EVOLUTIONARY ADAPTATION AND DOPAMINE MODULATED LEARNING IN SPIKING NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
MATERIALS SCIENCE AND NANOTECHNOLOGY

By
Abdurrezak Efe
July 2021

Evolutionary adaptation and dopamine modulated learning in spiking neural networks
By Abdurrezak Efe
July 2021

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Seymur Jahangirov(Advisor)

Aykut Erbaş

Alpan Bek

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

EVOLUTIONARY ADAPTATION AND DOPAMINE MODULATED LEARNING IN SPIKING NEURAL NETWORKS

Abdurrezak Efe
M.S. in Materials Science and Nanotechnology
Advisor: Seymur Jahangirov
July 2021

Amidst the rise of Deep Learning (DL) in the last ten years, many decades withstanding problems such as image classification, machine translation and text generation received superhuman level solutions. Combining DL with other paradigms such as Reinforcement Learning (RL) led to even more astonishing achievements such as solving the game of Go, 3D Protein Folding and scene reconstruction. However, not only Artificial Neural Networks (ANNs) used in DL require a massive amount of data as the problems get complicated (curse of dimensionality) but also the learning algorithms used in ANNs such as backpropagation are biologically not plausible. On the other hand, Spiking Neural Networks (SNNs) are computationally powerful nonlinear dynamical systems that are biologically more credible. Thus, approaching problems using SNNs promises the possibility of transferring methods and discoveries from Neuroscience. In this work, we utilized a well known genetic algorithm called NeuroEvolution of Augmenting Topologies (NEAT) on SNNs to train agents which can solve various nonlinear problems such as XOR, pole balancing and food chasing. Afterwards, we applied dopamine modulation on Spike-Timing Dependent Plasticity (STDP) and attested that dopamine modulated STDP can indeed solve harder problems such as discovering good and bad nutrition types while trying to catch and consume food. The gist of the problem is that the same food can be beneficial in one episode/trial and detrimental in another one. As a result, we have shown that NEAT applied to SNNs can solve various nonlinear tasks; however, it cannot solve problems where in-life adaptation and/or discovery is required.

Keywords: Spiking Neural Networks, Dopamine modulation, NEAT, Reinforcement Learning.

ÖZET

ATIMLI SİNİR AĞLARINDA EVRİMSEL ADAPTASYON VE DOPAMİN MODÜLASYONLU ÖĞRENME

Abdurrezak Efe
Malzeme Bilimi ve Nanoteknoloji, Yüksek Lisans
Tez Danışmanı: Seymur Jahangirov
July 2021

Derin Öğrenmenin son on yıldaki yükselişi ile birlikte, görüntü sınıflandırma, makine çevirisi ve metin oluşturma gibi on yıllardır çalışılan problemlere insanüstü düzeyde çözümler bulundu. Derin Öğrenmenin Pekiştirmeli Öğrenme gibi farklı paradigmalarla birleştirilmesi ise Go oyununu çözme, yaklaşık elli yıldır çözülemeyen üç boyutlu protein katlamayı çözme ve üç boyutlu manzara rekonstrüksiyonu oluşturma gibi daha etkileyici sonuçlara sebep oldu. Ancak Derin Öğrenme sadece çok büyük miktarlarda veriye ihtiyaç duymakla kalmaz aynı zamanda kullandığı geri yayılım gibi öğrenme algoritmaları da biyolojik olarak anlamlı olmaktan uzaktır. Öte yandan, Atımlı Sinir Ağları biyolojik nöronları neredeyse birebir taklit edebilen dinamik sistemler olduğundan; araştırmacılara Sinirbilimden metod ve fikirleri doğrudan simülasyonlara uygulama şansı tanımaktadır. Bu çalışmada, Atımlı Sinir Ağları üzerinde; XOR, direk dengeleme ve yemek yakalama gibi problemleri çözebilen sinir ağları eğitmek için iyi bilinen bir genetik algoritma olan Artırılan Topolojilerle Nöral Evrim algoritmasını kullandık. Daha sonra, Atım Zamanına Bağlı Plastisite algoritmasına dopamin modülasyonu uygulamanın yiyecekleri yakalarken faydalı veya zararlı olanları seçmek gibi daha zor problemleri çözmede etkin olduğunu gösterdik. Burada çözülen problemin can alıcı noktası yemeklerin fenotipleri değişmezken faydalı veya zararlı olmaları özelliklerinin değişebilmesidir çünkü bu, sinir ağlarının sadece genetik olarak yemek yeme arzularının olmasını değil ortamda keşif yaparak faydalı olan besini bulmalarını gerektirmektedir. Sonuç olarak, Atımlı Sinir Ağlarının Artırılan Topolojilerle Nöral Evrim ile eğitildiklerinde doğrusal olmayan problemleri çözebildiklerini ancak adaptasyon ve keşif gerektiren problemlerde dopamine modülasyonu olmadan yetersiz olduklarını gösterdik.

Anahtar sözcükler: Atımlı Sinir Ağları, Dopamin, NEAT, Pekiştirmeli Öğrenme.

Acknowledgement

First and foremost, I would like to express my infinite gratitude to my advisor Asst. Prof. Seymur Jahangirov. Who has been nothing but the perfect mentor, therapist and even a big brother from time to time. Conducting research can become staggering, yet he made it endurable if not delightful. Apart from Dr Jahangirov, I would like to thank Asst. Prof. Aykut Erbaş and Assoc. Prof. Alpan Bek who are on my thesis committee, for their invaluable critiques.

I would like to thank my brother Abdulmennan Efe, the best man and the best brother in the world, who always gave me the biggest support. I wouldn't have been able to get to this point without his presence and guidance. I would also like to sincerely thank my dearest angelic mother, father and my dear sisters Şeyma, Büşra and Zuhal, who constantly uplifted me and warmed me with their love.

I want to express my endless gratitude to my beloved, Nida, who raised my mood whenever I felt empty and hopeless on this thorny road. Her loving voice made even the most unsolvable problems trivial. And my deepest appreciation to Selim and Buğra, whom I call my brothers and who have been with me since my primary school years. Heartfelt thanks to my oldest friend, Muhammed, for always giving me wisdom and perspective mixed with love, even though we are miles apart. I am also sincerely grateful to Eda, whom I consider to be my fourth sister, for never leaving me alone on this quest and many more.

My merrymaking and Trabzon companions Yağız and Göktuğ are also deeply appreciated for their motivating and enlightening inputs. I would also like to thank my best friends Bado, Helin and Serra for accompanying and working with me during all those sleepless nights. Their smiles and companionship have always made me elated and ever at home. I also am blessed and grateful to have had my great office mates Merve Üstünçelik, Şahmurat Kazak, Kerem Kurban and Soheil Ershad Rad, who made the office feel more like home. I want to thank all of my friends and professors, whose names are not mentioned here, who have

supported me in UNAM and Bilkent in the last eight years. Also my wise friends from Interdisciplinary Journal Club, I love you all and am lucky to have you all.

Lastly, I would like to express my endless thanks to İnsan Doğramacı Bilkent University, my home, and UNAM, where cutting edge scientific research is conducted as a family. Also, I cannot describe how thankful I am to late Prof. İhsan Doğramacı, may he rest in peace, who made it all possible by establishing Bilkent University and much more.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The term Artificial Intelligence (AI) was first coined by John McCarthy in 1956 and the first building blocks of modern AI formed around the 1960s with the invention of peceptrons. Through the last fifty years, we have witnessed the first hype, winter and the rebirth of AI. Multiple paradigms have been utilized through the history of AI such as Reinforcement Learning (RL) where the agents are in constant interaction with their environments and receive rewarding signals whenever they act as preferred so as to enhance their model of the environment, Deep Learning (DL) where any agent consists of layers of neurons (hence "deep") whose connections are trained through mathematical optimization methods and Long Short Term Memory Networks (LSTMs) which are also neural networks of multiple layers with recurrent connections. However, none of the aforementioned approaches prospered before the last decade. Although the basic ideas of all these paradigms originated from cognitive and systems neuroscience, they diverged to a point where researchers are not simulating the methods from neural sciences but trying to realize the core algorithms with mathematical crafts and shortcuts. It works astonishingly in countless domains from protein folding [1] to image classification [2].

On the other hand, neuroscience-based AI research along with computational neuroscience has been progressing as well. One of the main constructs in the

latter field is known as Spiking Neural Networks (SNNs) where the units are not direct mathematical computational bricks like their DL counterparts but dynamical systems. There are multiple spiking neuron models such as Integrate and Fire (IF), Leaky-Integrate and Fire (LIF), FitzHugh-Nagumo, Izhikevich and spike response model each of which has different strengths and weaknesses in terms of biological plausibility, computational complexity etc [3]. Having multiple dynamical systems as units, SNNs are ensembles of dynamical systems which makes Dynamical Systems Theory applicable to them. Moreover, SNNs, being biologically more plausible, can be used to simulate real brain regions in silica. Indeed, there are multiple studies under the umbrella of the Human Brain Project that aim to simulate various functional units of the brain such as the hippocampus [4].

SNNs have been shown to be effective and in many cases, sometimes even better than their DL counterparts such as image classification [5], optimal control [6] and multi-sample online learning [7]. Other various applications of SNNs abound. However, although some promising work has been done [8], there is not a straightforward learning algorithm like backpropagation for us to learn the weights and other parameters of an SNN. Not only that, just like in DL it is not easy to find a good architecture for an SNN let alone determining the optimal input encoding. Finding a good architecture and a powerful/general learning algorithm for SNNs are the main setbacks but also adventurous challenges for researchers. Table 1.1 compares the properties of RL, DL and SNN agents in terms of their biological plausibility, optimal learning model and architecture. We took the adventure of training SNNs and utilized two methods to overcome the aforementioned setbacks. The methods will be briefly mentioned below and in more detail in the following chapters.

Common Paradigms vs SNNs			
Model	Optimal architecture	Learning algorithm	Biological plausibility
RL	doesn't require	TD(λ)	NO
DL	NO	SGD	NO
Deep RL	NO	SGD + TD(λ)	NO
SNNs	NO	STDP(proposed)	YES

Table 1.1: Here is a summary of three paradigms used in AI. The table shows that optimal architecture is not known for any of the methods where a network is employed. On the other hand, learning algorithms shown are the most common ones which are compatible to each other. The only models that are biologically plausible are SNNs.

There are limited methods to determine an almost-optimal architecture for a neural network. One of the most promising paths to follow is genetic-based algorithms. Genetic Algorithms (GAs) are metaheuristic search algorithms, generally guaranteed to reach a local minimum in a finite time. GAs can be utilized in numerous fields including but not limited to image processing [9], protein structure prediction [10], job scheduling [11], bioinformatics [12] and neural circuit evolution [13]. We use NeuroEvolution of Augmenting Topologies as our basis framework to train SNNs later on [13]. To handle the learning algorithm problem, we utilize dopamine based modulation on the plastic structure of the synaptic weights. Both plasticity and dopamine pathways are important bricks on the wall of the learning brain.

In this thesis, we present solutions to multiple problems using SNNs under evolutionary algorithms, specifically NeuroEvolution of Augmenting Topologies, along with dopamine based modulation. We show that evolutionary algorithms can solve various problems without plasticity such as XOR, pole balancing and food chasing. We reveal that even harder problems such as food chasing under

changing rules can be solved with unsupervised reward signals with the evolutionary approach. Finally, we present a harder case where the food chasing problem under changing rules need to be solved without sensory reward signals and as a result evolutionary algorithm fails to generalize because of the randomness of the rules; meanwhile, the dopamine modulated version solves it easily.

The remaining parts of this thesis is as follows. In Chapter 2 we give the motivation and mention the related works to our research. In Chapter 3, preliminary theory and the notation is provided. In Chapter 4, the experiments and results are presented in detail. And finally, In Chapter 5 the conclusive remarks are made and the future work is contextualized.

Chapter 2

Motivation and Related Work

In this chapter, our motivation to do this particular research on Spiking Neural Networks (SNNs) will be explained and then the related works in academia and in the industry will be summarized.

2.1 Motivation

Although there are many fields either clearly or vaguely under the umbrella of Machine Learning (ML), Deep Learning (DL) is the most popular one that has proven its powers and considered to be holding even more potential. DL employs deep neural networks, generally of many layers, to solve problems of various domains. However, whether DL is the answer to the search of general intelligence (or Artificial General Intelligence) is yet to be ascertained. Like many in this field, we also aim to create Artificial General Intelligence (AGI) at some point. And, we believe the path to AGI has to visit Neuroscientific concepts if not completely pass through Neuroscience. Thus we do not believe DL has all the answers. Neither do we hold the belief that we must simulate a neural unit with all of its details down to the molecular level. Rather, based on the idea that not the microscopic but the macroscopic behaviour of neuronal ensembles create

intelligence, we assume that a sufficiently detailed version of the neurons with a correct architectural and parametric configuration can take us one step closer to a realistic implementation of an artificial general intelligent system.

Granting the definition that intelligence is the adaptive success in a wide range of environments [14], we believe it is feasible to try out the structures from the only system we know to fit the definition, the human brain. Since SNNs are considerably closer to their biological counterparts, we think that they are an encouraging path to follow.

Considering the SNNs research is in its late infancy, we believe this research that explores their capabilities in various problems with several paradigms such as genetic algorithms and dopamine modulated STDP can be a promising early step. We presume that SNNs research will receive widespread interest in the following decades if not years as their computation becomes easier with the advances like SpiNNaker and more and more researchers flood into the field [15]. As will be explained in detail, there are multiple studies that follow the claim that transient and itinerant dynamics of the human brain can be simulated with SNNs [16], the field will be a hot haven for researchers from numerous fields ranging from psychology to mathematical machine learning.

We also believe that SNNs can open doors to develop synthetic neural networks that can be deployed as electro-mechanical systems which are constructed with Nanotechnology. We expect that in the near future, neural networks are going to be distributed with single purpose systems produced via Nanotechnology and Materials Science.

All in all, we conclude that SNNs are one of the most obvious keys to the gate of AGI that needs to be studied more and more. And we understand that our work will shed light not only on the capabilities of SNNs but also on the capabilities of genetic algorithms and dopamine based reinforcement learning.

2.2 Related Works

SNNs have been employed in many fields, the ones of which related to our research will be specified in this chapter. There are three major and one semi-major components of this thesis. Major ones are evolutionary training of the SNNs, Spike-Timing Dependent Plasticity (STDP), Reinforcement Learning (RL) on SNNs through dopamine modulation on STDP. Meanwhile, the semi-major component is the so-called Free Energy Principle (FEP).

2.2.1 Evolutionary Training of SNNs

Genetic Algorithms applied to SNNs bear fruitful results. Not to flood this section with sources, most related works done on this field are as follows. Authors in [17] show that the Grammatical Evolution algorithm applied on an SNN with three layers can solve classification problems with comparable results to Artificial Neural Networks (NNs). Another successful solution to classification problems can be found in [18] where the authors used Parallel Evolution Algorithm. Meanwhile, researchers in [19] solve nonlinear control problems such as pole balancing with NeuroEvolution of Augmenting Topologies (NEAT) on SNNs consisting of Izhikevich neurons. This result is particularly significant for us because we reproduce this experiment before getting into harder problems such as food chasing, although we do not use the mean firing rate as encoding.

