# ENHANCED REINFORCEMENT LEARNING WITH ATTENTIONAL FEEDBACK AND TEMPORALLY ATTENUATED DISTAL REWARDS

A Thesis

by

KUMARAN THULASIRAMAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2015

Major Subject: Computer Engineering

## ABSTRACT

This thesis presents a new reinforcement learning mechanism suitable to be employed in artificial spiking neural networks of leaky integrate-and-fire (LIF) or Izhikevich neurons. The proposed mechanism is upgraded from, and closely built upon the learning algorithm introduced by Florian, in which local synaptic plasticity is based on the relative spike-timing of the pre and post-synaptic neurons (STDP), and is modulated by a global reinforcement signal.

This work introduces and deals with multiple challenges identified in existing reinforcement learning schemes, that includes the *distal reward problem*, the *spatial credit assignment problem* and the *response numbness problem*. A number of improvements, that are inspired either from the biological elements or from similar implementations in non-spiking neural networks, are suggested to handle these challenges, and are validated through biologically-inspired experiments. The notion and implementation of attentional feedback that handles the spatial credit assignment problem during synaptic reinforcement are introduced. The effects of attenuated rewards, which gate network learning after satisfactory reinforcement is achieved, are also demonstrated. This aids in the exploration of the agent to discover other rewardable behaviors during learning. A spike-rate based input encoding scheme termed as *balanced-pair binary state* (BPBS) encoding, and a corresponding methodology for response selection are also introduced to improve network stability and confidence in response selection.

The proposed techniques are validated using multiple biologically-inspired single agent as well as multi-agent game-theoretic experimental tasks. The single-agent tasks include exclusive OR (XOR) function reproduction and a bot walking task. The multi-agent interactive and cooperative tasks demonstrated include the general-sum *iterated prisoners' dilemma* (IPD) game problem and the distributed *SensorNetwork* problem from the NIPS '05 reinforcement learning benchmarks.

The results and findings discussed in this work validate that the proposed improvements to existing implementations of reinforcement learning could, in fact, lead to better brain-like learning and behavior in artificial agents.

## DEDICATION

*I dedicate my thesis work to my loving parents,* ***Thulasiraman Krishnan*** *and* ***Valliammal Ramasamy****, whose support and prayers have made my perseverance and accomplishments possible.*

# ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Peng Li, for his continued support, guidance and mentorship towards this research effort. I feel honored to have worked with him. I also appreciate the feedback from Dr. Li's research group, which were helpful in refining my work and presentation.

I wish to express my gratitude to my advisory committee members, Dr. Yoonsuck Choe and Dr. Alex Sprintson, for their availability, patience and feedback.

I acknowledge and thank the Computer Engineering and Systems Group (CESG) at Texas A&M University, for providing access to the *dropzone* computer cluster for my research.

Special thanks to the LaTeX student users group of Texas A&M University for providing the LaTeX template for writing this document.

NOMENCLATURE

| | |
|---|---|
| CR | Conditioned Response |
| CS | Conditioned Stimulus |
| DA | Dopamine |
| DA-STDP | Dopamine-modulated Spike-Timing Dependent Plasticity |
| DR | Distal Reward |
| HH | Hodgkin-Huxley |
| IPD | Iterated Prisoner's Dilemma |
| LIF | Leaky Integrate-and-Fire |
| LTD | Long Term Depression |
| LTP | Long Term Potentiation |
| MARL | Multi-Agent Reinforcement Learning |
| MDP | Markov Decision Process |
| NIPS | Neural Information Processing Systems |
| RL | Reinforcement Learning |
| STDP | Spike-Timing Dependent Plasticity |
| UR | Unconditioned Response |
| US | Unconditioned Stimulus |
| VTA | Ventral Tegmental Area |
| XOR | Exclusive OR |

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1 Background and Motivation

Human beings are, and essentially every living organism is, inherently hedonistic. They continually engage in a pursuit for pleasure, and at every point in the course of their lives, the inherent goal of their mind is to promote or reinforce self behaviours that maximize pleasure, either short-term or long-term, while suppressing those that earn them punishments. They are programmed so, and it is that simple.

From the perspective of machine learning, approaches like supervised and unsupervised learning have been explored with brain-inspired systems like artificial neural networks. While supervised training provides labeled examples of target response for each observation or behavior, unsupervised learning lets the learning agent make models from training data without explicit expert supervision [1]. These approaches have led to learning techniques that handle precision problems, and perform human-like tasks to varying degrees of efficiency, concurrency and speed [2].

A critical notion that created a new branch of study in machine learning is that of an agent or an artificial system that mimics the way a human *learns*, instead of just the way he *performs* [3]. This is closely related to the reality of the existence of interaction between the brain and its environment. Interaction represents the transfer of feedback from the environment to the agent, for every action initiated by the agent, that affected the environment in some way. This feedback evaluates and positively reinfores the action performed by the agent in the context of the task at hand, and is in turn used by the agent to learn to act rewardably in similar situations within its environment. Thus *reinforcement learning* (RL), with this non-expert feedback from the environment that aids learning, sits in the middle of supervised and unsupervised

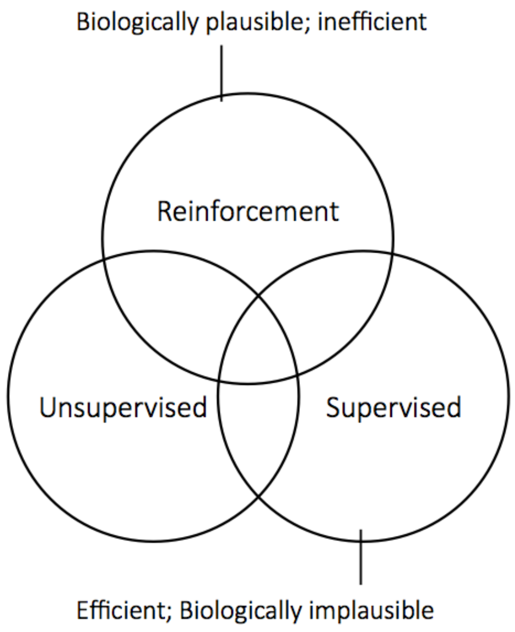

Biologically plausible; inefficient
Reinforcement
Unsupervised
Supervised
Efficient; Biologically implausible

Figure 1.1: Machine learning strategies

learning (Figure. 1.1).

While supervised learning has proved to be very efficient and precise, it is not very much biologically plausible. There is no *teacher* inside the brain, or outside, that always has and provides the precise action sequence that is expected from the person or his brain for every situation observed by him in his environment [3]. Also, in realistic tasks, which are interactive, obtaining exhaustive examples of correct behavior to teach a system might be unrealistic or very difficult. Reinforcement learning is also unique in the fact that, in many interesting applications, the future observations made by a system depend on the current action or set of actions that it chooses to perform. A system that learns this way has to explore its options to determine which action is rewarded in a specific situation, and in course of time, exploit this knowledge to maximize the rewards. This is the natural way of learning and survival in intelligent living organisms [4]. Understanding the human brain requires that the models and learning mechanisms that we use to study it be very much closely inspired by its structure and function. Since reinforcement learning is evidently the way the human brain learns behavior, it makes a lot of sense to pursue it as the strategy to make brain-inspired aritifical systems learn and function.

## 1.2 Biological Reward System

The reinforcement learning or the reward system in the brain has the Basal Ganglia (Figure. 1.2) at its core. This part of the brain controls responses to natural rewards like food and sex, that leads to motivation and drive. These rewards are stimuli that act as behavioral reinforcement agents. They cause an increase in the probability of a behavior if it is frequently associated with or followed by them. The reward pathways in the brain include a set of structures which play an important role in regulating internal activities, and thereby behavior, based on rewards. Dopamine (DA) is an extracellular neurotransmitter from the neurons of the ventral tegmental area (VTA) that is used as a chemical messenger across different internal structures of the brain to propagate the effects of these rewards [5]. Modeling of the dopamine pathways and the related mechanisms is therefore key to computationally implementing reinforcement learning.

### *1.2.1 Classical and Instrumental Conditioning*

Conditioning is the process of training a system to behave in a certain expected way [6]. A mouse trained using food to solve a maze, and a dog trained to salivate when a bell is rung are examples of conditioning in animals. Among the several types of conditioning, classical conditioning and operant or instrumental conditioning are interesting, especially from the perspective of training machines.

Classical conditioning is the processing of behavior modification in organisms brought about by the pairing or association of a neutral stimulus and a potent stim-

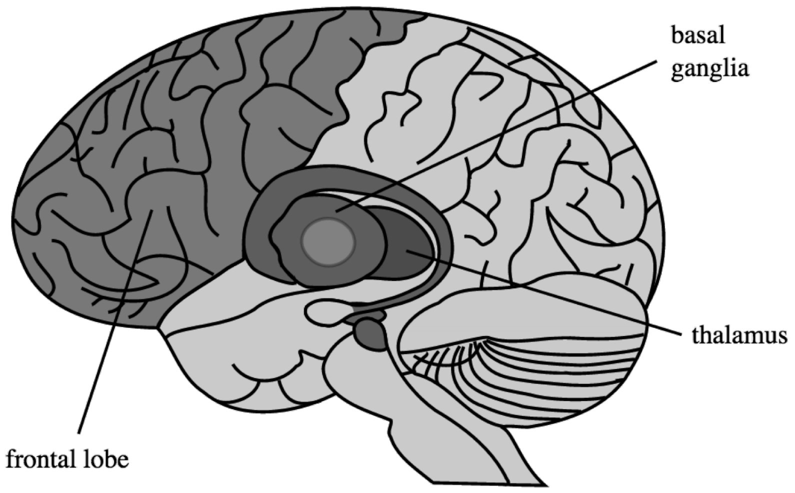


Figure 1.2: Basal Ganglia in the human brain

ulus to invoke a desired response, which was previously observed only in response to the neutral stimulus. This was demonstrated by Ivan Pavlov through his experiments with dogs [7]. Another example is the *Little Albert* experiment [8].

Table 1.1: Classical conditioning

| **Period** | **Stimulus** | **Response** |
|---|---|---|
| Before conditioning | US (food) | UR (salivation) |
| Before conditioning | NS (bell) | No response |
| During conditioning | US+NS | UR (salivation) |
| After conditioning | NS=CS | CR (salivation) |

Ringing of a bell is an example of a *neutral stimulus* (NS), which becomes the *conditioned stimulus* (CS). This is paired with an *unconditioned stimulus* (US), for instance, the pleasure from eating food. Pavlov's dog inherently has an *unconditioned* *response* (UR) to food (US), which is salivation. By frequently associating the ringing of a bell (CS) with the presence of food (US), Pavlov demonstrated that a conditioned response of salivation could be induced just by presenting the conditioned stimulus, though this effect is impermanent. Pavlov's dog would eventually start drooling whenever the bell is rung (Table. 1.1). In this case, the unconditioned stimulus or the food acts as a reward for the dog to learn the response to a previously neutral stimulus. Classical conditioning provides the principle to train or modify the behavior of learning agents through reinforcement learning.

Instrumental or operant conditioning is a bit complicated than classical conditioning due to the effects of positive and negative reinforcements that could affect behavior at the same time. Classical conditioning deals with only one of these reinforcements at any time. If a monkey is shut in a chamber with a lever, and could press the lever at any time to get a reinforcer like food or Heroin, the monkey is seen to get addicted to the reinforcer, and his lever-pressing action is reinforced [9]. This is an example of instrumental conditioning.

The idea is that such conditioning methods could be modeled and employed to train machines with a suitable learning mechanism.

## 1.3 Reinforcement Learning

Formally, reinforcement learning within a system could be defined as the process of developing a policy $\boldsymbol{\pi}$, internal to the learning agent, that maps observations or situations to actions, with the aim of maximizing a numerical global reward delivered to the system, in the long run, by the environment [3]. The environment of a learning agent includes passive response systems as well as other active agents that learn simultaneously while contributing to the learning of other agents. This opens up the interesting domain of multi-agent learning systems, and the context of multi-agent reinforcement learning (MARL). The fundamental differences in single-agent and multi-agent tasks are enumerated in Table. 1.2.

Table 1.2: Single-agent and multi-agent learning tasks

| **Single Agent Tasks** | **Multi-Agent Cooperative Tasks** |
|---|---|
| Agents function independently | Agents act cooperatively |
| Agents action directly determines the reward payoff | Environments payoff depends on the joint action of the participating agents |
| Centralized | Fault tolerant, decentralized |
| No interaction except with the environment | Interaction and simultaneous learning. One agents reactions (change in strategy etc.) affects the other agents. |

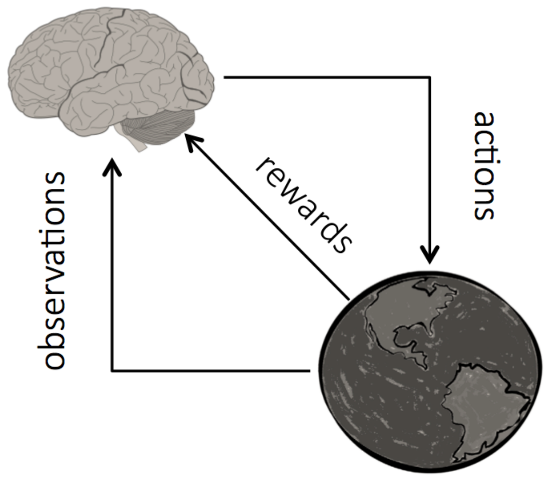
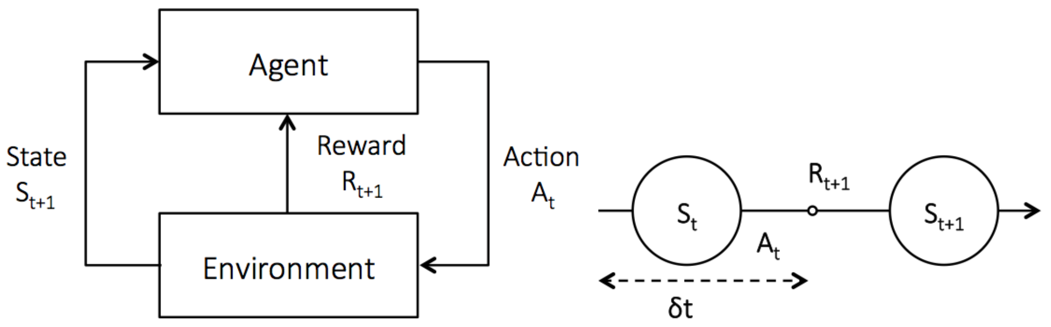


Figure 1.3: Interaction between a learning agent and its environment

In the reinforcement learning scenario, the interactions between a learning agent



(a) (b)

Figure 1.4: The formal reinforcement learning model

and its environment happen through three main steps (Figures. 1.3, 1.4a). At every time step $\boldsymbol{\delta(t)}$ (which need not be of fixed length), the agent performs an action $\boldsymbol{A_t} \in \boldsymbol{A(S_t)}$ that belongs to the set of actions applicable to the observed situation $\boldsymbol{S_t}$. This action affects the environment or itself in some way. Following this, it receives a new observation $\boldsymbol{S_{t+1}}$ from the environment along with a numerical reward $\boldsymbol{R_{t+\Delta t}}$ that evaluates the performed action in the context of the task at hand. In most realistic cases the reward is not delivered immediately following an action, but delayed by as much as few seconds. This is the single agent model of reinforcement learning.

The reward which leads to positive or negative reinforcement of performed actions is intended to help the agent learn a desirable observation-specific action. These rewards could be delivered instantaneously or be delayed. For example, in Pavlovian conditioning, the reward is presented to the dog after a delay, and the dog is able to successfully associate an observation to an action. Also, in interesting applications, an action at the present time could affect the observations provided by the environment in the future. This sequential decision making scenario resembles a Markov decision process (MDP), and can be modeled as one (Figure. 1.4b), with the condition that the transition probabilities and the reward probabilities are unknown or in other words stochastic from the perspective of the agent.

### *1.3.1 Principles and Elements*

Reinforcement learning is backed by the principle or notion that any real-life goal of an agent could be translated to the objective of maximizing its long-term reward. The internal goal of the agent is to create a policy $\boldsymbol{\pi}$ that maps observations to rewardable actions. Two processes aid the agent in this regard:

1. Exploration: The process of discovering and exercising several possible actions $\in \boldsymbol{A(S_t)}$, for a particular observation $\boldsymbol{S_t}$.
2. Exploitation: Maximizing reward from experience using a policy that maps observations to rewardable actions.

A good balance of *exploration* and *exploitation* results in efficient reinforcement and learning of observation-mapped behaviors in partially or completely known observations or environments.

