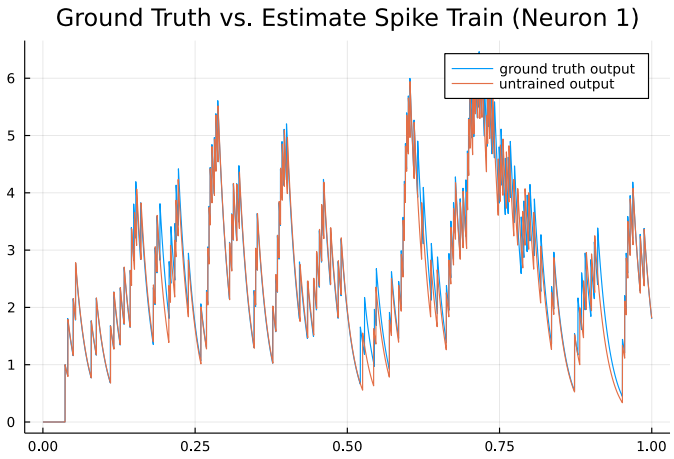


Figure 5-1: Pre-training van Rossum plot. The network that needs to be trained is already extremely close to the ground truth output spike train.

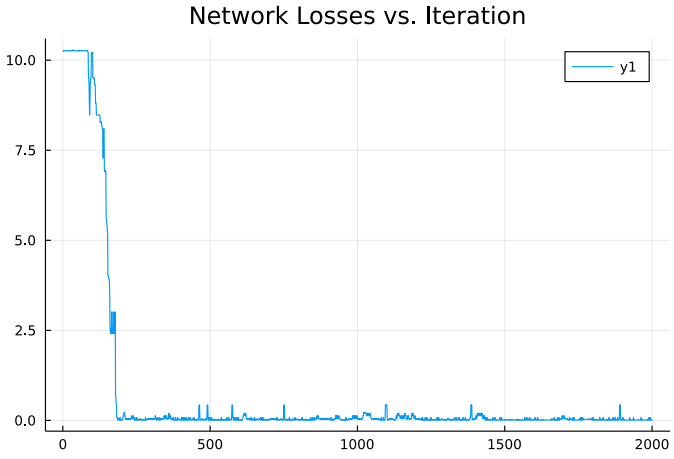


Figure 5-2: Loss function as the single input/single output system is trained.

We then train the system with simultaneous Q-learning and network updates. With $\alpha_{syn} = 0.001$, $\alpha_Q = 0.005$, exploration probability $= 0.3$, and $\gamma = 0.9$, we achieve the following network loss for each training iteration (see Figure 5-2).

We observe a fast decrease of the loss followed by fluctuation about the final value of zero. The network clearly obtains practically identical output as compared to the ground truth (Figure 5-3).

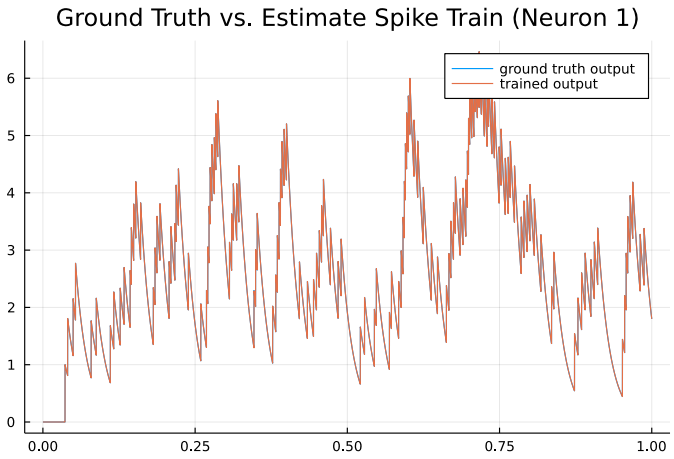


Figure 5-3: van Rossum plot after training for single input/single output network.

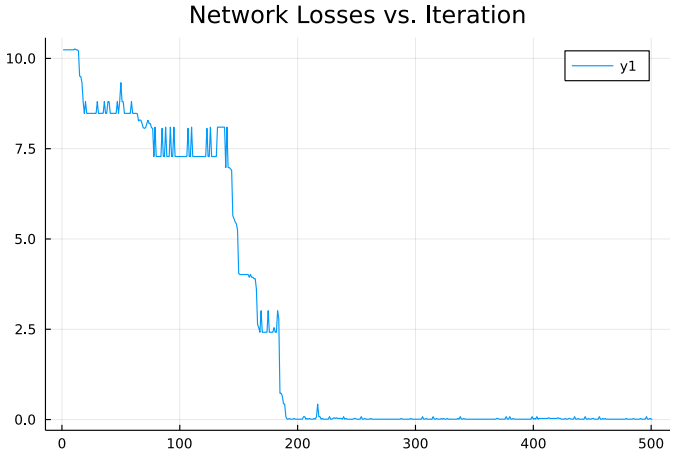


Figure 5-4: Loss function as the single input/single output system is trained with the static policy from previously, with same initialization and I/O problem.

### 5.1.2 Single Synapse Neuron: Reapplied Policy

We now re-apply the final policy from the previous experiment to the same problem, this time without any Q-learning (i.e., statically applied).

When applied to the exact same task as previous with the same initialization, the network also converges (see loss in Figure 5-4).

However, when the initialization is different (e.g., van Rossum loss is 60 upon initialization), the loss decreases by about 10 and remains stationary afterward, as though it were at some local optimum (see Figure 5-5).