2.2.2 STDP

STDP is an algorithm that originates from Hebbian Learning, so it has been around for a while. The following references are the ones that inspired our work the most. In [20], researchers proved that STDP can optimally represent the continuous environments. One of the significant results of their paper is that STDP can lead to a superb level of video frame recognition even better than ANNs, RNNs, CNNs and LSTMs. As we employ SNNs to recognize their scenes

with sensory inputs in continuous environments (2D rooms), this result provides invaluable insight for our work. Reference [21] shows that SNNs with STDP can actively apply Expectation-Maximization (EM) on their sensory inputs. EM is a latent variable model optimization problem that is widely used on variational inference problems. The aforementioned result not only is remarkable in that it displays how powerful STDP indeed is, but also it allows us to devise new tasks where the agents might need to separate their input sources. The last problem in Chapter 4, the matrix chase problem, is an example where the agents need to discriminate between the sensory inputs and make decisions. Another supporting result can be found in [5] where an unsupervised image classification task is solved by only employing STDP with a feedforward SNN.

2.2.3 Dopamine Modulation on STDP

Dopamine is known to be a strong modulator in the mammalian brain that is closely related to addiction, pleasure and learning. As an RL ingredient for SNNs, some authors have proposed third-factor modulators to be applied on top of STDP [22]. As the most studied and effective one, dopamine comes forward. Izhikevich shows an effective formulation followed by an application of dopamine modulated STDP in [23] where he shows shifting of an unconditional stimulus to a conditional stimulus with dopamine injection. More crucially, the dopamine rewards are given after a random duration of the action so as to present a framework where the distal reward problem is solved principally. This paper is one of the main parts of our study's foundation as we used Izhikevich's model of dopamine modulation with some simple nuances. In [24], authors have presented a robust SNN with inspiration from the anatomical structure of the reward mechanism of the human brain. They created an embodied agent within a drone that learns to fly through windows successfully. Finally, [25] shows a working example of an SNN simulation that actively performs chemotaxis. The author uses dopamine modulation on Izhikevich neurons.

2.2.4 FEP

FEP has increasing popularity among researchers from various fields such as systems neuroscience, AI, statistical physics and psychology. Our work touches on the ideas from FEP when we analyze the evolution of our agents under NEAT. Since FEP is a rather new framework, there are not many studies focused on the cross-section of evolutionary algorithms and FEP. However, [26] researched a genetic algorithm under the inspection of both FEP and the Integrated Information Theory (IIT), where they proposed a method to calculate so-called sensory surprisal. We used the same method to calculate the sensory surprisal in this thesis.

Chapter 3

Theory and Notation

Before getting into computational simulations and experiments, it is only convenient to provide the reader with some theoretical preliminaries. In the following sections, Izhikevich neuron model will be introduced and some properties of them will be shown. Then, Spiking Neural Networks (SNNs) that consist of multiple Izhikevich neurons will be mentioned to move on to the topic of synaptic plasticity, in particular Spike-Timing Dependent Plasticity (STDP). After mentioning dopamine modulation with STDP, we will move on to a brief introduction to Free Energy Principle and finally finish this chapter with NeuroEvolution of Augmenting Topologies.

3.1 Izhikevich Neuron Model

Izhikevich Neuron Model is a computational representation of a biological neuron, given as a nonlinear dynamical system. Izhikevich neurons can replicate the behaviour of their biological counterparts such as low-threshold spiking, chattering and fast-spiking [27]. The model consists of two differential equations and four parameters.

$$\begin{aligned} \dot{v} &= 0.04v^2 + 5v + 140 - u - I \\ \dot{u} &= a(bv - u) \end{aligned} \tag{3.1}$$

Equations in 3.1 represent an ordinary 2D dynamical system where v stands for the membrane potential, u represents the membrane recovery variable, I is the synaptic current and a, b, c and d are dimensionless parameters [27]. Different choice of a, b, c and d leads to different behaviours such as bursting, chattering etc.

$$\text{if} \quad v \geq 30 \quad mV, \quad \begin{cases} v \longleftarrow c \\ u \longleftarrow u + d \end{cases} \tag{3.2}$$

Equation 3.2 simply represents the emergence of an action potential when the membrane potential reaches its peak value. A more rigorous and detailed explanation of Izhikevich neurons can be found in the original text [28]. As a starting point, we reproduced a simulation of Izhikevich neurons. In Figure 3.1, an Izhikevich neuron with a randomly chosen configuration of a, b, c and d variables is simulated. Although the neuron shown in the figure is an Intrinsically Bursting (IB) neuron, adjusting the values of a, b, c and d allows one to simulate over 20 different neuronal firing patterns.

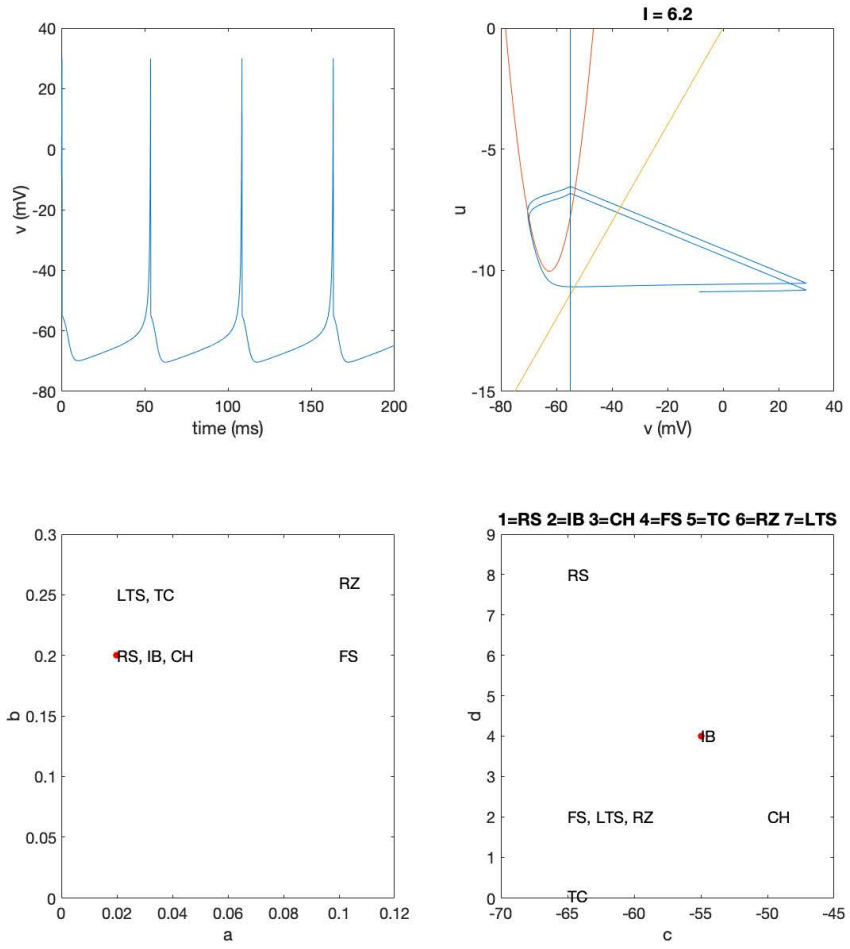

Figure 3.1: An Izhikevich model neuron's behaviour with current input $I = 6.2$ and parameters $a = 0.02, b = 0.2, c = -55$ and $d = 4$. This neuron acts as an Intrinsically Bursting neuron (IB). On top left, change of membrane potential v with time t is given. On top right, phase diagram of u vs v is given with $I = 6.2$. On bottom left, a basic map of neuron types with respect to their a and b values is given where the red dot represents the current example. At last, on bottom right, neuron types are given with respect to their c and d variables and again the red dot represents the current neuron type with given values.

Of course, multiple types of neurons can be simulated with different a, b, c and d values. Figure 3.2 shows some of the possible neuron types with changing values of the parameters.

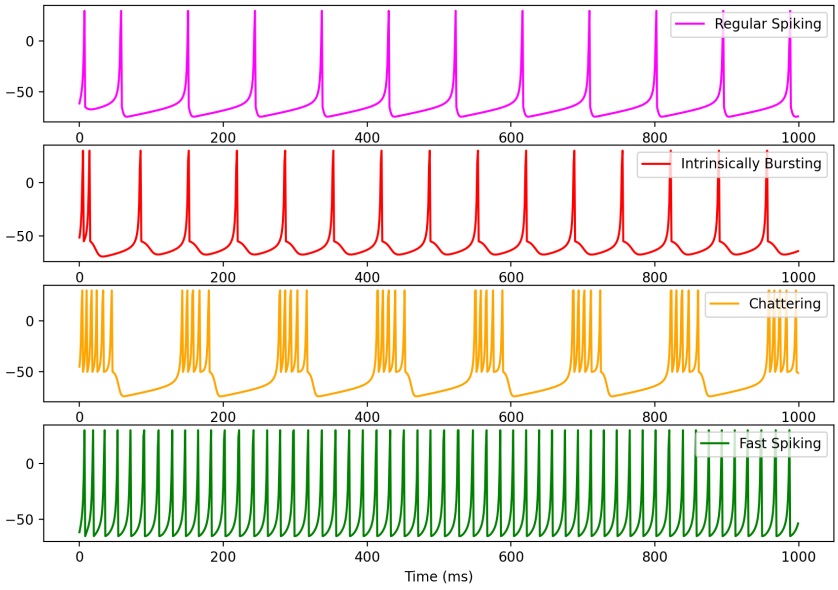

Figure 3.2: Here we show Regular Spiking (RS), Intrinsically Bursting (IB), Chattering (CH) and Fast Spiking (FS) neurons' response to constant $I = 10$ mA current for 1000 ms.

The version of the Izhikevich neuron model utilized in our experiments is a bit more complex. The model includes parameters that govern the short term depression and facilitation, namely Markram parameters [29]. The aforementioned parameters are R, W, U, D and F. Where R is responsible for depression and W is responsible for facilitation; meanwhile, U, F and D are conventional parameters obtained from cortical neurons.

$$\begin{aligned} \dot{R} &= \frac{1-R}{D} - RW\delta(t-t_n) \\ \dot{W} &= \frac{U-W}{F} + U(1-W)\delta(t-t_n) \end{aligned} \tag{3.3}$$

Equation 3.3 shows the dynamics of Markram parameters where RW represents the amount of neurotransmitter available at time t. If a presynaptic neuron fires a spike at t_n, the depression decreases by that amount and facilitation increases by $U(1 - W)$.

Having established the Izhikevich neuron model, we should lastly check how a synapse fits into this system. Then we can move on to the networks that are made of Izhikevich neurons, namely SNNs.

During the design of a synaptic connection, one can go into a tremendous amount of detail. However, our model follows the convention where the synapses are a means to convey signals with receptors and neurotransmitters present. The model has four components, namely $AMPA$, $NMDA$, $GABA_A$ and $GABA_B$. In neuroscience literature, $AMPA$ and $NMDA$ are Glutamate receptors that are known to depolarize the postsynaptic cell whereas $GABA_A$ and $GABA_B$ are receptors of gamma-aminobutyric acid (GABA) which hyperpolarizes the postsynaptic cell [30]. Thus, both $AMPA$ and $NMDA$ are excitatory while $GABA_A$ and $GABA_B$ are inhibitory. Conveniently, when an excitatory presynaptic spike arrives at the postsynaptic cell, we increase the amount of $AMPA$ and $NMDA$ while an inhibitory spike causes the increase of $GABA_A$ and $GABA_B$ parameters.

$$if \quad S_{ij} = \text{Excitatory} \begin{cases} \Delta g_{AMPA} = w_{ij} R_i W_i \\ \Delta g_{NMDA} = w_{ij} R_i W_i \end{cases} \tag{3.4}$$

$$if \quad S_{ij} = \text{Inhibitory} \begin{cases} \Delta g_{GABA_A} = w_{ij} R_i W_i \\ \Delta g_{GABA_B} = w_{ij} R_i W_i \end{cases} \tag{3.5}$$

Equations 3.4 and 3.5 summarizes the dynamics of receptors where S_{ij} represents the synapse from presynaptic neuron j to postsynaptic neuron i and w_{ij} represents the strength/weight of the synapse. Additionally, the value of any of the four parameters decreases exponentially with predetermined time constants.

Equation 3.6 shows a general form of the time evolution of any of the receptor parameters where g_x represents the parameter and τ_x represents the corresponding time constants that had been acquired by conducted experiments [29].

$$\dot{g_x} = \frac{-g_x}{\tau_x} \tag{3.6}$$

Before getting into SNNs, we need to establish how an input current affects the synaptic current. The amount of each neurotransmitter present will affect the synaptic current as well as the injected current.

$$I = g_{AMPA}v + g_{GABA_A}(v+70) + g_{GABA_B}(v+90) + g_{NMDA}\frac{(v+80/60)^2}{1+(v+80/60)^2}v - I_i \tag{3.7}$$

As Equation 3.7 shows, the injected current decreases the value of the synaptic current. One can feel as if it has a hyperpolarizing effect on the cell; however, when we revisit Equation 3.1, we see that it is indeed a depolarizing thing to have a negative synaptic current.

3.2 Spiking Neural Networks (SNNs)

Spiking Neural Networks are connected graphs of computational neuronal units. In our work, we have used the Izhikevich neuron model as it is computationally cheaper than more bio-physically reasonable models such as Morris-Lecar and Hodgkin-Huxley, and produces the same or even better results in simulations in terms of neuronal behaviour [3].

SNNs can be classified into different groups in terms of architecture or neuron types used. As it is a trivial distinction to divide them into groups concerning the neuron type used, we rather pay attention to architectural distinctions.

3.2.1 Feedforward SNNs

Feedforward SNNs are neural networks that do not have synapses from deeper layers to any past layers just like in Artificial Neural Networks (NNs). These are the SNNs that we made use of mostly in this project as the mathematical theory of recurrent SNNs is not well established yet.

3.2.2 Recurrent SNNs (RSNNs)

As the name suggests, RSNNs are neural networks where it is possible to have synapses directed from deeper layers to previous layers. There are many studies regarding RSNNs and it is shown that their biological counterparts play a major role in memory establishment in the human brain [31]. Recurrent Neural Networks (RNNs) have been playing a major role in Artificial Intelligence for decades and they are particularly good in fields such as Natural Language Processing, Time-series data analysis and Speech Recognition. Meanwhile, RSNNs are hard to train with known paradigms such as Spike-Timing Dependent Plasticity because they are ensembles of dynamical systems and their behaviour can easily slide to the realms of chaos. However, some studies show that they can be trained with backpropagation based methods such as BPTT [32].

3.3 Spike-Timing Dependent Plasticity (STDP)

Synaptic weights between neurons can get stronger or weaker through time depending on the relative activity of pre and postsynaptic neurons, a phenomenon called synaptic plasticity [33]. The notion of plasticity lies on the heart of Hebbian Learning, which states that neurons with consistent synchronous firing patterns get stronger connections as a result of this consistency [34]. A more causal model of synaptic plasticity is known as Spike-Timing Dependent Plasticity (STDP) which takes its roots from the same idea. STDP algorithm has two components,

Long-Term Potentiation (LTP) and Long-Term Depression (LTD); representing the increase of synaptic weight and decrease of synaptic weight respectively. Given the action potential(signal) s_{ij} from presynaptic neuron n_i to postsynaptic neuron n_j reaches to n_j at time t_0 and n_j fires after Δt seconds; LTP can be formulated as the following:

$$\Delta w_{ij} = A_+ e^{-\frac{\Delta t}{\tau_+}} \tag{3.8}$$

Where τ_+ is potentiation time constant and A_+ is a parameter that may depend on the value of the synapse at that moment [35]. The equation for LTD is unsurprisingly very similar to 3.8. If the postsynaptic neuron fires a spike just before the presynaptic neuron's action potential arrives at the postsynaptic cell, the synaptic weight decreases.

$$\Delta w_{ij} = -A_- e^{-\frac{\Delta t}{\tau_-}} \tag{3.9}$$

Where A_- is a parameter and τ_- is the time constant for LTD. Note that these changes are only for a single action potential pair. One can include only the most recent firings or all of the firings throughout the neurons lifetime depending on the convenience. For the sake of simplicity, our model uses only the most recent firings for both pre and postsynaptic neurons. More detailed versions of STDP can be found at [35].