For any interactive task, the corresponding reinforcement learning model of it includes a

1. State space ($\boldsymbol{S}$)
2. Action space ($\boldsymbol{A}$)
3. State transistion rules ($\boldsymbol{T_a(S, S')}$)
4. Reward policy ($\boldsymbol{R_a(S, S')}$)

The state space, $\boldsymbol{S}$, could include low-level sensations like temperature and position which could be observed by a mechanical agent, symbolic abstractions of configurations or an internal state of the agent, like happiness or surprise. The states could represent an agent's internal state or any change that occurs to its environment or other agents in its environment as a result of the agent's action(s). The action space, $\boldsymbol{A}$, could include low-level activations like voltage or current to the motors, high-level control operations like walking or rotating, or one that causes internal change of state like shifting focus to another task. As previously mentioned, in realistic reinforcement learning tasks that are perceived as Markov decision processes, the state transition rules $\boldsymbol{T_a}$ and rewarding policies $\boldsymbol{R_a}$ are usually stochastic to the agent.

## 1.4 Research Objectives

As discussed in this chapter, reinforcement learning is identified as a suitable mechanism to train machines in a brain-inspired way. Modeling and studying mathematical equivalents of reinforcement learning rules would improve our understanding of the brain. This research effort was initiated with the following objectives and goals.

1. Understand existing reinforcement learning frameworks that are applicable to spiking neural networks.

2. Evaluate these frameworks in the context of interactive tasks and applications, and identify problems and core limitations.

3. Improve the learning mechanisms with features that could handle the identified limitations.

4. Validate the enhancements for game-theoretic and control tasks, and provide future directions to further enhance and optimize the solutions.

# 2. COMPUTATIONAL NEURAL NETWORKS

The biological brain is a large neural network, and our goal is to model the learning agent as one. Neurons form the fundamental computational units in the brain, and they communicate with each other using spike trains, which are series of all-or-none signals at varying rates [10]. Several of these elements of the biological brain have to be modeled to implement brain-inspired learning techniques in machines.

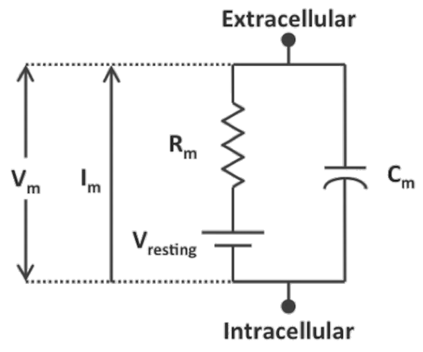
## 2.1 Spiking Single Neuron Models

An investigation of the existing computational models of neurons, which are the fundamental units of cognition in the brain, leads to the fact that third generation neural network models are particularly interesting because of their ability to add more realism to neural network simulation [11]. Networks composed of spiking neurons fall into this category. These models of spiking networks include, in addition to neuronal and synaptic states, the notion of time in their operation. There is a handful of popular spiking single-neuron models used in large scale network simulations. While the Hodgkin-Huxley (HH) model [12] is classic and good at reproducing significant diversity in the features of biological neurons, it is non-linear and complex with several model variables, which makes it tough to be simulated and analyzed mathematically. The leaky integrate-and-fire (LIF) neuron model [13], on the other hand, is very simple and computationally very inexpensive, but lacks the feature diversity. Izhikevich introduced a model [14] that replicates a wide set of features exhibited by diverse types of biological neurons, while being computationally simple. At this point, LIF and the Izhikevich neuron models are of interest to us especially because of their low computational complexity.

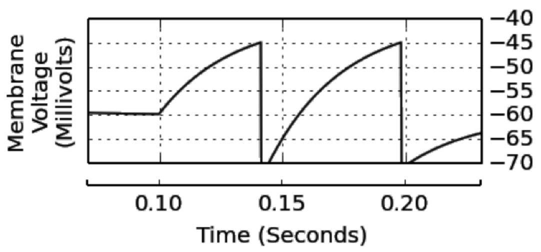
### *2.1.1 The LIF Model*

The leaky integrate-and-fire model models a spiking neuron as an RC circuit (Figure. 2.1). It accumulates aggregated current input with a leak until its voltage crosses a threshold, $\boldsymbol{v_{th}}$ (Eqn. 2.1). A spike is generated at the positive threshold-crossing (Eqn. 2.2), and at this point the membrane potential is reset to $\boldsymbol{v_r}$.

$$\begin{aligned} \boldsymbol{\tau_m \dot{v_m}} &= \boldsymbol{v_{\text{resting}} - v_m(t) + R_m I_m(t)} && (2.1) \\ \boldsymbol{v_m(t)} &= \boldsymbol{v_r if v_m(t) \geq v_{th}} && (2.2) \end{aligned}$$



(a)



(b)

Figure 2.1: Equivalent RC model of an LIF neuron

$\boldsymbol{v_m(t)}$ denotes the neuron membrane potential at time $\boldsymbol{t}$, $\boldsymbol{\tau_m}$ is the membrane time constant (=$\boldsymbol{R_m C_m}$), and $\boldsymbol{R_m}$ is the membrane resistance. While being computationally simple, this model does not generate biologically realistic spikes (Figure. 2.1b).

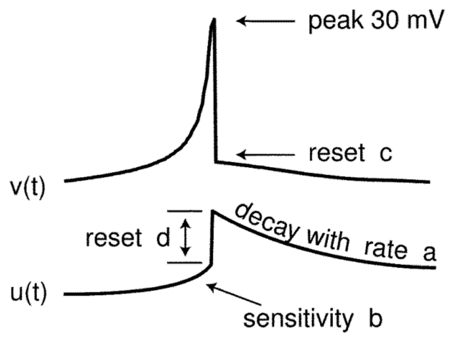


Figure 2.2: The Izhikevich single-neuron model features

### *2.1.2 The Izhikevich Model*

The Izhikevich model (Figure. 2.2) is comparable to the LIF model in terms of computational complexity [14]. But this model can exhibit spike patterns that closely mimic a variety of biological neurons (Figure. 2.3).

$$\begin{aligned} \dot{\boldsymbol{v}} &= \boldsymbol{0.04v^2 + 5v + 140 - u + I} && (2.3) \\ \dot{\boldsymbol{u}} &= \boldsymbol{a(bv - u)} && (2.4) \\ \boldsymbol{v} &= \boldsymbol{candu = u + difv \geq 30mV} && (2.5) \end{aligned}$$

Equations 2.3-2.5 summarize the model, where $\boldsymbol{v}$ represents the membrane potential of the neuron, and $\boldsymbol{I}$ represents the input current to the neuron through its dendrites. $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ are dimensionless parameters that give rise to spiking behaviors corresponding to different neuron types. Parameters $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ in the model could be adjusted to switch between different firing patterns (Figure. 2.3). The reset mechanism (Eqn. 2.5) is triggerred when the membrane voltage crosses 30 mV.

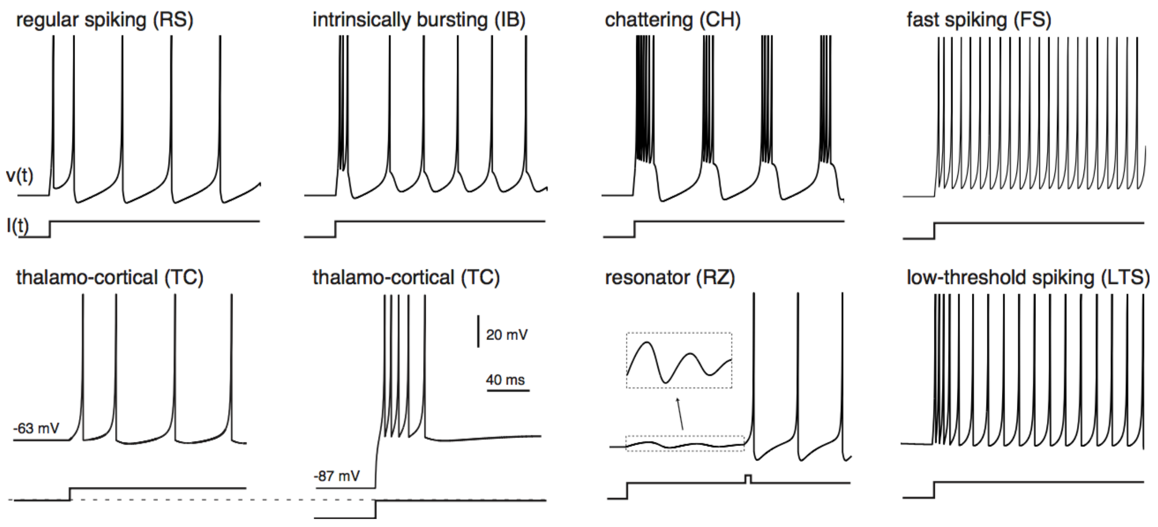


Figure 2.3: Spiking pattern diversity exhibited by Izhikevich's neuron model (*Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com*)

## 2.2 Synaptic Plasticity

A synapse represents the connection between two neurons, that permits one neuron to pass information to the other (Figure. 2.4). Biological neural networks are believed to learn memories and behaviors primarily by modifying their synaptic weights by an inherent ability called *Synaptic Plasticity*. Synaptic plasticity refers to the activity-dependent change in the strength of the synapses between neurons, and it is evidenced to be the primary cause that makes learning possible by rewiring the brain as interactions happen [15].

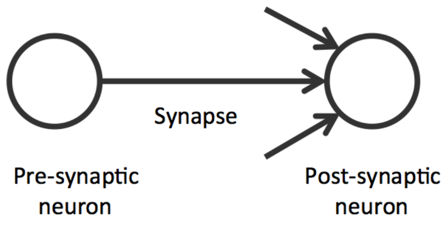


Figure 2.4: Representation of a synapse

Two terms become relevant here:

1. Long-Term Potentiation: The persistent strengthening of synapses that depend on recent patterns of neural activity.

2. Long-Term Depression: Activity-dependent weakening in the synaptic strength following neural activity or stimulus.

The interplay between potentiation and depression of synapses in a neural network leads to numerous configurations of the network, that results in different registrations of stimuli, and exhibition of behaviors by an organism.

### *2.2.1 Spike-Timing Dependent Plasticity*

In 1949, Hebb proposed that when a neuron persistently fires together with other neurons, the synaptic strength between this neuron and the others increases [16]. A more popular saying is, *Neurons which fire together, wire together.* A concept of synaptic plasticity called the *spike-timing dependent plasticity* (STDP) [17], introduced by M.M. Taylor in 1973, provides a substrate to Hebbian learning, in that it proposes that if synapses which have a pre-synaptic spike occur just before a post-synaptic one were potentiated more than those which have the reverse timing, there would be an efficient memory recording of the input patterns in the network. STDP

$$\mathrm{STDP}(\delta t) = \begin{cases} A_+ e^{-\delta t/\tau_+}, & \text{if } \delta t \geq 0 \\ -A_- e^{\delta t/\tau_-}, & \text{if } \delta t < 0 \end{cases}$$

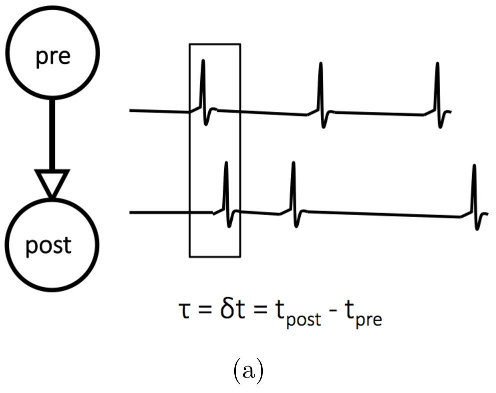
Figure 2.5: STDP function

is an experimentally verified classic technique in spiking networks of neurons [18]. It strengthens or weakens synapses based on the relative timing between a pre-synaptic and post-synaptic spike (Figure. 2.6).

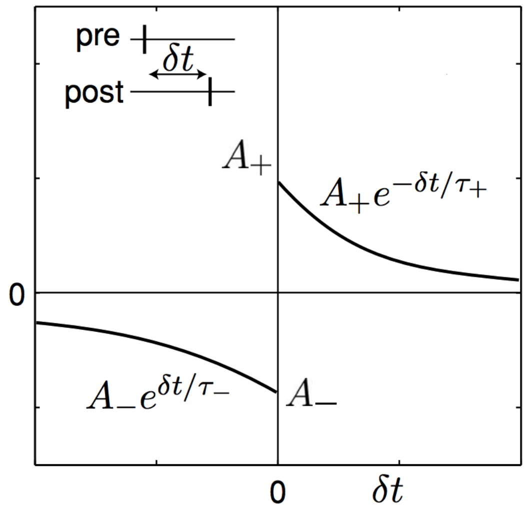
The change of strength, $\Delta w_j(t)$, of a synapse from a pre-synaptic neuron $j$ is dependent on the spike times at the presynaptic neuron and the post synaptic neurons. If $t_j^{n1}$ represents the arrival times of the presynaptic spikes, and $t_i^{n2}$ represents the arrival times of spikes at the $i^{th}$ post-synaptic neuron, then the total change in weight of the synapse is given by the equation 2.6. $n1$ and $n2$ represent the spike indices of the corresponding neurons [19].

$$\Delta w_j(t) = \sum_{n1=1}^{N} \sum_{n2=1}^{N} \mathrm{STDP}(t_i^{n2} - t_j^{n1}) \tag{2.6}$$

The STDP function is an exponential piecewise function (Figure. 2.5). The time constants $\tau_+$ and $\tau_-$ are usually the same, and are in the order of 20 ms. $A_+$ and $A_-$ scale the amount of potentiation and depression respectively, and are usually set to 1, although they could be made dependent on the current value of the synaptic strength. STDP has been used widely to selectively and meaningfully potentiate synapses in a network [20][21].



(a)



(b)

Figure 2.6: Spike-timing dependent plasticity (STDP)

# 3. REINFORCEMENT LEARNING IN SPIKING NETWORKS

## 3.1 Dopamine-modulated Synaptic Plasticity

The previous chapters introduced the fact that synaptic potentiations within the internal structures of the brain are influenced by the neuronal spike-timings as well as by the release and presence of a global neuromodulator called dopamine (DA). Dopamine plays an important role in the long-term potentiation (LTP) and depression (LTD) of synapses [22]. Reward-modulated synaptic plasticity or reinforcement learning in aritificial spiking neural networks is based on a simple hypothesis that STDP potentiation and depression could be modulated by a global reward signal (modeling the dopamine), and thus be made selective [25][26]. In other words, dopamine is hypothesized to gate STDP-induced synaptic potentiations. When STDP conditions (near-coincident firing of pre and post synaptic spikes) are met during spiking activity, whether or not a synapse is potentiated depends on the presence of the global, extracellular dopamine in the network.

Modulated-STDP or DA-STDP deals with the application of a reward (modeling

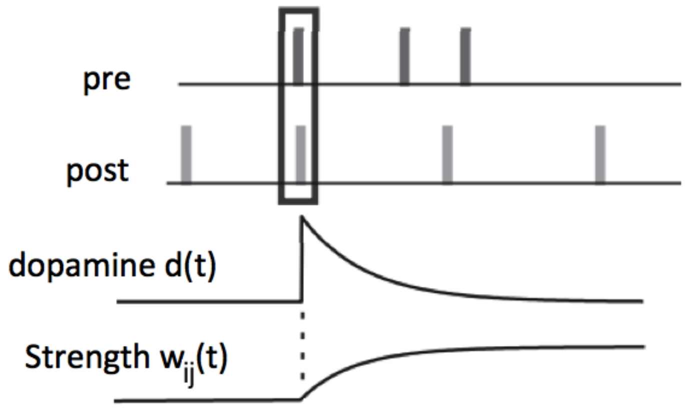


Figure 3.1: Dopamine-modulated synaptic plasticity

dopamine delivery) to enhance the potentiation of synapses by STDP. The idea is that, if synapses are potentiated only when the network receives a reward, activations related to the corresponding stimulus-action pair get potentiated, thereby strengthening a single response to a stimulus. The variability of the network is thus reduced leading to a registration of the input configuration. The effects of the reward are in terms of the long-term potentiations, which are exploited when a similar stimulus is presented later (Hebbian learning). This is exploitation of knowledge. When there is nil or minimum reward, the synapses do not get strengthened and the network is thereby allowed to behave on its own without any reinforcement. This permits variability in the response of the network, leading to exploration of behavior. This idea sits in the center of reward-modulated synaptic plasticity for reinforcement learning:

$$\frac{dw_{ij}}{dt} = \gamma \Delta w_{ij}^{\mathrm{STDP}} d(t) \tag{3.1}$$

$$\dot{d(t)} = \frac{-d(t)}{\tau_d} + \mathrm{DA}(t) \tag{3.2}$$

Figure. 3.1 illustrates the effects of dopamine in the modulated-STDP technique introduced in [25], exlcluding the influence of the *eligibility trace* (introduced later). The change in strength $\Delta w_{ij}$ of a synapse between neurons $j$ and $i$ is the product of a learning rate factor ($\gamma < 1$), the STDP-induced weight change ($\Delta w_{ij}^{\mathrm{STDP}}$), and the dopamine signal $d(t)$ (Eqn. 3.1). $\mathrm{DA}(t)$ models the actual source of extracellular dopamine in the network, while $d(t)$ represents its model of decaying delivery to the synapses. $\tau_d$ is the time constant for the decay, and is in the order of 20 ms [25].

