Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Sondre Haugen Elgaaen

Evolving Biologically Plausible Recurrent Neural Networks for Temporal Prediction

Master's thesis in Informatics
Supervisor: Zenas C. Chao
Co-supervisor: Keith L. Downing, Felix B. Kern, Shoji Takeuchi
November 2023

NTNU
Norwegian University of Science and Technology

東京大学
THE UNIVERSITY OF TOKYO

Sondre Haugen Elgaaen

Evolving Biologically Plausible Recurrent Neural Networks for Temporal Prediction

Master's thesis in Informatics
Supervisor: Zenas C. Chao
Co-supervisor: Keith L. Downing, Felix B. Kern, Shoji Takeuchi
November 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Sondre Haugen Elgaaen

Evolving Biologically Plausible Recurrent Neural Networks for Temporal Prediction

Master's thesis, Summer 2023

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Contents

List of Figures

List of Tables

Abstract

Inspired by research on temporal prediction in the human brain and a fascination with neuroevolutionary techniques, this thesis seeks to examine the potency of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm when tasked with evolving neural network structures that are minimalistic, biologically plausible and capable of predicting simple temporal sequences effectively. For this purpose NEAT's hyperparameters are tuned to evolve small Recurrent Neural Networks, which are combined with Hebbian learning. The networks are tested on a variant of the foreperiod task, dubbed the Ready-Go task, in which fitness is evaluated based on the ability to accurately predict the timing of a Go signal.

The thesis demonstrates that this approach makes it possible to evolve small biologically plausible networks that perform well on simple temporal tasks, but also identifies some inherent weaknesses in using NEAT for this purpose. Lastly, the thesis delves into analysis and interpretation of the networks, identifying key structures in the networks that facilitate temporal prediction. This analysis further reveals that the effect of Hebbian learning is inconsequential compared to the weights of the evolved networks, indicating that other neuroplastic processes may be better suited for the problem domain.

Preface

This master thesis was written in the summer of 2023 for the completion of the degree Master of Science in Informatics with a specialization in artificial intelligence at the Norwegian University of Science and Technology. The thesis was written during a one-semester exchange to the University of Tokyo, during which the research topic was decided. In this distractingly exciting and foreign country, this thesis was only made possible due to the academic, administrative and social aid of many people that I would like to thank.

First and foremost, from the International Research Center for Neurointelligence at the University of Tokyo, I would like to thank Zenas C. Chao and Felix B. Kern for their ardent guidance during my project, as well as Kantaro Fujiwara for his help with accessing and using their computational servers.

From the Biohybrid Systems Laboratory, I would like to thank Shoji Takeuchi for his patient guidance and allowing me to study at his lab, Kazuki Nishimoto for helping me transition to life in Japan, and Morie Watanabe for her help with official matters at the University of Tokyo.

From the Norwegian University of Science and Technology, I would like to thank Keith L. Downing for his sporadic yet precise guidance during my project, Pauline C. Haddow for her strict yet kind guidance before she had to go on leave, as well as my significant other, Emma L. Eikemo, for her motivating banter and help with proofreading.

Lastly, I would like to thank ChatGPT for reducing the burden on my tendonitis-riddled arms by providing a number of almost factually correct paragraphs to act as starting points for my writing.

Chapter 1

Introduction

1.1 Motivation and Background

Temporal sequence learning, the ability to recognize and reproduce information sequences over time, is a complex cognitive process widely observed in the animal kingdom. It underlies various intricate behaviors, from humans learning musical compositions to birds memorizing and recreating intricate song patterns. However, in the fields of Neuroscience and Artificial Intelligence, the mechanics of temporal sequence learning remains a scarcely understood research subject.

Several unique Artificial Neural Network (ANN) models made for temporal tasks have been suggested over the years, with various levels of success. However, balancing the demands of high-level performance and biological plausibility presents a challenge. Several effective machine learning methods such as LSTMs (Hochreiter et al., 1997) and other novel approaches (Chung et al., 2014, Lea et al., 2016) may not align with the constraints inherent in biological settings. Furthermore, as the complexity of models increases, their interpretability is often diminished, leading to the "black box" problem where the underlying workings of the networks are opaque.

Countering this trend, there has been a rise into research on biologically plausible neural networks the last few years. Two hand-crafted, and therefore relatively interpretable, models utilizing Hebbian Learning in Maes et al., 2020 and Maes et al., 2021 have shown good performance when faced with temporal tasks. However, these models are inflexible in regards to the problem domain, needing the tasks to be limited to specific trial lengths. Meanwhile, other similar research has primarily focused on the utilization of variants of backpropagation for learning, involve thousands of connections, or both.

An essential goal of this research is to investigate the possibility of generating biologically plausible ANNs that are both interpretable and flexible. Interpretable networks not only bolster the trust and verification process in the artificial intelligence realm, but also provide an understanding of the networks' decision-making process. This, in turn, provides an avenue to study the produced networks to gain a better understanding of the underlying processes in their biological counterparts. The proposed model for this purpose is designed as a semi-biologically plausible model serving as preliminary research into this subject, and will be referred to as NEAT HRNN (NEAT of Hebbian Recurrent Neural Networks) throughout the thesis.

1.2 Goals and Research Questions

This thesis is centered around the following goal and research questions.

Goal *Explore the potential of using NEAT to develop interpretable biologically plausible neural networks capable of temporal prediction*

There is a large volume of research on creating deep Recurrent Neural Networks and Spiking Neural Networks for temporal tasks. However, while these networks have been able to perform well on progressively more and more complex problems, their activity patterns are difficult to interpret. It's with this in mind that this thesis aims to create networks of small sizes, such that the structures underlying temporal prediction can be better understood.

RQ1 *What are the strengths of NEAT for the purposes of evolving biologically plausible and interpretable neural networks capable of temporal prediction?*

RQ2 *What are the limitations of NEAT for the purposes of evolving biologically plausible and interpretable neural networks capable of temporal prediction?*

RQ3 *How do RNNs encode prediction over time for temporal prediction tasks?*

RQ4 *How does changes in Hebbian weights correlate to different foreperiods?*

RQ1 and RQ2 ties directly into the goal, and are crucial to answer in the light of potential future research. RQ3 delves into questions of how both topology and connection weights affect networks' ability to store information over time. RQ4 deals with both exploring and questioning the role of Hebbian learning as a mechanism contributing to temporal prediction, and gives insight into how plasticity affects such tasks.

Chapter 2

Background Theory

This chapter establishes the core concepts and techniques needed to understand the contents of this thesis. 2.1 briefly introduces time series analysis, which leads directy in to 2.2 presenting key concepts in the domain of temporal prediction and the foreperiod task. 2.3 introduces Machine Learning (ML) as a concept and the particular branch of ML this thesis is focused on, which leads into 2.4 introducing Artificial Neural Networks (ANNs). 2.5 then delves into Evolutionary Algorithms, with 2.6 presenting the core algorithm of this thesis. Lastly, Hebbian Learning is introduced in 2.7.

It should be noted that parts of this chapter are based on previous joint work by me and Markus Hvidsten Kristoffersen (Kristoffersen et al., 2022).

2.1 Time Series Analysis

Time series analysis refers to the process of forecasting future events or states based on sequential data points observed over time, and is fundamental in various fields such as meteorology, finance, and neuroscience. A time series is a sequence of data points recorded at successive, usually equally spaced, time intervals. The essence of time series analysis lies in detecting regularities and trends in these sequences and using this information to make educated guesses about what the next entry in the sequence will be. This task can be extremely complex, depending on the nature of the data and the time scale over which predictions are made.

The nature of a time series is often divided into four distinct components; trends, seasons, cycles and irregularities. Trends represents the long-term progression of the series, either upward, downward, or stable, and indicate consistent increases or decreases over time. Seasons are regular, periodic fluctuations in a time series, often influenced by factors like seasons, quarters, or other recurring cycles. Cycles are long-term fluctuations without fixed periods, unlike seasonality, and can vary in duration. And lastly, irregularities are unpredictable, erratic fluctuations that cannot be attributed to trend, seasonality, or cycles.

In neuroscience, time series analysis is instrumental in studying brain activity patterns. Techniques like electroencephalography (EEG) and functional magnetic resonance imaging (fMRI) produce time series data reflecting neural activity. Analyzing these data can reveal insights into brain function, neural responses to stimuli, and patterns associated with different cognitive states or disorders.

2.2 Temporal Prediction

Temporal prediction is a subset of time series analysis used within neuroscience to describe predictions focusing on the timing of particular events, rather than output values at particular timesteps. In the biological world, the capacity for temporal prediction is crucial for survival, as temporal prediction is intimately linked to reaction time. In both human and animal behavior, the ability to predict when an event will occur enables a more rapid response, with research showing a strong correlation between the anticipation of an event occurring and the speed at which subjects react to the event (Janssen et al., 2005, Herbst et al., 2018). For example, a sprinter's reaction time at the start of a race is greatly enhanced by their ability to anticipate the starter's signal. This predictive timing streamlines the neural processing involved in initiating movement, allowing for quicker off-the-mark action. Similarly, a predator's ability to anticipate the movements of its prey can be the difference

between a successful capture and a missed opportunity. Reaction time, therefore, isn't just about raw speed but also about how effectively an organism can predict and then respond to events.

In the brain, there are several cortical regions that have been identified to be involved with temporal prediction. In particular, the prefrontal cortex and the inferior parietal cortex are believed to be integral to processing temporal information. Furthermore, regions such as the basal ganglia and cerebellum are involved with fine-tuning the timing of motor actions. The precise neural mechanisms underlying temporal prediction and reaction time are an area of ongoing research, but it is the interaction between these regions that is key to our understanding of temporal prediction and reaction time.

One of the missing pieces in understanding temporal prediction in the brain is the nature and topologies of the circuits in and connecting these cortical regions. Advancements in technology are enabling more accurate analysis and modeling of these regions, leading to the high-level communication between these regions being relatively well understood. However, the precise mechanisms of collaboration and mutual influence among these areas during temporal prediction tasks remain elusive, and is a vital area for research.

2.2.1 Hazard Function

The hazard function is a statistical concept that contributes to our understanding of how the brain might process temporal predictions. It represents the probability that an event will occur at a particular time given that it has not yet occurred. Unlike the probability density function (PDF), which describes the probability of an event at a specific time, the hazard function considers the history up to time t and provides the conditional risk of the event occurring in the next instant.

The hazard function is a crucial tool in various fields, including medical research for modeling patient survival times, engineering for understanding system failures, and in actuarial science for risk assessment. It is particularly valuable in cases where the risk of the event is not constant over time, which is common in real-world scenarios.

In neuroscience, the hazard function is an important tool for understanding the timing of events, such as neural spikes, responses to stimuli, or the occurrence of certain behaviors. It is especially relevant in the study of how neural systems process and predict temporal information. In particular, research suggests the hazard function may model the changing expectation of an event over time within the brain, influencing reaction times.

Mathematically, it can be expressed as a function the PDF divided by the survival function, which is given by:

$$S(t) = 1 - \int_0^t p(u)\,du \tag{2.1}$$

Which in turn gives the hazard function as:

$$h(t) = \frac{p(t)}{S(t)} \tag{2.2}$$

The survival function indicates the probability of an event not having occurred by timestep t. As an example, a computer program that has a uniform chance of crashing within 10 seconds of starting could be described by Figure 2.1.

From the figure, we can clearly see how the hazard function drastically increases over time, as the odds of a program instance still running decreases. The hazard function even clearly increases past 1 in the y-axis, which might seem weird at first glance. However, what the hazard function effectively encodes for in this example is the expected number of crashes within the next timestep, given that no crashes have occurred. In this example, the hazard function will eventually approach infinity, though it should be noted that distributions such as the one presented are extremely uncommon in real life, and most hazard functions will stay within more natural values.

2.2.2 Foreperiod Task

The foreperiod task is a widely used experimental paradigm in cognitive neuroscience and psychology research to investigate the neural mechanisms underlying temporal expectation and motor preparation. In this task, participants are required to make a response, usually a motor action like pressing a button, as soon as a certain stimulus is presented. The key feature of the task is the interval between

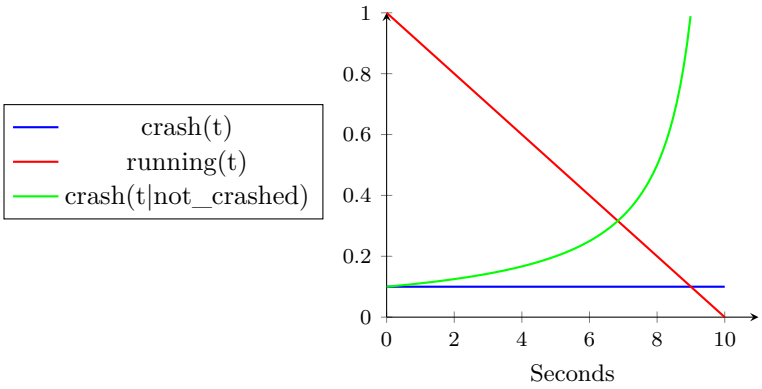


Figure 2.1: Probability Density Function $p(t)$, Survival Function $running(t)$ and Hazard function $crash(t|not\_crashed)$ of a program with a uniform chance of crashing between timesteps 0 and 10

a warning signal and the imperative stimulus (that prompts the response), which is known as the foreperiod.