Figure 3.3 shows two neurons that are connected as a loop. The first neuron which is drawn in blue receives a random input between 0 and 8 mA during the first half of the simulation; meanwhile the second neuron receives 0 mA. In the second half the roles are changed, the second neurons receives a random current input and the first neuron receives 0 mA.

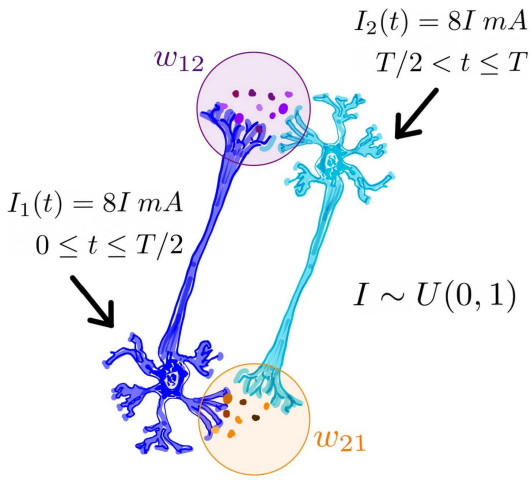

Figure 3.3: Two neurons as a loop is shown, the first one is blue and the second one is cyan. For the first half of the simulation, the first neuron is stimulated with a current $I_1(t) = 8I\ mA$ and the second neuron receives no injected current. On the second half of the simulation only the second neuron is stimulated with an injected current of $I_2(t) = 8I\ mA$ where I is sampled from the uniform distribution $U(0,1)$. The weights of the synapses from the first to the second neuron and from the second to the first neuron are w_{12} and w_{21} respectively.

Figure 3.4 shows a clear example of STDP where two excitatory neurons fire with causal and non-causal patterns. When the firing is causal, the weight of the first synapse' weight increases and when the firing pattern is non-causal, the weight of the second synapse decreases. This behaviour can be considered with an analogy. Imagine two friends generally making plans together, call them A and B. A gives ideas to B and B decides on whether to do it or not. If A repeatedly gives ideas that B finds reasonable and acts on accordingly, then B starts to listen A more carefully because his ideas make sense and B is acting with accordance to them. On the other hand, if A is coming up with ideas that B already discovered and acted upon, B thinks that A's ideas are no longer required. B can come up with them as easily. Thus when the idea propagation and action is causal, the connection gets stronger and when it is non-causal the connection gets weakened.

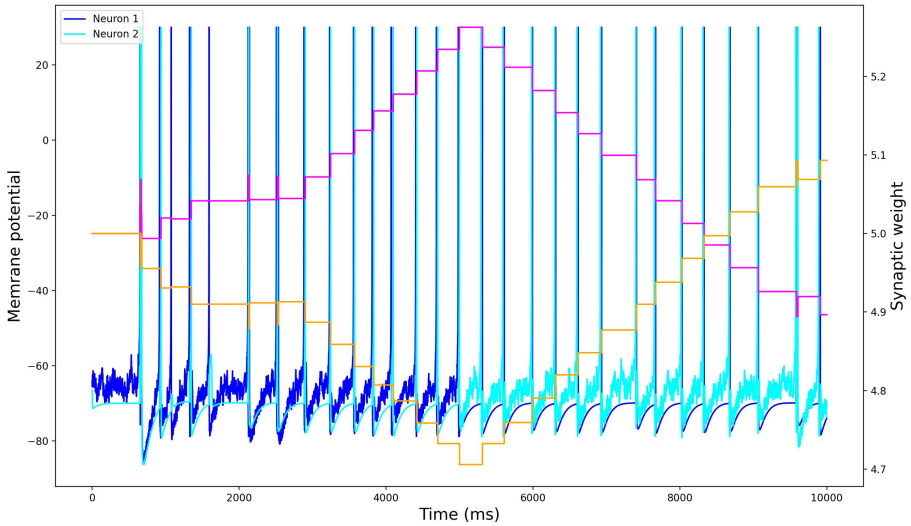

Figure 3.4: As the figure shows, the synaptic weight from Neuron 1 to Neuron 2 increases in the first half of the simulation due to the causal structure of the firings. However, on the second half the synaptic weight starts to decrease as the injected currents change. Of course the synaptic weight of the second synapse from Neuron 2 and Neuron 1 is affected by STDP as well, decreasing in the first half and increasing in the second. The Neuron1's potential is shown in Blue, and Neuron 2's potential in Cyan while the weight of the synapse from Neuron 1 to Neuron 2 is shown in Magenta and the other weight is shown in Orange.

3.4 Dopamine modulated STDP

Dopamine is a neurotransmitter accountable for the brain's reward and learning mechanisms. It is released from multiple regions but most of it originates from Substantia Nigra and Ventral Tegmental Area (VTA) in the midbrain. Dopamine has been shown to cause addiction, conditional learning and learning transfer in rodents [30, 36]. Moreover, dopamine is considered as one of the main elements in long-lasting behaviour learning in the human brain via modulating the plasticity of synapses [24]. However, since the molecular mechanisms of the modulation

are not yet fully understood, computational models of SNNs usually do not include any third-factor modulation of STDP let alone a modulation by dopamine. Therefore, there are multiple approaches towards achieving a working simulation of an SNN with dopamine modulated STDP that make use of the idea of synaptic eligibility traces [22, 23]. A synaptic eligibility trace is a variable that keeps track of the synapses that are to be affected by the release of dopamine.

In our work, we used the mechanism suggested by Izhikevich in [23]. If the amount of dopamine present is high, then any LTP or LTD process will be multiplied by a factor. Of course, in real life, the reward of an act is not always immediate or obvious. For instance, the decision a student makes to study a specific topic might bear results years later, the direction a rodent in a maze takes might lead to cheese many steps later. This is known as the distal reward problem or the credit assignment problem in Reinforcement Learning [37]. The model in [23] recommends using a synaptic eligibility trace that marks a synapse as eligible for some time after the LTP or LTD process responsible. Thus, if the reward (dopamine) is released after some time passed over the event, the synapse is still feasible to change its strength. The model introduces two new parameters on top of the Izhikevich neuron model, c, the eligibility trace of a synapse and d, the amount of dopamine present in the environment at the time. One important remark to make here is that these parameters should not be confused with Izhikevich neuron model parameters c and d given in 3.2. As these are the names of the parameters given in the original paper [23], we considered it to be convenient to use the same parameters here. Equation 3.10 manifests the exponentially decreasing dynamics of the eligibility trace where τ_c is the time constant for c, $\Delta w(t)$ is the change imposed by STDP at time t and t_f is any time a spike appears from either the presynaptic or the postsynaptic cell. Note that the eligibility trace changes whenever there is an STDP related event such as LTP or LTD.

$$\dot{c} = \frac{-c}{\tau_c} + \Delta w(t)\delta(t - t_f) \tag{3.10}$$

Meanwhile, the amount of dopamine rapidly increases whenever a reward is given to the agent which might be seconds later than a causal firing pattern that helped lead to the reward. Also, the amount of dopamine decreases exponentially with a given time constant. Of course, there is a source of dopamine, $DA(t)$, that is always providing the synapses with a modest amount so that they do not become fully non-plastic if no reward arrives for a long time. Equation 3.11 shows the change of d with time.

$$\dot{d} = \frac{-d}{\tau_d} + DA(t) \tag{3.11}$$

As a result, dopamine modulated STDP can help agents to acquire new behaviour if they are in a continuously changing environment. Indeed, we will later present some results that show that dopamine modulated STDP provides agents with the flexibility to change their behaviour in response to the same sensory input proving that they can adjust to the new conditions.

3.5 Free Energy Principle (FEP)

Free Energy Principle is an encompassing framework of existence and inference [38]. Although the full scope of FEP is out of topic for this thesis, we will introduce the core idea behind it and explain the parts relevant to this work. The following subsections will explain the essence of FEP, exhibit FEP equations and wrap up with the notion of making use of *surprisal*, an information theoretical concept that determines how unlikely an event is, given as $-\log(p)$ where p is the probability of concerning event.

FEP starts with the basic concept of existence. If something exists, it must be distinguishable from the environment it belongs in. The way to look at it is thoroughly statistical, by a construct called Markov Blanket. A Markov Blanket is a set of *states* that statistically insulates a subset of states from another [39]. The term state might refer to random variables, parameters or even events. For

the sake of argument, we will refer to neurons and their parameters in this work. The mathematical definition of a Markov Blanket of a set of nodes in a Bayes Network is the set of their parents, children and their children's [39]. Some examples of a Markov Blanket could be the cell wall of a bacterium, or the outer layer of oil molecules of an oil drop that isolate the internal molecules from the environment. Of course, the blanket states can be divided into two groups; the sensory states influenced by the environment but not by the internal states and the active states influenced by internal states but not influenced by the external states or environment. Figure 3.5 shows an example Markov Blanket where sensory, active, internal and external states are distinguished with various colours. To follow the convention, we use the letters used in [38]. Thus, sensory states are shown as s, active states with a, internal states as μ and the external states as η.

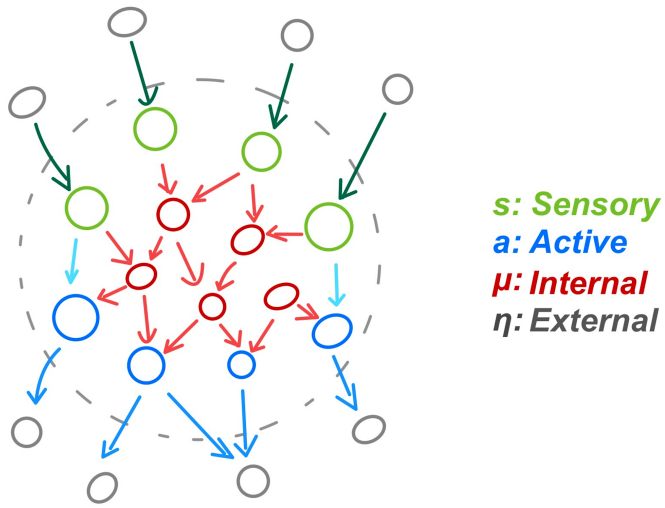

Figure 3.5: A Markov Blanket of a *thing* contained in the grey dotted line. The sensory (s) and active states (a) are highlighted with green and blue colours while internal states (μ) are shown in red and external states (η) are in grey.

Having established the definition of being a *thing*, FEP moves on to suggest that the internal states of any system that keeps its integrity on a non-equilibrium steady state (e.g. a brain) seem to minimize a free energy functional. Moreover,

FEP claims that minimization of free energy is equivalent to maximizing the model evidence of the environmental states the internal states encode [38]. Model evidence refers to the marginal likelihood of the external states given the values of blanket states. That is, the dynamics of the internal states at any time entail a probability density about the external states [40].

On the other hand, the probability density function generally does not represent the true distribution of the external states. Therefore, the system at the non-equilibrium steady state constantly ameliorates its beliefs. Note that the minimization of free energy goes hand to hand with enhancing the model accuracy (making better predictions about the environment).

So, there are two probability densities; the first one being the internal states' probabilistic representation of the external states $q_\mu(\eta)$ and the second is the real probability density of the external states given blanket states $p(\eta|s,a)$. Figure 3.6 portrays these probability density functions while depicting the Markov Blanket structure on a brain.

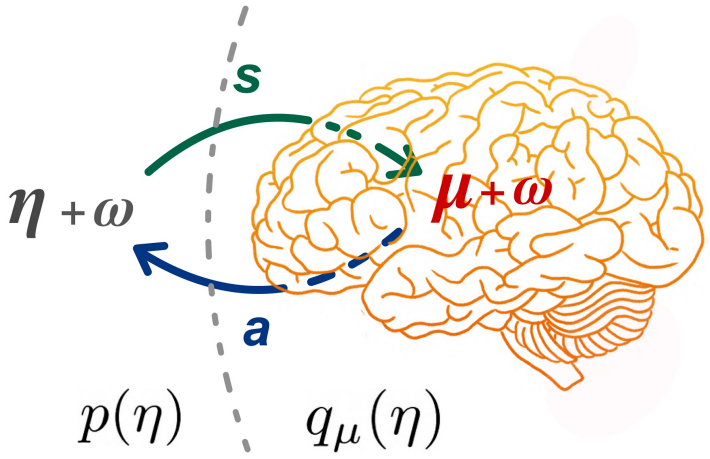

Figure 3.6: Showing the Markov Blanket of a brain where μ are internal states with random fluctuations ω only affecting the active states a, η are the external states with random fluctuations ω only influencing the sensory states s, $q_\mu(\eta)$ are internal beliefs about the external states and $p(\eta)$ is the true density of external states.

If both of the distributions are the same, the beliefs about the environment should be entirely correct. Thus, there should be nothing to minimize in the ideal case. Nevertheless, the external world can and will have random fluctuations. Hence, there is always going to be an open door for being *surprised*, even if you have the perfect model of the environment. Before writing down the equation of variational free energy, we must mention the Kullback-Leibler (KL) divergence between two probability density functions. KL divergence is a distance measure for probability density functions where $KL[p||q]$ is the expected difference between $\log(p)$ and $\log(q)$ weighted with p. In an information theoretical sense, this is the volume of information lost if q is utilised to approximate p [41]. Equation 3.12 displays the open form of KL divergence. Note that it can be shown that KL divergence is always non-negative and asymmetric. That is $KL[p||q] \geq 0, KL[p||q] \neq KL[q||p]$.

$$KL[p||q] = \int_{-\infty}^{\infty} p(x) \log(\frac{p(x)}{q(x)})\, dx \tag{3.12}$$

Now we can move on to present the formula for variational free energy in Equation 3.13 where $KL[q_\mu(\eta)\|\|p(\eta\|s,a)]$ denotes the KL divergence between the beliefs and the true distribution of the external states and $-\log p(s,a)$ denotes the surprisal of the blanket states.

$$F(\mu, s, a) = KL[q_\mu(\eta)||p(\eta|s,a)] - \log(p(s,a)) \tag{3.13}$$

Although the free energy equation can be put in various ways, Equation 3.13 is convenient for our work since we will employ surprisal minimization and it completely coincides with the Evidence Lower Bound (ELBO) quantity in the variational inference framework of Machine Learning. Indeed variational free energy is equivalent to negative evidence lower bound even though they are derived from completely different origins [42]. Equation 3.13 tells us that the difference between the agent's beliefs and the truth about the environment with the surprisal of sensory and active states determine how uncertain the agent is about the

environment. Also, as KL divergence is always non-negative free energy depicts an upper bound to surprisal. Equation 3.14 shows this relationship.

$$F(\mu, s, a) \geq -\log(p(s, a)) \tag{3.14}$$

Thus, if the agent minimizes the free energy, the surprisal will also be minimized. For example, one will have better expectations about what she will see when she turns her head to the right in a room she spent more time in. In other words, she will be less surprised about what she will see. This result is vital as we will show that the agents we train also have decreasing surprisals through generations.

3.6 NEAT

NEAT stands for NeuroEvolution of Augmenting Topologies. It is a widely used meta-heuristic evolutionary algorithm, it aims to evolve better architectures as well as better parameters for a neural network. The following parts shall explain the NEAT algorithm in detail as presented in [13]. Since the illustrations we use are Graph structures, the words node and neuron will be used interchangeably as well as the terms edge, connection and synapse.

While traditional evolutionary algorithms start with a predefined topology for neural networks, NEAT only determines the input and output layers of the neural networks at the beginning of evolution. This way, NEAT endeavours to solve the optimization task meanwhile keeping the topology as simple as possible.