This is a simple mathematical model of how dopamine could affect synaptic plasticity. There are multiple limitations in applying this learning rule to practical tasks, which are discussed in the forthcoming sections.

## 3.2 The Distal Reward Problem

In the hippocampal region of the brain, tetanus is seen to induce potentiation (LTP/LTD) of synapses, which are enhanced or modulated by the D1 receptors that release dopamine. It is observed that the effects of tetanus on the resulting long-term potentiations disappear if dopamine is delivered seconds later [23]. At a higher level, in practical applications, rewards are not delivered instantaneously to the learning agent. There are delays in evaluating the effects of actions performed, and hence in the delivery of rewards. For example, in Pavlovian and instrumental conditioning, rewards are delivered to the dog seconds after rewardable actions for potentiation, which are still effective in conditioning the dog's response. These delayed rewards are somehow effective in strengthening the correct set of synapses. The brain determines which firing patterns or synpases to reward when the patterns are no longer active at the rewarding instant. To persist these patterns the brain cannot freeze its state until rewards are delivered. The problem therefore is to pick or remember the reward-worthy synapses, for an earlier action, during an active episode of neural activity. This is also called the *temporal credit assignment problem*.

Learning rules represented by equations 3.1-3.2 clearly would not be effective in the potentiations of the correct synapses, if the rewards are delayed. This is because, the effect of STDP becomes negligent when the actual reward is delayed, and thus the potentiation could not be effectively modulated by the incoming reward.

## 3.3 Eligibility Traces

Dopamine modulation of tetanus-induced LTP in the Hippocampus [24] suggests that, instead of the dopamine delivery and modulation being expected instantaneously, there must be a short time window after the synaptic activations within which they are allowed to be delivered and thus effect modulation of the previously active synapses.

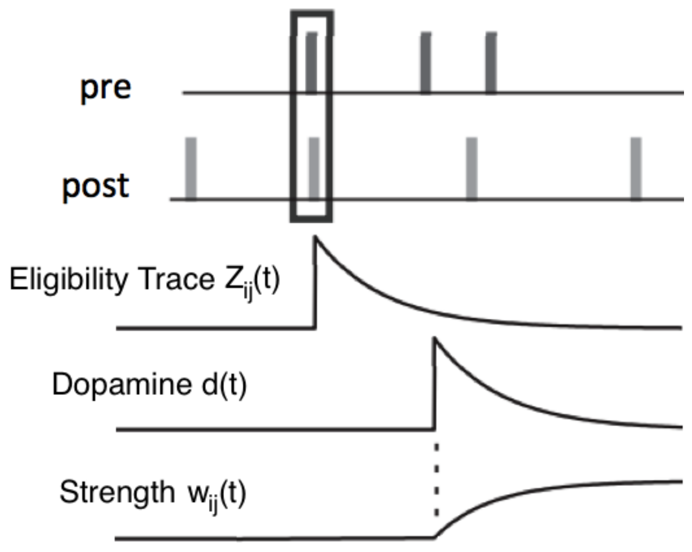


Figure 3.2: Dopamine-modulated synaptic plasticity with eligibility trace

Eligibility traces are a common and popular solution to the distal reward problem [25][26]. Each synapse in the network, between a presynaptic neuron $\boldsymbol{j}$ and postsynaptic neuron $\boldsymbol{i}$, is associated with a trace signal $\boldsymbol{Z_{ij}(t)}$, which is stepped up or down when near-coincident spikes occur, depending on the relative timing (Figure. 3.2). The traces decay with a time-constant $\boldsymbol{\tau_z}$. They aid the system in remembering active synapses during an earlier point in time, and thus lead to effective LTP/LTD when the reward/punishement is delivered with a delay. Of course, there is a finite waiting period, that depends on $\boldsymbol{\tau_z}$, during which the efficacy of this memory fades, and beyond which there is no effect of reward or punishment on those synapses. Therefore, the idea with eligibility traces is to combine STDP with a decaying memory trace of synaptic activations so that delayed rewards are still effective in potentiating synapses that were active a few seconds earlier.

Two related prior-arts implementing and validating the role of eligibility traces in DA-modulated STDP are detailed in [25] and [26]. They are summarized in the following subsection.

### *3.3.1 Izhikevich's Eligibility Trace*

#### *3.3.1.1 Implementation*

Izhikevich [25] defines two state variables for a synapse - the synaptic strength $\boldsymbol{w}$, and a synaptic tag $\boldsymbol{c}$ that models an enzyme whose activation plays an important role in plasticity (Figure. 3.3). The synaptic tag is a slow decaying process (Eqn. 3.3), and represents the eligibility trace.

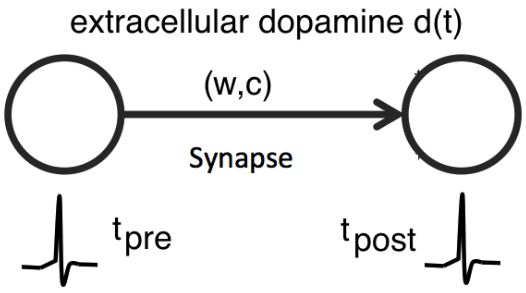


Figure 3.3: Izhikevich's model of a synapse with eligibility trace

$$\dot{\boldsymbol{c}} = \frac{-\boldsymbol{c}}{\boldsymbol{\tau_c}} + \mathbf{STDP}(\boldsymbol{\tau})\boldsymbol{\delta}(\boldsymbol{t} - \boldsymbol{t}_{\mathbf{pre/post}}) \tag{3.3}$$

$$\dot{\boldsymbol{w}} = \boldsymbol{cd} \tag{3.4}$$

$$\boldsymbol{\tau} = \boldsymbol{t}_{\mathbf{post}} - \boldsymbol{t}_{\mathbf{pre}} \tag{3.5}$$

The dynamics of the eligibility trace and synaptic strength are captured by equations 3.3-3.5. $\boldsymbol{d(t)}$ represents the extracellular concentration of dopamine in the network, as computed using Equation 3.2, and $\boldsymbol{\tau_d}$ (usually 0.2s) is its time constant for decay. Every reward delivery to the network is represented by a step increase of dopamine level by 0.5 uM (Figure. 3.2). $\boldsymbol{\delta(t)}$ is the dirac delta function which increases the synaptic tag by a step that is exponentially proportional to the difference in the relative timing of the pre and post synaptic spikes.

The synaptic tag $\boldsymbol{c}$ decays with a time constant of $\boldsymbol{\tau_c} = 1$s, and this models the sensitivity of the synaptic plasticity to a delayed reward.

#### *3.3.1.2 Experimental Setup and Validation*

A network of 1000 Izhikevich neurons was employed to validate the DA-STDP rule proposed in [25]. It consisted of **80%** excitatory neurons (regular spiking (RS); Figure. 2.3) and **20%** inhibitory neurons (fast spiking (FS)). Each neuron is connected to another neuron with a probability of 0.1 (100 synapses per neuron on average). The synaptic weights were limited to 0 to 4mV. For spike-timing dependent plasticity, $\boldsymbol{A_+}$ and $\boldsymbol{A_-}$ were set to 1 and -1.5 respecively. The neurons received random input currents between 0 and 30 mA at each timestep of 1ms.

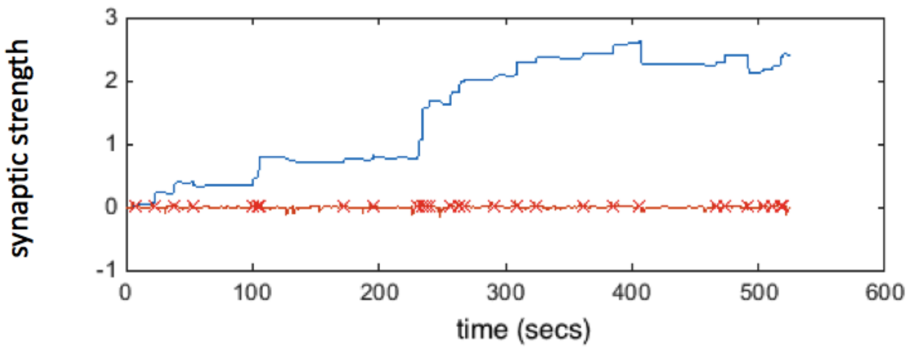


Figure 3.4: Synapse potentiated by Izhikevich's eligibility trace

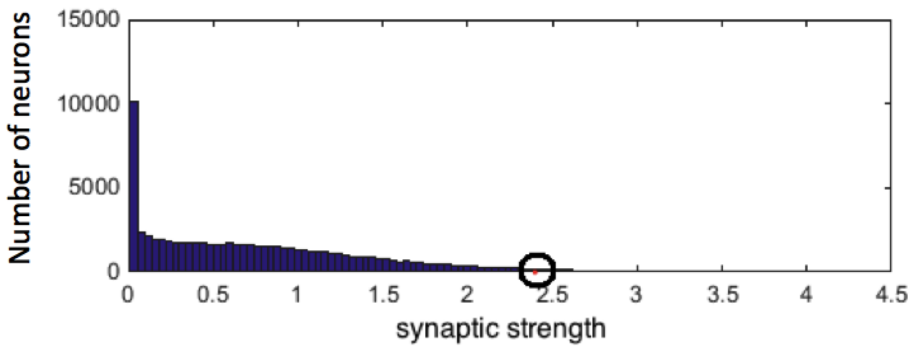


Figure 3.5: A histogram of the synaptic strengths in the network

The objective of this experiment was to validate dopamine-based reinforcement of a single synapse during spiking activity. A single excitatory synapse was randomly chosen from the network, its strength was set to 0, and was monitored. Every time the postsynaptic neuron fires within 10 ms from the firing of the presynaptic neuron, the dopamine concentration is stepped to 0.05 uM, not instantaneously, but with a delay that is randomly picked between 1 and 3 seconds. Such coincident firing would initially occur randomly, but due to reinforcement of the synapse, the frequency increases eventually, leading to a very strong response (maximum synaptic strength) at the end of the experiment (Figure. 3.4). The red curve represents the dopamine concentration $\boldsymbol{d(t)}$, and the red cross-marks represent time-points of dopamine delivery at source ($\boldsymbol{DA(t)}$) to the network. The blue curve is the strength of the synapse $\boldsymbol{w(t)}$, which is seen to get continuosly reinforced.

In Figure. 3.5, which is a histogram of the synaptic weights in the network, the blue circle points to the strength of the selected synapse, comparing it with the strengths of all the other synapses in the network.

### 3.3.2 Florian's Eligibility Trace

#### 3.3.2.1 Implementation

Another implementation of eligibility traces for handling the credit assignment problem is provided in [26]. The learning rule (Equations 3.6-3.7) was derived from the gradient-descent based online partially observable Markov decision process (OLPOMDP) algorithm introduced in [27]. OLPOMDP assumes that the interaction between the learning agent and its environment to be a partially observable Markov process.

$$\dot{w_{ij}} = \gamma R(t) Z_{ij}(t) \tag{3.6}$$

$$\tau_z \frac{dZ_{ij}(t)}{dt} = -Z_{ij}(t) + P_{ij}^{+}(t) f_i(t) + P_{ij}^{-}(t) f_j(t) \tag{3.7}$$

Aligning with the generic reward-modulated learing rule introduced in the previous section, the change in strength $w_{ij}$ of a synapse, between a presynaptic neuron $j$ and a post-synaptic neuron $i$, depends on the learning factor $\gamma$, the actual reward $R(t)$ and an eligibility trace $Z_{ij}$. In this implementation, STDP is integrated into the eligibility trace signal. $\delta t$ is the discrete time-step at which the network is simulated, usually set to 1 ms, and $\tau_z$ is the decay time-constant for the eligibility trace, usually set to 20 ms.

$$P_{ij}^{+}(t) = P_{ij}^{+}(t - \delta t) e^{-\delta t/\tau_{+}} + A_{+} f_j(t) \tag{3.8}$$

$$P_{ij}^{-}(t) = P_{ij}^{-}(t - \delta t) e^{-\delta t/\tau_{-}} + A_{-} f_i(t) \tag{3.9}$$

$P_{ij}^{+}(t)$ and $P_{ij}^{-}(t)$ track the influence of pre and post-synaptic spikes respectively by summing up their effects over time. For every spike of a presynaptic and postsynaptic neuron, $\boldsymbol{P_{ij}^{+}(t)}$ and $\boldsymbol{P_{ij}^{-}(t)}$ are updated according to equations 3.8 and 3.9. Also, for every presynaptic spike at time $\boldsymbol{t}$, $\boldsymbol{f_j(t) = 1}$, and for every postsynaptic spike, $\boldsymbol{f_i(t) = 1}$, these parameters otherwise being zero. The dynamics of the various variables in this mechanism is demonstrated in Figure 3.6, which provides an intuition to the way STDP and eligibility traces are integrated into this implementation.

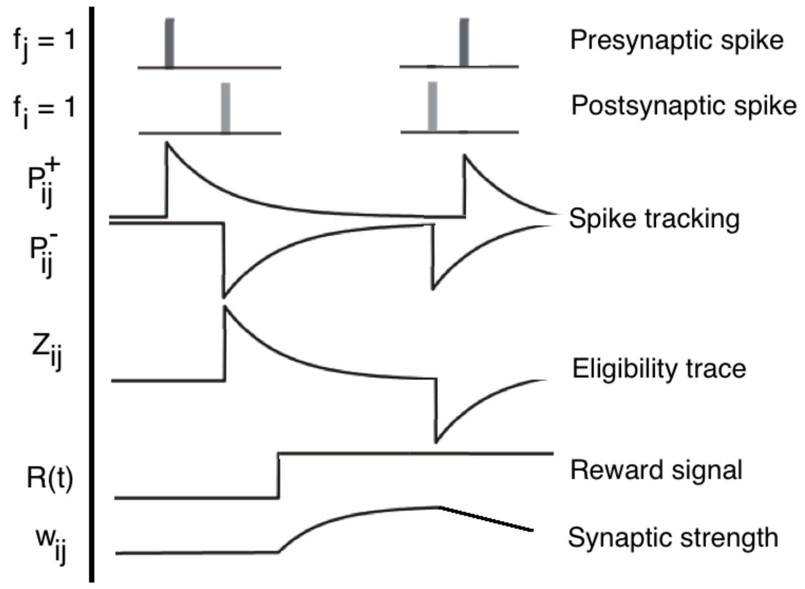


Figure 3.6: Dynamics of the Florian DA-STDP variables

From Figure 3.6, it could be inferred that the result of the eligibility trace implementation in [26] is quite similar to the one by Izhikevich in [25]. This DA-STDP mechanism has been demonstrated to work for single-action instrumental conditioning tasks, one of which is described in the following section.

### *3.3.2.2 Experimental Setup and Validation*

The implementation of the DA-STDP technique described in [26] consisted of a feedforward network of three layers composed of LIF neurons - 60 in the input layer, 60 in the hidden layer and 1 neuron in the output layer (Figure. 3.7). These neurons had a resting potential of -70mV, a time constant $\boldsymbol{\tau}$ of 20ms, and a spike threshold of -54mV. The network was simulated in a time resolution of $\boldsymbol{\delta(t)}$ =1ms. For spike-timing dependent plasticity, parameters $\boldsymbol{\tau_+}$ and $\boldsymbol{\tau_-}$ were set to 20ms, and $\boldsymbol{A_+}$= 1, $\boldsymbol{A_-}$=-1, while the eligibility trace time constant $\boldsymbol{\tau_z}$ was set to 20ms.

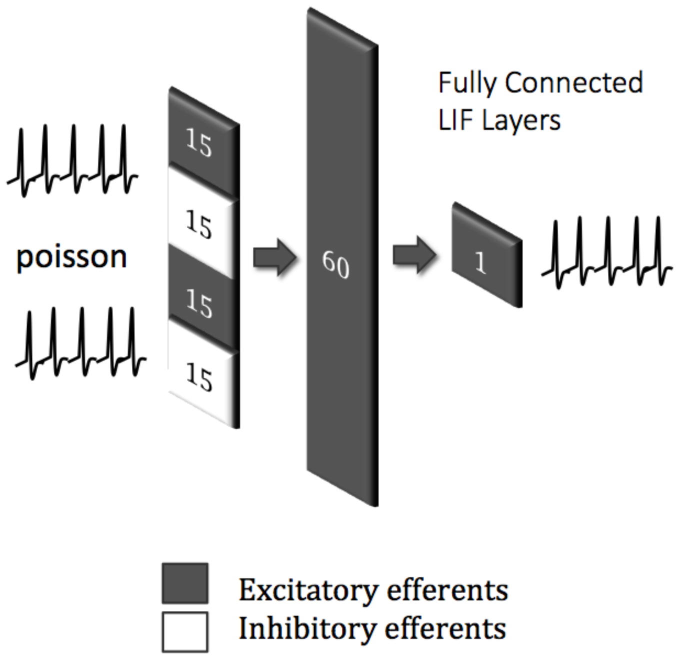


Figure 3.7: Network design for the XOR task

Each layer in the network was fully connected to its next layer. Each binary input was associated with a group of 30 neurons (50% inhibitory), and the binary value was coded as the spiking rate at the input. The absolute value of synaptic weights were limited to, and clipped at 2.5 mV. Poisson spike trains were generated at a specific rate depending on the input value, and were fed to the input layer. Binary value 0 was represented by a poisson spike rate of 0 Hz, and 1 by spikes 40 Hz. For example, for the (0,1) case, the first group of neurons were all fed poisson spikes at 0 Hz, and the second group of 30 neurons were fed spikes at 40 Hz.