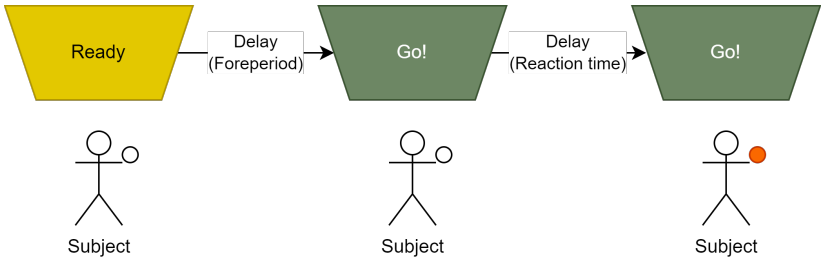


Figure 2.2: **The foreperiod task.** The Ready signal prompts the subject to predict the Go signal, and then the response time from the Go signal to the subject pressing the button is recorded.

A trial begins with a warning signal that alerts the participant to the impending imperative stimulus. This signal indicates that the participant should be ready to respond. After the warning signal, there is a waiting period before the imperative stimulus appears. This foreperiod can vary in length, either being fixed across trials or randomly varied to prevent the participant from predicting the stimulus onset. The imperative stimulus then follows the foreperiod, and requires the participant to perform a predefined action as quickly and accurately as possible. This marks the end of a trial, with data such as the participant's response time, accuracy and brain activity being recorded. Such trials are then repeated a number of times, usually with a small break between each trial.

Among many of the interesting discoveries made through the foreperiod task, is that brain activity closely follows the hazard function for different foreperiods after training (Bueti et al., 2010). If the foreperiod is fixed and known to the participant (prior probability), the hazard rate for the stimulus presentation increases rapidly near the end of the foreperiod. Meanwhile, with variable foreperiods (posterior probability), the hazard function initially increases as time progresses because the likelihood of stimulus presentation increases with the passage of time. However, if the foreperiod goes beyond the average or expected range, the hazard rate may plateau or even decrease, reflecting a drop in expectation or readiness caused by temporal uncertainty. This phenomenon is also observable in the participants' reponse times, as response time is highly correlated to the inverse of the hazard function (Bueti et al., 2010). Furthermore, researchers have found that different brain regions show increased activity when faced with prior or posterior probability tasks respectively (Coull et al., 2016), suggesting that the brain might process the two task types differently.

2.3 Machine Learning

Machine learning (ML) is a subfield of artificial intelligence (AI) that emphasizes the development of systems capable of learning from data without explicit programming. Essentially, the objective is to derive patterns or relationships from raw data, and then make predictions or decisions based on this derived knowledge. Machine learning is commonly divided into the three main subdomains

supervised learning, unsupervised learning, and reinforcement learning. For the scope of this thesis, the primary focus will be on unsupervised learning.

2.3.1 Unsupervised Learning

Unsupervised learning is a type of machine learning that trains on unlabeled data, striving to discern underlying patterns within it. The fundamental premise is to derive inherent structures from the data without any external guidance, that is, without the use of explicit labels or categories. Common algorithms under this umbrella include clustering, where data points are grouped based on similarity, and dimensionality reduction, where the aim is to reduce the number of variables in a dataset while preserving as much information as possible. The essence of unsupervised learning is to find representations for the data that can help in understanding its structure and subsequently using this understanding for various applications like anomaly detection or data visualization.

2.4 Artificial Neural Networks

At the intersection of biology and computation, artificial neural networks (ANNs) stand as computational models inspired by the brain's interconnected neuron structure. By mimicking the interconnected nature of neurons, ANNs provide a robust framework to tackle complex, non-linear problems. In essence, they are directed graphs where nodes (neurons) are connected by edges (synapses). These connections hold weights that get adjusted as the network learns. In the computational structure, the nodes are categorized into input, hidden, and output layers, and some might include bias nodes without any inputs for better performance. To calculate the output of the network, the input values are propagated through the hidden nodes to the output nodes. The value of each node's output when propagating through the network is given by

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.3}$$

where x_i are the values of the incoming connections, w_i are the corresponding weights, b is the bias and f is an activation function.

2.4.1 Recurrent Neural Networks

With standard ANNs being exclusively feed-forward, they cannot retain information from previous network inputs and outputs. Recurrent Neural Networks (RNNs) are a class of Artificial Neural Networks designed to recognize patterns in sequences of data, such as text, speech, or time series. Unlike feed-forward neural networks, RNNs possess connections that form directed cycles, creating an internal state that allows them to exhibit dynamic temporal behavior. However, traditional RNNs can struggle with long-term dependencies due to the Vanishing Gradient problem, making it difficult to learn from information in earlier time steps when using backpropagation or similar techniques to train the network.

2.5 Evolutionary Algorithms

Drawing inspiration from the process of natural selection, evolutionary algorithms (EAs) have established themselves as powerful optimization and search algorithms. The basic tenets revolve around the principles of selection, crossover (recombination), and mutation to find optimal or near-optimal solutions to problems. This section delves deeper into genetic algorithms, a popular subset of EAs, and explores some nuanced variations of it.

2.5.1 Genetic Algorithm

Genetic algorithms stand as testament to the power of bio-inspired computation. Based on the concepts of genetics and evolution, these algorithms iteratively evolve a population of candidate solutions to refine and approach an optimal solution. The process starts with the initialization of a random population of individuals. Over successive generations, individuals are selected based on their fitness, mated to produce offspring, and subjected to mutations to introduce variability. The

iterative process refines the population, aiming for convergence towards an optimal or near-optimal solution.

2.5.2 Speciation

Incorporating the concept of species from biology, speciation in genetic algorithms aims to promote diversity and protect innovative solutions. By categorizing solutions into different species based on similarities and ensuring competition within species rather than across them, speciation ensures that newly emerging solutions aren't prematurely discarded. This process leads to more exploration of the solution space, nurturing innovation and preventing premature convergence.

2.5.3 Neuroevolution

Neuroevolution is the intersection of ANNs and Evolutionary Algorithms, where an EA is used to evolve the weights and/or structure of an ANN. This concept mirrors how the structure of neurons in the natural brain has evolved over millions of years and is encoded in our DNA.

The field can be largely divided into two categories. In the first category, the ANN topology is predetermined, and the EA is only used to evolve the weights of the network. The second category deals with Topology and Weight Evolving Neural Networks (TWEANNs), where both the topology and weights are developed by the EA.

2.6 Neuroevolution of Augmenting Topologies

This section will focus on a specific neuroevolution algorithm called Neuroevolution of Augmenting Topologies (NEAT), introduced in the work of Stanley and Miikulainen (Stanley et al., 2002), and some of the algorithms that build upon it. NEAT is a TWEANN algorithm for evolving ANNs through a combination of the Genetic Algorithm and historical markings .

In NEAT, networks are expressed by a genome consisting of all connections in the network, with each connection receiving a unique historical marking upon creation. These markings allow for crossover between networks without loss of information, leading to a gradually complexifying search through network topologies, optimizing for those that perform best for a given task. This simple yet effective design has seen great success throughout the years and has given rise to a large family of algorithms that expand on it Papavasileiou et al., 2021.

2.6.1 The Competing Conventions problem

Before the introduction of NEAT, the predominant method of crossover for TWEANN was subgraph swapping. This technique is based on the idea that a segment of a neural network, or a subgraph, functions as a discrete unit. Swapping and merging subgraphs from varying networks seemed logical for performing crossover, as it essentially combined different functional units. However, the techniques of that era did not ensure that the chosen subgraphs were appropriate for producing functional offspring. Specifically, they did not take into account the competing conventions problem.

The competing conventions problem arises when two phenotypically similar networks can be represented by different genotypes. When this occurs, the resulting offspring can be compromised, losing some functionality inherited from the parents. Figure 2.3 exemplifies this: the crossover of the parents [A, B, C] and [C, B, A] yields an offspring [C, B, C], which lacks the information encoded by A in its parent structures. The challenge is further intensified with TWEANNs, as their flexible structure means equivalent networks might be represented by an even wider array of genotypes.

The competing conventions problem also causes issues when attempting to introduce speciation in TWEANNs. As described in section 2.5.2, there needs to be a compatibility function to compare the solutions in order to group solutions into species. The competing conventions problem makes it really hard to find this function because very similar phenotypes that should belong to the same species can have vastly different genotypes.

2.6.2 Genetic Encoding and Historical Markings

NEAT's solution to the competing conventions problem lies in its genetic encoding. The genome consists of two different types of genes, node genes and connection genes. Node genes are simple

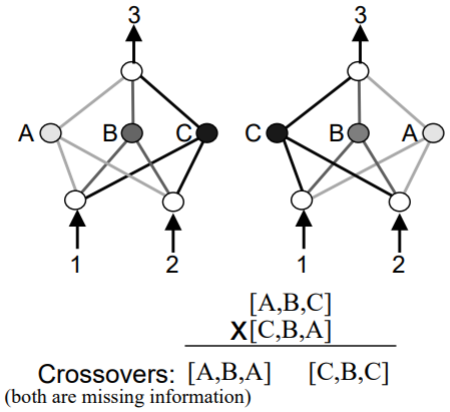


Figure 2.3: **The competing conventions problem.** Although the two networks have the same structure their chromosomes are different, which makes the offspring defect when performing crossover. (Stanley et al., 2002)

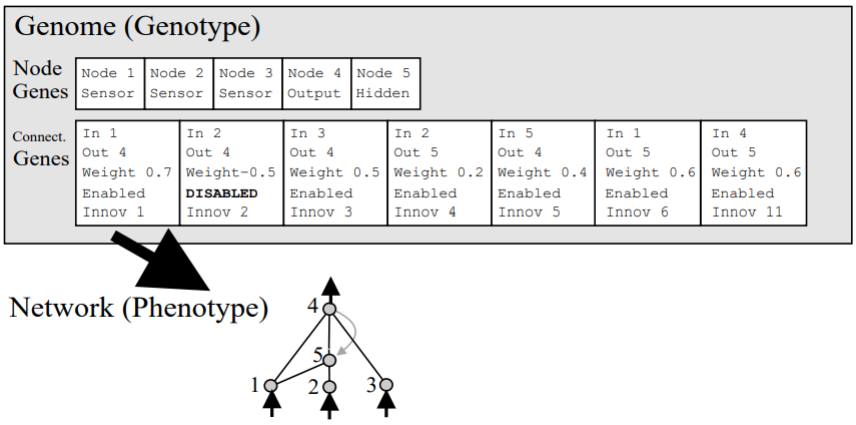


Figure 2.4: An example of a genome in NEAT and its corresponding phenotype network. Stanley et al., 2002

and only contain a number for the node and the type of node it is - input/sensor, output or hidden. Connection genes contain the node number for the two nodes it connects, the weight of the connection, a flag that allows the connection to be enabled or disabled, and an innovation number. The innovation number is a historical marking that is assigned when the connection is first created, and then never changed. The innovation numbers are kept track of globally so that a previously assigned number will never be assigned to a newly created gene in any individual. This is what allows NEAT to circumvent the competing conventions problem, as different individuals now have a historical marking for each connection that shows which connections originate from the same mutation and thus are equivalent. Figure 2.4 shows a visual representation of a NEAT genome and its corresponding phenotype.

The mutations that occur in NEAT affect the weights or structure of the network. Weight mutations are simple, with each weight either being changed or unchanged with a certain probability. There are four types of structural mutations, one for creating a connection between existing nodes, one for adding a node, as well as two for deleting connections and nodes in the network. When creating a connection between existing nodes a new connection gene is simply added to the genome. In the mutation for adding nodes, the node is always added by splitting an existing connection and adding the node in between. The previous connection gene is set as disabled and two new connection genes are added - one from the input node of the previous connection to the new node and one from the new node to the output node of the previous connection - as well as a new node gene. The connection into the new node gets a weight of 1, and the connection out gets the same weight as the previous connection. This ensures that the function of the network won't drastically change after applying this mutation.

2.6.3 Crossover

Historical markings, specifically innovation numbers, make it possible to identify corresponding genes in different individuals during the crossover process. During crossover, genes are categorized into: matching genes (present in both parents); disjoint genes (present in only one parent but within the range of the innovation numbers of both); and excess genes (those that are beyond the range of the other parent's innovation numbers). The matching genes are randomly selected from either parent. If the fitness levels of the two parents are equal, disjoint and excess genes are chosen randomly. Otherwise, they are taken from the fitter parent. An illustration of the crossover between two parents producing an offspring is shown in Figure 2.5.