While evolving neural networks, two candidate networks for mating may have the same functional characteristics but different permutations. Figure 3.7 shows an example of a situation where both of the networks give the same result because they have the same internal states. However, mating causes us to lose some of the information because the order of the genes is different. This is called the

competing conventions problem. NEAT proposes a rather simple solution for this, adding innovation numbers to each gene so that when two agents mate, the sorted order is used and no information is lost. For example, in the case shown in Figure 3.7 NEAT assigns innovation number $1, 2$ and 3 to genes A, B and C in order. Thus, when the cross-over happens the outcome is A, B, C as it should be.

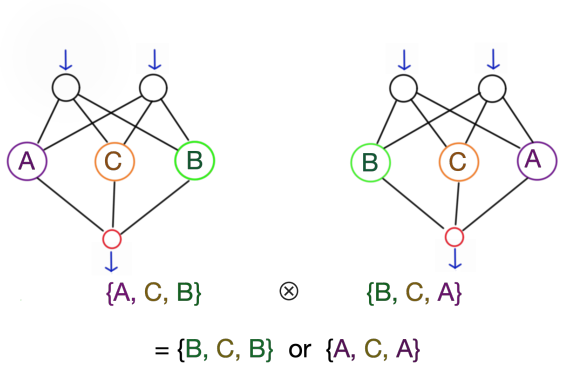

Figure 3.7: An example of competing conventions. Two networks work the same but their internal order is different, so when cross-over (shown with $\otimes$) is applied the possible offsprings lose a third of the information.

It is relatively more comfortable to mate two networks that have the same architecture or topology. However, when the first few generations are over, there will be agents with varying topologies. NEAT makes use of the same basic idea here, the innovation numbers. When the topologies are distinct, the genomes get merged with the help of innovation numbers. The example in Figure 3.8 displays two networks that have the same set of nodes but different connections. Where in the figure, the offspring gets node A from the first network with probability p_1 and from the second network with p_2. Then, the edges are conveyed with the same logic. However, if some genes (nodes or edges) are not common in parent networks, they are directly inherited by the offspring. For instance, edges $3, 4$ and 5 are not common in both networks.

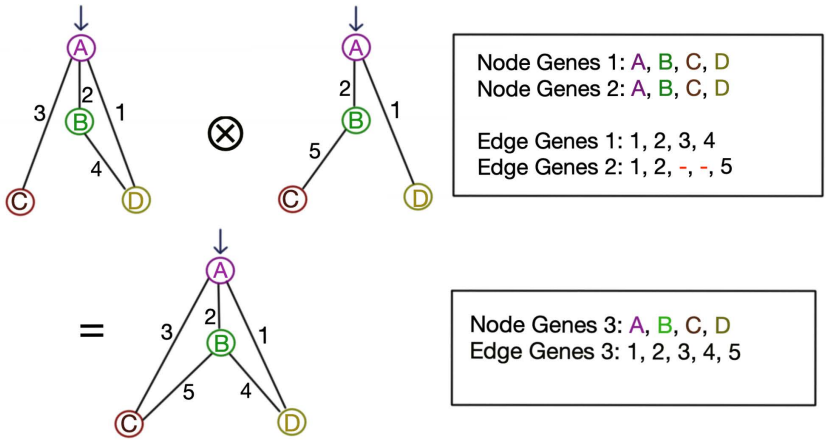

Figure 3.8: Cross over operation takes the list of nodes from both networks and merges them in sorted order with predetermined probabilities.

As the evolution progresses, the addition of neurons and edges will abound. Instead of creating a new (or possibly more) connection for the newly added neuron, to keep the behavioural properties of the network, NEAT places the new neuron on an existing edge. However, in order not to have multiple connections while adding a new neuron, NEAT disables the existing connection and creates two novel connections. Figure 3.9 displays a neuron addition operation. While disabling the existing connection, two new connections emerge. The first connection is from the presynaptic neuron to the new neuron and the second is from the new neuron to the postsynaptic neuron.

Also, since the behaviour of the networks is determined by their parameters, especially weights, the new weights become 1 and w_{ij} where w_{ij} is the weight of the disabled connection between presynaptic neuron i and postsynaptic neuron j. This helps keep the behaviour the same to some extent so that the new mutation does not get eliminated too fast.

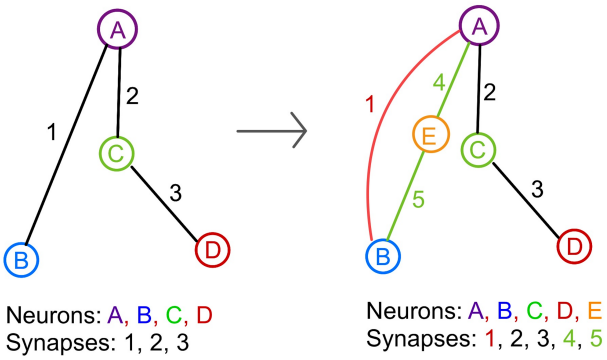

Figure 3.9: A new neuron, E is added between the neurons A and B. However, as there is already a connection between them, along with neuron E two new synapses, 4 and 5, emerge between A and E and E and B respectively. Of course, the synapse with innovation number 1 is disabled, shown in red on the right hand side.

On the other hand, when the mutations dictate an addition of connection to the network, a new synapse emerges between two existing neurons. The choice of the neurons is a design subtlety, one can pick the concerning nodes such that they do not have a direct connection at the moment or not. Figure 3.10 present a synapse addition operation where there was no existing connection between two neurons.

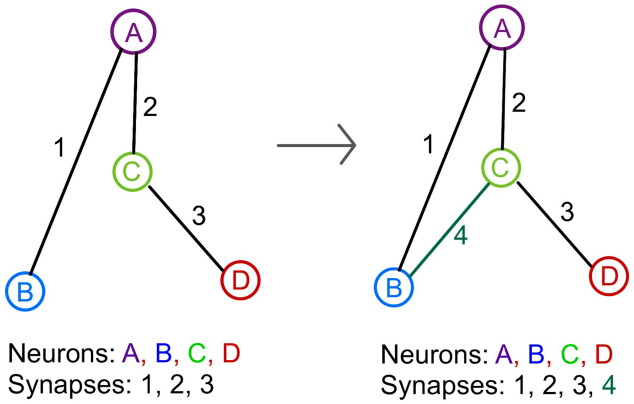

Figure 3.10: A new connection is added between neurons B and C and changed the genome accordingly.

The NEAT algorithm exploits one more idea from biological evolution, namely speciated evolution. That is, the agents in the population are distributed into groups that consist of agents similar to each other. In order to group the agents with their similarity, a metric should be designed but first, we need to define some gene types that shall prove useful such as disjoint, excess and common. A gene is disjoint, if not only it is present in only one of the parent genomes but also it has an innovation number within the genes that both of the parents have. Also, a gene is excess if it is present in one of the parent genomes but is not within the range of maximum innovation number of the other genome. For example, consider two parents, $P_1 = \{1, 2, 3, 4, -, 6, 7, 8\}$ and $P_2 = \{1, 2, -, -, 5, -, 7, 8, 9, 10\}$. The disjoint genes are $D_{P_1,P_2} = \{3, 4, 5, 6\}$ because all of them only belongs to either P_1 or P_2 (but not both) and they are smaller than or equal to 8 which is the largest innovation number both the parents has a gene with. While excess genes are $E_{P_1,P_2} = \{9, 10\}$ as they are greater than 8 and only one of the parents has them. Of course, the common genes are $C_{P_1,P_2} = \{1, 2, 7, 8\}$. Moreover, the common genes might have different values which should be accounted for while designing the metric. Thus, the distance between two genomes is determined with a metric that depends on the percentage of disjoint, excess and common genes. The Equation 3.15 presents a formula to calculate the distance between two genomes where c_1, c_2 and c_3 are arbitrary constants that determine the importance of disjoint, excess or common gene differences, N is the size of the union of both genomes and $\overline{W}$ is the average difference of common genes. For example, the values were $D = 4$, $E = 2$ and $N = 10$ in the example mentioned above while $\overline{W}$ is not obvious as we do not have the real values of common genes.

$$\delta(P_1, P_2) = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{3.15}$$

Speciation algorithm works iteratively, starting from a random agent, A_1, it immediately assigns a species to the agent S_1. After the species is assigned, the distance between A_1 and all of the remaining agents is calculated. If the distance is smaller than or equal to a threshold δ_T, the compared agent is also assigned with species S_1. After the first run, the process is repeated until there is no

remaining agent with an assigned species. The pseudo-code for speciation can be found below in Algorithm 1.

Algorithm 1 Speciate $P = \{A_1, A_2, A_3, A_4, .., A_n\}$

```
count ← 1
species ← 1
j ← 1
S(A_j) ← s_species
while count < n do
  if j == −1 then
    for i ← 1 to n do
      if S(A_i) == ∅ then
        j ← i
        species ← species + 1
        S(A_j) ← s_species
        break
      end if
    end for
  end if
  for i ← 1 to n do
    if S(A_i) = ∅ and δ(A_j, A_i) ≤ δ_T then
      S(A_i) ← s_species
      count ← count + 1
    end if
  end for
  j ← −1
end while
```

It is likely for a species to be better at the given task. In that case, if the fitness of the agents are directly compared, the species that is better will dominate the population in a couple of generations. However, an inferior species can evolve into a very successful species generations later. To protect the inferior species that might hold potential, NEAT applies a shared fitness calculation. Let's assume we have our agent A_i and it has a fitness f_{A_i}. The shared fitness is f_{A_i}/K where K is the size of species that A_i belongs to. Equation 3.16 presents the formal description of fitness sharing where f_i is the original fitness, n is population size and $\delta(A_i, A_j)$ is the difference between agent A_i and A_j.

$$
\begin{aligned}
f_i{}' &= \frac{f_i}{\sum_{j=1}^{n} \{\delta(A_i, A_j) \leq \delta_T\}} \\
\{\delta(A_i, A_j) \leq \delta_T\} &= 0 \text{ if } \delta(A_i, A_j) > \delta_T \\
\{\delta(A_i, A_j) \leq \delta_T\} &= 1 \text{ if } \delta(A_i, A_j) \leq \delta_T
\end{aligned}
\tag{3.16}
$$

This wraps up the preliminary theory and notation that will be used in the remaining parts of the thesis.

Chapter 4

Solving Nonlinear Problems with SNNs

In this chapter we will present systems, theories of which were given in Chapter 2, to evolve SNNs that can solve various problems such as XOR, pole balancing, food chasing and food chasing under shifting conditions. Firstly, we shall show the capabilities of SNNs trained with NEAT only. Then the networks that make use of external injection of dopamine will be exhibited.

4.1 Only NEAT

The NEAT algorithm is an efficient algorithm to evolve neural networks while keeping the topologies as simple as possible and preserving the distinction of species. In our case, the networks always have a constant number of input and output neurons in the first generation. Then, new neurons and synapses are added through mutations and the crossing over of agents.

4.1.1 XOR

The first problem we solve is the well-known logical binary exclusive or (XOR) operation. XOR takes two logical bits and returns *True* if and only if both of the inputs are different. XOR has been used as a benchmark for neural networks for multiple decades because of its nonlinear nature. Figure 4.1 shows this nonlinearity on a 2D cartesian plane. It is easy to show that there can never be a straight line that may divide the plane to classify outputs. Indeed, in 1969 Minsky and Papert proved that perceptrons cannot solve the XOR problem [43]. For comparison, logical AND and OR operations are given besides XOR in the same figure.

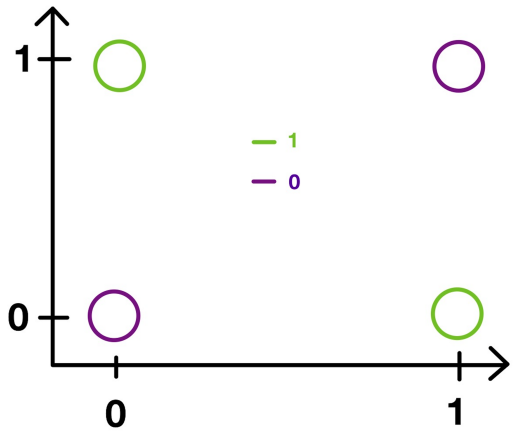

Figure 4.1: Binary XOR is a logical operator where the output of XOR operation is True if the inputs are different and the output is False if the inputs are identical. The phase space of XOR operator can be shown as above where there is no line that can separate the two output classes of XOR on the 2D space.

To solve XOR, we designed SNNs that have 10 sensory neurons for each of the input arguments. Our agents also have one output neuron that represents the output of the XOR operation. At first, there are some random connections between sensory and output neurons. Also, there is a neuron pool from where predefined neurons are chosen whenever a neuron addition mutation occurs. Figure 4.2 shows the initial architecture of our networks.

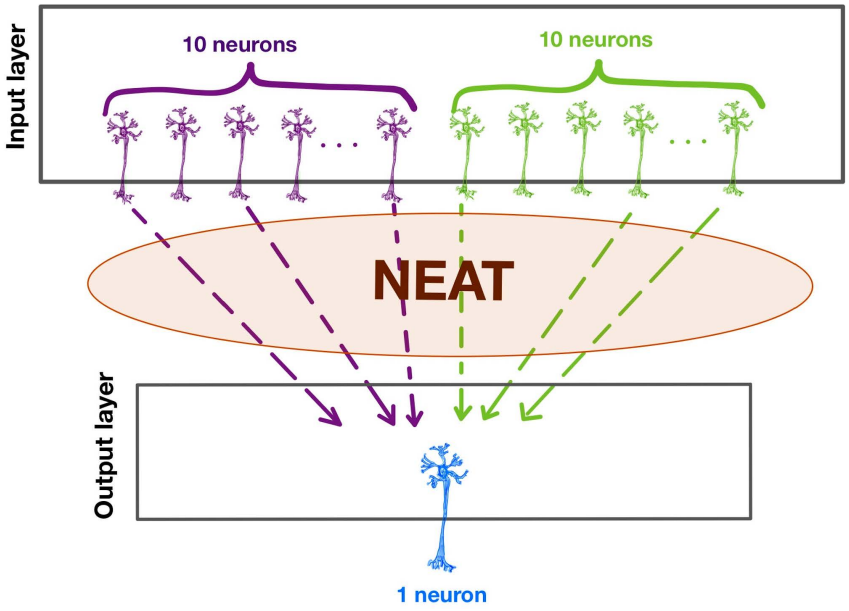

Figure 4.2: The design of an SNN initialized with random connections from input layer to output layer. The purple colored neurons represent the first input while the green ones represent the second input. There are only 21 neurons present at first. The hidden internal parts of the network is determined with NEAT.

The input is provided as a steady current to the sensory neurons. If the input is True, the neurons are stimulated with 8 mA current and if the input is False, the neurons are stimulated with 4 mA. The neurons are stimulated with respective inputs for 1000 iterations (0.5 seconds) and rested 4000 iterations (2 seconds) to get into refractive processes before the next trial. The output is True if the output neuron fires, and False otherwise. We showed that NEAT solves XOR fast. Within a couple of generations, it produces agents with %100 accuracies. Accuracy is the percentage of the correct outputs in 20 trials. Figure 4.3 shows ten trials of XOR operation tried on the best agent produced by NEAT, and the responses of the neurons as in action potentials. Note that, the figure only shows two input neurons one for each input, and the sole output neuron.