The input binary values coded as poisson spike rates were presented to the input layer, and as spikes arrived at the output layer, rewards were delivered according to the following rules. At the output neuron, the system was rewarded ($\boldsymbol{R(t) = +1}$) for every spike, for cases (0,1) and (1,0). There was a punishment of ($\boldsymbol{R(t) = -1}$) for every spike at the output neuron, for cases (0,0) and (1,1). The objective was to create a policy that results in the following input-output mappings: [(0,0)$\Rightarrow$0, (0,1)$\Rightarrow$1, (1,0)$\Rightarrow$1, (1,1)$\Rightarrow$0]. When the spike rate at the output neuron was higher than a threshold, the response was decoded as 1, and when it was low, it was decoded as 0. Each epoch, involving input presentation and decoding, lasted for 500ms. An episode, leading to satisfactory learning, consisted of 200 training epochs. At the end of the training phase, there was a testing phase where the learning rate was set to 0, and the number of spikes at the output neuron was counted for each case, as the inputs were presented. The number of spikes for (0,1) and (1,0) was expected to be more than that for the other cases. The results shown in Figure 3.8 validates that the system was successfully conditioned to learn the XOR function. The dynamics of the rewards delivered to the network, which is a measure of the total number of expected spikes, is displayed in Figure 3.9.

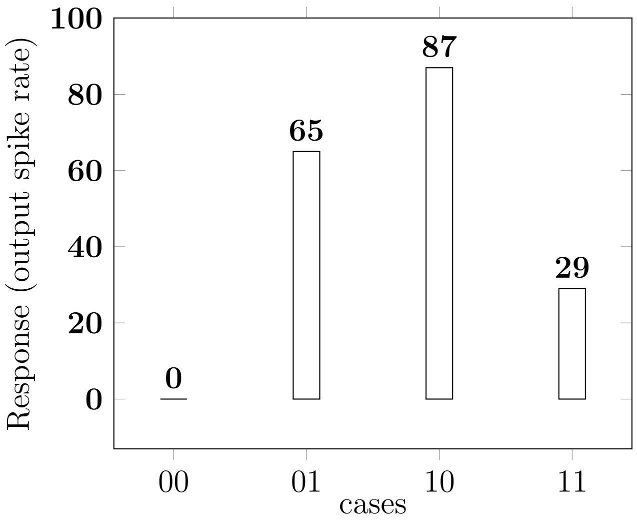


Figure 3.8: Output spike rates for the XOR task

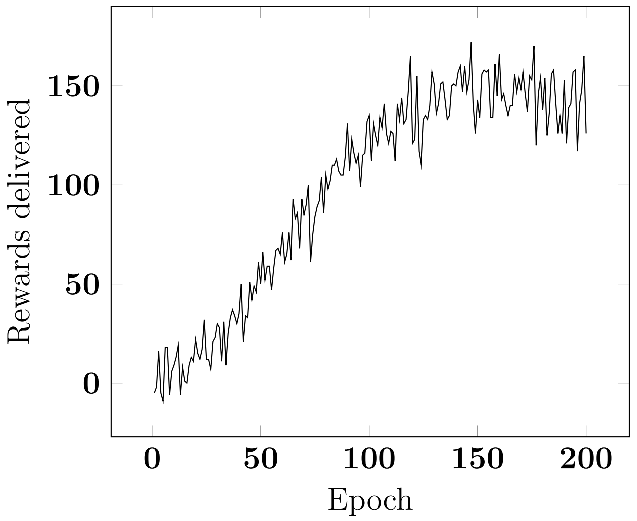


Figure 3.9: Reward profile for the XOR task

# 4. LEARNING LIMITATIONS

The dopamine-modulated STDP learning rules introduced in the previous chapter were validated to lead to network behaviors that resembled instrumental conditioning. However, the experiments involved in the said validations were simple, involving a small network with just one output neuron. In this chapter, we scale the complexity of the network to a small extent, to further study the efficacy of the DA-STDP techniques.

## 4.1 XOR Task Revisited

In this section, the XOR task discussed in the previous chapter to validate the Florian eligibility trace [26] is revisited. The network was upgraded to have two output neurons, with all the other parameters kept the same. The network decision was decoded as a 1 if the rate of spikes at output $\boldsymbol{O_1}$ was greater than the $\boldsymbol{O_2}$, and 0 if otherwise. Florian and Izhikevich's learning rules were separately applied for this task, with different network architectures, to validate the efficacy of the learning rules in this task. This task is interesting because of the need for spatial selectivity in potentiations. For each input cases, only one output neuron is expected to have a stronger response, and that means that the synapses contributing to spikes only at one output neuron must be reinforced.

### *4.1.1 Florian's Learning Rules*

The feedforward network, used for the validation in the previous chapter, was upgraded to have two outputs (Figure 4.1), with all the other parameters kept the same. If the rate at output $\boldsymbol{O_1}$ was greater than the $\boldsymbol{O_2}$ the response was 1, and 0 if otherwise. The network was simulated in a time resolution of $\boldsymbol{\delta(t)}$ =1ms, and the learning rate $\boldsymbol{\gamma}$ was set to 0.01.

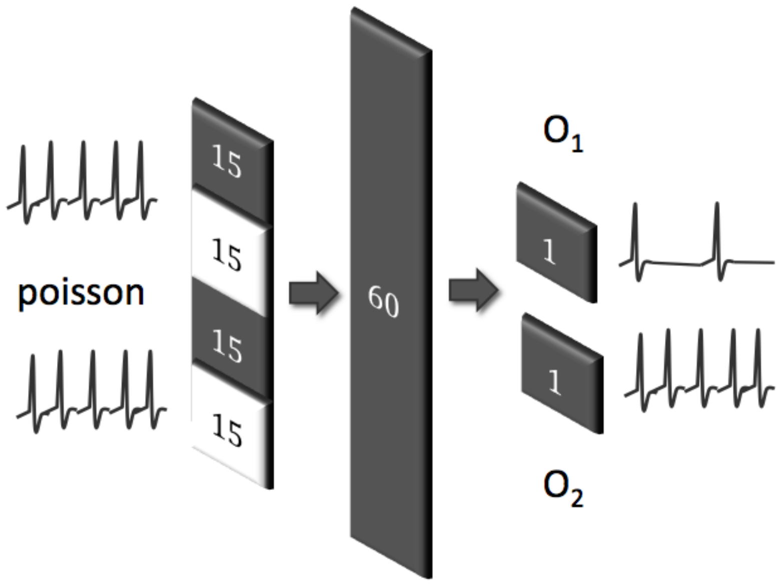


Figure 4.1: Upgraded network design for the XOR task

The rewards were instantaneous. For cases (0,1) and (1,0), for each spike at $\boldsymbol{O_1}$ a reward of +1 was delivered to the network, and for each spike at $\boldsymbol{O_2}$, a reward of -1 was delivered, with the goal of strengthening the response at $\boldsymbol{O_1}$, which represents an output value of 1. For cases (0,0) and (1,1), spikes at $\boldsymbol{O_2}$ were rewarded, while those at $\boldsymbol{O_1}$ were punished. The synaptic strengths were modulated based on the Florian learning rules (Equations 3.6-3.9).

The simulation was run for 200 epochs, with each epoch lasting for 500ms. The resulting spike rates at outputs $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$ for each of the input cases are shown in Figure. 4.3.

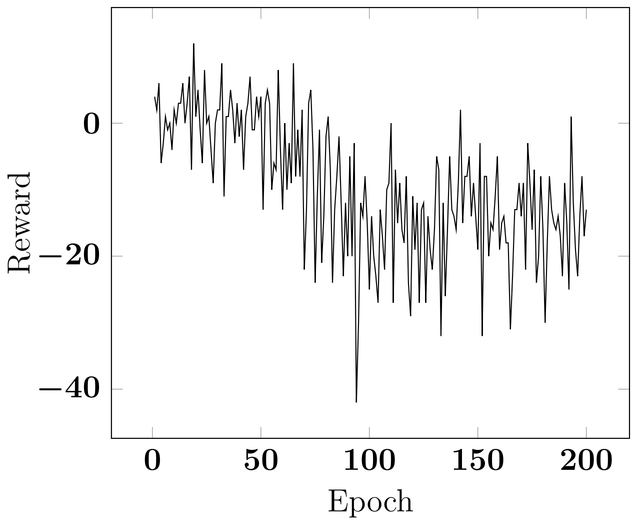


Figure 4.2: Reward profile for XOR task with 2 outputs

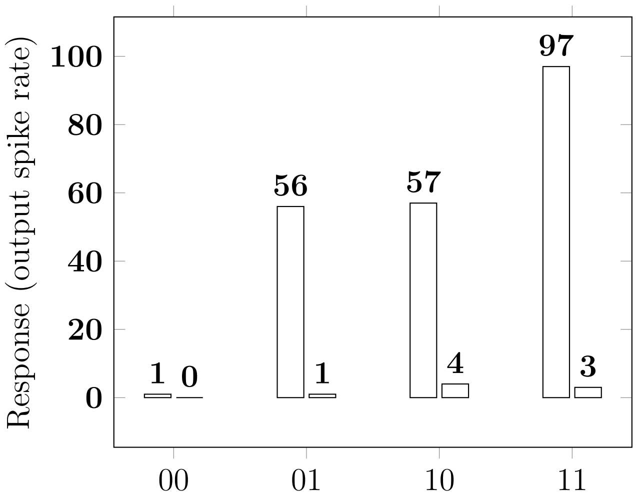


Figure 4.3: Output spike rates for XOR task with 2 outputs

From Figure 4.2 we see that the reward profile keeps oscillating, indicating that the network struggles to learn contradicting responses for spatially separated neurons in the case of instantaneous rewards. Figure 4.3 shows that the network learning gets biased over time, leaning towards one specific response. Since the reward is global, a neuron behaving incorrectly can be rewarded because most of its neighbors behaved correctly, and vice versa.

### *4.1.2 Izhikevich's Learning Rules*

In another similar experiment with Izhikevich's learning rules [25], a network of 1000 Izhikevich neurons were used to implement the XOR task. The network consisted of **80%** excitatory neurons and **20%** inhibitory neurons. Each neuron was connected to another with a probability of 0.1 (100 synapses per neuron on average). The synaptic weights were limited between 0 to 4mV. For spike-timing dependent plasticity, $\boldsymbol{A_+}$ and $\boldsymbol{A_-}$ were set to 1 and -1.5 respecively. The neurons received random input currents between 0 and 30 mA at each timestep of 1 ms.

Four groups, $\boldsymbol{S1}$, $\boldsymbol{S2}$, $\boldsymbol{A}$ and $\boldsymbol{B}$, of 50 neurons each were selected from the network. These groups represented the stimulus X, stimulus Y response 0 and response 1 respectively. Each epoch involved a presentation of stimulus, a superthreshold current, to the groups $\boldsymbol{S1}$ and $\boldsymbol{S2}$, for one of the four input cases. The number of spikes in groups $\boldsymbol{A}$ and $\boldsymbol{B}$ were counted during a 20 ms window, and were used to decide on the response from the network. The network response was marked 0 if the number of spikes in group $\boldsymbol{A}$ was twice more than that in group $\boldsymbol{B}$ ($\boldsymbol{|A| > 2|B|}$). Similarly, the response was marked 1 if the number of spikes in group $\boldsymbol{B}$ was twice more than that in group $\boldsymbol{A}$ ($\boldsymbol{|B| > 2|A|}$). Rewards, delayed by 1-3s, were delivered according to the expected outcomes [(0,0)⇒0, (0,1)⇒1, (1,0)⇒1, (1,1)⇒0]. A positive reinforcement of 0.05 uM was delivered for the correct response, and a negative reward of 0.05 uM for the wrong response. The network was trained for 250 seconds with each epoch starting every 10 seconds.

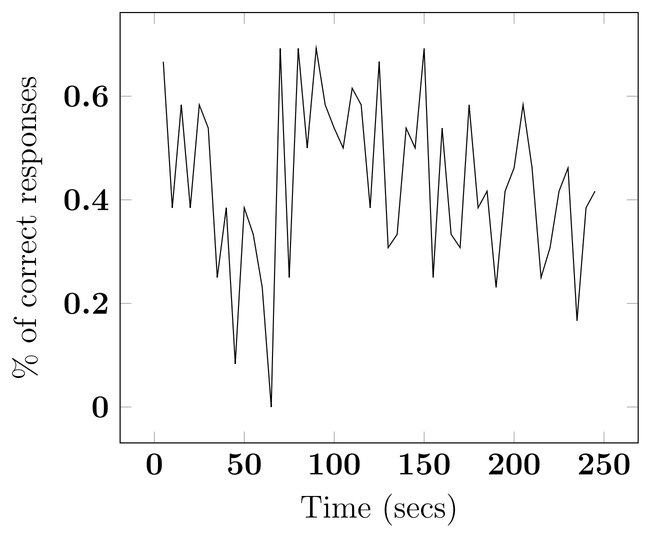


Figure 4.4: Learning performance

The percentage of correct responses for each set of four cases is plotted in Figure. 4.4. Quite similar to the Florian experiment, this network also failed to learn the mappings, and it continually oscillated between learning contradictory responses.

## 4.2 Spatial Selectivity in Potentiation

Input specificity is the property of a neural network by which it stores knowledge via synaptic transmissions that are driven just by input-specific changes [28]. This makes it possible for the network to have a high storage capacity, given that there are a large number of neurons ($\mathbf{10^{11}}$ in the brain) and synapses per neuron ($\mathbf{> 10^4}$) in the network. In reinforcement learning, such effective storage of patterns is largely dependent on the changes in input patterns and rewards affecting only a subset of synapses that actually contribute to an action or storage of a memory pattern. During neural activity, several synapses across the network become active. While a subset of them contribute strongly leading to rewardable or punishable behavior, the other synapses do not affect the decisions of the network. These synapses (Figure.

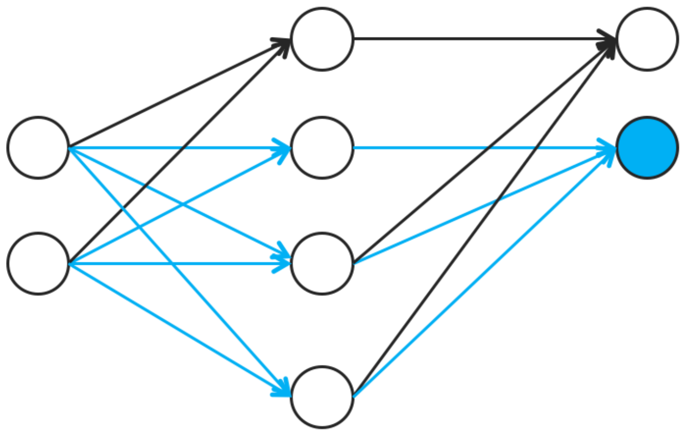


Figure 4.5: Spatial selectivity in potentiation

4.5 in black) ideally should not be potentiated or depressed because of the decisions of the network as a whole. Only the synapses that contribute to the network decision (Figure. 4.5 in blue) must be made to face the consequences of those decisions. This idea forms the core of the problem that is discussed in the following section.

The experiments discussed illustrate the logical network partitioning challenge for credit assignment, which remains to be solved. A bad neuron cannot be rewarded just because its neighbors are correct, and vice versa. This is the *spatial credit assignment problem*, dealing with making credit assignments precisely and selectively, and it is currently an important direction of research in reinforcement learning.

## 4.3 Response Numbness Problem

Exploration offers the only chance for a learning agent to discover rewards before exploiting knowledge. Response numbness is a phenomenon in reinforcement learning systems, in which due to inactivity or continuous supression of a response, neurons belonging to that particular action group are never allowed to explore enough to get rewarded. They remain supressed. The underlying reason is that the strength of synapses between these neurons saturate at their lower limit from learning. Spike variability is thus prevented, thereby creating a hurdle for exploration and learning. In an interactive task, this also happens when some states are observed by the system very rarely, when compared to several other states.