It appears that the policy requires a greater exploration probability in order to

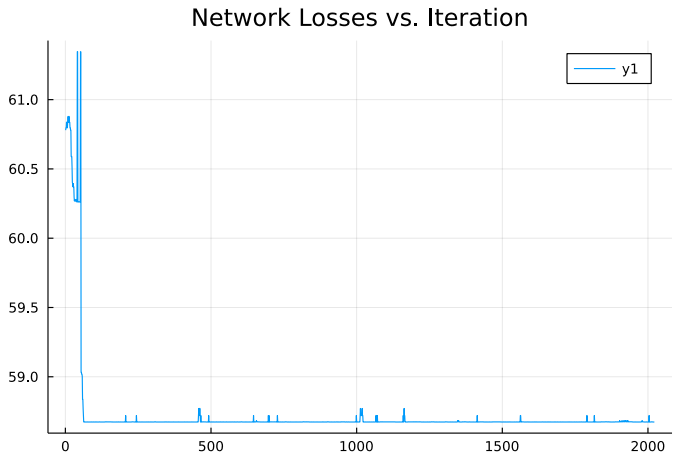


Figure 5-5: Loss function as the single input/single output system is trained with the static policy from previously, with different initialization but the same I/O problem.

be applied statically to novel problems. It may be that the Q-learning aspect of the policy pulls more computational "weight" in the spiking network regime, thus making the system more susceptible to halting when applying a policy without Q-learning. Figure 5-6 shows a plot for a randomly instantiated network being trained to perform the same I/O mapping with exploration probability 0.4 that appears to have more success in breaking out of the plateaus suffered in Figure 5-5.

### 5.1.3 1 input, 10 output neurons: Applied and Trained Policy

Since it appears that the Q-learning may have "overfit" to the 1-unit task, we now apply the methodology to a new 1 input, 10 output matching task (see spike rasters for input, output, and desired output before training in Figure 5-7).

The range for the weights in the ground truth and trained networks were chosen to be only 1, with a mean value of 3 in order to stay in the regime of high similarity. The policy from the first single input/single output experiment was used as a "starting point" for training the Q-function in this experiment. The loss (Figure 5-8 does generally decrease, but there are some notable plateaus and dips. Optimally, a consistent and stable policy would arise from the training that enables smooth convergence in such tasks. After all, each synapse is functionally independent, receiving a (very noisy) loss signal indicating improvement or deterioration from one iteration to the next.

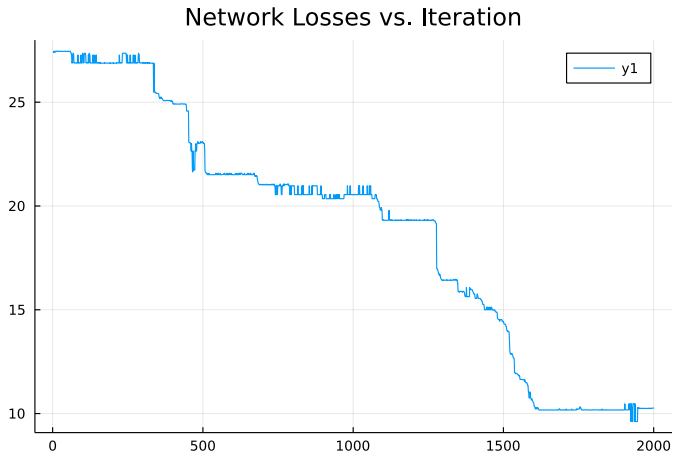


Figure 5-6: Loss function as the single input/single output system is trained with the static policy from previously, with different initialization but the same I/O problem. Static policy. Exploration probability is set to 0.4, apparently enabling the system to break from plateaus.

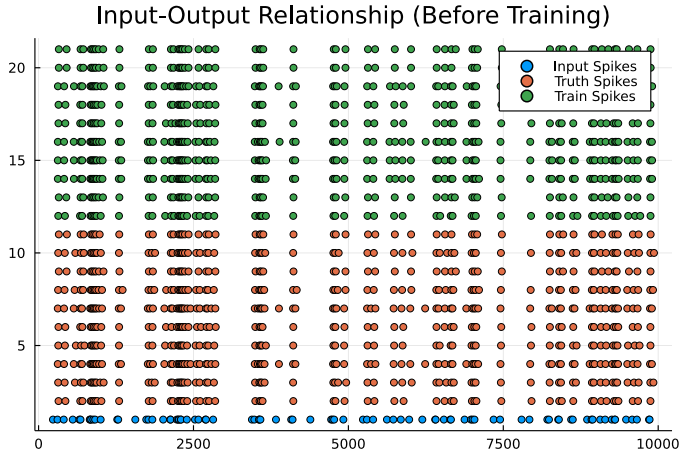


Figure 5-7: Spike rasters before training for 1 input 10 output task.

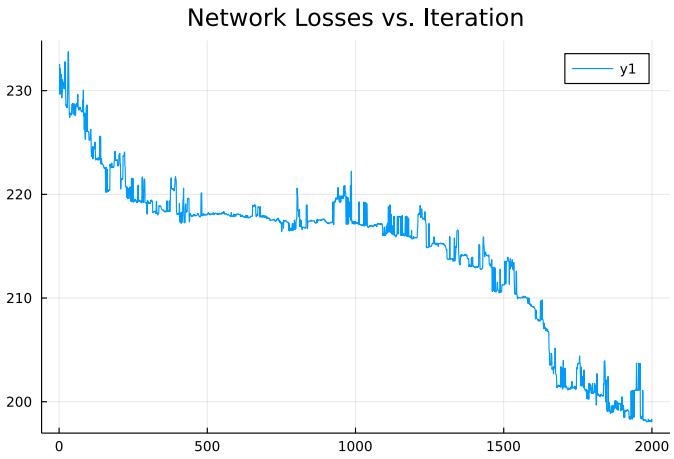


Figure 5-8: Loss curve for the 1 input 10 output network matching task.