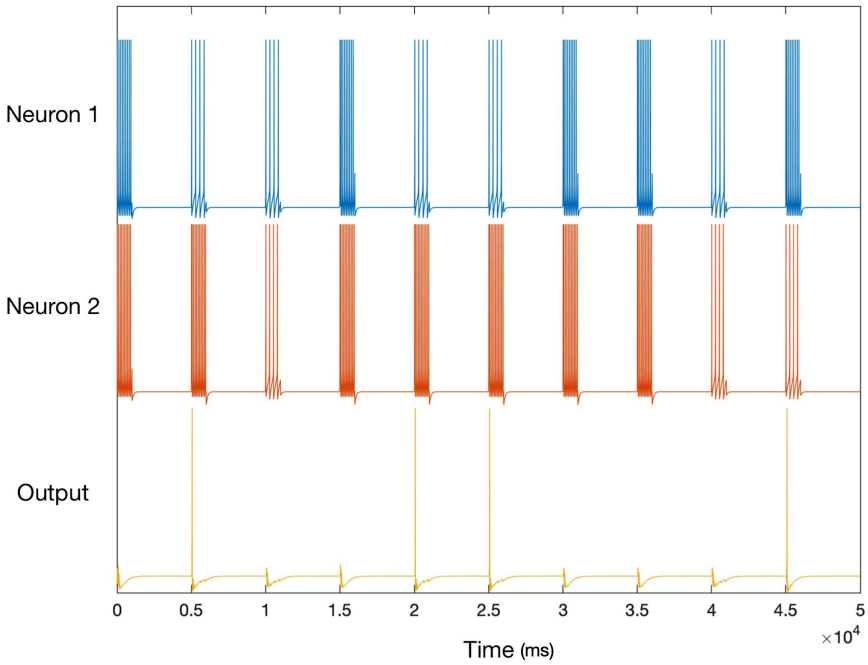

Figure 4.3: The neurons are stimulated with randomly chosen inputs from $\{0, 1\}$ and the result is read from the third neuron. Whenever the inputs are different, the output neuron fires and stays silent when the inputs are the same.

The evolution is initialized with 100 agents, 400 excitatory and 100 inhibitory neurons in the pool. 20 trials is applied to each agent and the accuracy is calculated with the percentage of the correct outcomes. Figure 4.4 displays the results in a chart. The average score almost monotonically increases and after only 15 generations we always have at least one agent with % 100 accuracy. In fact, there were over 60 agents with % 100 accuracy after 20th generations.

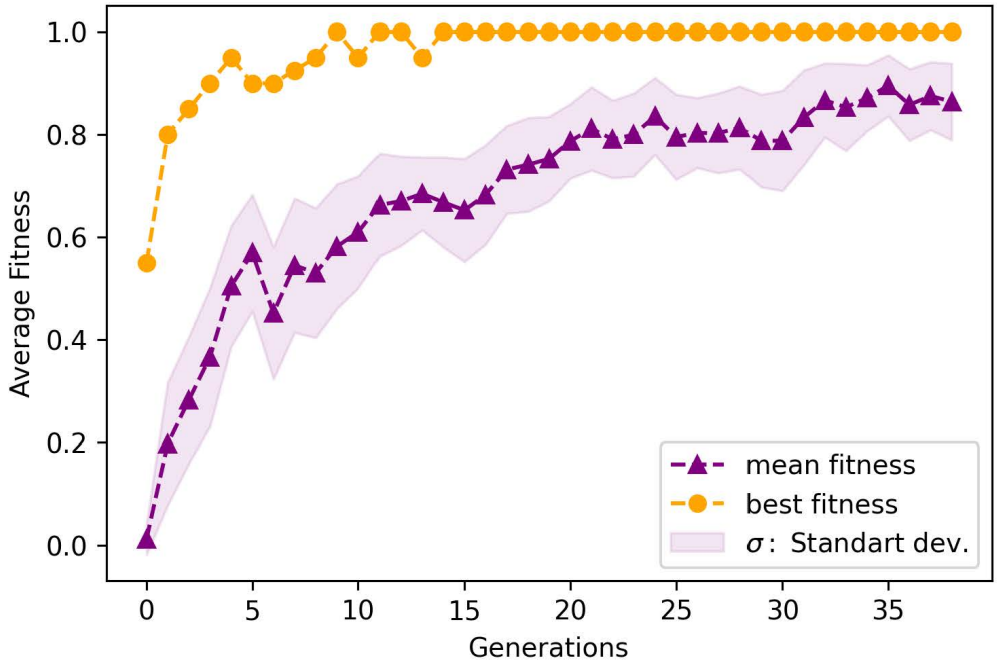

Figure 4.4: NEAT algorithm solves XOR problem within 15 generations. The figure shows the mean score of agents in each generation in purple and the best agent's score in orange. The purple shade is the standard deviation of agents' scores.

4.1.2 Pole Balancing

Having solved the XOR problem, the next test for our networks was Pole balancing. Pole balancing is a widely used benchmark problem especially famous in the Reinforcement Learning paradigm. No need to add that pole balancing is also a nonlinear problem and requires complex calculations if one tries to solve it analytically. For the curious, an analysis can be found in [44]. A pole is positioned on a cart that can move to either right or left. The goal is to keep the pole balanced as long as possible while applying force to the cart. Figure 4.5 displays a sample that has a cart and a pole positioned with an initial deviation from the balanced configuration.

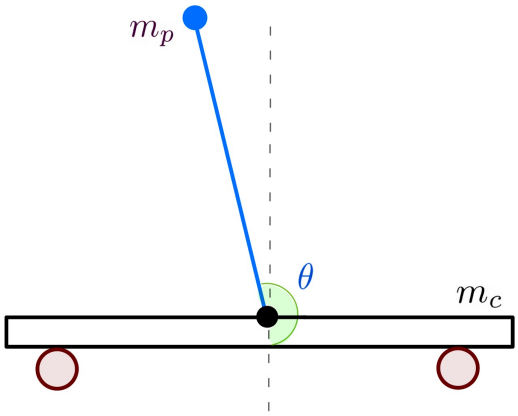

Figure 4.5: A pole with a mass of m_p on a cart with a mass m_c is deviated with an angle of $\delta = |\theta - \pi|$. The cart can be moved by force applied from either of its sides. The success of a position can be calculated by the smallness of this deviation, δ.

The value of a position can be assessed by the magnitude of the deviation from the middle point, which is the vertical axis that passes through the point the pole is attached to the cart. Equation 4.1 displays the formula we utilized to score any configuration at time step t between 0 and T, inclusive. And the overall value of a cart pole episode would be the average of the values at all steps. Other formulations are also possible where only the upper positions of the pole have positive values; meanwhile, bottom ones are considered to be 0 valued.

$$\begin{aligned} s(t) &= 1 - \frac{|\theta(t) - \pi|}{\pi} \\ V &= \frac{1}{T} \sum_{t=0}^{T} s(t) \end{aligned} \tag{4.1}$$

Our agents designed to solve pole balancing have six sensory neurons and four motor neurons. The first two sensory neurons represent the deviation from the right and left respectively. Their stimulation is provided as an injection current $I = I_b + c(\pi - \theta)/\pi$. Where I_b is the base current value for any of the sensory neurons and c is a constant that regularizes the importance of the deviation from

the fully balanced $\theta = \pi$. The value of c is positive if the pole is on the right-hand side and negative if it is on the left-hand side because the term $\pi - \theta$ will be negative if the pole is on the left.

The third and fourth sensory neurons represent the velocity of the cart at the time. If the velocity is positive (towards the right), the third neuron is injected with $I = I_b + cv_c$ while the fourth neuron merely gets $I = I_b$. The opposite happens if the velocity is negative, or towards the left side.

The fifth and sixth sensory neurons symbolize the angular velocity of the pole at the time. Following the same logic, if the pole has a positive angular velocity, the fifth neuron is injected with a current $I = I_b + c\omega_p$ while the sixth neuron gets the base value as an input. Meanwhile, if the pole has a negative angular velocity, the sixth neuron is stimulated with the higher current and the fifth neuron gets the base current input.

The four motor neurons collectively determine the force applied to the cart. The first and the third motor neurons contribute to the force applied towards the right meanwhile the second and the fourth neurons make up the force towards the left. Equation 4.2 displays the calculation of the applied force where α and β are constants that determine the importance of crude (motor neurons one and two) and sensitive (motor neurons three and four) components of the applied force. And m_i represents the number of action fires produced by ith motor neuron within the last 500 iterations (0.25 seconds).

$$F = \alpha(m_1 - m_2) + \beta(m_3 - m_4) \tag{4.2}$$

Figure 4.6 shows the sensory and motor neurons and their respective inputs. Just like the XOR example, the internal structure is determined by the NEAT algorithm. The cart pole system is simulated using Euler forward equation with $dt = 0.2$. In the mentioned figure, c_i are the coefficients of motor neuron firings, where $c_1 = -c_2 = \alpha$ and $c_3 = -c_4 = \beta$. $\alpha = 9$ and $\beta = 3$ are the values used in our simulations.

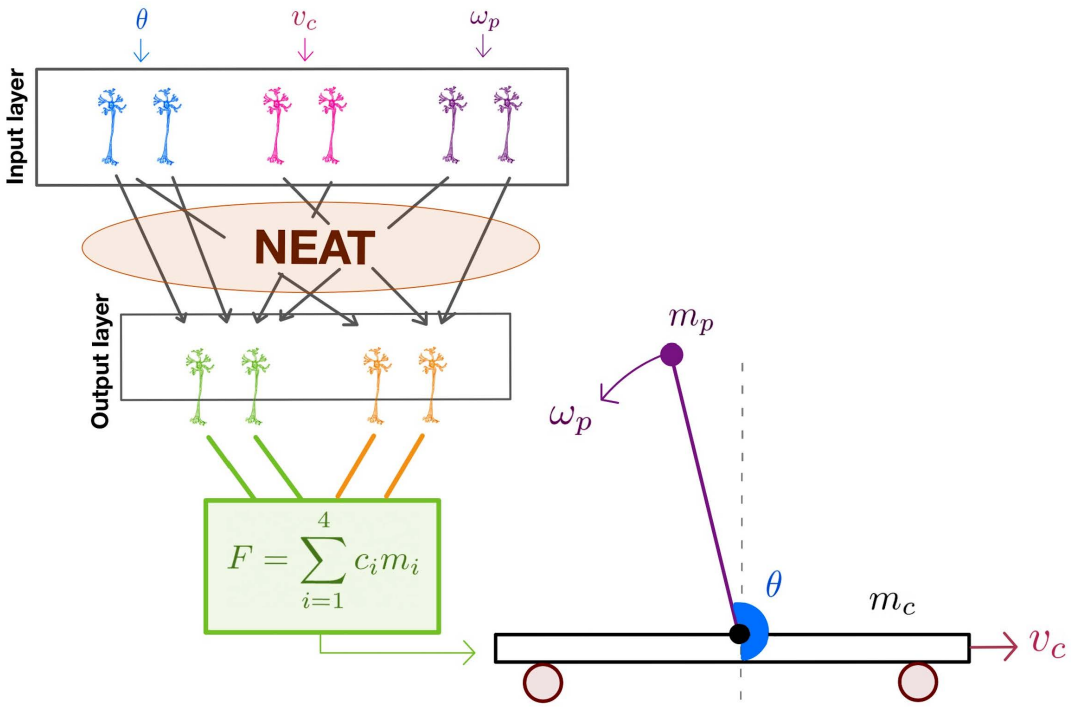

Figure 4.6: Initial architecture of the pole balancing network and the respective inputs and outputs taken from the cart pole system. Three main inputs are the deviation of the pole (θ), velocity of the cart (v_c) and the angular velocity of the pole (ω_p). Only output is the force that will be applied to the cart.

Our implementation of the NEAT algorithm has 100 SNNs in each generation which are simulated with different initial configurations of cart pole system. All of the simulations are run for 1200 time steps (300 seconds)each of which has 500 iterations (0.25 seconds) for the neurons. Each time step's first 0.1 seconds are allocated for the stimulation determined by the configuration of the cart pole system and the last 0.15 seconds are for the network to run and potentially rest to reach a decision that dictate the applied force.

We have solved the pole balancing problem within 15 generations. In fact, after five generations, in each population at least one of the networks reached to 0.99 score. Figure 4.7 depicts the evolution of the networks in 30 generations.

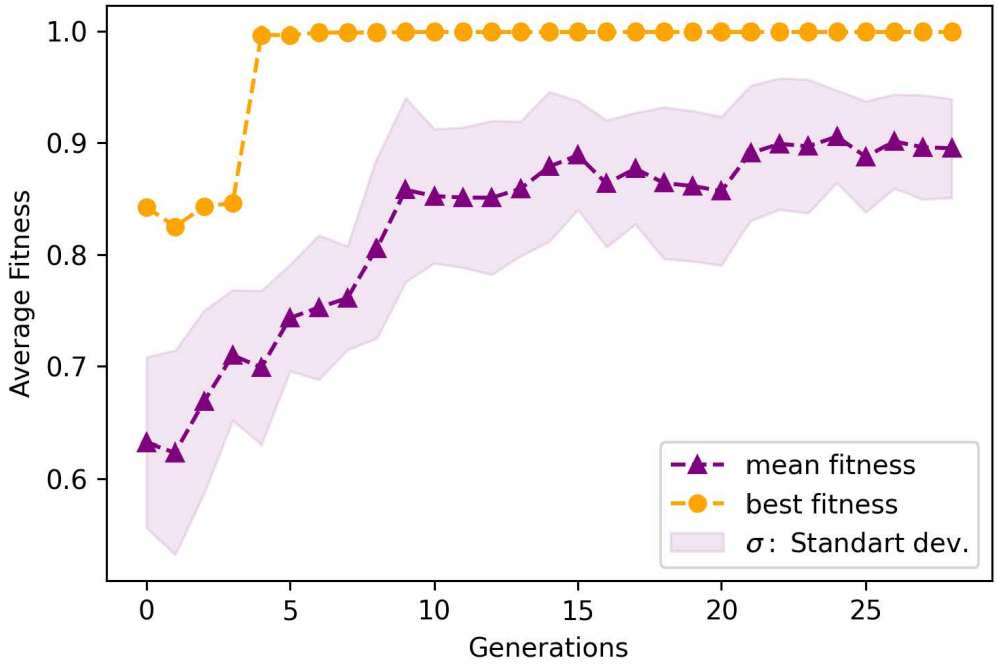

Figure 4.7: NEAT reaches 0.99 in the pole balancing problem within 15 generations.

4.1.3 Food Chasing

All the living somehow exploit their environments to supply energy ample enough to survive and pass genes. Of course, this requires learning the environment or at least the dynamics of the environment. We chose food chasing as our main task where an agent should find and get closer to the food source. The fitness or score of an agent is the number of foods it consumes. This problem is very convenient for our purposes because it is a game that provides the agent with an unlimited number of states and the agent can freely roam around the state space by just moving. Being a Reinforcement Learning friendly problem choice, we will show that SNNs will be solving this problem's variations with different methods such as dopamine modulation in later parts of this thesis.

A single food chasing episode is of a finite length of 1000 steps. The agents are in varying, empty 2D environments with accompanying foods and expected to

move towards them given their sensory inputs. Every agent is run in multiple 2D rooms in each generation to assess its fitness well. At the end of each generation, the best agents get chosen to mate and produce the offsprings.

Firstly, let's introduce the architecture of the networks as initialized. Our networks have two "eyes", each of which receives some projection of the food in the environment if present. Figure 4.8 shows a simple drawing of the seeing event including the network architecture. As always, the internal neurons will be placed by the NEAT algorithm.

Our agents have 60 retina cells, three motor unit cells, 30 sensory neurons that represent olfactory input, 30 neurons that represent hunger as if it is a sensation directly given from outside, and 30 neurons that represent the distance between the agent and the food at the moment. Thus, as a total, we have 150 sensory neurons and just three motor neurons that dictate the movement to right, left and/or forward directions. Each sensory cell gets stimulated for 100 iterations in each step, and the agent changes its position by firing the motor neurons.