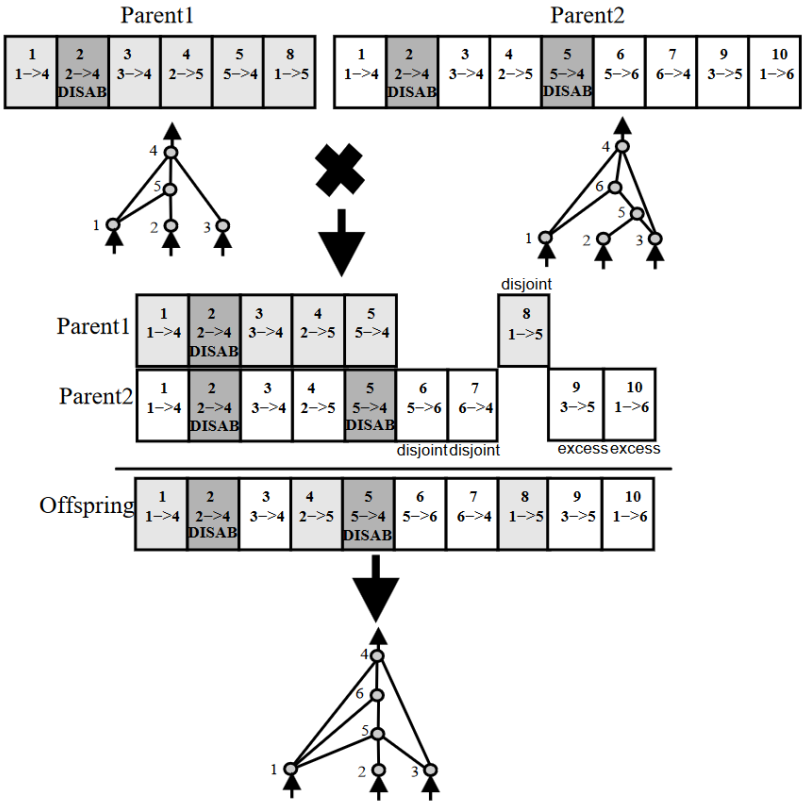


Figure 2.5: An example of crossover between two genomes in NEAT. Matching genes are chosen randomly, while disjoint and excess genes are taken from the fittest parent. In this example fitness is assumed equal, so disjoint and excess genes are also chosen randomly. (Stanley et al., 2002)

Speciation

The introduction of historical markings solves the issue of finding a compatibility function for TWEANNs, because they say something about how much genetic history two individuals share. In NEAT, the compatibility distance δ is a linear combination of the number of excess genes E and disjoint genes D, and the average difference in the weight of the matching genes, $\overline{W}$:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W} \tag{2.4}$$

The coefficients c_1, c_2 and c_3 gives a weight for the importance of each factor, and N is the total number of genes in the largest of the two genomes.

To decide whether two individuals belong to the same species the value of δ is compared to a compatibility threshold δ_t. Every individual is placed into a species at the beginning of each generation. The existing species are taken from the previous generation, with each species getting a random genome belonging to it in the previous generation as a representative. Every genome of the new generation then iterates through these representatives and are placed in the first species

for which the compatibility distance is below the threshold. If no compatible species is found for a genome, a new one is created with that genome as its representative.

NEAT uses explicit fitness sharing (Goldberg et al., 1987), meaning the fitness is shared between the members of a species, preventing one species from taking over the population. The formula for the adjusted fitness function $f_i^{'}$ for individual i is

$$f_i^{'} = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))}. \tag{2.5}$$

$sh(\delta(i,j))$ is a sharing function whose value is 0 if δ is greater than δ_t or 1 if δ is less than δ_t, meaning $\sum_{j=1}^{n} sh(\delta(i,j))$ is the number of individuals belonging to the same species as i. Each species is given a number of offspring proportional to the sum of adjusted fitness of its members. The individuals with lowest fitness are then removed, and the members of the species for the next generation are produced through reproduction between the remaining individuals.

2.6.4 Incremental Growth from Minimal Structure

Another hallmark of NEAT, setting it apart from other TWEANN methodologies, is its commitment to maintaining solutions as streamlined as possible. This is achieved by beginning with simplistic solutions and methodically adding complexity as required. Starting with overly complex random initializations usually results in many non-functional structures. Since mutations tend to either retain or add complexity, these ineffective structures persist, rendering the resultant ANNs needlessly intricate. By launching with a bare-bones structure, NEAT ensures that only those structural mutations which prove advantageous during the evolutionary process influence the final ANN design.

2.7 Hebbian Learning

Hebbian learning, a biological learning principle stating that "neurons that fire together, wire together," has found valuable applications in training ANNs, RNNs and SNNs. It is an abstraction of synaptic plasticity, the ability of synapses to strengthen or weaken over time, which is crucial for learning and memory formation in the brain. This approach facilitates unsupervised learning by adjusting synaptic weights based on the correlation of activity between pre-synaptic and post-synaptic neurons. When applied to RNNs, it allows learning without falling into the Vanishing Gradient problem.

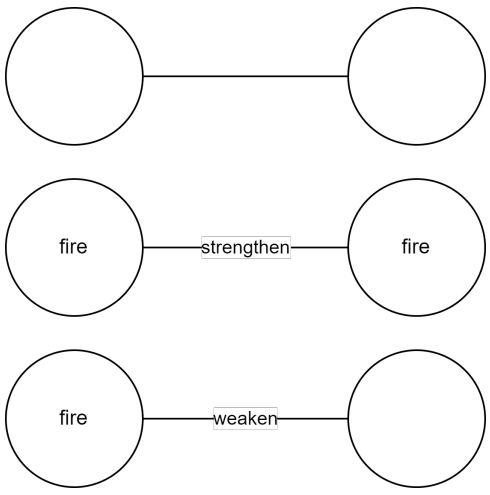


Figure 2.6: A simple representation of the Hebbian learning rule, with a form of Long Term Depression for uncorrelated firing.

There are many variations of the Hebbian learning rule, with different strengths, weaknesses, and application areas. Highly biologically accurate models such as Spiking Neural Networks (SNNs) typically evaluate Hebbian updates based on the timing of spiking patterns between nodes. Meanwhile, ANNs normally consider the output values of the presynaptic and postsynaptic nodes at the same timestep t. However, most implementations can be described by the following equation:

$$Hebb_{i,j}(t+1) = (1-l)Hebb_{i,j}(t) + lP_{i,j}(t) \tag{2.6}$$

Where l is the network's Hebbian learning rate, and $P_{i,j}(t)$ signifies the pre- and post-synaptic activities of the connection. A key notion of most implementation of this rule is that the magnitude of change in the Hebbian weights scales with the proximity of rapid firing between the neurons.

In its basic form, Hebbian learning continually strengthens the synaptic weights between neurons that are simultaneously active. However, without a mechanism to control this strengthening, synaptic weights can grow indefinitely, leading to issues like saturation where a neuron becomes overly responsive. For this purpose, a multitude of weight normalization techniques are employed, which can be roughly divided into multiplicative normalization and subtractive normalization techniques.

Multiplicative normalization involves adjusting the synaptic weights entering a neuron so that their total strength remains within a magnitude range. Examples include, vector normalization, in which each connection weight is divided by the sum of all incoming weights to the neuron, and Oja's rule, in which Hebbian updates are inhibited by abnormally strong node activity (Oja, 1982). Meanwhile, subtractive normalization adjusts the threshold needed for Hebbian weight changes to take place. This threshold can either be static or dynamic, and could the same for the whole network or have different values for each neuron, such as in the BCM rule (Bienenstock et al., 1982).

Chapter 3

Related Work

This chapter covers especially relevant research, algorithms and frameworks for the development of the proposed model. First is an outline of the literature review process is outlined in 3.1, before the subsequent sections delve into related work. 3.2 introduces τ-NEAT, a variant of NEAT with modifications to improve performance in temporal tasks. While 3.3 introduces a hand-crafted biologically plausible Recurrent Spiking Neural Network for sequence learning.

3.1 Systematic Literature Review

A literature review was conducted both in the preparatory phase of the thesis, as well as during review to find potential relevant papers that were missed. This was conducted with the following goals:

1. Establish a good understanding of temporal prediction within the field of AI
2. Obtain an overview of NEAT usage within the field of temporal prediction
3. Obtain an overview of biologically plausible AI models

Two separate techniques were used sequentially in the search for relevant papers; a general breadth-first search, followed by a more pointed depth-first search. During the breadth-first search phase, relevant keywords such as "NEAT", "temporal prediction" and "biologically plausible" were input into Google Scholar. By searching wide in this fashion, several unique papers and ideas would present themselves during the search. This was followed by filtering the found papers to a few core papers, from which a depth-first search was made by following the citations found in the papers, as well as in the cited papers. In doing this, a deeper understanding of the core concepts used in the thesis could be attained. During both phases, when considering a paper for inclusion, each paper was evaluated by referring to the literature review goals.

3.2 τ-NEAT

Caamaño et al., 2015 is one of very few papers that explore using NEAT in a purely temporal domain, highlighting it as a paper of particular interest for this thesis. The paper introduces τ-NEAT, a specialized variant of the NEAT algorithm designed to enhance the performance of neural networks in handling temporal tasks, developed for use in time-series analysis. τ-NEAT achieves this enhancement by introducing variable delays within the synapses of the networks constructed by NEAT. The variable synaptic delays are introduced into the NEAT chromosome and are modeled through a FIFO buffer containing the last n input values to a synapse, thus adding a temporal dimension to the synaptic interactions. The length of each buffer can be modified through mutations in the genome, within a predetermined range, making it part of the evolutionary process of τ-NEAT.

In the paper, the authors apply τ-NEAT to three different time-series analysis tasks of increasing complexity: the logistic map chaotic series, the Henon attractor series, and the Mackey-Glass time series. In all three tasks τ-NEAT outperforms NEAT, though NEAT just barely falls behind for the Henon attractor series, as seen in Figure 3.3.

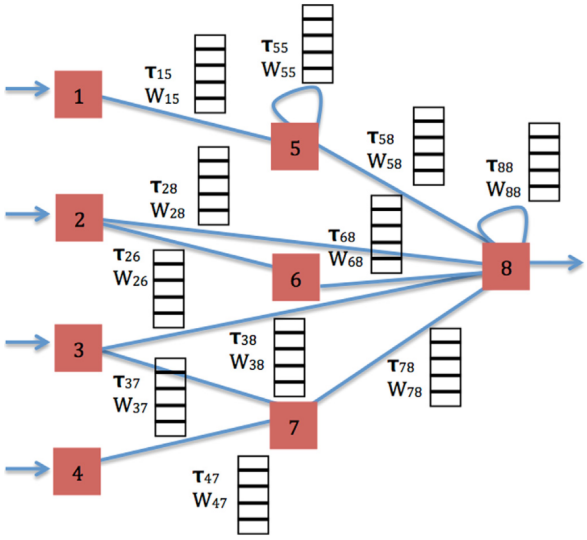


Figure 3.1: **Example τ-NEAT network.** Each connection in the network has both a weight and a variable-size FIFO buffer, representing the synaptic delays. (Caamaño et al., 2015 - with permission)

	Logistic	Henon	MG
tau-NEAT	9600E−05	5163E−05	9228E−06
NEAT-NU	2138E−04	8395E−05	1457E−04
NEAT	4555E−04	2271E−04	5452E−05

Figure 3.2: **MSE of best τ-NEAT and NEAT networks** (Caamaño et al., 2015 - with permission)

These experiments and their results provide a good window into the performance of NEAT for temporal tasks. Furthermore, since the temporal prediction tasks in this thesis can be modeled as a subset of time series analysis, a lot of relevant knowledge for my thesis can be extracted from the paper. One key takeaway is that their results make it clear that NEAT can perform well when faced with temporal tasks. However, some limitations of NEAT are also made apparent throughout the τ-NEAT paper.

Their study aims for optimal performance in their chosen problem domains, potentially at the cost of increased network complexity and size. While the τ-NEAT models in the paper achieve good results in the designated tasks, there is no discussion of the developed topologies. However, by inspecting the chosen hyperparameters, as seen in Figure 3.3, it becomes clear that the networks likely are of significant size. In particular the absence of the `remove node mutation`, and the high difference between `add connection rate` and `remove connection rate` is telling. This focus on performance might lead to network bloat, where evolved topologies become large and complex, potentially obscuring their functional transparency.