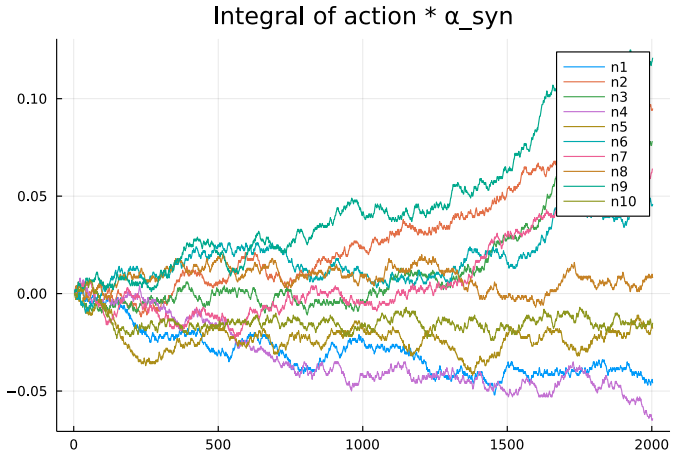


Figure 5-9: Trajectory of each weight over the course of training. Note the lack of clear correlation over the collection of synapses.

To find trends in the selected action (e.g., the system becomes biased toward 1 action depending on what benefits the most synapses at a given time), plots showing synapse trajectory (Figure 5-9) of each synapse were generated.

As well, the evolution of the policy over training was visualized in Figure 5-10 that shows the number of policy decisions each policy along the way during training "agrees" on with the final policy. The plot indicates that the final policy only agreed with about 30% of the original "starting policy". Perhaps this indicates some overfitting.

**Applying the Policy:** We now apply the 1 input 10 output policy to a new random 1 input 10 output task statically. The same narrow bounds are set on the

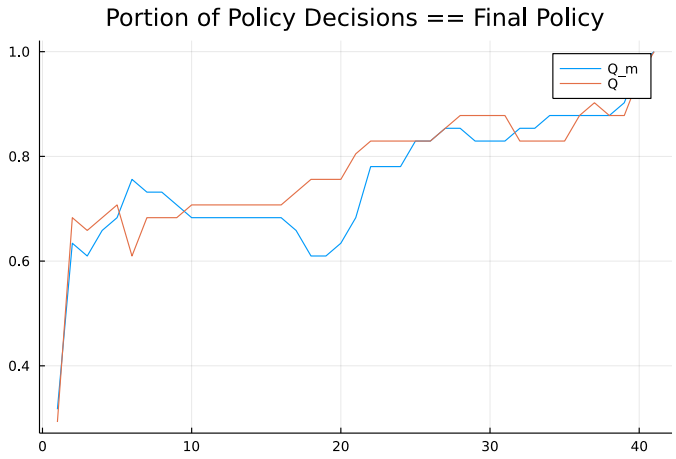


Figure 5-10: Number of decisions in agreement with the final policy over the course of training.

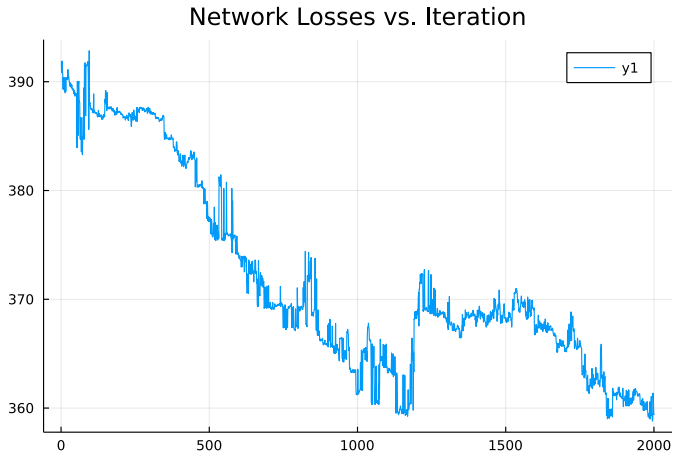


Figure 5-11: Loss over time for the statically applied 1 input 10 output task.

ranges of the initialized networks as before.

Figure 5-11 shows the loss generally decreasing with time, and Figure 5-12 shows the relatively uncorrelated weight trajectories with time. Overall, it appears that the statically applied policy is met with moderate success on a new network matching task with a differing objective and initialization.

Unfortunately, applying the policy to more complex tasks (e.g., where output neurons have more than 1 input synapse) yields poor convergence, motivating additions to the state space of each synapse to better inform weight updates.

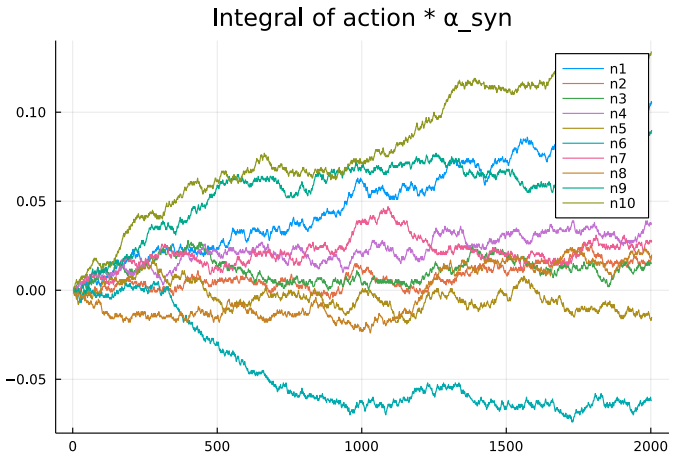


Figure 5-12: Weight trajectories for the statically applied 1 input 10 output task.