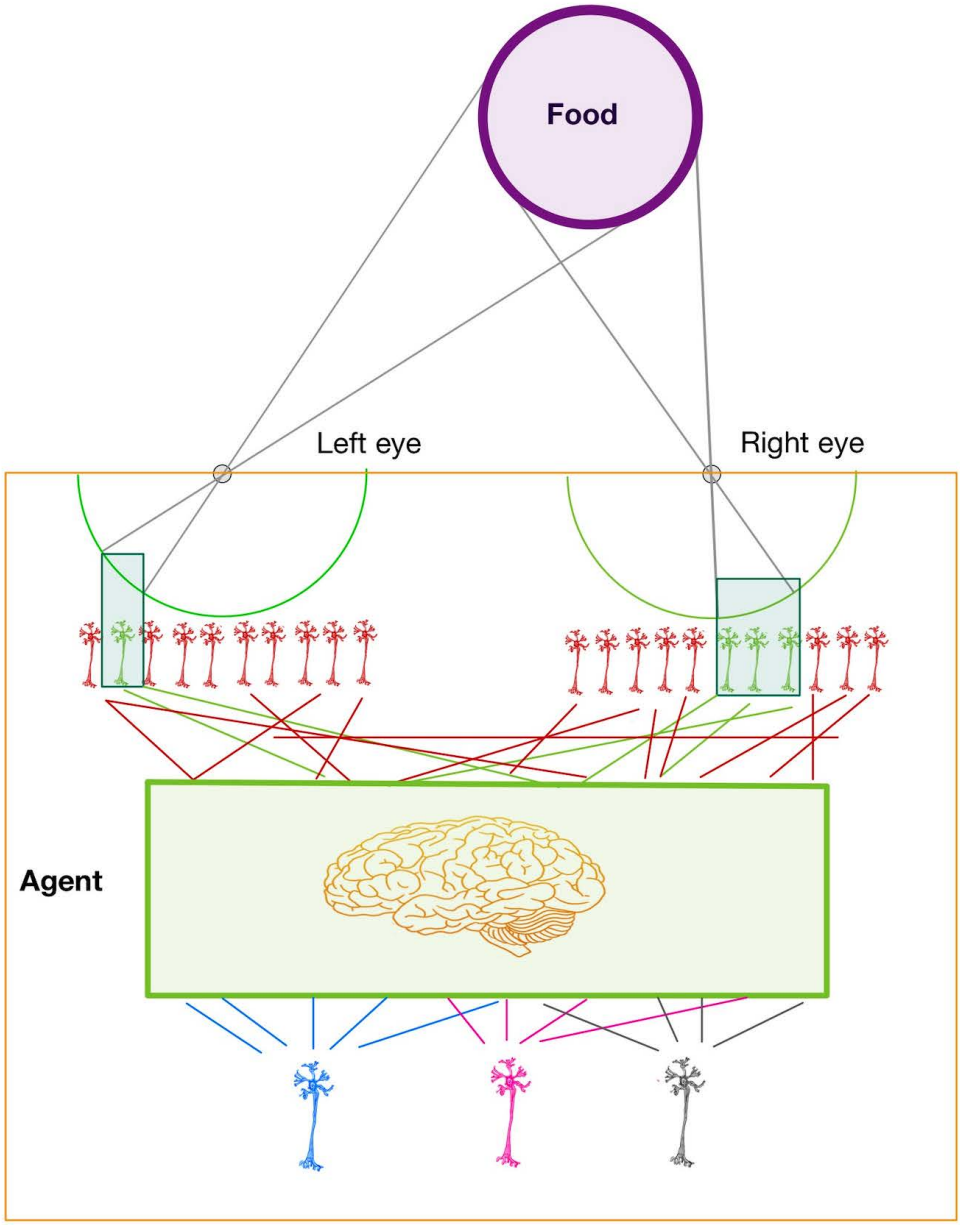

Figure 4.8: Architecture of the agents used in food chasing game with visual sensory units. The food's projection falls into the retinal cells and stimulates the respective fields. The stimulation causes firing of some neurons (green) more than the others (red). The sensory firings then stimulate the internal network, which will be determined by NEAT, and the output is given with three motor neurons shown in Blue, Magenta and Gray.

Using only the NEAT algorithm, we solved food chasing in only 30-35 generations. The agents fastly discover strategies that enhance their chances to consume more food and in turn better odds to pass their genes. For instance, when the food was unseen in the early generations, that is, it fell behind the eyes, the agents tended to fire right and left motor neurons repeatedly. However, trying to move left and right at the same time does not work well if you want to see and follow the food. This behaviour was dumped around the 29th generation which can be seen in the behaviour maps printed, also sensory surprisal makes a sudden jump at the same generation. Sensory surprisal of the population suddenly changes because behaviours determine the future sensory inputs. For instance, if you tend to turn left whenever you do not see the food it will be less surprising for you in the future because you will be even better at it. However, the moment you decide going right is the right way, it will be surprising for you because the food will get into your visual field from the right, not the left as it before did before. Figure 4.9 displays the evolutionary process of the agents.

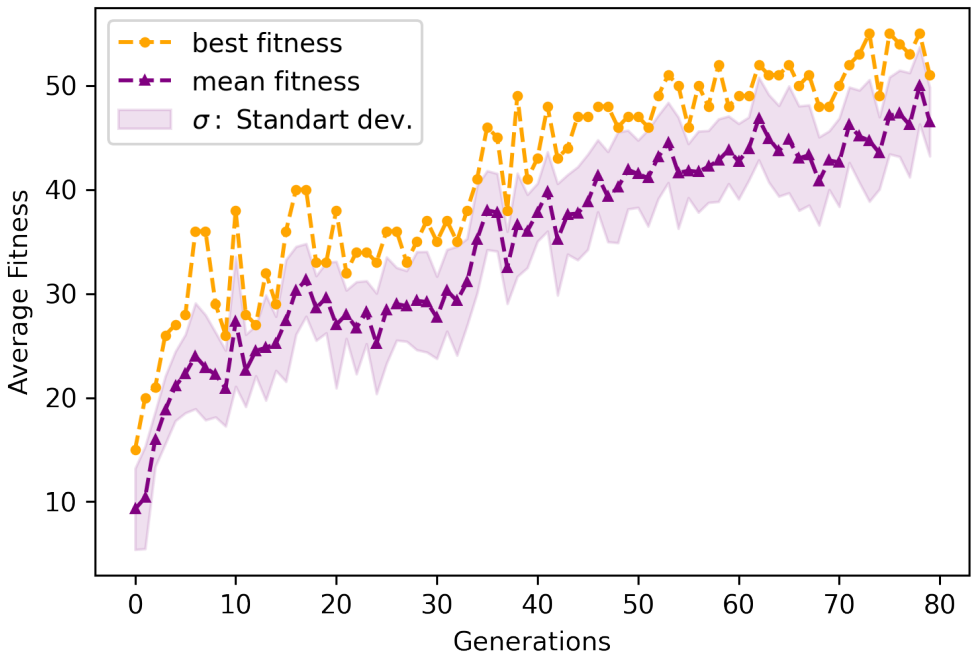

Figure 4.9: The average fitnesses of the agents increase in a consistent way. Fitness of an agent is the number of foods the agent consumes within its lifetime.

To see the behavioural evolution, we allotted lookup tables that keep the data of actions taken whenever a particular situation is encountered. For instance, the number of times the agent decides to move right when the food is on the left-hand side with more than 15 degrees deviation. The behavioural map accounts for four different situations and eight different actions. The four situations are the food being directly in front of the agent, on the left, right and behind. As for the actions, we used binary encoding to represent different firing patterns of the motor neurons. To elaborate, if none of the motor neurons fires more than one spike then the encoding of the action is $0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 = 0$. Or, if the forward motor neuron fires more than one spike but the others do not, the encoding becomes $0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 = 2$. This encoding scheme helps us measure 32 state-action pairs the agent can be found in. Also, we can calculate the entropy and the surprisal of the states which informs us about the randomness or precision of the actions taken. Figure 4.10 shows the average behaviour maps of populations on generations 1, 28, 29 and 80. There is a noticeable change between 28th and 29th generation where agents learn to turn right whenever the food is on the right. More importantly, they learn to turn right and move forward if the food is not visible. To see the behavioural maps of all generations check Appendix B.

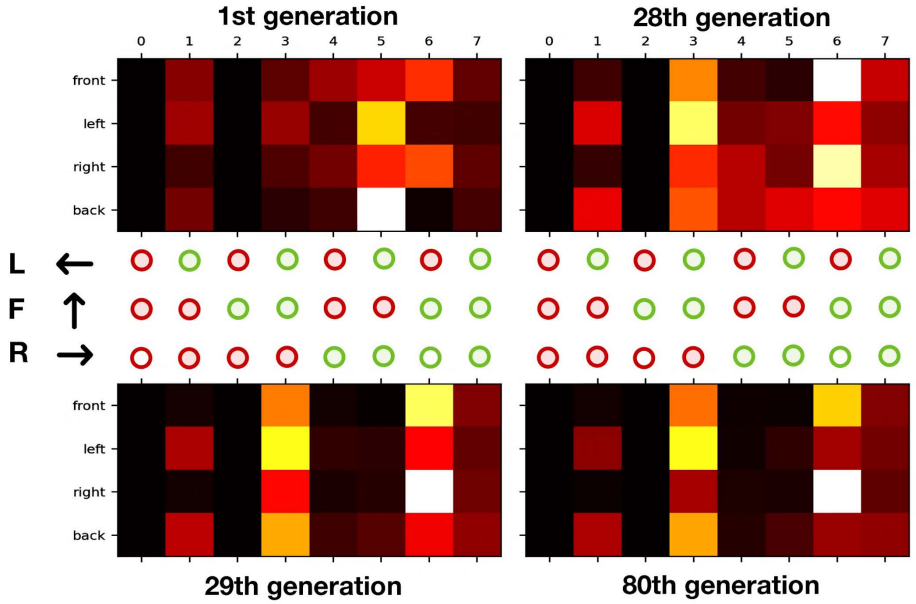

Figure 4.10: Behavioural maps of 1st, 28th, 29th and 80th generation as 4x8 matrices. Each row represents the situations where the food falls with respect to the agent while each column represents the response given by the agent as a motor activity. The eight responses are binary coded configurations each of which has three components: left, forward and right motor neurons' firings. Each cell represents a probability where the rows are normalized, i. e. sum of any row is equal to 1. The red and green circles represent whether left (L), right (R) and/or Forward (F) neurons fired when encountered with the situation on y axis. The cells are hot color coded, black is the lowest and white is the highest. Notice that the agents prefer to fire left and right when the food is on the back at 1st generation but this behaviour is almost totally abolished when we look at the 29th generation.

Although this behavioural change is not obvious in the scoring plot, after two generations (around 30th generation) there is a jump in mean scores. The reason for this is that the behavioural change firstly occurs in a few of the agents which get higher scores. Scoring high leads provides them with higher odds to have off springs and as a result, the next generations get more agents with the new behaviour. Moreover, this behaviour change, or rather strategy discovery, can be

seen in Figure 4.11 where the average number of time steps the food is visible or on the back are shown. Realize that although the possibility of the food being back decreases dramatically as the agents learn to position the food in front of them, there is a large deviation on the 28th generation where the probability of the food being on the back suddenly increases. The reason for this increase is the transition between the previous and the novel strategy the agents entail. This behavioural change coincides with the architecture change of the best agents between the 28th and the 29th generation. The architectures of all of the best agents are displayed in Appendix B Figure B.2.

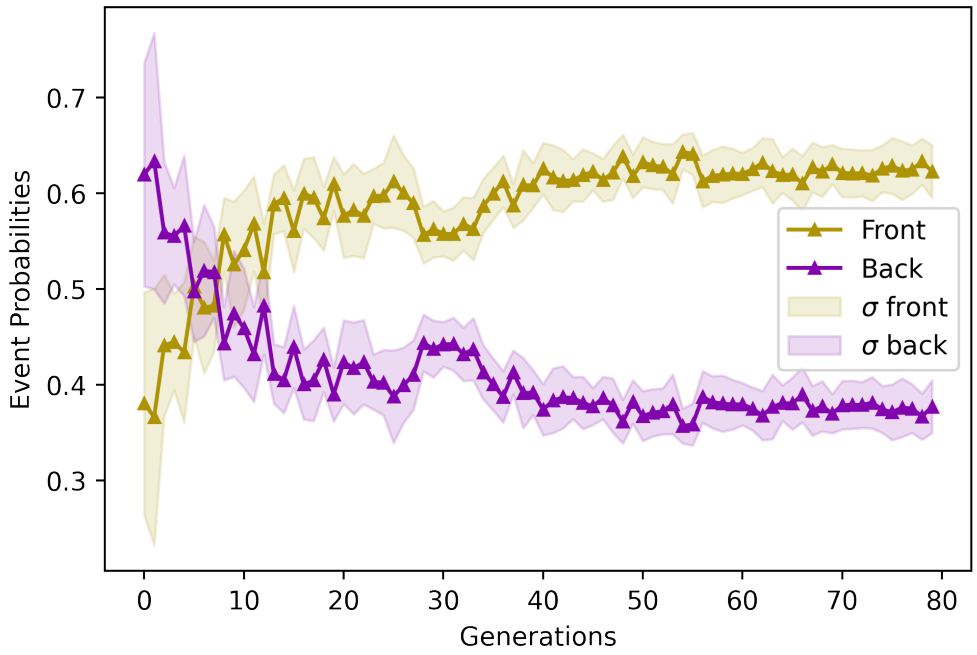

Figure 4.11: The agents learn to take the food in front of them after a while. Although a couple of fluctuations appear when the agents discover a new strategy, the decreasing regime continues.

Moreover, Figure 4.12 shows the decreasing regime of the sensory surprisal during the evolution. As mentioned before, this is due to agents getting better at determining where the food is and preferring certain configurations such as taking the food in front of them. However, the behavioural change also affects surprisal because a new strategy means a new set of configurations preferred. There is a

recent study showing similar results on a very simple Tetris playing agent. When the agent discovers new strategies it's sensory surprisal jumps [26].

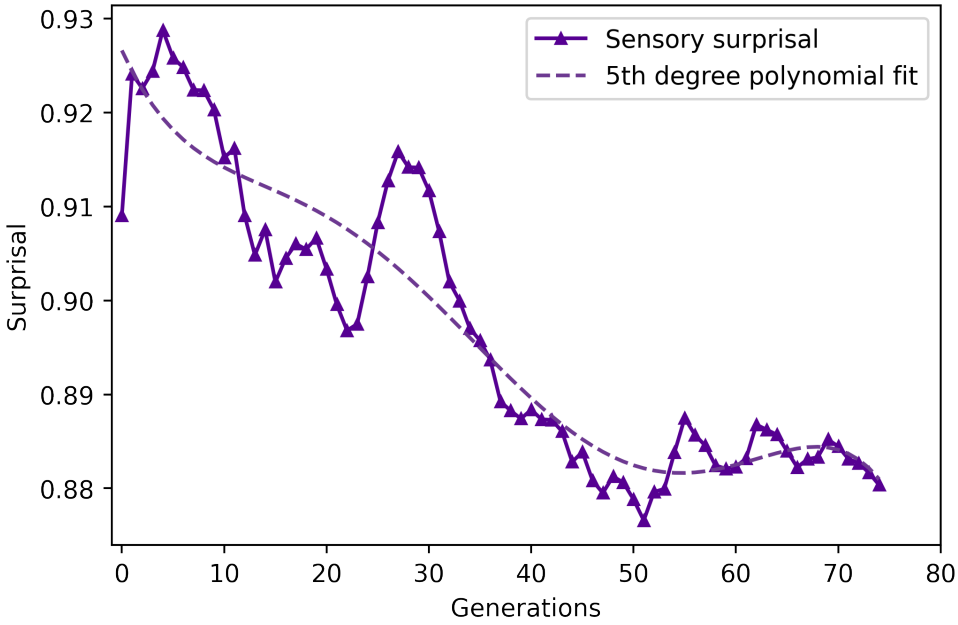

Figure 4.12: The average sensory surprisal of the agents decreases through generations. A 5th degree polynomical is fit to the original plot so that the decreasing is obvious. The fluctuation around the 28th generation is due to the discovery of the new behaviour.