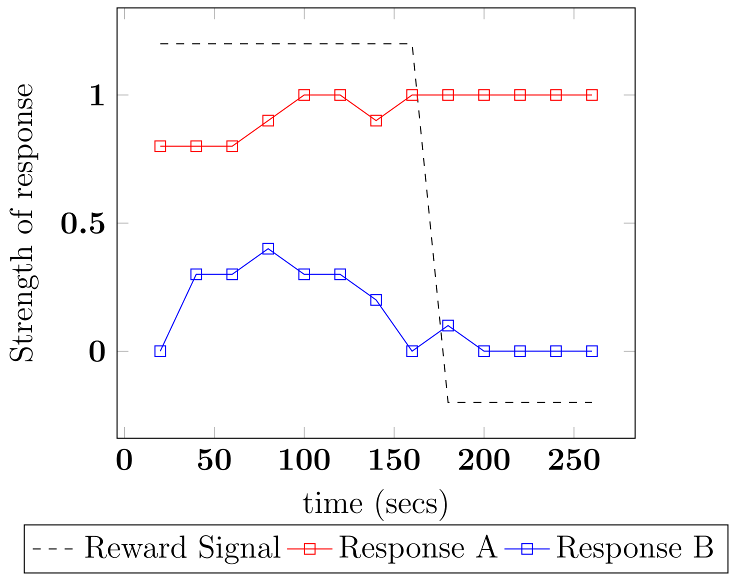


Figure 4.6: Response numbness

A network of 1000 neurons was used to perform an instrumental conditioning experiment. The network consisted of **80%** excitatory neurons and **20%** inhibitory neurons. Each neuron was connected to another with a probability of 0.1 (100 synapses per neuron on average). The synaptic weights were limited to 0 to 4mV. For spike-timing dependent plasticity, $\boldsymbol{A}_+$ and $\boldsymbol{A}_-$ were set to 1 and -1.5 respecively. The neurons received random input currents between 0 and 30 mA at each timestep of 1 ms.

Three groups, $\boldsymbol{S}$, $\boldsymbol{A}$ and $\boldsymbol{B}$, of 50 neurons each were selected from the network. These groups represent the stimulus, response $\boldsymbol{A}$ and response $\boldsymbol{B}$ respectively. Each epoch involved a presentation of stimulus, a superthreshold current, to the group

$\boldsymbol{S}$. The number of spikes in groups $\boldsymbol{A}$ and $\boldsymbol{B}$ were counted during a 20 ms window, and were used to decide on the response from the network. The network response is marked $\boldsymbol{A}$ if the number of spikes in group $\boldsymbol{A}$ is twice more than that in group $\boldsymbol{B}$ ($|\boldsymbol{A}| > \mathbf{2}|\boldsymbol{B}|$). Similarly, the response is marked $\boldsymbol{B}$ if the number of spikes in group $\boldsymbol{B}$ is twice more than that in group $\boldsymbol{A}$ ($|\boldsymbol{B}| > \mathbf{2}|\boldsymbol{A}|$). Until 180 seconds, the network was rewarded dopamine (with a delay between 1-3s) for response $\boldsymbol{A}$, and after 180 seconds it was rewarded for response $\boldsymbol{B}$. By the time this reverse reinforcement started, the network had already learnt response $\boldsymbol{A}$ very strongly, that it became difficult for it to exhibit diversity after 180 seconds to explore and learn the other response ($\boldsymbol{B}$) (Figure. 4.6). The network will never be rewarded as long as its response was very strong for $\boldsymbol{A}$. This illustrates the response numbness problem.

The challenge here, therefore, is to find a mechanism to stop the agent from learning once an acceptable strength of desired behavior is reached during learning. This would prevent complete supression of responses during other observations.

# 5. ENHANCED DA-STDP WITH ATTENTIONAL FEEDBACK

Several enhancements made to the DA-STDP technique to handle the limitations discussed in the previous chapter are introduced here. The network design and learning methodologies that are part of this proposal are detailed in the following sections. The next section describes the type of neuron models and the network structure used. The section that follows it details the input encoding method, which is followed by an explanation of features developed to deal with some of the other problems with modulated-STDP scheme that were explained in the previous chapter.

## 5.1 Network Architecture

A feedforward network, with three or more layers, is proposed to be employed to model the learning agent. Feedforward networks are simpler than recurrent networks in terms of simulation complexity, and in terms of applying the proposed reinforcement learning rules. The learning strategies are not affected by the type of spiking neuron model, and therefore both LIF and the Izhikevich models could be used. A comparison of performances with these models is reported in Appendix A. The structure and arrangement of neurons in the input layer are discussed in the next section. The input layer ($\boldsymbol{N_i}$ neurons) and the first hidden layer ($\boldsymbol{N_h}$ neurons) are fully connected, whereas the last hidden layer and the output layers ($\boldsymbol{N_o}$ neurons) are sparsely connected to handle the spatial credit assignment problem. This is discussed later in this chapter. Each output neuron is connected to a neuron in the hidden layer with a probability of $\boldsymbol{1/N_o}$, where $\boldsymbol{N_o}$ is the number of output neurons (Figure. 5.1).

For complex tasks, there could be several hidden layers in the network modeling the agent. In these cases, between the hidden layers, the sparsity is set to 0.5. This means that a neuron in one hidden layer is connected to a neuron in the next hidden layer with a probability of 0.5. For tasks that require a winner-takes-all strategy at the output layer, for evaluating a single response, lateral inhibition is set up at the output layer. Inhibition is implemented using inhibitory synapses (negative weights) between pairs of output neuron groups. A certain percentage of neurons in one group are randomly connected to the other group in the pair through inhibitory synapses.

In tasks which require inclusion of the previous response of the agent in the evaluation of the next response (an example is provided in Chapter 5), partial recurrence is implemented by feeding the decision from the output layer to the input layer.

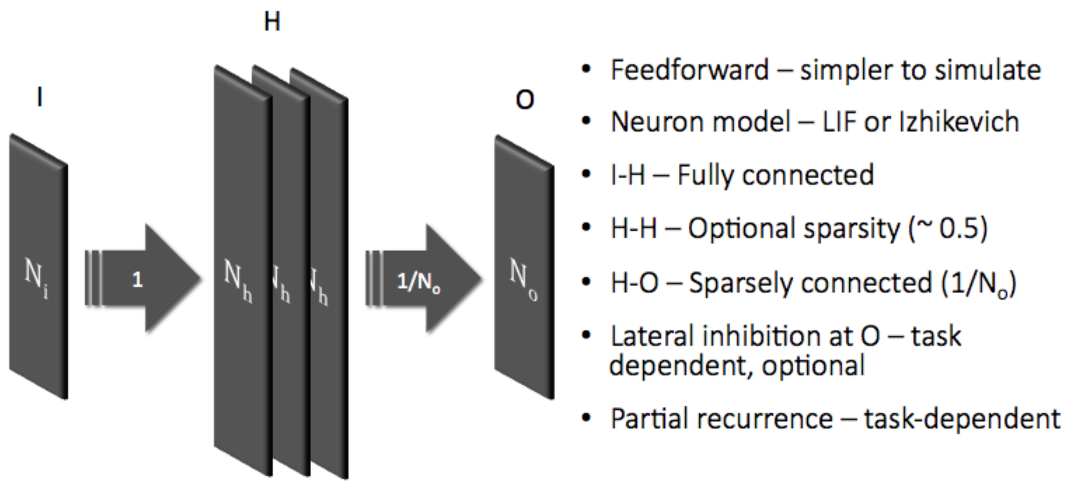

H
I
O
$N_i$
1
$N_h$
$N_h$
$N_h$
$1/N_o$
$N_o$
• Feedforward – simpler to simulate
• Neuron model – LIF or Izhikevich
• I-H – Fully connected
• H-H – Optional sparsity (~ 0.5)
• H-O – Sparsely connected ($1/N_o$)
• Lateral inhibition at O – task dependent, optional
• Partial recurrence – task-dependent

Figure 5.1: Proposed network architecture

## 5.2 Balanced-Pair Binary State Encoding

Since the applications of the proposed reinforcement learning rules are focused around game playing and control tasks, and not continuous-value precision problems, the input states are encoded using a binary spike-rate based encoding. Each variable

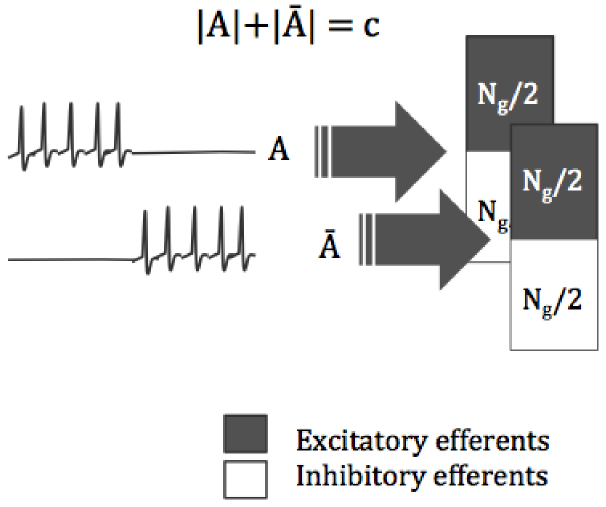


Figure 5.2: Balanced-pair binary rate encoding

or bit in the binary representation of the state or input to the network is assigned two neuron groups (Figure. 5.2) consisting of $\boldsymbol{N_g}$ neurons each. 50 percent of the neurons in a group are inhibitory, while the rest are excitatory. This is inspired from the arrangement of sensory neurons in the retina of the human eye [29]. All the synapses elsewhere are excitatory.

If a bit in the input representation of a state is *turned on* (value = 1), one of the neuron groups is fed a poisson current at F Hz (represented by $\boldsymbol{A}$ in Figure. 5.2), and the other is set to 0 Hz (no input spikes; represented by $\tilde{\boldsymbol{A}}$). If the variable is *turned off* (value = 0), the groups are assigned frequencies the opposite way.

An example of the encoding scheme for an XOR task is illustrated in Table. 5.1. $\boldsymbol{A}$ and $\boldsymbol{B}$ refer to the two input variables in the XOR function, and with $\boldsymbol{F} = \mathbf{50}$, for each input case it is seen that the total number of spikes per second remains constant ($= \mathbf{100}$). This arrangement is utilized to normalize the rate of the spikes fed into the network across all the observations or states experienced by the agent. The number of spikes that are fed into the network via the input layer for each

Table 5.1: Encoding inputs for XOR with the balanced binary state encoding

| **Case** | $\boldsymbol{A}$ | $\boldsymbol{\tilde{A}}$ | $\boldsymbol{B}$ | $\boldsymbol{\tilde{B}}$ | **Total** |
|---|---|---|---|---|---|
| (0,0) | 0 | 50 | 0 | 50 | 100 |
| (0,1) | 0 | 50 | 50 | 0 | 100 |
| (1,0) | 50 | 0 | 0 | 50 | 100 |
| (1,1) | 50 | 0 | 50 | 0 | 100 |

state remains constant this way, aiding in confident decoding based on firing rate at the output. This is because, across all the states, if the spike rates are constant at the input layer, then the spike rates at the output neurons do not tend to vary very much. Otherwise, if for different states there are different rates of spikes at the input, deciding on a threshold to decode a response at the output becomes difficult.

## 5.3 Attentional Feedback

From the upgraded XOR task experiment that was discussed in the previous chapter, it became evident that the DA-STDP method lacks a way to filter out the synapses at the early layers of the network that contributed significantly to the observation-action mapping, for potentiation. This was termed the spatial credit assignment problem. To handle this problem, we introduce *feedback synapses* (Figure. 5.3) in the network from each post-synaptic neuron to its pre-synaptic neurons. Evidence for such feedback is biologically validated between motor and sensory layers of the cortex, especially in the visual areas of the cortex [30]. These synapses aid in selective potentiation [31], as explained here in this section.

As illustrated in Figure. 5.3, each post-synaptic neuron has a feedback synapse to each of its pre-synaptic neurons. Feedback synapses are not plastic; their strengths are set to a constant value of 1. These synapses do not carry the spikes that are carried by the feedforward synapses. Instead, they form a separate channel in the

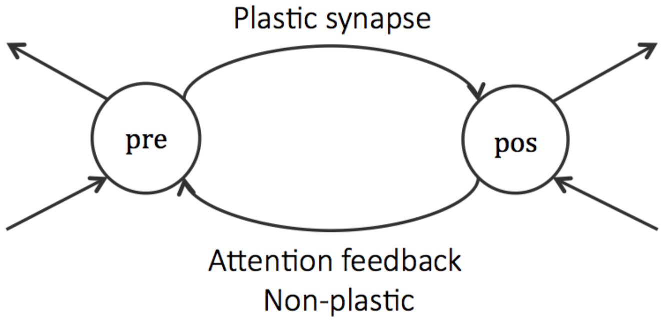


Figure 5.3: A feedback synapse

network, that carry only the feedback signal which are binary-valued in nature. The primary objective of using this feedback is to let the network identify synapses that have structurally causal relationship to the output neuron whose activity was responsible for any decision decoded from the network. Figure. 5.4 illustrates this concept. Only the blue synapses, which structurally lead to the *winning* neuron at the output through their connectivity, could have influenced the activity of that neuron. Since the activity of this neuron was primarily responsible for the decoded decision, only the blue synapses should be reinforced. Thus, the feedback signal from the winning neuron to the earlier layers, that carry this causality information, could be used to modulate the already-modulated STDP potentiations (Equation 5.1)

$$\boldsymbol{\frac{dw}{dt} = \gamma R(t) Z(t) \Theta(t)} \tag{5.1}$$

$$\boldsymbol{\Theta(t)} = \begin{cases} \boldsymbol{O_k(t),} & \text{for output-hidden synapses} \\ \boldsymbol{\sum_{k=1}^{N_o} w_{kj} O_k(t),} & \text{for hidden-input synapses} \end{cases}$$

The feedback signal is from the response-selection stage $\boldsymbol{O}$ to all of the previous layers in the network. Potentiations for synapses follow equation 5.1. $\boldsymbol{\Theta(t)}$ represents

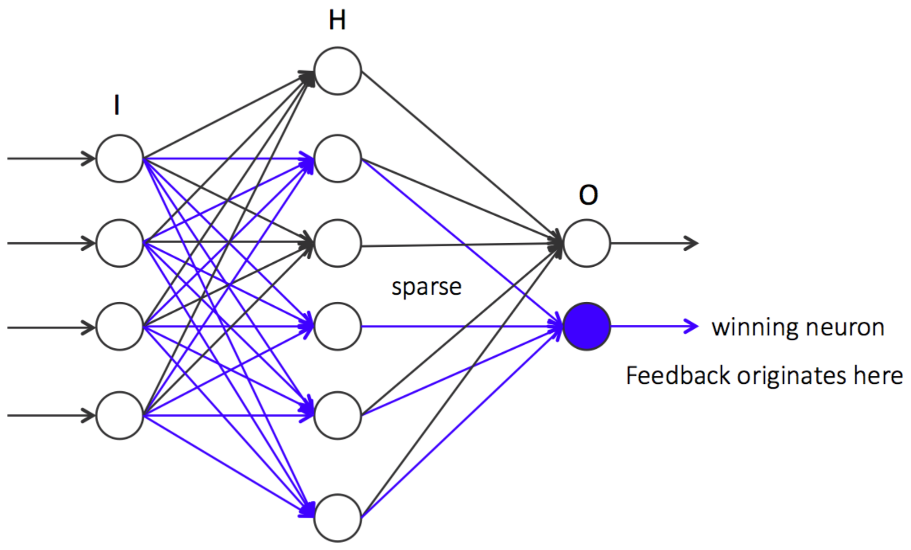


Figure 5.4: Attentional feedback

the feedback signal. $\boldsymbol{i, j, k}$ represent input, hidden and output neurons respectively. $\boldsymbol{O_k}$ is a binary value that represents the response of the network. For instantaneous rewards, $\boldsymbol{O_k(t) = 1}$ means that output neuron $\boldsymbol{k}$ spiked at that instant. This ensures that the potentiations due to rewards for this output neuron's decisions are only effective on the synapses that form a path from input layer to this output neuron (Figure 5.4). The other synapses that contributed to the spikes of the other output neurons are not potentiated at this point. For the synapses between hidden layer and input layer, the sum of ouput responses is used instead. $\boldsymbol{w_{kj}}$ is the strength of the feedback synapse, and it is set to 1 for all connected neurons, otherwise 0.

The functioning of this feedback signal is inspired from *error backpropagation* mechanisms in non-spiking neural networks as well [31]. This enhancement solves the spatial credit assignment problem to a good extent, and makes a lot of sense in spiking neural networks. The mechanism is validated in the following section.

### *5.3.1 XOR Task with Attentional Feedback*

The XOR task with two output neurons is revisited. The network is composed of three layers of LIF neurons. Attentional feedback synapses are implemented between each connected pair of neurons in the network, and the two input variables are encoded with the BPBS scheme (with $\boldsymbol{F = 40}$) discussed in the previous section (Table. 5.1). Rewards during the episodes were instantaneously delivered to the synapses. This means that for cases $\mathbf{(0,0)}$ and $\mathbf{(1,1)}$, for each spike at neuron $\boldsymbol{O_1}$, a reward of $\mathbf{+1}$ was delivered to the network, and for each spike at $\boldsymbol{O_2}$, a reward of $\mathbf{-1}$ was delivered. The sign of the rewards was reversed for cases $\mathbf{(0,1)}$ and $\mathbf{(1,0)}$. The episode lasted for 200 epochs, which represents 50 sets (each epoch includes 4 cases of inputs) of input presentations.