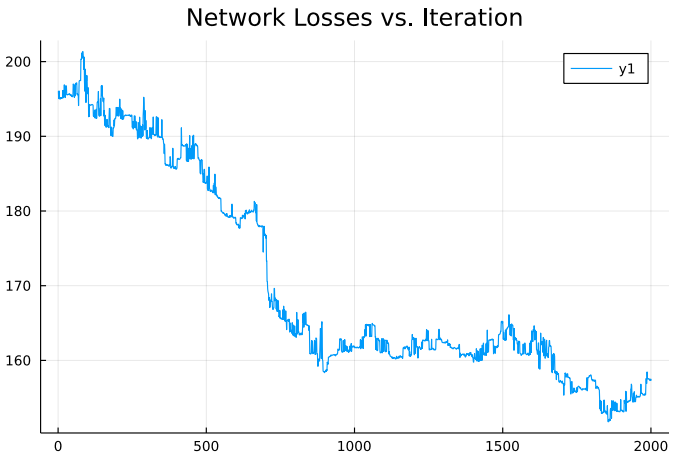


Figure 5-13: Loss function for the 1 input 10 output task with STDP in the state space.

## 5.2 SynRL & STDP

We start by training a 1 input 10 output model with the same initialization distribution as before with STDP signals as part of the state space.

Based on Figures 5-13 and 5-14, it does not appear as though STDP offers substantial improvement relative to the original SynRL state space.

**Reapplying the static policy:** When re-applying the policy, we observe much the same behaviour as without STDP. Namely, the loss decreases with slightly more noise and bottoms out before reaching zero (Figure 5-15).

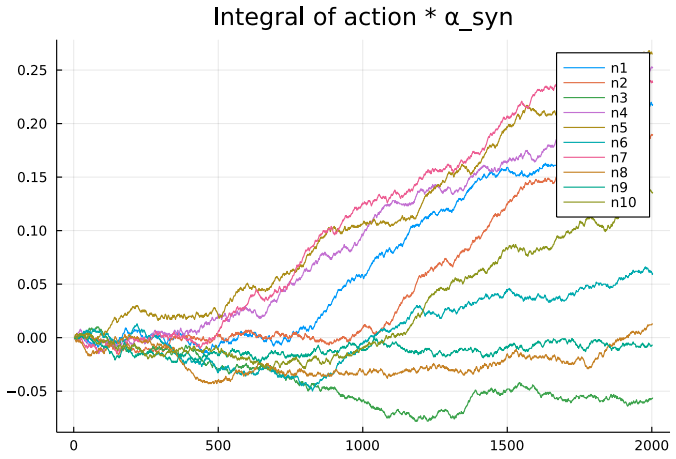


Figure 5-14: Weight trajectories for the 1 input 10 output task with STDP in the state space.

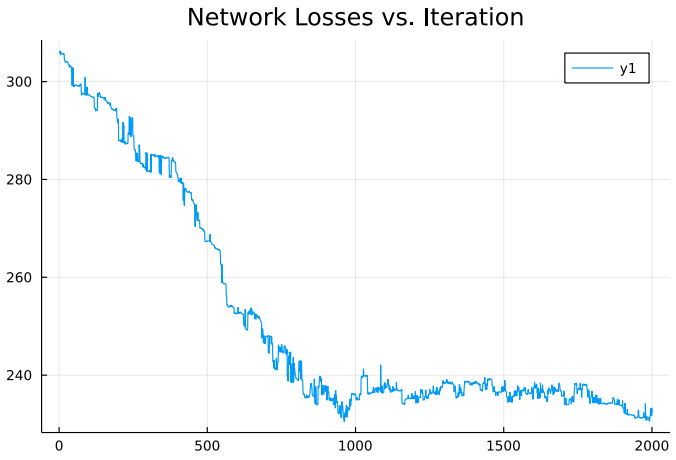


Figure 5-15: Loss function for the 1 input 10 output task with STDP in the state space, reapplied static policy from the previous test.

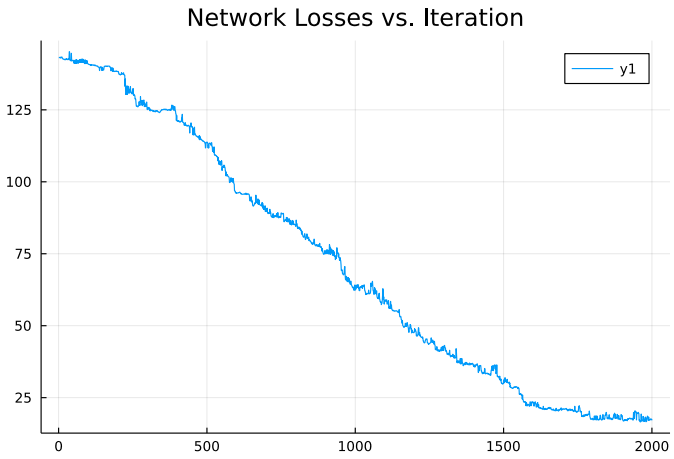


Figure 5-16: Loss over time for the 1 input 10 output task with message passing.

Overall, it was found that the addition of STDP to the state space had negligible improvement on shallow and small SNN's trained using this method.