This is not a problem given their research goals, however my research seeks to identify minimalistic network structures capable of predicting simple temporal patterns, that could possibly be repeated throughout the brain. As such, this size and complexity not only hinder the interpretability

Generations	1000
Population size	200
Topology mutation	Classic
Add connection rate	0.1
Remove connection rate	0.01
Remove connection max weight	5.0
Add neuron rate	0.1
Prune mutation rate	1.0
Weight mutation	Non uniform mutation
Weight range	[−10.0:10.0]
Survival rate	0.1
Elitism	True
Selection operator	Roulette
Elitism species minimum size	1

Figure 3.3: **τ-NEAT and NEAT parameters.** (Caamaño et al., 2015 - with permission)

of the networks, a critical aspect when aiming for biologically plausible models, but also means that the developed networks likely aren't the sort of minimal networks my research is focused on. Furthermore, although the concept of synaptic delay exists in the brain, the difference in delay length between different neurons normally vary in small amounts. Meanwhile, the synaptic delays in the τ-NEAT model are limited to discrete timesteps, meaning there could be several orders of magnitude in difference in synaptic delay between connections.

In conclusion, the τ-NEAT model offers valuable insights into harnessing NEAT for temporal tasks, which aren't easily found elsewhere. It could serve as an inspiration for future research of my model, particularly if standard NEAT proves insufficient. Nevertheless, given their results showing that standard NEAT already is capable of performing impressively in temporal tasks prompt a reconsideration of the need for the added complexity of such large synaptic delays. With that said, given the hypothesis that generalized motifs for temporal prediction in the brain are likely small and simplistic, and considering the typically short synaptic delays in neuronal systems, it seems prudent to explore a more streamlined approach.

3.3 Learning Spatiotemporal Signals Using a Recurrent Spiking Network That Discretizes Time

In search of biologically plausible neural networks dealing with temporal prediction, Maes et al., 2020 stood out as particularly interesting. Their work introduced a handcrafted biologically plausible Spiking Recurrent Neural Network (SRNN) capable of learning spatiotemporal patterns, which inspired parts of this thesis. Their work focused on developing a computational model to understand how the brain learns and encodes spatiotemporal sequences, addressing the gap in current models which often lack realistic, biologically-plausible learning mechanisms.

The model is designed to address the challenge of bridging the gap between the millisecond time scale at which individual neurons operate and the longer behavior time scales spanning several milliseconds to hundreds of milliseconds. This is crucial because many neural tasks and behaviors involve learning and producing flexible spatiotemporal sequences, such as the sequential motor tasks in songbirds and other animals. Their model is compromised of two main modules; a driver SRNN consisting of excitatory and inhibitory neurons, and a read-out layer.

The SRNN is organized into clusters of excitatory neurons and a central cluster of inhibitory neurons, designed to encode the dimension of time, discretizing it into sequential intervals. The excitatory neurons follow adaptive exponential integrate-and-fire dynamics, while the inhibitory neurons follow a leaky integrate-and-fire model. The output of the driver SRNN is then mapped onto another dimension, such as space or phase, through all-to-all feedforward connections from the exitatory neurons of the SRNN to the read-out layer.

The read-out layer consists of neurons that are driven by these excitatory neurons. These neurons are not interconnected among themselves, but are solely driven by output from the SRNN, and encodes for the spatial dimension of the problem domain.

For learning, the model leverages Spike-timing-dependent plasticity (STDP), a variant of the Hebbian learning rule, allowing for the learning and encoding of various spatiotemporal patterns in a biologically plausible manner. They train the model using STDP in two distinct stages; learning to exhibit sequential dynamics, and learning the target sequence in the read-out layer.

In the first stage of learning, the network learns to generate sequential activity that operates as a 'neuronal clock,' providing a temporal backbone that drives the learning of the read-out layer. During this stage, all neurons in each cluster of the recurrent network receive the same input sequentially. This process creates the sequential dynamics of the clusters, effectively discretizing time in the recurrent network.

After establishing the sequential dynamics of the SRNN, network is ready for the second stage of learning. During this stage, neurons in the read-out layer receive additional input which modulate their activity in tune with the desired sequence at the appropriate timesteps. For example, to teach the network to output A during $[t_i, t_{i+1}]$, the corresponding read-out neuron would receive activation signals during training for that slice of time, but inhibitory signals otherwise. This updates the read-out weight matrix, representing the weights of the connections between the SRNN and the read-out layer, according to STPD. After training, the relationship between SRNN cluster activity and read-out layer output according to the desired sequence will be embedded in the weight matrix, and the additional inputs to the read-out layer are turned off.

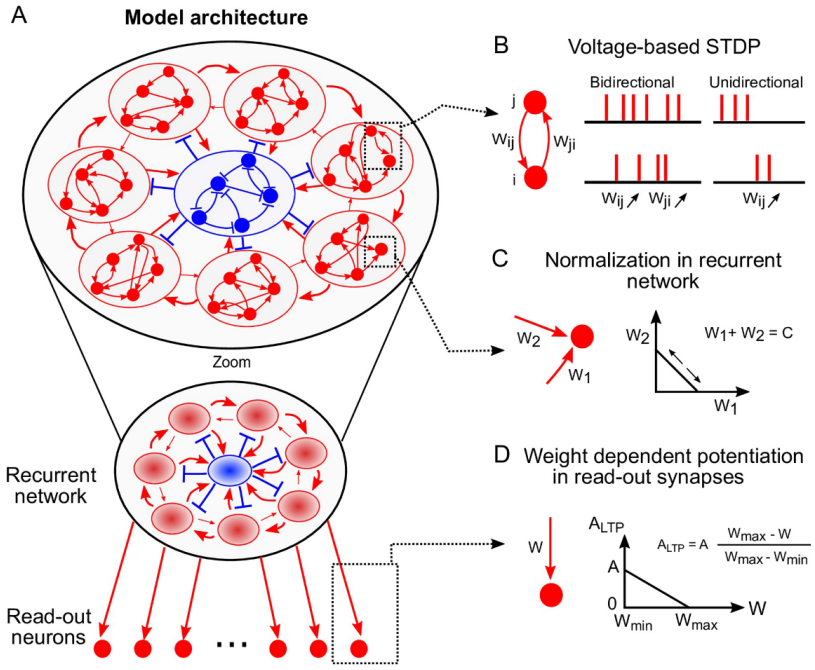


Figure 3.4: **Maes et al. model architecture.** (A) The driver RNN, consisting of clusters of excitatory (red) neurons and a single cluster of inhibitory (blue) neurons. (B) An example of voltage-based STDP in the model. When two neurons j and i fire together a lot, they will form a bidirectional connection strenghtening both W_{ij} and W_{ji}. When neuron j fires before i, W_{ji} will be unidirectionally strengthened. (C) The sum of all incoming exitatory connections to a node will be normalized, keeping the sum of all excitatory weights to the node constant. (D) Potentiation of read-out synapses are linearly dependent on weight. This, combined with weight normalization, gives a soft upper bound for the weights. (Maes et al., 2020)

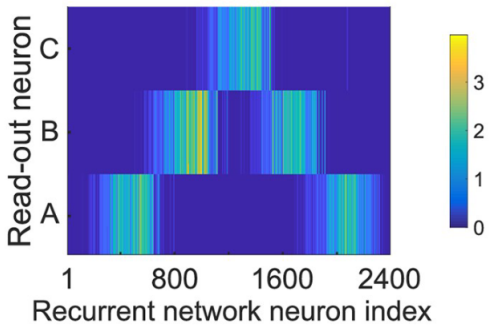


Figure 3.5: Read-out weight matrix for the sequence ABCBA after training. (Maes et al., 2020)

They claim that this setup makes the network capable of learning non-Markovian sequences, which requires that information about previous network states are stored in the network. With only learning one sequence, limited in duration by the sequential dynamics embedded in the network, it could be argued that explicit information about previous network states do not need to be encoded in the network activity to learn Markovian sequences. However, the network was able to learn two different sequences when presented to it sequentially, given that each sequence had its own dedicated read-out neurons. This suggests that the network activity does in fact encode for previous network states.

However, there is still one further limitation that the sequence must adhere to. That is, it cannot exceed the duration of one cycle of the sequential dynamics of the SRNN. In their follow-up study (Maes et al., 2021), they expand on the model by simulating two neuronal clocks, one fast and one slow. Furthermore, they replace the read-out neurons with small networks encoding sequence motifs, which are short sequences synced with the fast clock. Although these changes alleviate the problems native to the model, they are still not eliminated.

Lastly, although their model was able to learn a single sequence when each cluster was reduced

in size to just one neuron, the topology of the networks are highly specific in terms of function. Especially in regards to the limitations imposed by the neuronal clock. This makes it unlikely that their handcrafted topology is repeated throughout the brain, as such microcircuit motifs should necessarily be relatively general in their function.

Nevertheless, their model offered three key inspirations for the development of this thesis. Firstly, the property of these networks to encode information of previous network states reinforced the idea of using RNNs for prediction. Secondly, the paper showcased the power of Hebbian learning clearly, inspiring its use in this thesis. And lastly, the desire to overcome some of the limitations posed by the neuronal clock helped shape the Ready-Go task.

Chapter 4

Methodology

The model proposed in this thesis combines the techniques introduced in the Background Theory section. The NEAT of Hebbian Recurrent Neural Networks (NEAT HRNN) model is developed to create biologically plausible and interpretable RNNs. Its primary aim is to aid in the understanding of temporal sequence learning processes within biological neural networks.

4.1 Network Implementation

The networks developed by this model are all RNN networks utilizing Hebbian learning. An essential aspect I considered when designing the model is that it should not aim to find the most optimized networks for specific problems, but to uncover networks that are biologically plausible, interpretable, and perform satisfactorily. The reason being that developing such networks facilitates the emergence of network topologies that can be studied to gain insights into the related biological processes.

For this reason, Hebbian learning was incorporated to introduce plasticity into the networks, allowing the weights to adapt over time in a manner that is biologically plausible, while the network topology remains constant. For the networks in my model, the Hebbian weight is defined as:

$$Hebb_{i,j}(t+1) = (1-l)Hebb_{i,j}(t) + l((y_i(t) - y_{th})(y_j(t) - y_{th})) \tag{4.1}$$

Where l is the network's Hebbian learning rate, $y_i(t)$ and $y_j(t)$ indicate the output of the connected nodes, and y_{th} indicates the firing threshold for a node to be considered firing. Weight changes only apply if at least one of the nodes' outputs exceed the firing threshold, meaning that the updates can be both positive and negative. This implementation is a form of subtractive weight normalization, where the firing threshold is subtracted from the node activities when performing weight updates. In doing this, the magnitude of the positive and negative changes are kept in balance, as negative changes would otherwise naturally be lower than positive. Note that only excitatory connections have their weights change through Hebbian learning, while inhibitory have static weights. This design decision was made based on the prevalence of plasticity in positive connections in the brain, relative to that of negative connections. Although these aspects are quite standard for implementations of the Hebbian learning rule, there are some nuances to how these updates are applied to the network.

Notably, the model batches together all the Hebbian weight updates of a trial, instead of applying them immediately. With each trial of the Ready-Go experiment being evaluated over several timesteps, as opposed to instantaneously, updating the Hebbian weight at every timestep could affect the network output during the trial. To avoid any unstability this might cause, the updates for each timestep of the trial τ are simply summed together and applied before the evaluation of the subsequent trial $\tau + 1$:

$$Hebb_{i,j}(\tau+1) = \sum_{t \in \tau} Hebb_{i,j}(t) \tag{4.2}$$

Furthermore, I have taken inspiration from an implementation detail of Miconi et al., 2018, in which each Hebbian weight is multiplied by a corresponding Hebbian magnitude factor evolved for each connection. As in their implementation, the Hebbian weights are limited within the range $[-1, 1]$, though in my implementation, the Hebbian magnitude factor is instead evolved for each node. This fine-tunes the influence the Hebbian factor imparts on node outputs and showing similarities to how

different neurons are affected differently by plasticity. Consequently, a connection in the network is evaluated on the basis of three components: static weight, Hebbian magnitude factor, and Hebbian weight, giving the following equation for the connection activity level at time t:

$$p_{i,j}(t) = w_{i,j} + \alpha_j Hebb_{i,j}(\tau) \tag{4.3}$$

Where $w_{i,j}$ is the weight of the connection between node i and j, α_j is the node's Hebbian magnitude factor, and $Hebb_{i,j}(t)$ is the connection's Hebbian weight at trial τ. Lastly, when evaluating a node output, the sum of the acticity level of each connection is provided as input to the node's activation function, σ:

$$y_j(t) = \sigma\left(\sum_{i \in \text{ inputs to } j} p_{i,j}(t)\right) \tag{4.4}$$

With σ being the logistic function, as implemented in the NEAT-Python library (McIntyre et al., n.d.), in which the exponent of e is modulated to make the function converge faster towards 0 or 1:

$$\sigma(x) = \frac{1}{1 + e^{-5x}} \tag{4.5}$$

Do note that there is no holdover of node activity between timesteps. Meaning that the node activity at one timestep is not taken into account when evaluating the same node at subsequent timesteps. This decision was made with the assumption that if holdover is important for temporal prediction, NEAT should be able to find through mutations that add excitatory self-recurrent connections.