Before wrapping up the section we must mention one more point. While training, in multiple versions of our genetic algorithms, agents tended to move drawing cycles. Even though we used different number of motor neurons and sensory neurons, this behaviour kept repeating. Finally, NEAT found the solutions where the agents directly move towards the food. Then as of late, we encountered studies which show that the cyclic movement is common because it allocates far more volume in the state space than the direct movement [45].

Having displayed the results of our food chasing agents, we are now confident that our training method does indeed help SNNs solve various nonlinear problems. However, until this point we have not used any plasticity algorithm such as STDP or a third factor modulator on STDP such as dopamine. In the following section,

not only shall we show that dopamine Modulated STDP can enhance our models to solve harder problems but also that without dopamine modulated STDP, the networks cannot solve tasks that require adaptive agency without sensory clues. In the next section we will solve another nonlinear problem with two variations in order to show that NEAT can solve problems when it is provided with sensory clues without explicit reward but cannot solve adaptivity requiring tasks without in-life adaptive capabilities i.e. dopamine modulated STDP.

4.2 NEAT with Latent and Injected Reward

As mentioned at the end of the previous section, genetic algorithms cannot solve problems where agents need to adapt to randomly chosen rules unless they are given as a signal during lifetime. By its nature, genetic algorithms assume an invariant set of rules. That is, the dynamics of the environment hardly change let alone changing dramatically. For example, our food chasing environment had invariant dynamics with an infinite state space. However, the distributions parameterized by the weights, delays and other variables of the SNNs were able to approximate a local minimum/maximum in the state space. It was possible because the algorithm worked by iteratively choosing the best agents concerning a well-defined fitness function, and by passing the genes (weights, delays etc.) which narrowed down the search space. Additionally, mutations helped the distributions to cover a larger volume of the search space because otherwise, the algorithm becomes analogue to a scarce greedy search which is known to be inferior to other methods such as Monte Carlo, Dynamic Programming and TD(λ) in most of the settings in Reinforcement Learning [46].

Now, we can formulate what genetic algorithms for SNNs do in a general sense as in Equation 4.3. The equation basically tells us that we choose the best parameters for the current task concerning a fitness function, $\mathcal{R}$. Considering every trial/episode takes k iterations, we sum up the scores for each iteration and take the average at the end of the trial. However, it is crucial to note that the fitness can be assessed directly at the end of the trial in some variations as well.

$p(\mathcal{A}|E_i, w_t)$ is the probability density or mass function of choosing actions from the set of all possible actions $\mathcal{A}$ given the current state of the environment E_i and parameters w_t. However, $\mathcal{A}_i$ is the action chosen on the ith iteration within the generation. Ω represents the cross over operation. Note that the update of the environment depends on the action taken by the agent and the current state itself under a dynamical rule γ in each iteration within the trial. Thus, the environment constantly gets updated but the parameters get updated at the end of the generation. Finally, some random fluctuation is present, shown with r for both updates. The $argmax$ comes from the fact that we choose the best agents for the given task, which is the definition of $argmax$ in the sampling sense. That is, if we apply infinitely many trials, our agents will have almost definite scores/fitnesses for a given task and choosing the best ones will be totally equal to $argmax$ operation.

$$\begin{aligned} w_{t+1} &\leftarrow \Omega(\underset{w_t}{\operatorname{argmax}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{R}(p(\mathcal{A}|E_i, w_t))) + r \\ E_{i+1} &\leftarrow \gamma(E_i, \mathcal{A}_i) + r \end{aligned} \tag{4.3}$$

It would be convenient to think of $\mathcal{R}$ as the expected reward the agent can get under a policy defined by the current values of the parameters w_t. Some remarks should be made before going any further. The first is that the equation describes a stochastic approach where the agents can choose between tasks while our agents are mostly deterministic. However, the above formulation works either way because the deterministic case can be considered stochastic with collapsed probability densities. Thus, the equation is just a more general form. Secondly, one should pay attention that $\mathcal{R}$'s value at any iteration does not depend on the action taken. It depends on the probability density function of all the actions that can be possible -hence policy. This idea is analogous to the expected future reward calculation in reinforcement learning, the Good Regulator Theorem and the Free Energy Principle. Latter two of which claim that any agent should have a model of its environment [38], [47]. Even more importantly, as our systems are deterministic within a lifetime, the set of all possible actions given the environment and the current values of the parameters consists of just one action in each

step.

On the other hand, Equation 4.3 has an obvious setback. The parameters w_t are never updated during the trials, which means if w_t is not general enough or if the environment gets a rule change either now or in the next trial/episode, the agent will fail. For example, an agent trained with only Earth terrain may have a hard time managing underwater or on the Moon's surface. The solution is to update the parameters both at the end of the trials and during the episodes. For example, allowing the agent to grow a caudal fin or providing it with online learning, regarding the above example. To do that, we attached dopamine modulated STDP to our agents and made the food chasing a bit harder. Before getting into the task, details and the results we should check Equation 4.4.

$$\begin{aligned} w_t(i+1) &\leftarrow \underset{w_t(i)}{\text{STDP}}[\mathcal{R}(p(\mathcal{A}|E_i, w_t(i)))] + r \\ E_{i+1} &\leftarrow \gamma(E_i, \mathcal{A}_i) + r \\ w_{t+1}(0) &\leftarrow \Omega(\underset{w_t(k)}{\text{argmax}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{R}(p(\mathcal{A}|E_i, w_t(i)))) + r \end{aligned} \tag{4.4}$$

Equation 4.4 introduces only one more update component, which is where the parameters (specifically the weights) are updated after each iteration. In our case, the update is ruled under the STDP algorithm modulated by dopamine described in Chapter 2.

To sum up, updating the parameters within the trials allows the agents to update their internal models of the environment. The anchor idea here is to make use of the generalizing properties of the STDP and reinforcing properties of the dopamine modulation. STDP is proved to be inherently applying Expectation Maximization for its inputs [21]; meanwhile, we mentioned at least two sources that show dopamine modulation on top of STDP is a computational framework for reinforcement learning such as [24] and [23].

Now, to see whether NEAT can produce agents that can tackle adaptation requiring tasks with only latent sourced sensory rewards, we solved the following

task using a population of agents without STDP and another population with dopamine modulated STDP separately.

4.2.1 Matrix Food Chasing with Sensory Reward

The new problem we need our agents to solve is to chase food and consume it again. However, this time there are always two choices to follow. There are one red and one blue pill, hence the name "Matrix Food Chasing". Two classes of agents are initialized. The first are the SNNs without STDP and the second; SNNs with STDP.

The agent needs to learn to find the pills, choose the correct one and consume it to get a reward and higher fitness at the end of the episode. The hardness of this problem comes from the changing state of the pills. In an episode, one of the pills is chosen to be the good pill totally randomly and uniformly. The agents are not provided with this information, i.e. they do not know which one is the good pill and which is not. But they get sensory stimuli regarding whether the pill was the good one or not. The sensory neurons that are stimulated secrete positive or negative dopamine respective to the pill as well. However, dopamine cannot affect the internal structure of the first class of the agents that are evolved with NEAT only; meanwhile, the second class of agents can change their internal structures with the secreted dopamine. Regardless, both of these classes of agents need to explore the choices and supposedly start to follow the good pill repeatedly.

In order to solve Matrix Food Chasing, we designed SNNs similar to the ones we used for Food Chasing. Our agents in both of the classes have 120 sensory neurons 80 of which are responsible for seeing the pills and distinguishing the colors of them. 20 sensory neurons are allocated for dopamine modulation where they are stimulated whenever a pill is consumed. If the pill is the good pill some subset of these cells are stimulated which secrete positive dopamine. If the pill is not the good one, the subset responsible for secreting negative dopamine is stimulated. Note that, in the original model of Izhikevich, there was no negative values allowed for dopamine [23]. Lastly, the last 20 sensory neurons represent

hunger which get stimulated with an increasing current until the agent consumes a pill. The idea of hunger is not to let the SNN get into a sub-critical regime which generally lead to death of the network -no spiking activity.

In the second class of agents, so as to keep the integrity of the genetically determined circuitry, we randomly choose some of the internal connections to be unchangeable by STDP. The reason for that is to have an underlying behavioural basis for the agents to intuitively follow the food(s). Meanwhile, the connections that can be updated via STDP + dopamine are to allow the agents to adapt to the environmental rules such as determining which pill is rewarding. Figure 4.13 sketches the common structure of the agents' architecture in both classes in the first generation and the event of seeing. The hunger input is not drawn for simplicity.

We initialized 100 SNNs in the initial generation of both classes, each of which is run for 2000 iterations (1 second) for 56 episodes in 28 of which the red pill is the good pill and in the other 28, the blue pill is the good pill. The agents are simulated for 100 generations. At first, in the second class, all of the weights are plastic, that is dopamine modulated STDP can be applied on them to update their values but in later generations, the connections can get mutated to non-plastic also new connections can be added by NEAT.

The agents learn to catch pills swiftly like the previous food chasing agents. As we need the agents to consume pills but determine the good pill as well, the scoring has a slight difference. We count the number of times the good pill is captured and subtract the number of times the bad pill is captured from it when the Blue pill is the good one. We repeat the same calculation for the episode where the Red pill is the good pill. Considering results are f_1 and f_2 in sorted order, Equation 4.5 shows the formula to calculate the overall score. The reason f_1 is squared is that we do not want to favor the agents that favor one particular color of food.

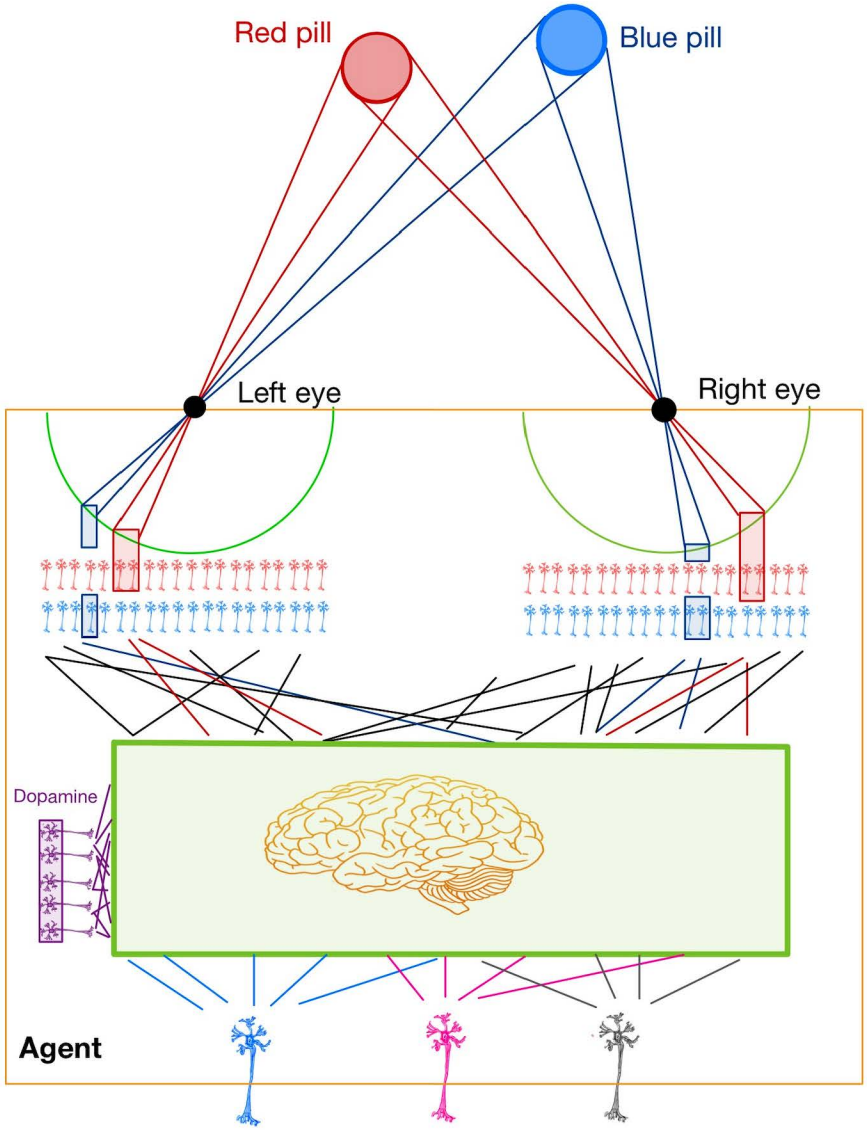

Figure 4.13: An agent in its initial form, has two eyes and 40 retina cells each. In each of the eyes, there are two layers of sensory neurons responsible for capturing red and blue input signals shown in Red and Blue. Another difference from classical food chasing networks we designed, there is another sensory layer responsible for reward inputs. This layer consists of dopaminergic neurons and when they fire dopamine is secreted through them to the internal neurons. Just like the previous version there are three motor neurons responsible for moving to right, forward and left.

$$S = (f_1{}^2 f_2)^{1/3} \tag{4.5}$$

Figure 4.14 shows the results of both classes of networks. Unsurprisingly both of the classes of agents successfully solve the task though STDP agents are advantageus to their no STDP counterparts.

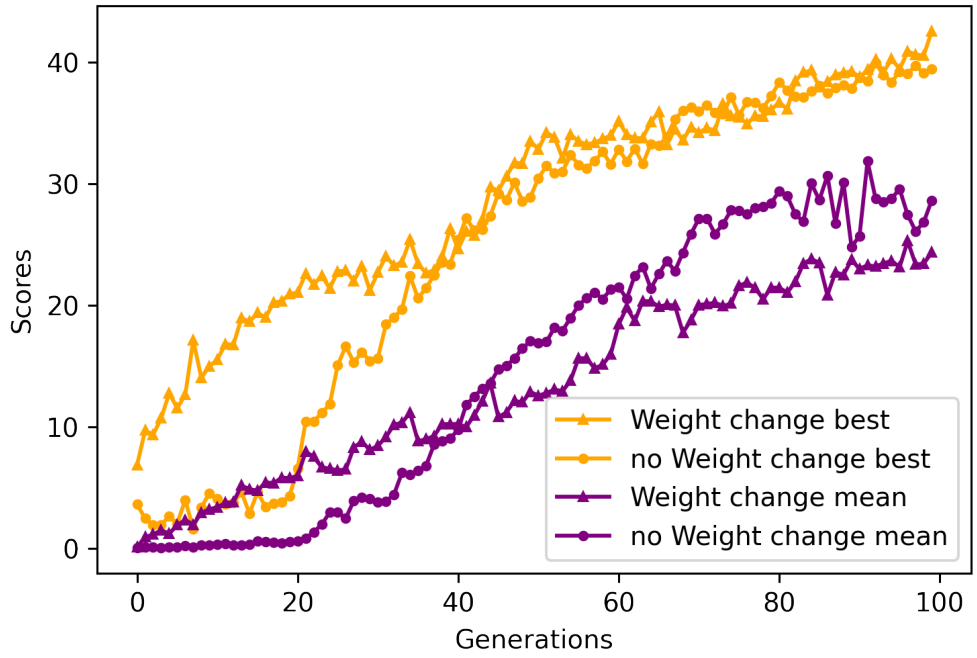

Figure 4.14: Simulation results of two populations, the first contains agents without STDP and the second with STDP. Both of the populations solve the matrix chase problem where the agents are supposed to catch the correct pill out of two pills. The first population is given with Circles and the second class is in Triangles.