## 5.3 SynRL + Message Passing

We now investigate the addition of a message passing protocol to the original SynRL methodology. For these preliminary trials, messages can take on values in the set $\{1, 2, 3\}$. For the output neurons, they receive messages based on whether they outputted more spikes than expected (3), fewer (1), or the correct number (2). We hypothesize that, particularly for single-layer networks, this will substantially improve performance.

### 5.3.1 1 input, 10 output

As a sanity check, the 1 input/10 output structure ought to benefit substantially from this message passing addition. The results in Figure 5-16 confirm this as the loss very quickly and consistently begins to decrease. Figure 5-17 also shows a smooth convergence of the policy to its final value, and the weight trajectories indicate a lack of correlation between the weights.

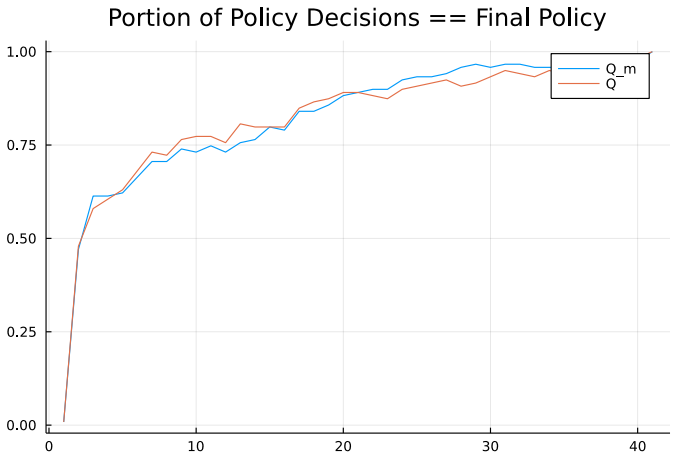


Figure 5-17: Number of agreements in policy decision with the final policy for the 1 input 10 output training with message passing.

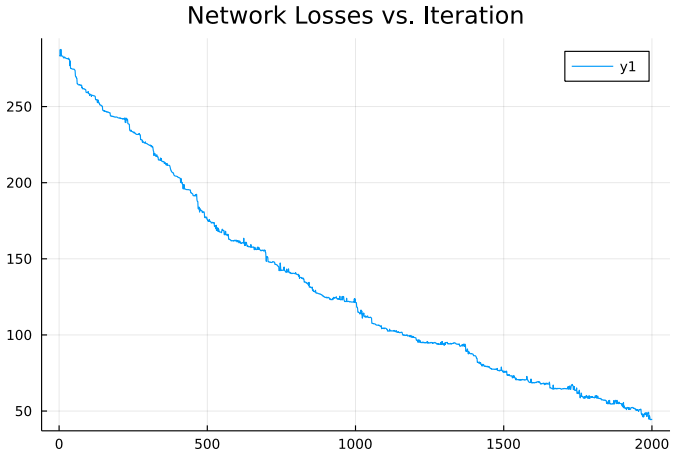


Figure 5-18: Loss curve for statically applied policy to 1 input 10 output task with message passing.

**Static application:** We now apply the learned policy from above to a new problem of the same form but with different initialization and ground truth. If anything, Figure 5-18 indicates faster convergence than before.

### 5.3.2 2 inputs 5 outputs

Notably, the previous task had only 1 input synapse per output neuron. We now investigate if the message passing scheme can generalize to multiple input synapses by training a 2 input/5 output system with the policy from the previous experiment as the starting point. We hypothesize that the additional state information will allow

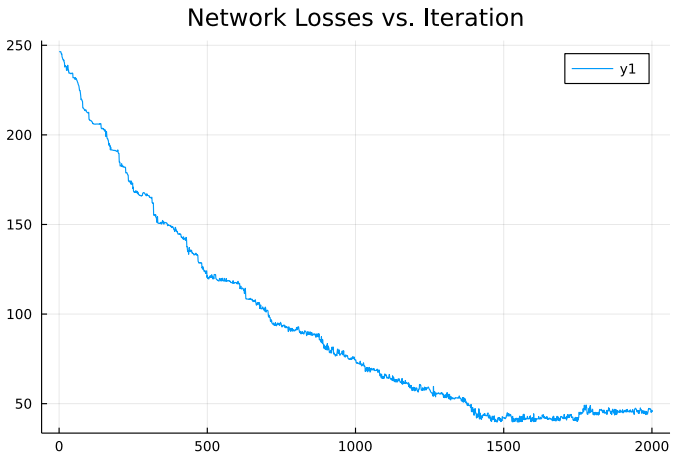


Figure 5-19: Loss curve for 2 input 5 output model with message passing.

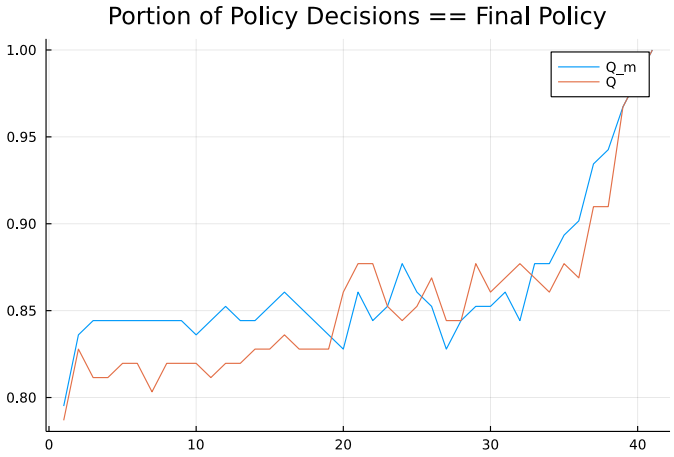


Figure 5-20: Policy self-agreement for 2 input 5 output model with message passing.

this formulation to succeed where the messageless one failed.