When making these node evaluations, the network progresses one step at a time for each timestep. This is done to ensure that information in the network doesn't travel through every node at each activation, as that would make time-sensitive predictions trivial. With this in mind, the network is also limited to avoid direct connections between the input and output nodes.

Finishing off, a fairly common implementation detail of this model is keeping the Hebbian weight seperate from the original weight values, ensuring they remain unchanged. Maintaining the Hebbian factor as an independent entity from the weights allows for resetability between runs and generations, ensuring non-interference with the operations of the NEAT algorithm, and preserving the integrity of the evolutionary process.

4.1.1 Model Limitations

Although the model aims to generate biologically plausible networks, there are some limitations that hinder this goal. Most notably, the model evolves Recurrent Neural Networks, rather than Recurrent Spiking Neural Networks. An RSNN could give more accurate simulations and is included in the Neat-Python library the project is based on, but would also be more difficult to set up. As the aim of this project was to examine the feasability of using Neuroevolution to find biologically plausible neural network topologies, a decision was made to reduce the complexity of the model in favor of ease of development.

Secondly, the model does not distinguish between different types of neurons. Although the behaviour of different neuron types might be loosely modeled through node attributes, the model places no restrictions on the outgoing connections of nodes to be either positive or negative. This is in contrast with biological neurons, which have the sign of all outgoing connections determined by the neuron type. While this deviation from biological plausibility exists, a theoretical split of each node into two imaginary nodes following this model's rules would yield equivalent outputs, mitigating concerns over biological coherence.

Although modifying the NEAT implementation to develop networks with signed nodes was considered, it was quickly discovered that such a change would be incompatible with the algorithm. An effect of adding a new node through mutation is that a connection is split in two, with a connection with a weight of 1 being connected from the outgoing node of the original connection to the new node. This is done to avoid perturbing the function of the network too much when such mutations are made, ensuring stable and progressive evolution. However, it also means that adding a node along the outgoing connection of a negative node would either require the negative node to swap sign, or the mutation would perturb the network function a great deal. As such, the implementation is left without this modification to ensure its stability.

4.2 NEAT Implementation

The model employs NEAT due to its well-recognized effectiveness as an evolutionary algorithm. I chose to base the implementation on the NEAT-Python library due to three factors. First and foremost, I had already explored using the Pureples library, which is based on the NEAT-Python library. This is because some very early testing was done with ES-HyperNEAT (Risi et al., 2010), before it was concluded that standard NEAT would converge faster towards good solutions for the Ready-Go task. Secondly, my familiarity with Python and its active use in AI-related fields would make it easy to develop in. And lastly, the library had a variant of NEAT that develops RNNs, which are well-suited to address problems in timing- and history-sensitive domains, and are more analogous to biological neural networks than standard ANNs.

The networks, during the evaluation phase, are subjected to one or multiple temporal sequence learning tasks, where connection weights are adjusted following the Hebbian learning rule. In early experiments, I observed that the networks can perform well on single foreperiod tasks without the need for Hebbian learning. However, to ensure that the networks maintain flexibility and adaptability in response to changes in the problem domain, such as adjusting the foreperiod, the implementation of Hebbian learning was deemed necessary.

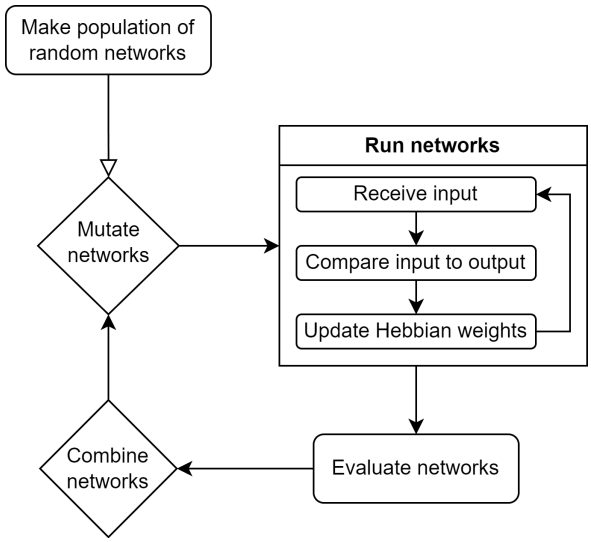


Figure 4.1: Procedure of the NEAT HRNN model

In addition to changing the implementation to include Hebbian learning, I made a few other alterations to the NEAT-Python implementation. The `response` parameter of NEAT-Python, normally being an evolved scalar that node output of specific nodes is multiplied by, was left unused. Due to this, and the ease of conversion, I repurposed it as the Hebbian magnitude factor. Futhermore, in order to guarantee the needed network limitation that no direct connections between the input and output nodes were made, the mutation that add new connections was adjusted accordingly. This restriction was necessary, as a direct connection between input and output nodes would trivialize the problem domain of this thesis, as well as many other timing-based problem domains.

4.2.1 Hyperparameters

Certain hyperparameters can have a large impact on the size and diversity of the generated networks. In order to control the generation of networks towards diverse populations of interpretable networks, it was important to generate both a wide range of species and compact individuals.

To achieve this, I took inspiration from Bloat Free NEAT (BF-NEAT) (Trujillo et al., 2014) when configuring the hyperparameters for NEAT. The parameters available between the NEAT implementation used by BF-NEAT and the NEAT-Python library differs slightly, so some flexibility was needed. With this in mind, the hyperparameters in Table 4.1 have been used when configuring the NEAT algorithm.

The odds of adding and removing connections and nodes are controlled by `conn_add_prob` and `conn_delete_prob`, and `node_add_prob` and `node_delete_prob` respectively. These parameters were low to allow individuals to explore different balances between weight, hebbian magnitude and bias for

Hyperparameter	Value
compatibility_disjoint_coefficient	1.0
compatibility_weight_coefficient	0.0
conn_add_prob	0.2
conn_delete_prob	0.3
node_add_prob	0.1
node_delete_prob	0.15
compatibility_threshold	3.0
max_stagnation	100
species_elitism	10
elitism	10
survival_threshold	0.2

Table 4.1: Core NEAT hyperparameters used for generating networks

each topology. Furthermore, deleting connections and nodes usually disrupt the network more than adding them, and as such those mutations are often less favored, leading to more complex network topologies over time. By having a higher chance of delete mutations rather than add mutations, networks with reduced complexity are explored more often, increasing the odds of simpler networks surviving.

The compatibility coefficient parameters control speciation, by affecting how much certain factors contribute to genetic distance. Here, `compatibility_disjoint_coefficient` affects genetic distance gained from disjointed and excess genes, while `compatibility_weight_coefficient` affects genetic distance gained from the weight, bias, and node response factors. As finding unique topologies is prioritized, the `compatibility_weight_coefficient` factor is set to 0 to avoid differentiating between weights.

Speciation between differing topologies is then taken advantage of by implementing a relatively low `compatibility_threshold`, which is the genetic distance threshold for differentiating between different species, and high `species_elitism`, which ensures the top n species don't go extinct even if they stagnate. This combination ensures a diverse population of species are kept each run, and that new species are discovered frequently. It should be noted that such a low `compatibility_threshold` diverges from the standard BF-NEAT hyperparameters, but given the reduced size of the networks developed using this model, the genetic distance between networks will generally be much lower to begin with. Lastly, by having a relatively high `max_stagnation` parameter, which sets the threshold for how many generations a species can live without improving, newly discovered species have time to explore many solutions before potentially going extinct.

In addition to the hyperparameters in focus here there are many more that vary between runs. These can be found in the NEAT configuration files alongside the relevant results in the GitHub repository[1].

4.3 Ready-Go Task

The Ready-Go task was made as a problem domain for this research, based on the foreperiod task. When adapting the foreperiod task to an AI environment, I split the Ready and Go signals into two separate input nodes. Both the Ready and Go nodes will output 0 at all timesteps except for when they are set to fire, at which the respective node will output 1.

The Ready signal marks the start of a trial, representing the cue, and prepares the network for the coming Go signal. Then the Go signal, being the target for prediction, is provided after a delay selected from a predetermined distribution. Lastly, a random amount of delay selected from a uniform distribution is inserted before the next trial begins. In time series analysis terms, this could be seen as introducing irregularities into the dataset, but in the temporal dimension. This is done to ensure that the generated network responds to the Ready and Go signals respectively, instead of generating networks that simply output a static pattern. The networks are then given 5 trials for the Hebbian weights to update before being evaluated based on their average fitness across all trials thereafter.

Such trials are repeated a number of times for each foreperiod, called a block. The Hebbian weights and network state is kept between each block, and measures are taken to ensure the networks

[1] https://github.com/SondreElg/thesis

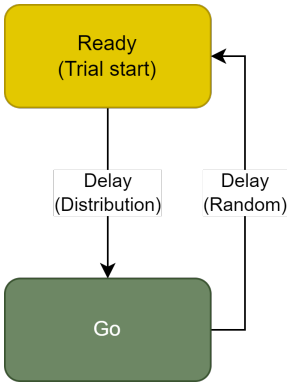


Figure 4.2: Structure of each trial in the Ready-Go task

adapt properly to each block, rather than developing patterns based on the block order. First, the ordering of the blocks are randomized, but never in an order of strictly rising or sinking foreperiod. Then the reverse ordering of the blocks are appended to the blocks, ignoring the last entry. This minimizes the advantage given to certain species and individuals during evolution caused by ordering, and ensures proper adaptation to the problem domain.

Lastly, an omission trial is ran using an updated copy of the network after each block. The goal of these omission trials is to get data on how the network behaves in the absence of a Go signal, providing insight into the inner workings of the network activity. These trials are ran using only a copy of the network to avoid affecting the internal network state and Hebbian weight updates in the event one wants to inspect the first few trials at the start of a block. By keeping the omission trials separate from the other trials, it's fully possible to observe how the network output and Hebbian weights naturally adapt to new foreperiods.

When running these experiments, various parameters took on different values depending on the goal, chosen from a limited list of options. For the networks produced in this thesis, the parameters found in Table 4.2 were used.

Experiment parameter	Values used
Foreperiods	[1, 2, 3, 4, 5]
Block length	50
Trial length	6
Trial delay	[0, 3]

Table 4.2: Parameters used for the experiment

The fitness of each block was determined by taking the mean squared error of the expected and actual output of the network at each timestep. Since only one output during a trial was expected to not be 0, networks could achieve relatively high fitness when only outputting 0. However, this also meant that good network mutations could improve the relative network performance quickly. When determining how to evaluate the fitness of the network as a whole, two main strategies were considered: the minimum block fitness, and the mean of block fitnesses. A key observation for making this choice was that the ordering of foreperiod blocks could affect some networks more than others during evaluation. With this in mind, mean block fitness was chosen to minimize the impact ordering of foreperiod blocks would have on the network fitness and evolution.

Chapter 5

Experiments and Results

Before the model was finished, I performed preliminary tests using NEAT to develop RNNs without Hebbian learning. These networks were tested on a variant of the Ready-Go task where the Go signal followed a random distribution, rather than given at a single point in time. These early results seemed promising, with the network topologies for some different distributions being identical (Appendix A). This suggested that weight changes, such as those caused by Hebbian learning, could allow the evolved networks to learn different distributions with the same topology. Shortly afterwards I implemented Hebbian learning, but saw lackluster results compared to what was expected (Appendix A).