However, some observations can be made here such as the fact that the plot of the second class, NEAT with STDP, falls into local minima more often (i.e. generation 1oth, 24th and 35th) meanwhile NEAT with STDP fluctuates more often. This is most likely caused by the changing weights of the agents in the first class where we have STDP since the agents can detect which one is the good pill during the episode, they can perform well and even in the early generations.

Nevertheless, a more important result of this experiment is that the agents in the first class with no STDP can learn this task without being told which pill is good. The reason for this is that the NEAT algorithm determines the fact that some of the sensory stimuli are indeed correlating with the fitness of the agents at the end of the generation. Thus, NEAT starts to favor agents that get the particular kind of stimuli which in turn lead to agents that follow the sensation that is correlated with dopamine secretion in their second class counterparts.

We have shown that both NEAT without STDP and with dopamine modulated STDP can solve a task where the reward is given as a latent sensory stimulus. Both of the algorithms can detect the rewarding behaviour and improve through generations. Of course, one should not forget that agents with dopamine modulated STDP improve their behaviour during episodes as well. Although agents that have no STDP seem inferior to the ones with STDP, their scores still increase and they learn the task as successfully as others.

4.2.2 Matrix Food Chasing with Injected Reward

Now, we can move on to the last part of this chapter where we simulate two populations again. Nevertheless, this time dopamine will not be given by sensory stimuli but rather will be given via direct external injection. The difference between the two tasks is that, if run correctly like the one in the previous section, a genetic algorithm such as NEAT can capture the meaning of the sensory inputs whether they cause direct update of the internal weights or not, whereas it cannot capture the meaning of injected stimuli as it is completely blind to in-life rewards.

NEAT algorithm or any other genetic algorithm alone cannot solve this problem because they do not allow the agents to change their internal structures during their lifetime. In turn, the policy their parameters entail does not update itself. So, in a sense, an agent either chooses to follow the red pill or the blue pill intrinsically. When the blue pill is the "good" pill, the agents that choose the blue pill will get rewarded; however, if they get simulated in the environments where the red pill is the good one, their performances will suffer. As the rewards earned

by consuming the blue and the red pill are opposite (1 and -1) in our problem setting, the expected value of the accuracy score of an agent trained with only NEAT will be close 0.5. Indeed, the expected score will get arbitrarily close to 0.5 as the number of trials increases.

To be clearer, the agents in the first class have connections that can be modified through dopamine modulated STDP. The dopamine is be provided whenever a good or bad pill is caught by the agent. The value of the dopamine is either negative or positive depending on the pill's type. Whereas, the agents on the second class have no connections that can be modified. Thus, the injected dopamine has no affect on their internal configurations. The only difference between this problem and Matrix Food Chasing with Sensory Reward that we solved above is the manner dopamine is provided. Impressively, even without STDP, when dopamine is provided as sensory stimuli, in spite of the fact that the connections do not get affected, the agents still can learn to catch the good pills. We can consider this scenario with the following analogy. Imagine a population of animals that have no in-life learning ability, and you want to teach them to choose between two pills. When they choose the good pill you show them a particular scenery image. On the other hand, if they choose the bad pill you show them a particular art work. The ones that choose the good pill more get a right to reproduce. The evolutionary process will teach them that the scenery image is a good sign. Intrinsically they will want to choose the pill which after consuming they see the scenery image. This exactly what happens in the previous problem. Now, however, the animals does not even see a scenery or an art work. They are just eliminated by random because there is no indication of a pill being good or bad.

In the simulations, in each of the populations, we have 100 agents that are run for 2 trials/episodes in each generation where every episode lasts for 1000 steps. The simulation is run for 100 generations. Figure 4.15 shows the results where the second population does not learn the task. Meanwhile, the agents that have STDP learn the task and continue to improve.

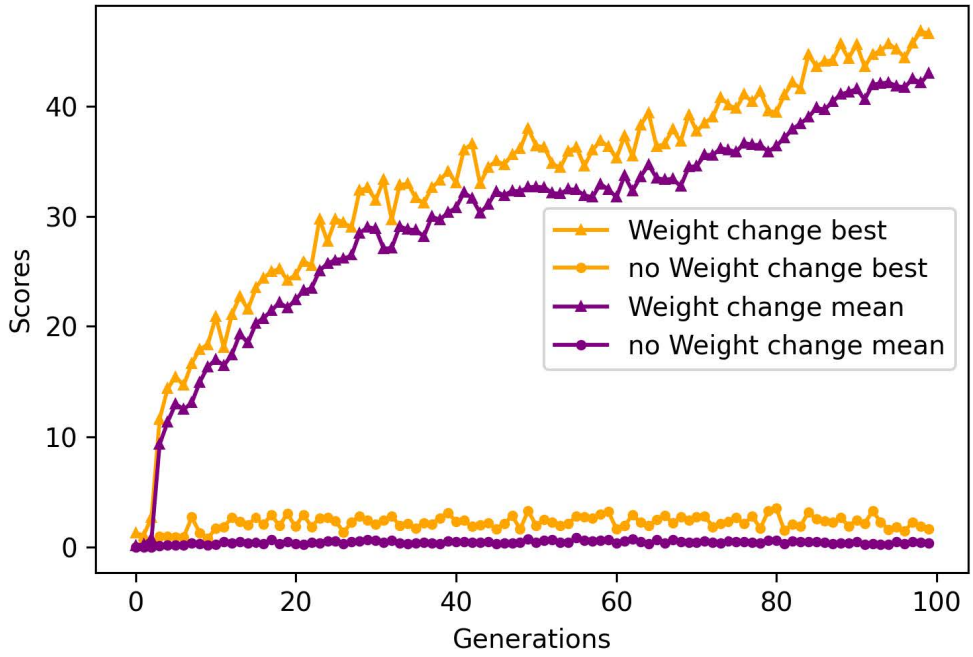

Figure 4.15: Direct external dopamine injection results. The agents with STDP learn the task where agents with no STDP show no improvement from the start.

The reason agents with STDP perform a bit better than their previous versions with sensory stimulated dopamine is that their dopamine injection is direct. That is, in the previous version where we used sensory stimulation of dopaminergic neurons, NEAT had to optimize the weights related to sensory dopaminergic neurons as well. Meanwhile, in this case, dopamine is directly given without any need to be fine-tuned.

Also note that the number of sensory neurons in each of the described simulations were chosen to keep the balance between overall performance and available computational time we had.

This sums up the current chapter where we have solved nonlinear problems with varying difficulties. We have shown that NEAT is a powerful metaheuristic search algoritm and can solve problems even when the fitness is correlated with sensory clues of a randomly changing latent factor.

Chapter 5

Conclusion and Future Work

We have designed, implemented, trained and analyzed SNNs that can solve non-linear problems such as XOR, pole balancing, food chasing and matrix food chasing. Our research shows that NEAT algorithm is a powerful method to train SNNs even when the problem at hand does not explicitly provide sensory input that is correlated with the fitness function. The agents can learn to position themselves, control their motor behaviours and decrease their sensory surprisals. Meanwhile, dopamine modulated STDP offers a promising aspect of training for SNNs that allow them to enhance their abilities within their lifetimes. Our results show that dopamine modulated STDP and NEAT make a successful pair to train SNNs that can solve problems even when there is no sensory clue for the agents about the rewarding actions.

Our work sketches a promising framework for possible extensions in SNNs research. We aim to continue our studies on SNNs by tackling different problems such as attention and scene reconstruction.

Bibliography

[1] A. W. Senior *et al.*, "Improved protein structure prediction using potentials from deep learning," 2020.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.

[3] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE Transactions on Neural Networks*, vol. 15, pp. 1063–1070, Sept. 2004.

[4] K. Amunts *et al.*, "The human brain project—synergy between neuroscience, computing, informatics, and brain-inspired technologies," 2019.

[5] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," 2015.

[6] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," 2018.

[7] H. Jang and O. Simeone, "Multi-sample online learning for spiking neural networks based on generalized expectation maximization,"

[8] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida, "Deep learning in spiking neural networks," 2018.

[9] A. C. D. S. Paulino, J.-C. Nebel, and F. Flórez-Revuelta, "Evolutionary algorithm for dense pixel matching in presence of distortions," 2014.

[10] K.-C. Wong, K.-S. Leung, and M.-H. Wong, "Protein structure prediction on a lattice model via multimodal optimization techniques," 2010.

[11] A. George, B. B. Rajakumar, and D. Binu, "Genetic algorithm based airlines booking terminal open/close decision system," 2012.

[12] C. Notredama and D. G. Higgins, "Saga: sequence alignment by genetic algorithm," 1996.

[13] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *The MIT Press Journals*, 2002.

[14] S. Legg and M. Hutter, "Universal intelligence: A definition of machine intelligence," 2007.

[15] X. Jin *et al.*, "Modeling spiking neural networks on spinnaker," 2010.

[16] M. Rabinovich, K. Friston, and P. Varona, *Principles of Brain Dynamics: Global State Interactions*. MIT Press, 2012.

[17] G. L. Vazquez *et al.*, "Evolutionary spiking neural networks for solving supervised classification problems," 2019.

[18] N. G. Pavlidis, G. Nikiforidis, and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms," 2005.

[19] H. Qiu, M. Garratt, D. Howard, and S. Anavatti, "Evolving spiking neural networks for nonlinear control problems," 2019.

[20] T. Moraitis, A. Sebastion, and E. Eleftheriou, "Short-term synaptic plasticity optimally models continuous environments," 2020.

[21] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," 2013.

[22] A. Foncelle and A. Mendes, "Modulation of spike-timing dependent plasticity: Towards the inclusion of a third factor in computational models," 2018.

[23] E. M. Izhikevich, "Solving the distal reward problem through linkage of stdp and dopamine signaling," 2007.

[24] F. Zhao, Y. Zeng, and B. Xu, "A brain-inspired decision-making spiking neural network and its application in unmanned aerial vehicle,"

[25] R. V. Florian, "A reinforcement learning algorithm for spiking neural networks," 2005.

[26] P. T. Waade, C. L. Olesen, M. M. Ito, and C. Mathys, "Consciousness fluctuates with surprise: An empirical pre-study for the synthesis of the free energy principle and integrated information theory," 2021.

[27] E. M. Izhikevich, "Spike-timing dynamics of neuronal groups," *Cerebral Cortex*, vol. 14, pp. 933–944, Mar. 2004.

[28] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1569–1572, Nov. 2003.

[29] H. Markram, Y. Wang, and M. Tsodyks, "Differential signaling via the same axon of neocortical pyramidal neurons," *Proceedings of the National Academy of Sciences*, vol. 95, no. 9, p. 5323–5328, 1998.

[30] E. Kandel, S. Siegelbaum, and S. Mack, *Principles of Neural Science 5th Edition*. 2013.

[31] R. P. Costa, Y. M. Assael, B. Shillingford, N. de Freitas, and T. P. Vogels, "Cortical microcircuits as gated-recurrent neural networks," 2018.

[32] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *bioRxiv*, 2019.

[33] J. R. Hughes, "Post-tetanic potentiation," *Physiological Reviews*, vol. 38, pp. 91–113, Jan. 1958.

[34] D. O. Hebb, "Organization of behavior. new york: Wiley, 1949, pp. 335, $4.00," *Journal of Clinical Psychology*, vol. 6, pp. 307–307, July 1950.

[35] J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Scholarpedia*, vol. 5, no. 2, p. 1362, 2010.

[36] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," 1997.

[37] R. Sutton, "Temporal credit assignment problem in reinforcement learning," 1984.

[38] K. Friston, "A free energy principle for a particular physics," *arXiv e-prints*, p. arXiv:1906.10184, June 2019.

[39] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. 1988.

[40] K. Friston, "The free-energy principle: a unified brain theory?," 2010.

[41] K. P. Burnham and D. R. Anderson, "Practical use of the information-theoretic approach. model selection and inference," 1998.

[42] M. J. Beal, "Variational algorithms for approximate bayesian inference," 2003.

[43] M. Minsky and S. Papert, *Perceptrons*. MIT Press, 1969.

[44] R. Tedrake, *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)*. 2021.

[45] R. A. Watson, S. G. Ficici, and J. B. Pollack, "Embodied evolution: Distributing an evolutionary algorithmin a population of robots," *Robotics and Autonomous Systems*, vol. 39, 2002.

[46] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari, "Convergence results for single-step on-policy reinforcement-learning algorithms," 2000.

[47] R. C. Conant and W. R. Ashby, "Every good regulator of a system must be a model of that system," 1970.

Appendix A

Code and Reproducibility

All of the simulations and experiments in this thesis are implemented in MATLAB and Python 3.9. We mean to open source the codes on GitLab after we finish a related journal article that is in progress. However, until then, we will be glad to share the codes for the simulations and the plots upon request.

All of the figures are either drawn or digitally created using MATLAB and Python scripts implemented by Computational Modeling Group at Bilkent University, UNAM.

Appendix B

Evolving Networks

Figure B.1: The behavioural maps of all 80 generations for the food chasing experiments.

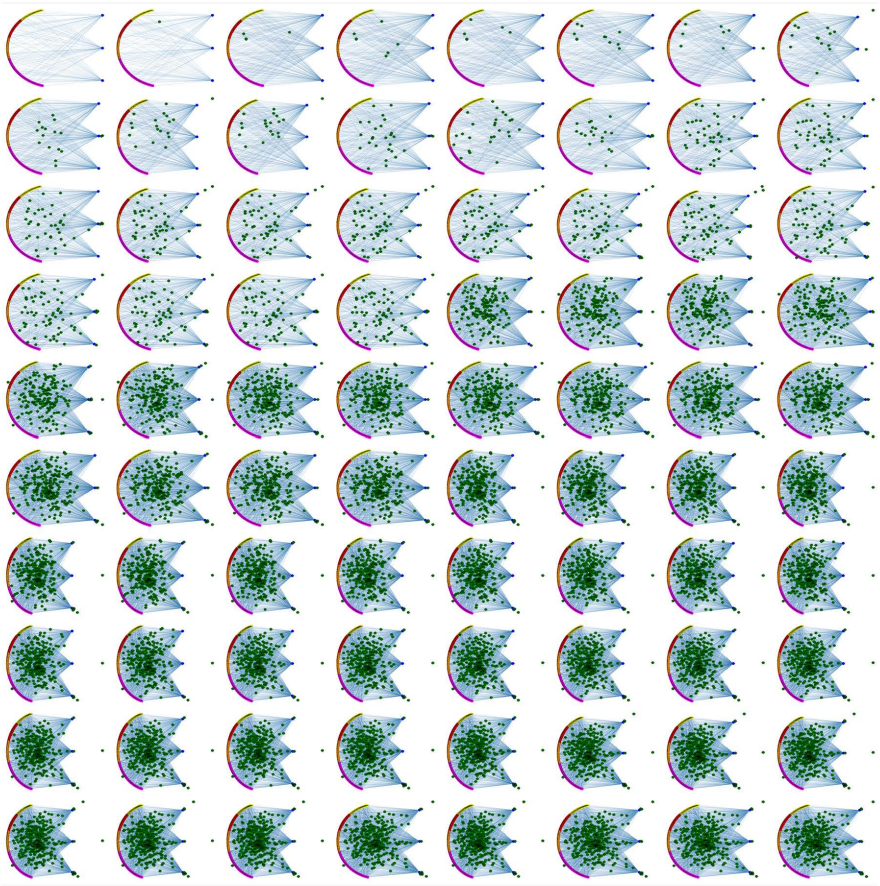

Figure B.2: The best agents of each generation throughout 80 generations for the Food chasing problem. The architecture of the best network considerably changes on the 29th generation which coincides with the behavioural change as well.