Promisingly, the network converges (Figure 5-19) and the policy from before was already about 85% in agreement with the final policy learned in this new task (Figure 5-20)

### 5.3.3 2 inputs, 3 hidden units, 5 output units

To gauge the model's ability to generalize from bespoke output neuron messages to internal messages, we apply the policy from the previous experiment as a starting point for an experiment with 2 inputs, 3 hidden units, and 5 output units.

The results from Figure 5-21 and Figure 5-22 imply that the network does, in fact,

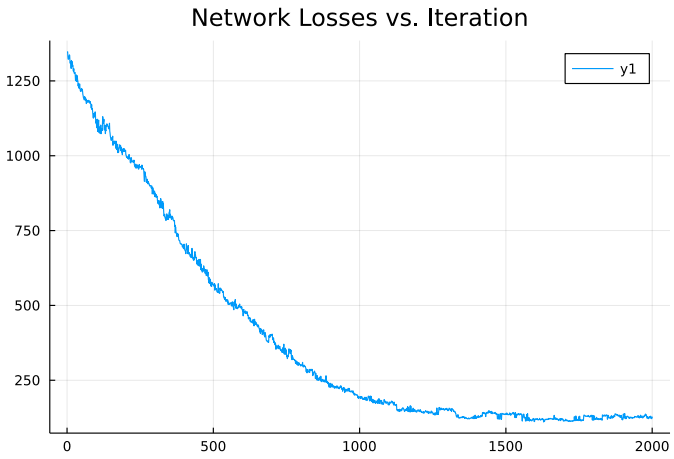


Figure 5-21: Loss curve for 2 input 5 output model with message passing.

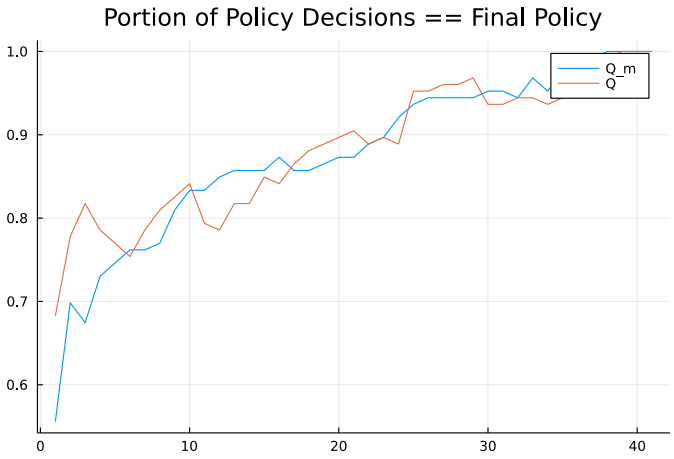


Figure 5-22: Policy self-agreement for 2 input 5 output model with message passing.

take advantage of this newfound message passing ability, substantively changing its message passing policy $Q_m$ and reaping performance benefits.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 6

# Discussion & Conclusions

The above results indicate that message passing, even when discrete and noisy as it is in this system, vastly improves convergence on SNN training. Without additions to the state space, the conventional SynRL methodology struggles to scale to the challenges of spike-based computation. Additional specificity in feedback signals, as well as high exploration probabilities, appear to aid in addressing this struggle.

This issue likely results from the discontinuous loss landscape. Adjusting a SNN's synapse weights yields changes in the placement and in the number of output spikes. While changing the placement of spikes is a continuous operation, adding a wholly new spike is a discontinuous one. It seems likely that the smoothness properties of rate-based networks is largely what enabled the original SynRL paper to perform with such a simple state space formulation.

The work, however, bodes well for the future of this research program. In particular, the ability for the reinforcement learning model to learn effective and balanced policies with message passing indicates that further work on automated information propagation designs via RL. Results from Figure 5-21 are extremely promising when considering that the policy's form remains tabular.

That said, it appears that STDP features alone in shallow networks do not enable substantively improved performance as compared to the original SynRL methodology. It may be that these features will improve performance as deeper networks are experimented with, or as more extrapolative Q-function approximators are implemented.

## 6.1 Next Steps

In light of the results gathered in this report, the next steps in the research program outlined in Section 3.5 have been reorganized. At this point, the highest priority is to implement more extrapolative Q-function approximators as the combinatorial explosion brought on by concatenating additional features into the original state space was untenable to learn in a reasonable time complexity. This will also have the added benefit of allowing for continuous state space variables, thus increasing the information transmissibility within the model.

As well, the efficiency of the training process for these tests was a limiting factor for conducting larger scale and longer time-horizon experiments. In addition to conventional supercomputing infrastructure, neuromorphic computing modules are of interest to dramatically increase the scale and scope of the experiments.

Finally, changing the training process to be more dynamical is of great interest for generating efficient, biologically feasible rules. Since biological neurons do not appear to have well-defined training iterations, online learning is a key next step for understanding their computational properties. Moreover, the reinforcement learning methodology proposed herein is very well suited for generating such rules since the rate of actions can simply be increased over the course of the training as in [80].

# Appendix A

# Tables

THIS PAGE INTENTIONALLY LEFT BLANK

# Appendix B

# Figures

THIS PAGE INTENTIONALLY LEFT BLANK

# Bibliography


[1] Larry F Abbott. "Lapicque's introduction of the integrate-and-fire model neuron (1907)". In: *Brain research bulletin* 50.5-6 (1999), pp. 303–304.

[2] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. Vol. 4. AMLBook New York, NY, USA: 2012.