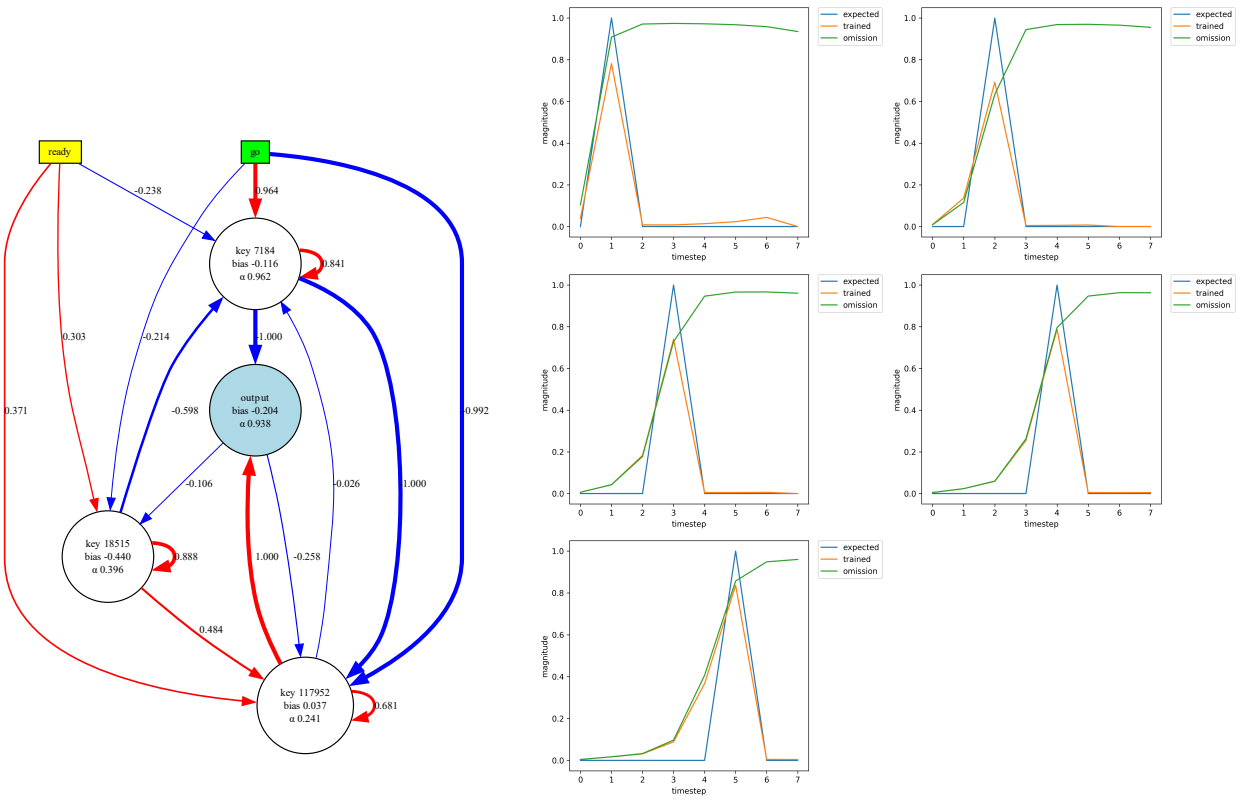
Realizing that the difference between the optimal output for the different distributions may be too small to easily learn, it was at this point that I adjusted the Ready-Go experiment to test simple foreperiods. The idea being that compared to random distributions, the optimal output for different foreperiods would be easier to distinguish both during training and when interpreting the results. Around the same time, I was granted access to the computing servers of the International Research Center for Neurointelligence (IRCN) at the University of Tokyo. With these new resources available, I was able to scale up the experiments, with every run of NEAT from this point onwards running over 2500 generations for a population of 2000 individuals.

Many runs of my model, with various unique combinations of hyperparameters and experiment setups, have yielded a diverse array of network architectures. Having established the foundational parameters and settings for my experiments, the next phase of my research centered on exploring the capabilities of these evolved networks. In the following sections, I will start with presenting a single network, network A, for the purpose of introducing typical performance and output, to get an idea of what these look like. Then, when delving into inspecting Hebbian values and topology, I will expand the selection with two more networks, networks B and C, totalling three networks from three distinct species. The networks were chosen based on a subjective selection process factoring in their fitness, structural complexity, and ensuring all networks run on the same model parameters for the sake of a fair comparison and ease of interpretation.

5.1 Model Performance

In order to start the network analysis, let's take a look at what some typical network outputs look like. To achieve a comprehensive understanding, I will initially focus on the per-trial outputs of a representative network, Network A, before delving deeper into the ouputs of its omission trials. This particular network was selected for its intermediate positioning in terms of structural complexity and fitness among the selected networks. This choice ensures that the results presented in this section reflects the normal characteristics and behaviors of networks developed by the model, providing a balanced perspective on its capabilities and limitations. Nevertheless, the same results as those discussed for network A are made available for networks B and C in Appendix B and C respectively.

From these graphs, we can see that the actual output matches the expected output quite well, as expected given the fitness. However, it's clear to see that leading up to the Go signal, the network deviates somewhat from the expected output in the form of ramp-up behaviour. This behavior, characterized by the gradual increase in output magnitude as the foreperiod extends, diverges from the anticipated pattern based on the hazard function of a prior distribution. Interestingly, the emergence of this behaviour might not be entirely negative, as it could be argued that it's reminiscent of the subjective hazard function caused by temporal uncertainty.



(A) (B)

Figure 5.1: **Network A.** (A) Topology and weights of Network A, with a fitness of 0.980. (B) Network output for each foreperiod trial of network A, in order.

Meanwhile, the omission trial for each of the foreperiods generally follow the same patterns as the network output until the expected Go signal. Following the Go signal, the output keeps increasing until it plateaus near 1 and starts to slowly sink. This is one of two frequently seen pattern of behaviours among evolved networks, with the other pattern being a delayed rapid decrease in output after the expected foreperiod (Appendix B). The emergence of this sort of pattern was usually linked to a strong inhibitory connection directly affected by the activation levels of the output node.

These patterns don't affect the fitness of the network, as the omission trials are performed outside of training and evaluation. However, as research into the hazard function has revealed similar responses as the second patterns in human subjects, whether or not networks display this pattern should be taken into consideration when selecting networks for analysis.

5.2 Interpreting The Hebbian

Having established and explored what typical network output of the evolved networks might look like, it's time to shift focus to one of the most integral components of the thesis, that is, the Hebbian mechanisms within these networks. Exploring several aspects of the Hebbian weights such as magnitude, correlation with the foreperiod and rate of change can provide insight into how the Hebbian weights shape the changes in network output. As such, this section will first explore the Hebbian changes across trials for network A, before establishing the variance of the Hebbian weights in the connections of networks A-C.

We can see from Figure 5.2 that most Hebbian weights settle very quickly at the start of each trial and stay relatively stable near that value. Among the connections in the figure, especially the connection between the Ready input and Node 117952 and the self-excitatory connection of Node 7184 are of particular interest. The connection from Ready to Node 117952 shows a clear increase in weight when the foreperiod is short, causing a more rapid spike in output. Meanwhile, the self-excitatory connection of Node 7184 see the most change over time, and fluctuate wildly at times. This could indicate that the learning rate of the network is too high, as well as that the internal

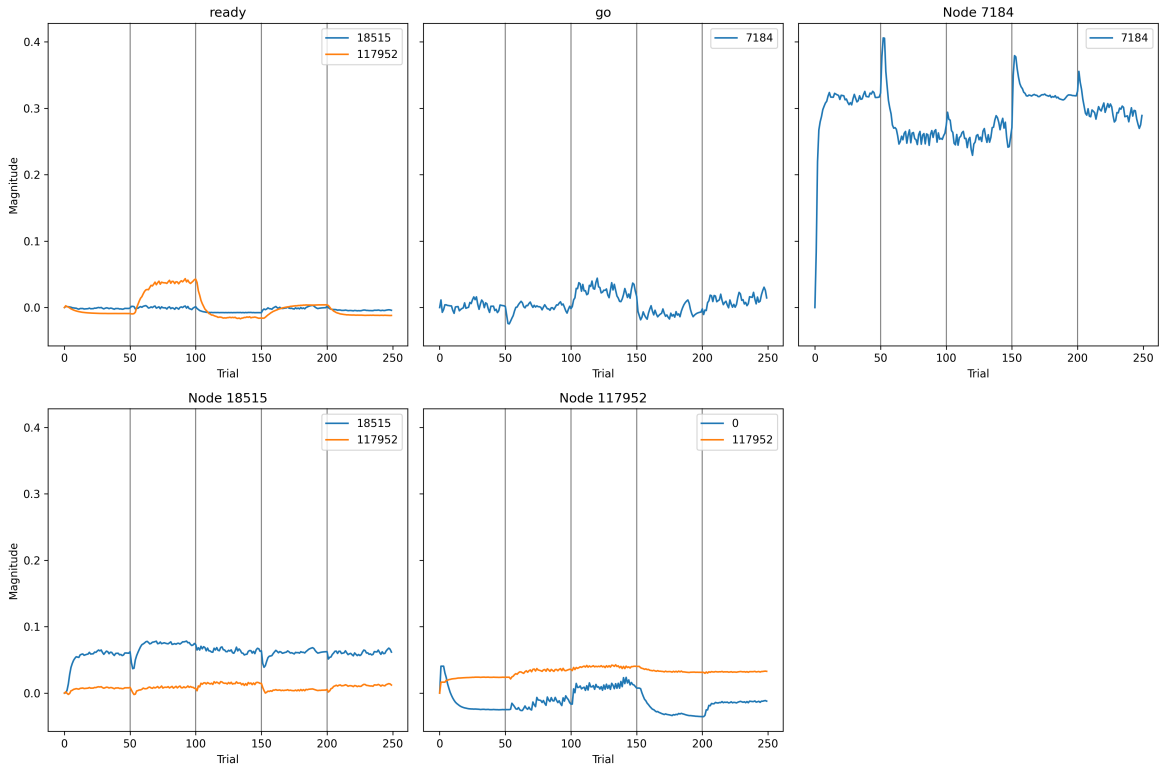


Figure 5.2: Hebbian weight change for network A. Each graph indicates the change in Hebbian weight of the outgoing connections from the node in the graph title to the node indicated by the plot color. Only positive connections are included and values are scaled based on the input node's Hebbian magnitude factor. Vertical bars indicate the start of a new foreperiod block, which are in the order [3, 1, 5, 2, 4].

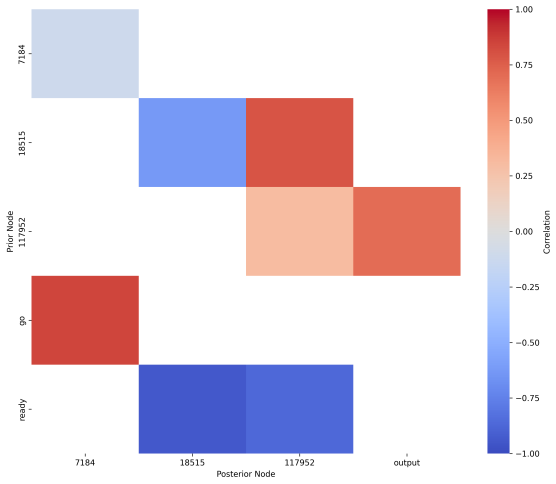


Figure 5.3: **Correlation of Hebbian weights and foreperiod.** Correlation is calculated by taking the average Hebbian weight of the last 40 trials of each block for each connection and comparing those averages to the foreperiod.

dynamics of the network is greatly affected by the random delay between each trial.

However, when inspecting Figure 5.3, the connection between the Ready node and node 1815 stands out as having very high correlation to the foreperiod. This correlation is not immediately clear from Figure 5.2, as the weight changes are very slight. Despite this, the high correlation

might indicate that the very slight changes in weight have a significant effect on the network output. However, we can expand on this with the observation that the Hebbian weight changes are overall very small. Aside from Node 7184's connection maxing out near 0.4 for a brief moment, but most weights max out near 0.1.

Moving on to the correlation plot between the Hebbian weights and foreperiods in Figure 5.3, a number of interesting patterns stand out. First of all, it's clear that the self-excitatory connections have little correlation, a pattern is common among most of the generated networks. Furthermore, the very high correlation of the connection from Go to 117952 reflect the higher need for inhibition after Go for long cycles. This is reinforced by the highly negative correlation from Ready to nodes 18515 and 117952. Node 117952's function is very clear as the sole node with an excitatory connection to the output, as well as an inhibitory connection to node 7186, lessening the inhibition it imparts on the output. Meanwhile, node 18515 acts as a balancer of the activity levels between nodes 7186 and 177952. Considering this, the high negative correlation to foreperiod length makes a lot of sense, as you for longer foreperiods want slower ramp-up behaviour.

Expanding on this, we can see that the overall pattern of correlation favors positive correlation for connections that directly or indirectly excite the output, but favors negative correlation for connections that directly or indirectly inhibit the output. With this, it would seem that Hebbian learning have an overall positive effect on the ability of the networks to perform temporal prediction, as their signs and, to a lesser degree, their magnitude directly correlate to what would be expected from the foreperiod.

Given these observations, looking at a more holistic view of how the Hebbian weights change over time could provide further insight into how they affect the output. At the same time, introducing more networks for comparison can help with interpreting these results, as there are many factors that can affect the need for and impact of Hebbian weight changes in a network. With this in mind, it is time to examine the standard deviation for each connection between all trials for networks A-C, as detailed in Figure 5.4.

Inspecting these figures, it's obvious that the Hebbian weight changes in each of the networks are overall quite small. While it is true that small changes in network weights can cause ripple effects leading to large changes to the output, especially in RNNs, I still found these results curious. Moreover, by performing the same analysis of the correlation heatmaps for network B and C (see Appendix B and C), I noticed that the Hebbian weight changes in network B seemed sporadic, and might even hurt the performance of the network. It was with this in mind that I decided to try running the same networks without any Hebbian learning rules, and observe how they perform.

5.3 Running Without Hebbian Learning

Since the preliminary tests showed that the same topologies could learn different distributions by adjusting the network weights, I had worked with the assumption that Hebbian learning was needed in order for the networks to adapt to different foreperiods. However, in making this assumption, I disregarded RNNs' inherent ability to encode historical information in network activity. By stripping away the Hebbian mechanisms, we can directly observe and assess the inherent capabilities and limitations of the network topologies themselves. This comparison between network A-C and what I will call networks A'-C' will offer a clearer perspective on whether the observed outputs and fitness levels were predominantly a product of the evolved network structures, or a result of the dynamic adaptations afforded by Hebbian learning.