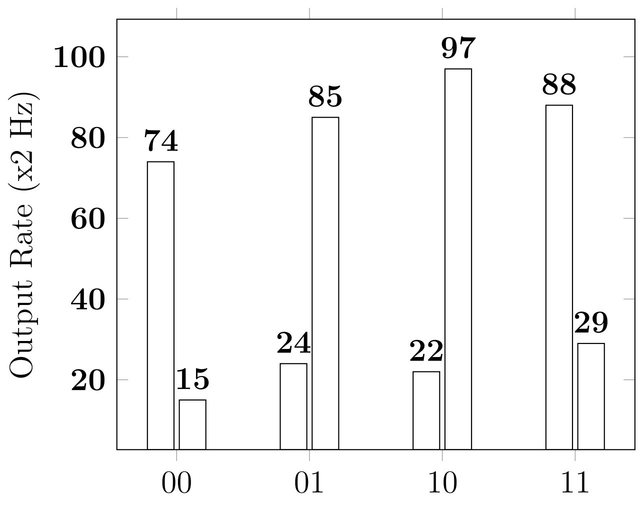


Figure 5.5: Output spike rates (40 Hz poisson input)

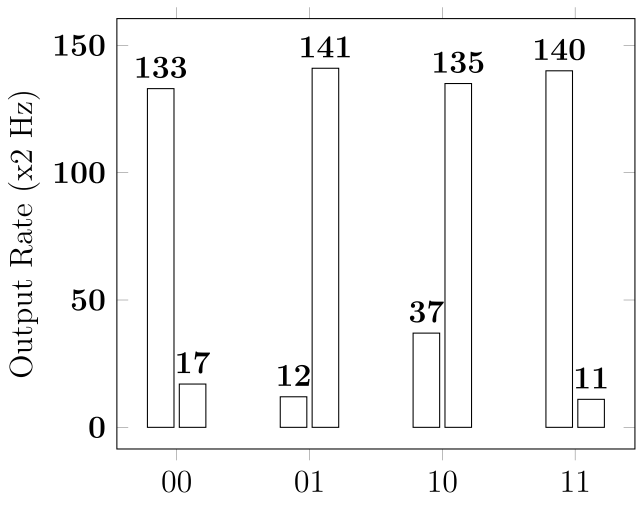


Figure 5.6: Output spike rates (100 Hz poisson input)

Spike rates at the output neurons for the four cases are shown in Figure. 5.5 which corresponds to F = 40 Hz poisson input, and in Figure. 5.6 for F = 100 Hz input. Also, the reward profiles for the task over 200 epochs with 40 Hz and 100 Hz input are compared in Figure. 5.7. The higher reward profile for the 100 Hz input is expected because it provides more opportunity for exploration since the spike rate is higher. The results validate the notion of attentional feedback in resolving the spatial credit assignment problem.

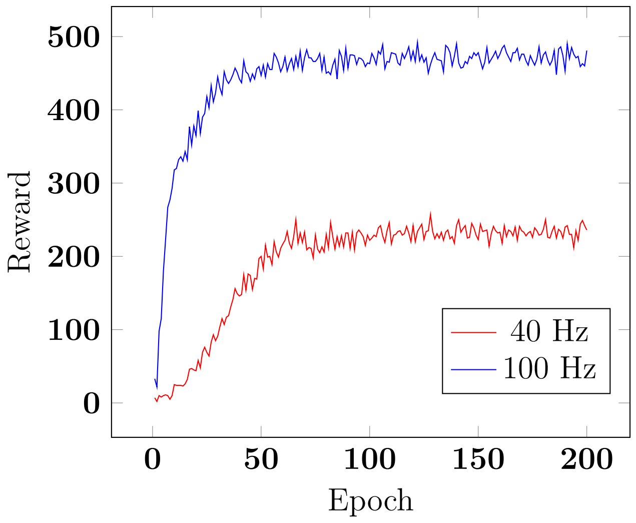


Figure 5.7: Reward profile for the XOR task with attentional feedback

## 5.4 Temporally Attenuated Rewards

Attenuating rewards over time is a way to deal with the response numbness problem. We do not always learn at the same rate for the same rewards from the environment, and the excitement or the level of pleasure fades away with time due to several reasons. From another perspective, there is no need to learn something which has already been learned enough. So the internal perception of pleasure could be diminished slowly over time, so that after a behavior is learnt sufficiently well, the rewards for the same behavior do not affect as much as it used to. This is the idea behind attenuated reward profiles introduced here. This approach reduces the amount of reward actually delivered to the synapses as the strength of an observation-action mapping becomes stronger. This works for tasks involving a finite set of states. For generic tasks, a biological mechanism and its simulation counterpart that remember this profile are yet to be discovered.

For each state $\boldsymbol{S}$ observed by the system, an expected reward factor $\boldsymbol{E_S(t)}$ is associated with the network. These are global factors, not local or synapse-specific. This state-specific factor is to ensure attention to each state or task that the system is involved with, so that performance and rewards for each input state could be tracked separately. $\boldsymbol{E_S(t)}$ is initially set to 0, and updates to this factor follow equation 5.2. $\boldsymbol{R(t)}$ represents the actual reward from the environment and $\boldsymbol{\tau_S}$ determines the speed of decay or saturation (Figure. 5.8).

$$\boldsymbol{E_S(t) = E_S(t) + \frac{R(t) - E_S(t)}{\tau_S}} \tag{5.2}$$

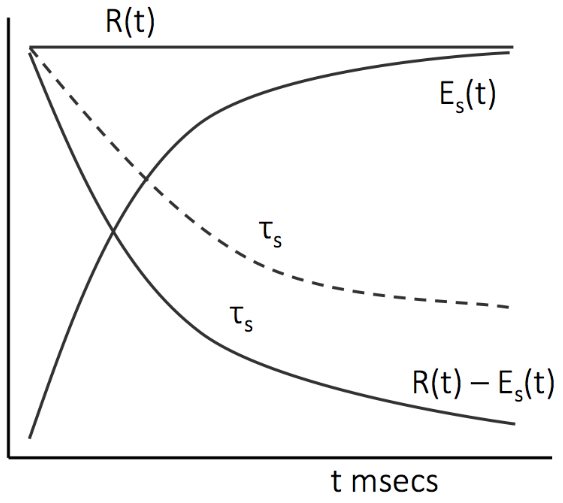


Figure 5.8: Temporally-attenuated reward profile

The reinforcement signal delivered to the system is the difference between the expected reward $\boldsymbol{E_S(t)}$ and the received reward $\boldsymbol{R(t)}$. With an attenuated reward system, the learning rule in equation 5.1 modifies to equation 5.3.

$$\boldsymbol{\frac{dw_{jk}}{dt}} = \gamma(\boldsymbol{R(t) - E_S(t))Z_{jk}(t)\Theta(t)} \tag{5.3}$$

For specific choices of $\boldsymbol{\tau_S}$ this mechanism prevents complete supression of responses that prevent exploration, because the system is never forced to learn or unlearn any mapping completely. Also, for rewards that span multiple levels for various degrees of reinforcements, this keeps the output rates stable. These are demonstrated in the later chapters.

### *5.4.1 XOR Task with Attenuated Rewards*

A feedforward network with two output neurons, 80 hidden neurons and 80 input neurons was used in this experiment to model the learning agent (Figure. 3.7). The balanced-pair binary encoding explained previously was used to encode the input states (with $\boldsymbol{F = 100}$). The epochs were 2000ms long, and the rewards were attenuated with a time constant of 10ms. The reward profile and average output spike rates are shown in Figures 5.10 and 5.9.

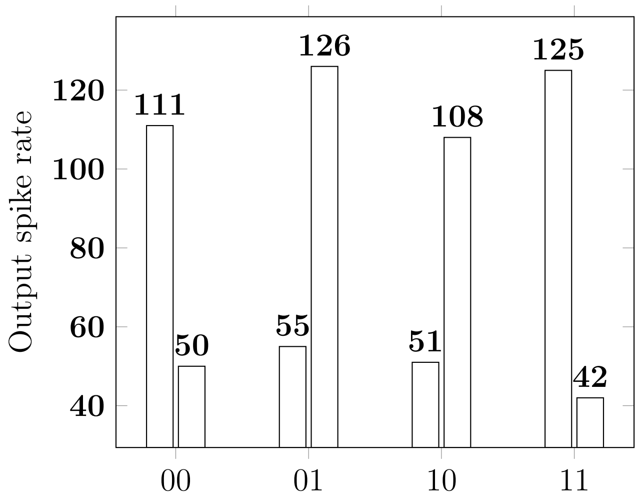


Figure 5.9: Output spike rates with attenuated rewards

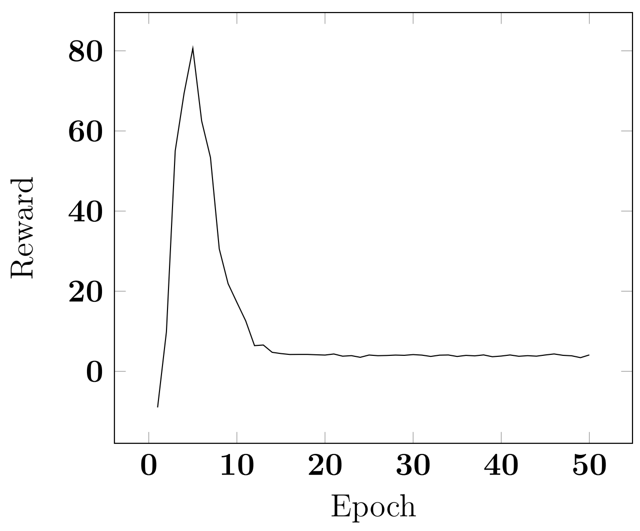


Figure 5.10: Reward profile for the XOR task with attenuated rewards

It could be seen that the network has handled the spatial credit assignment problem very well, and that the response rates are now more stable than before, because the attenuation of the rewards prevent changing the strengths of the synapses once they have learnt a particular mapping well.

## 5.5 Fitness Proportionate Response Selection

This concerns the response selection part of the network model. The rate of firing of the output neurons of the network represents the strength of the response. For tasks with multiple, equally probable response expectations from which one has to be picked, fitness proportionate selection or the *Roulette Wheel* selection [32] was used to select a response in tasks where only one action was required to be selected. The *softmax* function (Equation 5.4) was used to normalize the output rates before the selection. If $\boldsymbol{O_k(t)}$ represents the number of spikes at the output neuron $\boldsymbol{k}$ during an epoch, then the softmax normalization happens according to equation 5.4. The normalized rate of all the output neurons $\boldsymbol{\dot{O_k}(t)}$ is then used with the *Roulette Wheel* selection to pick one winning neuron, based on the the fitness value represented by $\boldsymbol{\dot{O_k}(t)}$. This approach improves exploration in the network because there is a finite probability that every action in the set of actions applicable to the input case will be selected.

$$\boldsymbol{\dot{O_k}(t) = \frac{e^{O_k(t)}}{\sum e^{O_k(t)}}} \tag{5.4}$$

For simpler cases, where only one action is expected to be reinforced very strongly, the output neuron with the maximum number of spikes during an epoch could be selected as shown in equation 5.5. Here $\boldsymbol{O_k(t)}$ represents the array of $\boldsymbol{O_k(t)}$ for $\boldsymbol{k = 1..N_o}$.

$$\textbf{winning neuron} = \textbf{argmax}(\boldsymbol{O(t)}) \tag{5.5}$$

Fitness proportionate selection works here too, but is computationally a bit more expensive. Also, if one action is always reinforced stronger, then the probability of the other actions being selected becomes zero here, thus leading to less exploration. However, for simpler tasks involving 2-3 actions, this method is simpler and viable.

## 5.6 Summary

The learning rules with all the proposed enhancements are summarized here. If $\boldsymbol{w_{ij}}$ is the strength of the synapse between a post-synaptic neuron $\boldsymbol{i}$ and the pre-synaptic neuron $\boldsymbol{j}$, then local synaptic plasticity happens according to the rules in equations 5.6-5.10.

$$\frac{dw_{ij}}{dt} = \gamma(R(t) - E_S(t))Z_{ij}(t)\Theta(t) \tag{5.6}$$

$$\tau_z \frac{dZ_{ij}(t)}{dt} = -Z_{ij}(t) + P_{ij}^{+}(t)f_i(t) + P_{ij}^{-}(t)f_j(t) \tag{5.7}$$

$$P_{ij}^{+}(t) = P_{ij}^{+}(t-\delta t)e^{-\delta t/\tau_+} + A_+ f_j(t) \tag{5.8}$$

$$P_{ij}^{-}(t) = P_{ij}^{-}(t-\delta t)e^{-\delta t/\tau_-} + A_- f_i(t) \tag{5.9}$$

$$E_S(t) = E_S(t) + \frac{R(t) - E_S(t)}{\tau_S} \tag{5.10}$$

$$\Theta(t) = \begin{cases} O_k(t), & \text{:for output-hidden synapses} \\ \sum_{k=1}^{N_o} w_{kj}O_k(t), & \text{:for hidden-input synapses} \end{cases}$$

$R(t)$ is the reward signal, and $E_s(t)$ is the actual attenuated reward corresponding to state $S$, that is delivered to the network. $\tau_s$ is the decay time constant for the rewards. This is usually set to 500 ms for instantaneous rewarding schemes. $Z_{ij}(t)$ is the eligibility trace computed by tracking the occurrences of post and pre-synaptic spikes. $f(t) = 1$ represents a spiking event, it is $0$ for no spike during time $t$. $\tau_+ = \tau_- =$ 20 ms is the decay constant for these tracking signals. $\tau_z$ is the time constant for the eligibility trace, which is set to 20 ms.

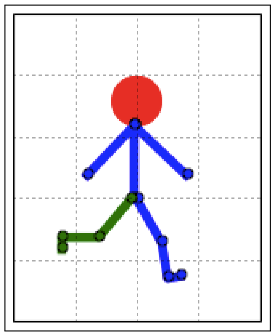
$\Theta(t)$ is the feedback signal that is propagated from the response selection stage (output layer) to the input layer, through the hidden layers. This signal is computed as given in the equations, where $O_k$ represents the number of spikes at the output neuron $k$ during an epoch and $w_{kj}$ denotes the synaptic strength of the non-plastic feedback synapse between output neuron $k$ and the hidden neuron $j$. The time step $\delta(t)$ of simulation is usually set to 1 ms.

# 6. EXPERIMENTAL TASKS AND RESULTS

The effectiveness of the proposed methodology was studied using several experiments, involving both a single learning agent and multiple agents in a cooperative environment. The experiments involved both instantaneous and delayed rewarding schemes for reinforcements. The tasks, except for the XOR task, were picked to be game-playing or control problems involving interactions with the environment and/or other agents. These problems, as mentioned before, are better suited for reinforcement learning, since they involve a lot of interaction, and there are no easily programmable solutions to these tasks.

## 6.1 Bot Walking Task

The objective of this task is to apply the enhanced DA-STDP technique, as a control algorithm, to make a two-legged bot (Figure. 6.1a) always move forward. This is a single-agent task. Earlier implementations have involved Q-learning in a non-spiking environment [33] to accomplish this. The task involves creating a suitable network and a reward strategy for the agent to learn to walk forward. To make the task simple, balance and other dynamics are ignored. Only the two legs of the agent are the motor areas that will be controlled by the network. There are four possible motor actions for each leg - [back-up, back-down, front-up and front-down]. Since each leg can be controlled independently, there are a total of 16 possible states that the agent can be in. Actions 0,1,2 and 3 toggle the state of one control each, as given in Table. 6.1b. For example, if the right leg of the bot is up, action 0 toggles this state and brings the leg down.