[3] Ralph Adolphs. "The unsolved problems of neuroscience". In: *Trends in Cognitive Sciences* 19.4 (2015), pp. 173–175.

[4] Edgar D Adrian and Yngve Zotterman. "The impulses produced by sensory nerve-endings: Part II. The response of a Single End-Organ". In: *The Journal of physiology* 61.2 (1926), pp. 151–171.

[5] Karl Johan Åström. "Optimal Control of Markov Processes with Incomplete State Information I". eng. In: Journal of Mathematical Analysis and Applications 10 (1965), pp. 174–205. ISSN: 0022-247X. DOI: 10.1016/0022-247X(65)90154-X. URL: https://lup.lub.lu.se/search/files/5323668/8867085.pdf.

[6] Bruno B Averbeck, Peter E Latham, and Alexandre Pouget. "Neural correlations, population coding and computation". In: *Nature reviews neuroscience* 7.5 (2006), pp. 358–366.

[7] Atilim Gunes Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Journal of machine learning research* 18 (2018).

[8] Samy Bengio et al. "On the optimization of a synaptic learning rule". In: *Optimality in Biological and Artificial Networks?* Routledge, 2013, pp. 281–303.

[9] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

[10] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[11] Aman Bhargava, Mohammad R Rezaei, and Milad Lankarany. "Gradient-Free Neural Network Training via Synaptic-Level Reinforcement Learning". In: *arXiv preprint arXiv:2105.14383* (2021).

[12] Christopher Bishop. *Pattern recognition and machine learning*. New York: Springer, 2006. ISBN: 978-1-4939-3843-8.

[13] James Bower. *The Book of GENESIS : Exploring Realistic Neural Models with the GEneral NEural SImulation System*. New York, NY: Springer New York, 1998. ISBN: 978-1-4612-1634-6.

[14] Romain Brette. "Philosophy of the spike: rate-based vs. spike-based theories of the brain". In: *Frontiers in systems neuroscience* 9 (2015), p. 151.

[15] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[16] Guy C Brown. *The energy of life: The science of what makes our minds and bodies work*. Free Press New York, 1999.

[17] Natalia Caporale and Yang Dan. "Spike timing–dependent plasticity: a Hebbian learning rule". In: *Annu. Rev. Neurosci.* 31 (2008), pp. 25–46.

[18] Matthew Chalk, Gasper Tkacik, and Olivier Marre. "Training and inferring neural network function with multi-agent reinforcement learning". In: *bioRxiv* (2020), p. 598086.

[19] François Chollet. "On the measure of intelligence". In: *arXiv preprint arXiv:1911.01547* (2019).

[20] Ami Citri and Robert C Malenka. "Synaptic plasticity: multiple forms, functions, and mechanisms". In: *Neuropsychopharmacology* 33.1 (2008), pp. 18–41.

[21] Samuel Frazer Cooke and Timothy VP Bliss. "Plasticity in the human central nervous system". In: *Brain* 129.7 (2006), pp. 1659–1673.

[22] EM Dogo et al. "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks". In: *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE. 2018, pp. 92–99.

[23] Edgar Galván and Peter Mooney. "Neuroevolution in deep neural networks: Current trends and future challenges". In: *IEEE Transactions on Artificial Intelligence* (2021).

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

[25] Wulfram Gerstner et al. "Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules". In: *Frontiers in neural circuits* 12 (2018), p. 53.

[26] Wulfram Gerstner et al. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[28] Wenzhe Guo et al. "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems". In: *Frontiers in Neuroscience* 15 (2021), p. 212.

[29] Robert Haslinger, Kristina Lisa Klinkner, and Cosma Rohilla Shalizi. "The computational structure of spike trains". In: *Neural computation* 22.1 (2010), pp. 121–157.

[30] Jeff Hawkins and Subutai Ahmad. "Why neurons have thousands of synapses, a theory of sequence memory in neocortex". In: *Frontiers in neural circuits* 10 (2016), p. 23.

[31] Donald O Hebb. "The first stage of perception: growth of the assembly". In: *The Organization of Behavior* 4 (1949), pp. 60–78.

[32] Jay A Hennig et al. "How learning unfolds in the brain: toward an optimization view". In: *Neuron* 109.23 (2021), pp. 3720–3735.

[33] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[34] Eric Hunsberger, Matthew Scott, and Chris Eliasmith. "The competing benefits of noise and heterogeneity in neural coding". In: *Neural computation* 26.8 (2014), pp. 1600–1623.

[35] Rodrigo Toro Icarte et al. *The act of remembering: a study in partially observable reinforcement learning*. 2020. arXiv: `2010.01753 [cs.LG]`.

[36] Kei Ito et al. "A systematic nomenclature for the insect brain". In: *Neuron* 81.4 (2014), pp. 755–765.

[37] Eugene M Izhikevich and Niraj S Desai. "Relating stdp to bcm". In: *Neural computation* 15.7 (2003), pp. 1511–1523.

[38] Jakob Jordan et al. "Evolving interpretable plasticity for spiking networks". In: *Elife* 10 (2021), e66273.

[39] Jonathan Kadmon, Jonathan Timcheck, and Surya Ganguli. "Predictive coding in balanced neural networks with noise, chaos and delays". In: *Advances in Neural Information Processing Systems* 33 (2020).

[40] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[41] Eric R Kandel. "The molecular biology of memory storage: a dialog between genes and synapses". In: *Bioscience reports* 24.4-5 (2004), pp. 475–522.

[42] George M Kapalka. *Nutritional and herbal therapies for children and adolescents: a handbook for mental health clinicians*. Academic Press, 2009.