Looking at the plots and fitness results in Figure 5.5, led to a shocking discovery. Not only was the same networks, trained for temporal prediction with Hebbian learning, able to predict the Go signal nearly as well without Hebbian learning. Network A' even outperformed network A by a slight margin, demonstrating that Hebbian learning could some times worsen the results of these networks. There are multiple possible explanations for why this phenomenon occurs.

The Hebbian weights might be too large, creating unstable dynamics within the networks. Although, it's already been established that the weights are relatively small, it should be noted that the only network that performed noticably better with Hebbian learning, network C, also had the smallest max weight for the Hebbian weights (see Appendix C), so there is some credibility to this theory.

Hebbian learning, or this thesis' implementation of it, might be ill suited for the Ready-Go task. As the Hebbian learning rule is based on mutual activity between nodes, with the very sparse input of the task, there simply may not be enough reliable activity that the rule can act on. Even though

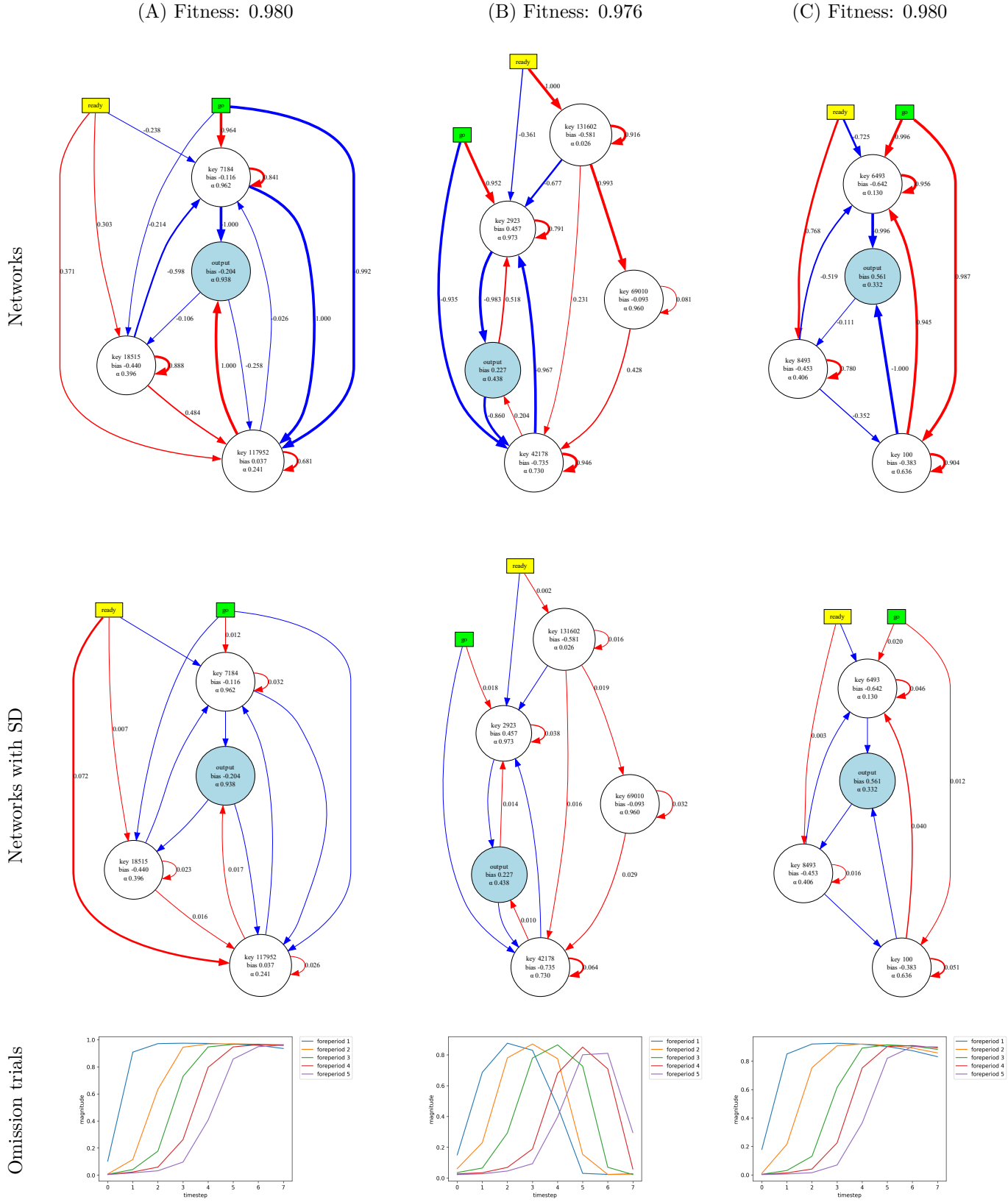


Figure 5.4: **Network A-C weight, Hebbian standard deviation and omission trials.** From top to bottom for A-C: Network with weights, network with standard deviation (SD) of Hebbian weights across all blocks, output of omission trials. When calculating the SD, the weights were divided into 10-trial bins in which the weights were averaged. The first bin of each trial was discarded. This minimizes the impact of intra-trial fluctuations and ensures the SD values focuses on the difference between different foreperiods.

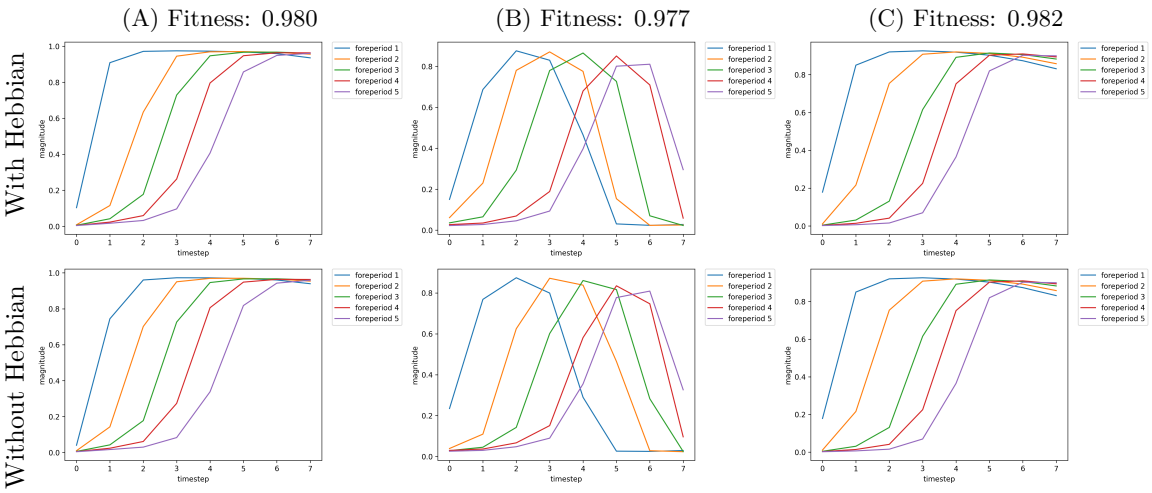


Figure 5.5: Omission trials of networks with and without Hebbian learning

Hebbian learning performs well in many RNN models and in Spiking Neural Networks, commonly used for various temporal tasks, it might be that the combination of using a regular RNN for a temporal task is an ill fit for Hebbian learning.

Nevertheless, this means that even a tiny network of just 3 hidden nodes is able to encode for network history in such a way that it remembers how long to wait between the Ready and Go signals. However, before making this conclusions, one other alternative must be crossed out.

What if the networks don't time their output based on the Ready signal, but rather by waiting for an entire cycle after the Go signal? This sort of behaviour would've easily been possible if each trial stopped after the maximum foreperiod, but that is not the case for given the design of the Ready-Go task. With the random delays introduced at the end of each trial, the output of the networks would've fluctuated a great deal, sometimes having a clear spike before the Go signal. When inspecting the network output for each trial in Figure 5.1, some effect from the variable delay is visible in the discrepancy between the ramping behaviour of the network during the last trial and the omission trial. However, this discrepancy is much smaller than what would be caused by this phenomenon, and it can therefore be ruled out.

With this, it's clear that the network itself is able to encode temporal information within its activity patterns. Even though the Hebbian weights might help fine-tune the connection weights in some cases, the networks are fully capable of learning to predict the Go signal without plastic learning rules. As such, a deeper dive into the topology, as well as the node and connection parameters of these networks is a natural next step.

5.4 Interpreting the Networks

When inspecting these networks, there are a few common points that stand out. First of all, excitatory self-recurrent connections are extremely common, with every single hidden nodes having one across all three networks. Moreover, all but one of these connections have relatively high weight. This could indicate the importance of ramp-up behaviour, where the network builds up its output when nearing the foreperiod. Futhermore, the connections act as a sort of memory structure by keeping a large portion of the last node activity for the next timestep, essentially implementing holdover of previous node activity. When combined with outgoing inhibitory connections, these sort of self-recurrent connections could also be important for balancing the network activity.

Interestingly, the output node has no self-recurrent connection. This is highly surprising, as I had expected to see the output node having an inhibitory self-reccurent connection to help quickly lower output after predicting the Go signal. However, it might be that the output of the output node helps regulate more of the overall network activity, meaning that inhibiting it could cause other parts of the network to fire uncontrollably.

Furthermore, each network has some level of mirroring between the outgoing from the Ready and

Go inputs. At least one node in each network has an excitatory or inhibitory connection from Ready, and the opposite sort of connection from Go as input to it. This is especially apparent in network A, in which every connection from one has a mirrored connection. This makes a lot of sense, as the networks have learned that the Ready signal means it should fire soon, while the Go signal means that it should cease firing, meaning the signals provide the opposite effect from each other.

One more observation relating to the topology itself is that each network has at least one node that doesn't connect to the output directly, but rather seems to modulate the output of the nodes that do. Node 188515 has already been discussed in Section 5.2, and the balancing holds true when running without Hebbian learning. In network B, node 131602 performs this function in the same manner, matching the positive-positive and negative-negative connections from it to the output node through the nodes it modulates, but differs in that there is no direct connection from the Go input. Meanwhile, network C diverges slightly from the pattern set by A and D, by having only inhibitory connections to the output node, which is only excited by its own bias. A modulating node is also present in the network in the form of node 8493, but as there are no excitatory connections to the output, it instead has two negative-negative chains to the output through the nodes it modulates.

Lastly, the node bias is generally quite large in each of the evolved networks. Network A sees the least bias out of them all, but the bias magnitude across the networks overall averages to about 0.4, a significant amount of activity for each timestep. It could be argued that the bias factor reduces biological plausibility and complicates interpretation of the networks. However, by imagining the bias as the result of a weighted connection from an always-on input, which in this case can be interpreted as the brain being in "trial mode", alleviates the first concern. As for the second concern, I did run some simulations with no bias factor to ascertain their efficacy. The produced networks performed as well as those selected, but most were considerably larger, and therefore more difficult to interpret.

In conclusion, these observations of the topology reveal a few core patterns that stand out as being important for the networks to function properly. Self-recurrent connections are important for hidden nodes in order to keep a sort of local history of previous activity. Furthermore, a mirroring between the Ready and Go inputs ensure the network responds properly to each of these signals, and one or more modulatory nodes reinforces this behaviour and scales the network's ramp-up behaviour over time. Lastly, high node bias indicates the importance of an always-on input to the network, which can be likened to the brain keeping different regions active for different tasks.

Chapter 6

Discussion and Future Work

6.1 Discussion

For discussing the results of this thesis, let us return to the research questions posed at the beginning.

RQ1 *What are the strengths of NEAT for the purposes of evolving biologically plausible and interpretable neural networks capable of temporal prediction?*

Since NEAT starts from a minimal structure which is progressively complexified as the population evolves, it is possible to create very compact networks that lend themselves more easily to interpretation compared to networks generated by other techniques. Furthermore, as NEAT evolves both the topology and the weights of the network together, no further training is required to establish good weights after evolution.

Lastly, the speciation schemes utilized in NEAT make it possible to simultaneously evolve a number of different well-performing species. This is a great boon when searching for potential topologies, as a single run of the algorithm can reveal multiple possible solutions. These solutions can then be compared to find commonalities between the networks, aiding the analysis of what makes certain topologies effective.

RQ2 *What are the limitations of NEAT for the purposes of evolving biologically plausible and interpretable neural networks capable of temporal prediction?*

The performance of NEAT heavily depends on the choice of hyperparameters, such as mutation rates and speciation thresholds. Finding the right set of parameters often requires extensive experimentation, and the optimal settings might not be clear, especially when aiming for biological plausibility.