(a) A two-legged agent

| Action | Control |
|---|---|
| 0 | Right leg up/down |
| 1 | Right leg back/forward |
| 2 | Left leg up/down |
| 3 | Left leg back/forward |

(b) Action space

Figure 6.1: The walking task: Agent model and action space

### *6.1.1 Task and Network Design*

A network of LIF neurons, composed of three layers, was used to model the bot. There were 160 neurons in the hidden layer, 160 neurons in the input layer (16 input groups) and 4 output neurons to select one of the four actions per state. Each configuration of the bot was assigned a number, which represents the value of the state corresponding to that configuration. The state transition matrix, which determines the next state of the agent, given the current state and the performed action is provided in Table. 6.2a. A reward matrix (Table. 6.2b) was created from the actual movie sequence that corresponds to the bot walking forward. Rewards are set for a next state that immediately follows the current state in the movie sequence. Backward state transitions are manually given a negative reward to negatively reinforce those transitions.

| $S_t$ | $A_0$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|---|
| 0 | 1 | 3 | 4 | 12 |
| 1 | 0 | 2 | 5 | 13 |
| 2 | 3 | 1 | 6 | 14 |
| 3 | 2 | 0 | 7 | 15 |
| 4 | 5 | 7 | 0 | 8 |
| 5 | 4 | 6 | 1 | 9 |
| 6 | 7 | 5 | 2 | 10 |
| 7 | 6 | 4 | 3 | 11 |
| 8 | 9 | 11 | 12 | 4 |
| 9 | 8 | 10 | 13 | 5 |
| 10 | 11 | 9 | 14 | 6 |
| 11 | 10 | 8 | 15 | 7 |
| 12 | 13 | 15 | 8 | 0 |
| 13 | 12 | 14 | 9 | 1 |
| 14 | 15 | 13 | 10 | 2 |
| 15 | 14 | 12 | 11 | 3 |

(a) State transition matrix for the walking task

| $S_t$ | $A_0$ | $A_1$ | $A_2$ | $A_3$ |
|---|---|---|---|---|
| 0 | 0 | -1 | 0 | -1 |
| 1 | 0 | 0 | -1 | -1 |
| 2 | 0 | 0 | -1 | -1 |
| 3 | 0 | -1 | 0 | -1 |
| 4 | -1 | -1 | 0 | 0 |
| 5 | 0 | -1 | 0 | -1 |
| 6 | 0 | -1 | 0 | -1 |
| 7 | -1 | 2 | 0 | 0 |
| 8 | -1 | -1 | 0 | 0 |
| 9 | 0 | -1 | 0 | -1 |
| 10 | 0 | -1 | 0 | -1 |
| 11 | -1 | 2 | 0 | 0 |
| 12 | 0 | -1 | 0 | -1 |
| 13 | 0 | 0 | -1 | 2 |
| 14 | 0 | 0 | -1 | 2 |
| 15 | 0 | -1 | 0 | -1 |

(b) Reward matrix for the walking task

Figure 6.2: State transition and reward lookup for the walking task

The initial state was picked at random, and the state values were encoded in binary. These binary values were converted to spikes via the balanced pair encoding scheme that was discussed, with the Poisson spiking rate set to 100 Hz. For every action chosen by the agent, the reward and the next state was chosen by the environment according to the state transition matrix (agent-internal) and the reward lookup (environment) matrices. These are explicit models in this task. Rewarding was instantaneous; for every spike at the output corresponding to a rewarded action, the network was rewarded, otherwise punished according to the reward matrix. Each state was assigned an attenuation time constant of 500ms, and the rewards for these states were attenuated accordingly. The learning episode lasted for 20 epochs, with each epoch lasting for 500ms.

### *6.1.2 Results*

Rewards earned by the agent for making actions when its current state was 2 is shown, for instance, in Figure. 6.3. The final sequence of states resulting from the learning is shown in Figure. 6.4. This sequence, when animated, corresponds to the forward walking of the agent, which validates the learning effectiveness of the proposed method a moderately simple task. The agent was able to always walk forward while avoiding backward steps in less than 20 epochs.

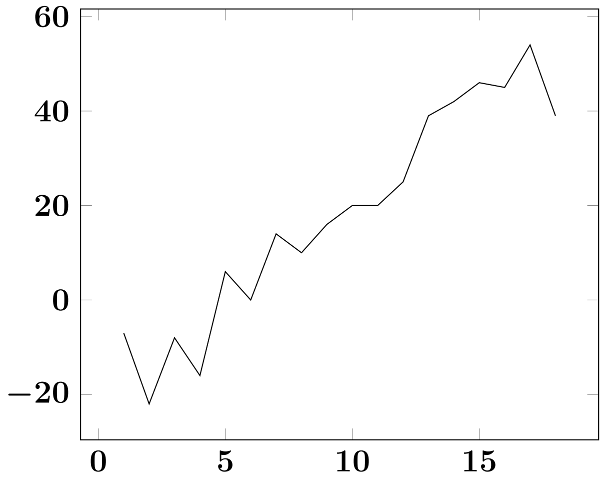


Figure 6.3: Reward profile for state 2

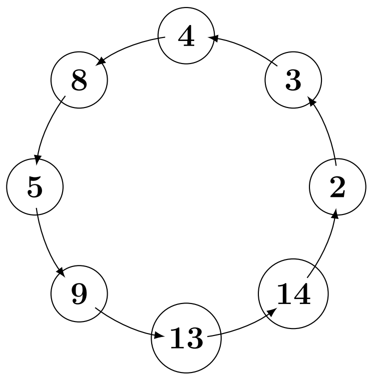


Figure 6.4: Converged state sequence

## 6.2 The Iterated Prisoners' Dilemma

The iterated prisoners' dilemma (IPD) is a symmetric, general-sum game between two agents. The reward structure (Table. 6.1) is such that both the agents will have to play greedy and for the team's benefit at the same time. This contradictory behavior requirement, demonstrating cooperative intelligence of a multi-agent system makes this interaction interesting to study [34].

Table 6.1: Reward payoff structure

| Action | Cooperate(C) | Defect(D) |
|---|---|---|
| Cooperate(C) | (4,4) | (-3,5) |
| Defect(D) | (5,-3) | (-2,-2) |

The premise is that two prisoners commit a crime together, and are questioned separately. Each of these prisoners can choose to either cooperate (C) or defect (D) at each timestep. The reward structure (Table. 6.1) clearly shows that from an individual agent's perspective, irrespective of the other agent's decision, defecting maximizes its reward. However, during the interplay of exploration and exploitation the agents learn that mutual cooperation maximizes the reward of the system as a whole. No prisoner can improve his chances of getting rewarded by changing his strategy, and mutual cooperation is the only Nash equilibrium of this game [35]. The iterated version of this interaction involves making the agents play multiple times. During these iterations the agents learn the what actions make the game's outcome rewardable, and how their independent actions affect the system.

### *6.2.1 Task and Network Design*

A partially recurrent network is implemented to model the system with two players or agents (Figure 6.5). Each agent was modeled using a 3-layered network of LIF neurons. The common input layer consisted of 4 groups of neurons with 20 neurons in each. The inputs to this layer encode the decisions of the two agents in the previous iteration. The hidden layer in each agent consists of 80 neurons, and is fully-connected to the input layer. That is, each neuron in the input layer is connected to all neurons in the hidden layer. The output layer in each agent consists of two neurons, one for each decision (C,D) which are fed back to the input layer. The hidden and the output layers are sparsely connected. Between any neuron in the output layer and any neuron in the hidden layer, the probability of connection is 0.5. The reward policy shown in Table. 6.1 models the interaction between the agents and the environment, and the output feedback represents the agent-agent interaction. This task involved delayed rewards, as the rewards and punishments are delivered to the agents after the end of each epoch, whose length was 500 ms.

The initial input responses fed to the network were random. After the end of an epoch, if say, the response of agent 1 was C and that of agent 2 was D, then a

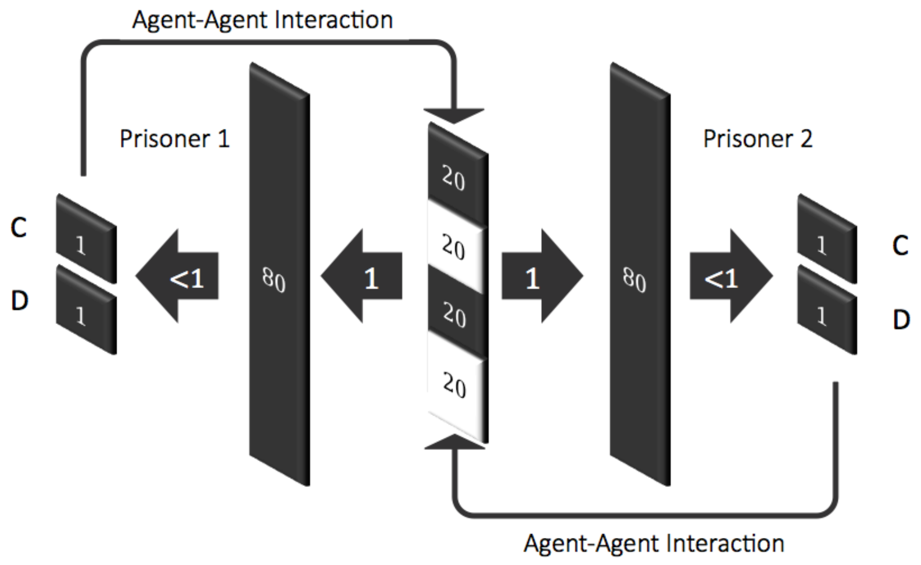


Figure 6.5: Network design for the IPD task

reward of -3 was delivered to agent 1, and a reward of +5 was delivered to agent 2 throughout the next epoch. The cooperative system of twp agents was trained for 100 epochs.

### *6.2.2 Results*

The system converged in about 40 epochs. Both the agents chose mutual cooperation (C,C) consistently which is the nash equilibrium state required for the system to maximize its reward. The average reward and strength profiles for 100 iterations are shown in Figures 6.6 and 6.7. *Response strength* is the difference between the number of spikes at the deciding output neuron and the other output neuron, at the end of an epoch.

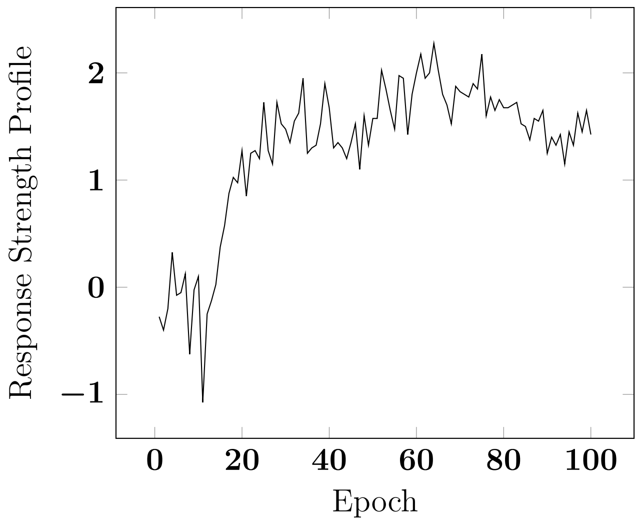


Figure 6.6: Response strength profile for the IPD task

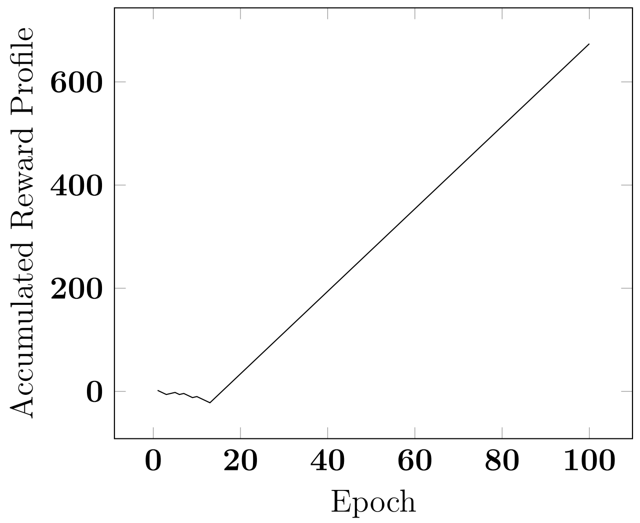


Figure 6.7: Accumulated reward profile for the IPD task

The strength-weighted reward profile (Eqn. 6.1) of both the agents are plotted in Figure. 6.8. This is computed by weighing the rewards and the punishments of the output neurons by the strength of their responses (number of spikes at the end of the epoch).

$$\boldsymbol{R_s(t) = R(t)N_t + R^c(t)N_t^c} \tag{6.1}$$

$\boldsymbol{R(t)}$ is the reward signal, $\boldsymbol{N_t}$ is number of spikes at the expected output neuron, $\boldsymbol{R^c(t)}$ is punishment for the wrong output neuron and $\boldsymbol{N_t^c}$ is number of spikes at the wrong output neuron. The profiles show that the system of two agents cooperated while maximizing the reward delivered to the combined system as a whole.

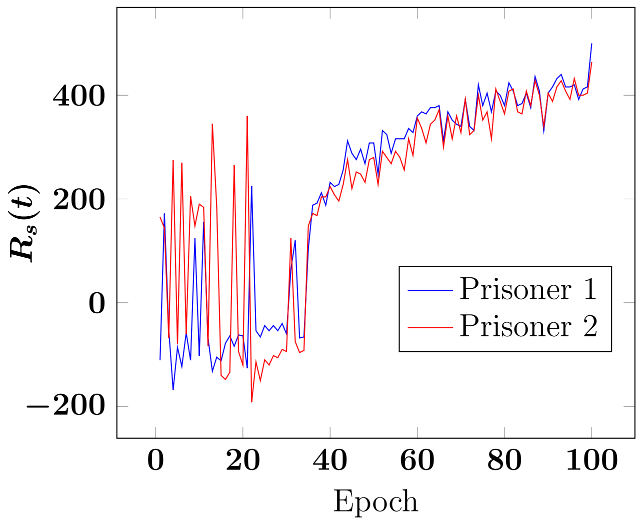


Figure 6.8: Strength-weighted reward for the IPD task

## 6.3 Multi-agent Distributed Sensor Network

This task is one of the NIPS 2005 multi-agent reinforcement learning benchmarks [36]. It is a sequential decision making variant of the distributed constraint optimization problem described in [37]. A sensor network composed of two arrays of sensors parallel to each other enclosing cells within them is shown in Figure. 6.9. Every cell is enclosed by exactly 4 sensors. There are N targets within the network, only one of which can occupy a cell at any instant. The targets move randomly to a neighboring cell or remain in their current cell with equal probability.

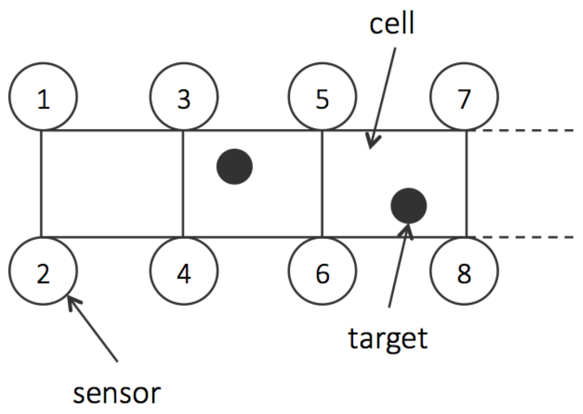


Figure 6.9: A sensor network configuration with 8 sensors

Each sensor can perform three actions - track a target in its left cell (action 1), track a target in its right cell (action 2) or do not track (action 3). Every tracking action is associated with a cost. There is no cost for performing action 3 (no tracking). Every target $\boldsymbol{t}$ has an initial energy $\boldsymbol{E_i^t = 3}$. When a target is tracked by 3 or more sensors at a time, then it is said to have been hit, which leads to a decrease of the said target's energy level by 1. A target is captured by the system when its energy level becomes 0, at which point the target vanishes from the system. The goal of this distributed system is to capture all the targets as soon as possible. The ideal (quickest) number of steps to capturing all the targets is 3.

### *6.3.1 Task Complexity and Network Design*

The details of this task, its complexity and state space are listed in Table. 6.2. The task involved a configuration of the sensor network with 8 sensors and N = 2 targets. Each of the eight sensors was modeled as an independent agent with 12 input

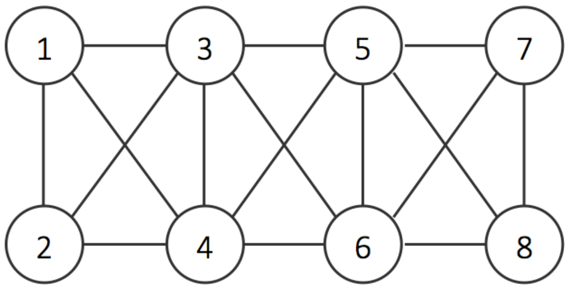


Figure 6.10: Interaction relationship graph for the sensors

neuron groups of 30 neurons each (LIF), 3 output neurons and 30 hidden neurons. Thus, in total, the system had 360 output neurons, 720 hidden neurons and 24 output neurons. The state of the system, which represented the position and energy of the targets, was encoded as a vector of energy values $\boldsymbol{E_t}$. For example, the state [3,2,0] represented a target with energy 3 at cell 1, and another target with energy 2 at cell 2. The encoding of the state was done over the 12 inputs. Each cell was assigned 4 input neurons, and the energy level $\boldsymbol{i}$ was assigned the $\boldsymbol{i^{th}}$ position in the 4-input array. For example, state [3,2,0] was encoded as [0,0,0,100][0,0,100,0][100,0,0,0], where 100 represented the firing rate of that neuron. At the output, fitness-proportionate selection was used to decode the action response based on the number of spikes at each output neuron, at the end of each epoch.

Rewards to the sensors could be instantaneous or delayed. In the case of instantaneous rewarding, for each tracking action, the network was negatively rewarded (reward = -1). And, for each cooperative action that led to a hit of a target (resulting from 3 or more sensors tracking it), each of the participating sensors was given a reward of +5. In the case of delayed rewarding, the hit reward and tracking costs were delivered to the networks in the following epoch (with a delay of 500ms).