[43] Christian Keysers and Valeria Gazzola. "Hebbian learning and predictive mirror neurons for actions, sensations and emotions". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 369.1644 (2014), p. 20130175.

[44] Benjamin James Lansdell, Prashanth Ravi Prakash, and Konrad Paul Kording. "Learning to solve the credit assignment problem". In: *arXiv preprint arXiv:1906.00889* (2019).

[45] Benjamin James Lansdell, Prashanth Ravi Prakash, and Konrad Paul Kording. "Learning to solve the credit assignment problem". In: *arXiv preprint arXiv:1906.00889* (2019).

[46] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[47] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[48] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback". In: *PLoS computational biology* 4.10 (2008), e1000180.

[49] Timothy P Lillicrap et al. "Random synaptic feedback weights support error backpropagation for deep learning". In: *Nature communications* 7.1 (2016), pp. 1–10.

[50] William A Little. "The existence of persistent states in the brain". In: *Mathematical biosciences* 19.1-2 (1974), pp. 101–120.

[51] Yuhan Helena Liu et al. "Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network". In: *Proceedings of the National Academy of Sciences* 118.51 (2021).

[52] Pietro Mazzoni, Richard A Andersen, and Michael I Jordan. "A more biologically plausible learning rule for neural networks." In: *Proceedings of the National Academy of Sciences* 88.10 (1991), pp. 4433–4437.

[53] Gregory Morse and Kenneth O Stanley. "Simple evolutionary optimization can rival stochastic gradient descent in neural networks". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. 2016, pp. 477–484.

[54] Amir Mosavi et al. "Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics". In: (2020). DOI: 10.20944/preprints202003.0309.v1. URL: http://dx.doi.org/10.20944/preprints202003.0309.v1.

[55] Kevin Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass: MIT Press, 2012. ISBN: 9780262018029.

[56] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[57] Shohei Ohsawa et al. "Neuron as an Agent". In: *ICLR 2018 : International Conference on Learning Representations 2018*. 2018.

[58] Jordan Ott. "Giving Up Control: Neurons as Reinforcement Learning Agents." In: *arXiv preprint arXiv:2003.11642* (2020).

[59] Michael Pfeiffer and Thomas Pfeil. "Deep learning with spiking neurons: opportunities and challenges". In: *Frontiers in neuroscience* 12 (2018), p. 774.

[60] Robert Plonsey. *Bioelectricity : a quantitative approach*. New York, NY: Springer, 2014. ISBN: 978-1-4899-8408-1.

[61] Roger L Redondo and Richard GM Morris. "Making memories last: the synaptic tagging and capture hypothesis". In: *Nature Reviews Neuroscience* 12.1 (2011), pp. 17–30.

[62] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408.

[63] Jonathan E. Rubin et al. "The credit assignment problem in cortico-basal ganglia-thalamic networks: A review, a problem and a possible solution". In: *European Journal of Neuroscience* 53.7 (2021), pp. 2234–2253.

[64] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[65] Neil Savage. "How AI and neuroscience drive each other forwards". In: *Nature* 571.7766 (2019), S15–S15.

[66] Lauralee Sherwood. *Human physiology : from cells to systems*. Toronto, Ontario: Nelson, 2019. ISBN: 978-0176744847.

[67] Sumit Bam Shrestha and Garrick Orchard. "SLAYER: Spike Layer Error Reassignment in Time". In: *NeurIPS*. 2018.

[68] Dan Simon. *Evolutionary optimization algorithms*. Chichester: Wiley-Blackwell, 2013. ISBN: 978-0-470-93741-9.

[69] Stephen J Smith et al. "Single-cell transcriptomic evidence for dense intracortical neuropeptide networks". In: *Elife* 8 (2019), e47889.

[70] Daniel Soudry, Itay Hubara, and Ron Meir. "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights." In: *NIPS*. Vol. 1. 2014, p. 2.

[71] Courtney J Spoerer, Patrick McClure, and Nikolaus Kriegeskorte. "Recurrent convolutional neural networks: a better model of biological object recognition". In: *Frontiers in psychology* 8 (2017), p. 1551.

[72] Charles F Stevens and Anthony Zador. "Neural coding: The enigma of the brain". In: *Current Biology* 5.12 (1995), pp. 1370–1371.

[73] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[74] Claudio Turchetti. *Stochastic models of neural networks*. Vol. 102. IOS Press, 2004.

[75] Robert Urbanczik and Walter Senn. "Reinforcement learning in populations of spiking neurons". In: *Nature neuroscience* 12.3 (2009), pp. 250–252.

[76] Joaquin Vanschoren. "Meta-learning". In: *Automated Machine Learning*. Springer, Cham, 2019, pp. 35–61.

[77] Zhipeng Wang and Mingbo Cai. "Reinforcement Learning applied to Single Neuron". In: *arXiv preprint arXiv:1505.04150* (2015).

[78] James C.R. Whittington and Rafal Bogacz. "Theories of Error Back-Propagation in the Brain." In: *Trends in Cognitive Sciences* 23.3 (2019), pp. 235–250.

[79] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3 (1992), pp. 229–256.

[80] Friedemann Zenke and Surya Ganguli. "Superspike: Supervised learning in multilayer spiking neural networks". In: *Neural computation* 30.6 (2018), pp. 1514–1541.

[81] Friedemann Zenke, Ben Poole, and Surya Ganguli. "Continual learning through synaptic intelligence". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3987–3995.

[82] Friedemann Zenke and Tim P Vogels. "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks". In: *Neural Computation* 33.4 (2021), pp. 899–925.