The sensors could either explicitly interact with the other neighboring sensors as

Table 6.2: DSN task complexity for 8 sensors and 2 targets

| | |
|---|---|
| States space cardinality | 37 |
| Combined action space cardinality | 6561 |
| Initial states | [3,3,0],[3,0,3] or [0,3,3] |
| Terminal State | [0,0,0] |
| Tracking cost | -1 |
| Hit reward (for each responsible sensor) | +5 |

shown in the graph in Figure. 6.10, or implicitly only with the environment. The explicit interaction graph in Figure. 6.10 shows that each sensor could interact with 5 of its neighbors, which influence the hit of the targets in the cells enclosed by them. However, this configuration makes the system much more complex than the one with implicit interaction. Thus, this task was modeled to have interactions only between an individual agent and the environment.

### *6.3.2 Results*

The system converged in about 15 epochs. The ideal number of iterations for the network to reach the terminal state is 3. The multi-agent network that learned to lock the targets using the proposed reinforcement scheme was able to converge to 6 and 4 steps to the terminal state for the instantaneous and delayed reward policies respectively. The profile of the number of steps required to reach the terminal state, over the epochs, is shown in Figure. 6.11. This result is comparable to the results from the non-spiking implementation of the distributed sensor network discussed in [38], which converged to nearly 7 steps for reaching episode termination where all the targets disappear from the network cells.

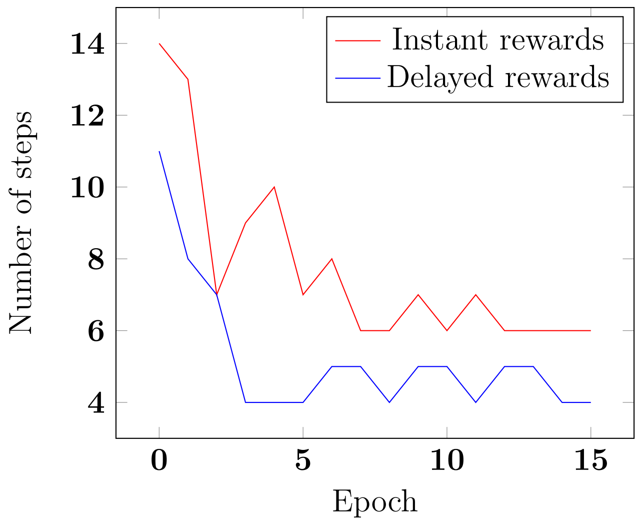


Figure 6.11: Number of steps to episode termination

# 7. CONCLUSIONS

This thesis aimed to provide an overview of reinforcement learning in spiking neural networks, enumerate and explain theoretical challenges in existing techniques, provide solutions to handle those challenges, and evaluate them in psuedo-realistic interactive environments.

## 7.1 Summary of Contributions

A reinforcement learning mechanism that includes various improvements over the existing Dopamine-modulated STDP (DA-STDP) techniques [25][26], to handle the spatial credit assignment problem and response numbness problem is presented. These improvements are inspired from biological elements as well as existing techniques in non-spiking neural networks. *Attentional feedback* from the output layer to the earlier layers in the network is seen to improve the spatial credit assignment during learning, by gating the synaptic updates of those synapses that are not involved in the decisive actions at the output. This makes learning attentive or goal-oriented. *Attenuated rewards* that decay with time help improve the chances of exploration of the learning agent, and also solve the response atrophy problem. The *balanced-pair binary encoding* scheme, and the techniques involved in response selection are also part of the proposals in this work.

The proposed improvements have been evaluated for single-agent learning tasks like the Exclusive OR function reproduction and the walking task, and multi-agent cooperative learning tasks like the *iterated prisoner's dilemma* and the *distributed Sensor-Network* task. These involved different configurations like instantaneous reward and delayed reward polcies, cooperative and independent action based learning etc. These experiments and their results validate the improvements proposed with the techniques discussed in this work.

## 7.2 Future Directions and Perspectives

A basic reinforcement learning framework with features that make it applicable for pseudo-realistic tasks has been developed. Additional studies are needed to optimize the implementations for tasks, and also to maximize efficiency and performance.

### *7.2.1 Network Size Optimization*

In this study, the network size for the XOR task was referred from the experiments in [26]. This has not proven to be optimal, or minimal. However, considering the XOR network to be a fundamental computational unit, the network was scaled for other tasks depending on the complexity. For example, the number of neurons in the input layer of the XOR network was 20 per input variable. So if a task requires $\boldsymbol{N}$ input variables, then the network was scaled to have $\boldsymbol{20N}$ neurons at the input layer. The number of neurons in the hidden layer is kept the same as in the input layer. If $\boldsymbol{M}$ different outputs can be expected from the network, the network had $\boldsymbol{M}$ output neurons, one for each response.

Studies have to be conducted to optimize the size of the layers, depending on the task. Though, the more complex a network is, the more complex patterns it could store, the network has to still be optimized to avoid computational complexity and processing delay. Several directions could be taken to handle this.

1. **Dynamic scaling** - The network could be initialized with a minimal size, and be scaled by adding layers or neurons to the layers during training. Algorithms like Cascade Correlation allow this type of dynamic network scaling [39]. A suitable algorithm to scale the network based on the amount of punishments that the learning agent receives over time, could be created to achieve this.

2. **Experimental evaluation** - This method involves making several experiments on a basic task like the XOR function reproduction, and identifying the number of layers and neurons that optimize the performance of the network, in terms of the maximum reward acheived and the strength of the response. This could be used as a unit network, and be scaled for more complex tasks, depending on the number of inputs and outputs.

3. **Analytical approach** - The learning rules could be analyzed mathematically, and the network size that would theroretically maximize performance could be computed. A similar approach has been made in [40] to determine an optimal size for the backpropagation learning.

### *7.2.2 Reward Attenuation*

The reward attenuation mechanism in this study involved attenuating them based on the difference between the actual and the expected reward. A suitable time constant has to be hand-picked before the experiment, by estimating the time that the agent would take to sufficiently learn the task. This mechanism is to ensure that the weights of the synapses do not get saturated, during learning, due to continued supression from negative rewards. By decreasing the rewards during an epoch or episode, the weights could be prevented from saturation. However, this approach does not guarantee that the saturation would not happen.

The learning rules could be altered to keep the balance between exploration and exploitation a constant, thereby making the network always explore to some extent. This would guarantee that there is always room for exploration, which means that any single response could not be supressed forever. The fitness-proportionate response selection mechanism is one example of realizing this, because it selects response based on firing rate probabilistically. Since the probability of zero spikes happening at any neuron consistently is very less, this approach definitely improves exploration. Otherwise, random noise could be fed to the network, that is different from the input spikes, which would keep this probability of zero spikes at the output neurons even less.

# BIBLIOGRAPHY

[1] Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Cambridge, MA: MIT, 2012. Print.

[2] Smith, Michael R., and Tony Martinez. *Improving Classification Accuracy by Identifying and Removing Instances That Should Be Misclassified.* The 2011 International Joint Conference on Neural Networks (2011): n. pag. Web.

[3] Sutton, Richard S., and Andrew G. Barto. *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT, 1998. Print.

[4] Shteingart, Hanan, and Yonatan Loewenstein. *Reinforcement Learning and Human Behavior.* Jerusalem: Center for the Study of Rationality, 2014. Print.

[5] Arias-Carrion O, and Poppel E. *Dopamine, Learning and Reward-Seeking Behavior.* Act Neurobiol 2007, Exp 67 (4): 481488.

[6] Shettleworth, Sara J. *Cognition, Evolution, and Behavior.* New York, NY: Oxford UP, 2010. Print.

[7] Pavlov, I. P.. *Conditional Reflexes.* New York: Dover Publications, 1927/1960

[8] John B. Watson. *Big Ideas Simply Explained: The Psychology Book.* London: Dorling Kindersley Publishing, Inc., 2012.

[9] Goldstein, Avram. *Heroin Maintenance: A Medical View. A Conversation between a Physician and a Politician*, Journal of Drug Issues, 9, 341347, 1979.

[10] Gerstner, Wulfram, and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge, U.K.: Cambridge UP, 2002. Print.

[11] Maass, Wolfgang. *Networks of Spiking Neurons: The Third Generation of Neural Network Models.* Egham, Surrey: Royal Holloway, U of London, 1996. Print.

[12] Hodgkin, A. L., and A. F. Huxley. *A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve.* The Journal of Physiology 117.4 (1952): 500-44. Web.

[13] Tuckwell H. C.. *Introduction to Theoretic Neurobiology.* Cambridge University Press, 1988.

[14] Izhikevich, E.m. *Simple Model of Spiking Neurons.* IEEE Trans. Neural Netw. IEEE Transactions on Neural Networks 14.6 (2003): 1569-572. Web.

[15] Hughes, John R.. *Post-Tetanic Potentiation.* Physiological Reviews 38 (1): 91113. PMID 13505117, 1958.

[16] Hebb, D. O. *The Organization of Behavior; a Neuropsychological Theory.* New York: Wiley, 1949. Print.

[17] Taylor MM. *The Problem of Stimulus Structure in the Behavioural Theory of Perception.* S. African J. Psychology 3: 2345, 1973.

[18] Markram, H. *Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs.* Science 275.5297 (1997): 213-15. Web.

[19] Kempter, Richard, Wulfram Gerstner, and J. Leo Van Hemmen. *Hebbian Learning and Spiking Neurons.* Physical Review E Phys. Rev. E 59.4 (1999): 4498-514. Web.

[20] Gutig, R., Aharonov, R., Rotter, S. and Sompolinsky, H.. *Learning Input Correlations through Nonlinear Temporally Asymmetric Hebbian Plasticity.* J. Neurosci. 23:3697-3714, 2003.

[21] Dan, Yang, and Mu-Ming Poo. *Spike Timing-Dependent Plasticity of Neural Circuits.* Neuron 44.1 (2004): 23-30. Web.

[22] Schultz W. *Predictive Reward Signal of Dopamine Neurons.* J Neurophysiol. 80:1–27, 1998.

[23] Hull, Clark Leonard. *Principles of Behavior, an Introduction to Behavior Theory.* New York, London: D. Appleton-Century, Incorporated, 1943. Print.

[24] Otmakhova NA, Lisman JE. *D1/D5 Dopamine Receptor Activation Increases the Magnitude of Early Long-Term Potentiation at CA1 Hippocampal Synapses.* J Neurosci. 16:7478–7486, 1996.

[25] Izhikevich, E. M. *Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling.* Cerebral Cortex 17.10 (2007): 2443-452. Web.

[26] Florian, Rzvan V. *Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity.* Neural Computation 19.6 (2007): 1468-502. Web.

[27] Baxter, J., Bartlett, P. L., and Weaver, L. *Experiments with Infinite-Horizon, Policy-Gradient Estimation.* Journal of Artificial Intelligence Research (2001), 15, 351381.

[28] Markram, Henry. *A History of Spike-timing-dependent Plasticity.* Front. Syn. Neurosci. Frontiers in Synaptic Neuroscience 3 (2011): n. pag. Web.

[29] Bloomfield, Stewart A., and Ramon F. Dacheux. *Rod Vision: Pathways and Processing in the Mammalian Retina.* Progress in Retinal and Eye Research 20.3 (2001): 351-84. Web.

[30] Desimone, R. *Neural Mechanisms of Selective Visual Attention.* Annual Review of Neuroscience 18.1 (1995): 193-222. Web.

[31] Roelfsema, Pieter R., and Arjen Van Ooyen. *Attention-Gated Reinforcement Learning of Internal Representations for Classification.* Neural Computation 17.10 (2005): 2176-214. Web.

[32] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1989. Print.

[33] Orjan Ekeberg, *DD2431 Machine Learning. Lab 4: Reinforcement Learning.* Available online at csc.kth.se/utbildning/kth/kurser/DD2431/ml10/rl.pdf

[34] Christodoulou, Chris, Gaye Banfield, and Aristodemos Cleanthous. *Self-control with Spiking and Non-spiking Neural Networks Playing Games.* Journal of Physiology-Paris 104.3-4 (2010): 108-17. Web

[35] Osborne, Martin J., and Ariel Rubinstein. *A Course in Game Theory.* Cambridge, MA: MIT, 1994. Print.

[36] Dutech, A. et al. *Reinforcement Learning Benchmarks and Bakeoffs II*, Workshop at the NIPS conference, 2005. Available online at
cs.rutgers.edu/ mlittman/topics/nips05-mdp/bakeoffs05.pdf

[37] Ali, Syed, Sven Koenig, and Milind Tambe. *Preprocessing Techniques for Accelerating the DCOP Algorithm ADOPT.* Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems - AAMAS '05 (2005): n. pag. Web.

[38] Jelle R. Kok, Nikos Vlassis, *Collaborative Multiagent Reinforcement Learning by Payoff Propagation*, Journal of Machine Learning, 2005, Vol. 7, 1789-1828.

[39] Fahlman, Scott E., and Christian Lebiere. *The Cascade-Correlation Learning Architecture.* Pittsburgh, PA: Carnegie Mellon U, Computer Science Dept., 1989. Print.

[40] Lawrence, Stephen Robert., C. Lee. Giles, and Ah Chung. Tsoi. *What Size Neural Network Gives Optimal Generalization?: Convergence Properties of Backpropagation.* College Park, MD: U of Maryland, 1996. Print.

# APPENDIX A

# IZHIKEVICH MODEL VS LIF MODEL

An experiment to evaluate and compare the performances of Izhikevich neurons and LIF neurons in the context of the proposed methodologies was performed. The task was to functionally reproduce the Exclusive OR behavior. The network employed consisted of 80 neurons in the hidden layer, and 80 neurons in the input layer. 100 Hz poisson spike inputs were fed into the network using the balanced-pair binary rate encoding. The simulation was run for 200 epochs runtime (simulation time: 100 seconds at 500ms per epoch). The run-times of the simulation are listed in Table. A.1. Details of simulation environment are provided by Table. A.2.

Table A.1: Run times: Izhikevich vs LIF

| Model | Simulation runtime |
|---|---|
| Izhikevich (a=0.02/ms, b=0.2/ms) | 23.975 minutes |
| LIF ($\boldsymbol{\tau}$ = 20ms) | 17.476 minutes |

Table A.2: Simulation environment

| | |
|---|---|
| Processor | Intel XEON 3.2GHz 2MB L2/L3 Cache |
| Cores | 8 |
| Operating System | Ubuntu 14.04.2 LTS |
| Language | Python 2.7 |
| Framework | Brian 1.0 (Spiking Neural Network Library) |
| Multi/Parallel Processing | No |

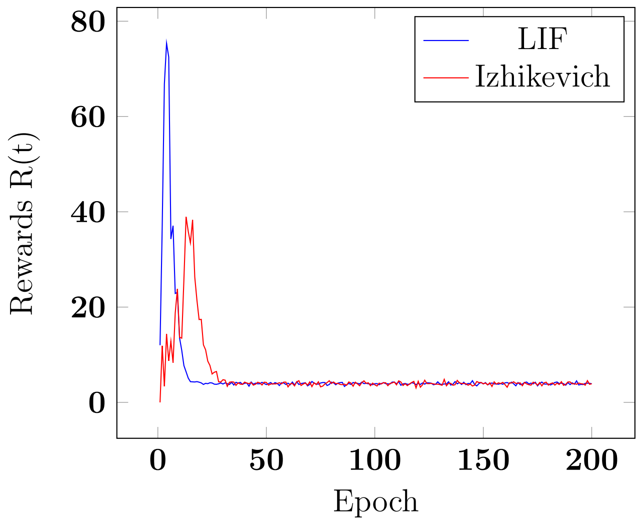


Figure A.1: Reward profile comparison

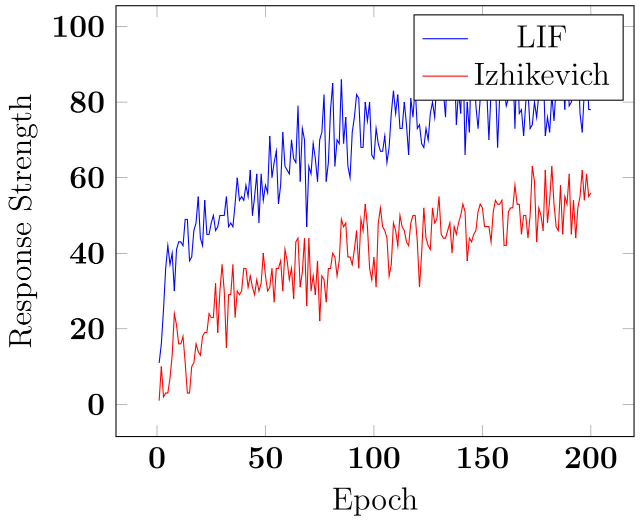


Figure A.2: Response strength profile comparison

The reward profiles with LIF and Izhikeivch neurons are compared in Figure. A.1, and the strength of response exhibited by the networks is compared in Figure. A.2. According to the results, the leaky integrate-and-fire neurons exhibit better response and performance while being computationally less intensive as well, though not by much compared to Izhikevich, which makes both the models suitable for implementation with the proposed reinforcement learning methodology.