Furthermore, the algorithm being incompatible with the notion of excitatory and inhibitory nodes, as opposed to excitatory and inhibitory connections, introduces extra steps for the interpretation of any evolved network. A proposed interpretation of the networks, in which each node is split into a positive and negative node, retaining the corresponding outgoing connections of the original node was mentioned in the model section. Nevertheless, generating networks in which outgoing node connections are exclusively positive or negative depending on the node could have several benefits. Most notably, it could reduce the complexity of the evolved networks when compared to the interpreted networks of the current implementation. In addition, implementing such a restriction might cause the network to evolve in a more natural manner.

RQ3 *How do RNNs encode prediction over time for temporal prediction tasks?*

It is clear from the results that the evolved networks' structure inherently facilitated the encoding of temporal information, demonstrating RNNs' capability to process and predict temporal sequences. These predictions are made possible through a complex relationship of a number of factors. Mirrored connections from the Ready and Go nodes mark the start and end of trials, and cause large changes to the network activity. A combination of always-on signals in the form of node bias, as well as activity holdover in the form of excitatory self-recurrent connections help keep the network state stable. Meanwhile, one or more modulatory neurons assist with this, while excerting control over the ramp-up behaviour of the networks.

RQ4 *How does changes in Hebbian weights correlate to different foreperiods?*

The investigation into Hebbian weight changes in relation to different foreperiods revealed that these changes were generally very small. Although some Hebbian weights displayed high correlation to the foreperiod, the positive effect granted by these changes are dubious, as Hebbian learning in general provided small but unpredictable changes to fitness. Given the complex nature of RNNs and temporal prediction, linear correlation might not be the best measure of how Hebbian weight changes contribute to adaptation in such domains, a notion that is reinforced by the mixed results of running the networks with Hebbian learning. However, the results highlight that there might be other neuroplastic techniques that are better suited for temporal domains.

Futhermore, although the networks' performance generally didn't change much with or without Hebbian learning, they all utilized Hebbian learning during evolution. As such, it's possible that running NEAT with similar hyperparameters, but without Hebbian learning, could result in networks that perform even better when evaluated without Hebbian learning.

Such an experiment has actually already been made, resulting in a best fitness of 0.986 at the cost of having twice as many hidden nodes, outperforming the best of the networks presented in this thesis. I will not go into a detailed discussion of the network, as that is outside the scope of this thesis. However, it is included for reference as Appendix D for those interested.

6.2 Conclusion

Although there are still many ways to improve the biological plausibility of the model, the experiments conducted in this thesis demonstrate that NEAT can effectively be used to evolve biologically plausible neural networks capable of temporal prediction, which are also relatively interpretable. The ability of these networks to adapt to different foreperiods both with and without minimal Hebbian learning highlights NEAT's efficacy in evolving robust networks where the architecture itself is primarily responsible for learning and prediction in temporal domains, aligning well with the objectives of biologically plausible neurocomputation. This was made possible through a combination of optimizing the hyperparameters of NEAT for creating compact networks, together with making the foreperiods of the Ready-Go task clearly distinct from each other. However, given the restrictions of NEAT, exploring different neuroevolutionary techniques for network generation should be considered in the future.

6.3 Future Work

There are many possible paths this research can take in the future. However, based on the finding in this thesis, there are a few immediate steps that can be taken to better understand how small RNNs perform temporal prediction.

The networks presented in this thesis were all primarily tested on a single problem domain, utilizing the same foreperiods. Experimenting not only with more variations of foreperiod patterns, lengths and spacings, but also more distributions, could grant deeper insight into the adaptability of the networks generated by this model, and how this adaptability is expressed in the topology.

However, to better understand the networks, one of the most significant drawbacks of NEAT for this thesis should be adressed. That being the inability of NEAT to develop networks with signed nodes. Either finding a solution to make NEAT evolve networks with signed nodes, or exploring other neuroevolutionary techniques would be a natural step towards generating more biological plausible and interpretable networks.

Lastly, seeing as the networks were able to perform well even without employing Hebbian Learning, doing more experiments utilizing other potential forms of learning could grant insight into how different neuroplastic techniques affect temporal prediction.

Bibliography

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, *2*(1), 32–48. https://doi.org/10.1523/JNEUROSCI.02-01-00032.1982

Bueti, D., Bahrami, B., Walsh, V., & Rees, G. (2010). Encoding of Temporal Probabilities in the Human Brain. *Journal of Neuroscience*, *30*(12), 4343–4352. https://doi.org/10.1523/JNEUROSCI.2254-09.2010

Caamaño, P., Bellas, F., & Duro, R. J. (2015). T-NEAT: Initial experiments in precise temporal processing through neuroevolution. *Neurocomputing*, *150*, 43–49. https://doi.org/10.1016/j.neucom.2014.04.077

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014, December 11). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv: 1412.3555 [cs]. https://doi.org/10.48550/arXiv.1412.3555

Coull, J. T., Cotti, J., & Vidal, F. (2016). Differential roles for parietal and frontal cortices in fixed versus evolving temporal expectations: Dissociating prior from posterior temporal probabilities with fMRI. *NeuroImage*, *141*, 40–51. https://doi.org/10.1016/j.neuroimage.2016.07.036

Goldberg, D. E., Richardson, J., et al. (1987). Genetic algorithms with sharing for multimodal function optimization. *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, *4149*.

Herbst, S. K., Fiedler, L., & Obleser, J. (2018). Tracking Temporal Hazard in the Human Electroencephalogram Using a Forward Encoding Model. *eNeuro*, *5*(2), ENEURO.0017–18.2018. https://doi.org/10.1523/ENEURO.0017-18.2018

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, *9*, 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

Janssen, P., & Shadlen, M. N. (2005). A representation of the hazard rate of elapsed time in macaque area LIP. *Nature Neuroscience*, *8*(2), 234–241. https://doi.org/10.1038/nn1386

Kristoffersen, M. H., & Elgaaen, S. H. (2022, December 31). *Combining des-hyperneat with gradient descent* [NTNU Project Assignment].

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016, August 29). *Temporal Convolutional Networks: A Unified Approach to Action Segmentation*. arXiv: 1608.08242 [cs]. https://doi.org/10.48550/arXiv.1608.08242

Maes, A., Barahona, M., & Clopath, C. (2020). Learning spatiotemporal signals using a recurrent spiking network that discretizes time. *PLOS Computational Biology*, *16*(1), e1007606. https://doi.org/10.1371/journal.pcbi.1007606

Maes, A., Barahona, M., & Clopath, C. (2021). Learning compositional sequences with multiple time scales through a hierarchical network of spiking neurons. *PLOS Computational Biology*, *17*(3), e1008866. https://doi.org/10.1371/journal.pcbi.1008866

McIntyre, A., Kallada, M., Miguel, C. G., Feher de Silva, C., & Netto, M. L. (n.d.). *neat-python*.

Miconi, T., Rawal, A., Clune, J., & Stanley, K. O. (2018). Backpropamine: Training self-modifying neural networks with differentiable neuromodulated plasticity. Retrieved August 30, 2023, from https://openreview.net/forum?id=r1lrAiA5Ym

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*(3), 267–273. https://doi.org/10.1007/BF00275687

Papavasileiou, E., Cornelis, J., & Jansen, B. (2021). A Systematic Literature Review of the Successors of "NeuroEvolution of Augmenting Topologies". *Evolutionary Computation*, *29*(1), 1–73. https://doi.org/10.1162/evco_a_00282

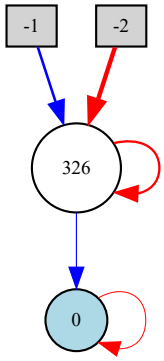
Risi, S., Lehman, J., & Stanley, K. O. (2010). Evolving the placement and density of neurons in the hyperneat substrate. *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 563–570. https://doi.org/10.1145/1830483.1830589

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811

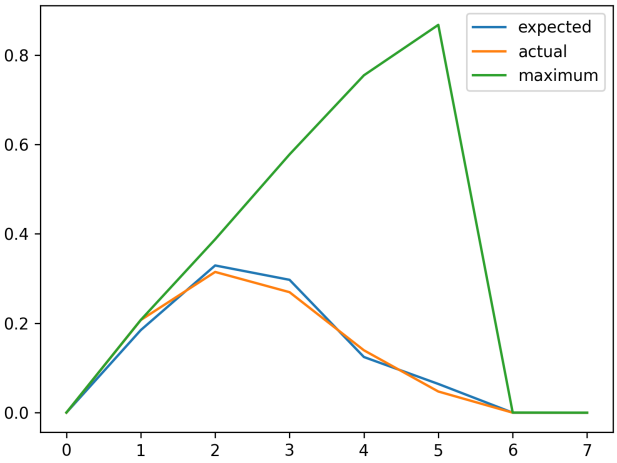
Trujillo, L., Muñoz, L., Naredo, E., & Martínez, Y. (2014). NEAT, There's No Bloat. In M. Nicolau, K. Krawiec, M. I. Heywood, M. Castelli, P. García-Sánchez, J. J. Merelo, V. M. Rivas Santos, & K. Sim (Eds.), *Genetic Programming* (pp. 174–185). Springer. https://doi.org/10.1007/978-3-662-44303-3_15

Appendix A

Results of Preliminary Testing



(A) The nodes labeled -1 and -2 represent the Go and Ready nodes respectively, while the 0 node is the output node. Red connections have positive weight, blue connections have negative weight.



(B) Y-axis represents output magnitude, X-axis represent time steps

Figure A.1: Preliminary network and corresponding network output for a binomial distribution

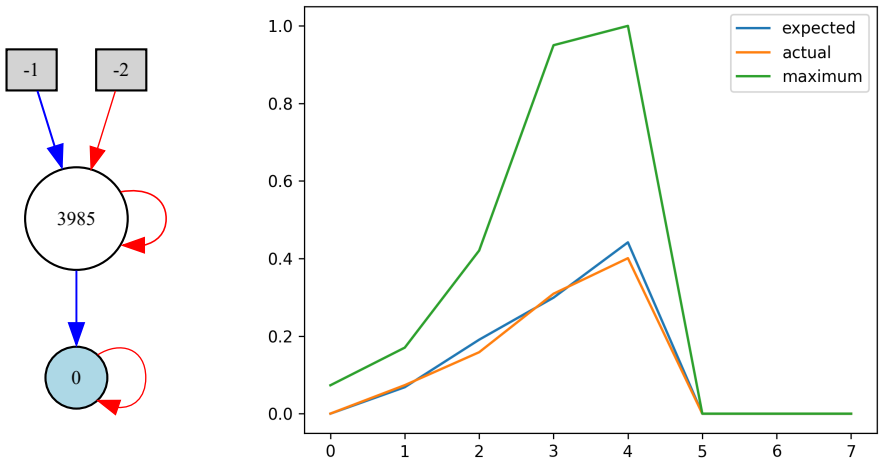


Figure A.2: Preliminary network and corresponding network output for a rising linear distribution

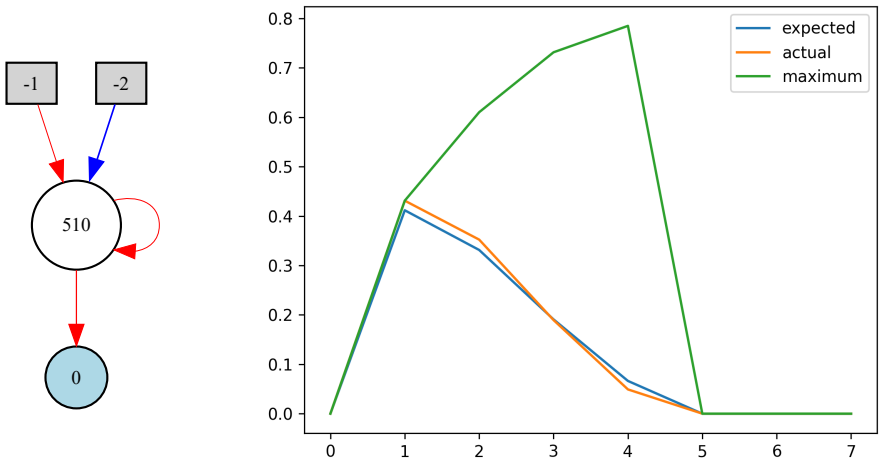
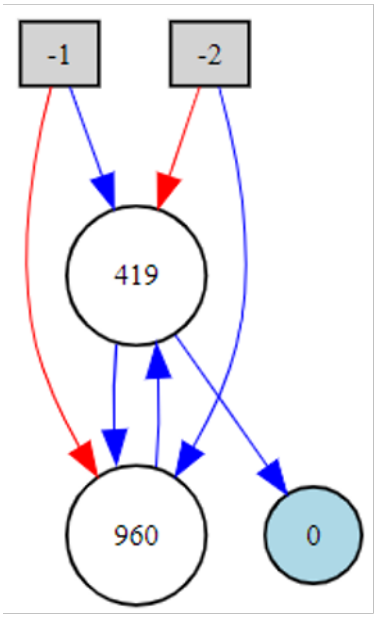


Figure A.3: Preliminary network and corresponding network output for a sinking linear distribution



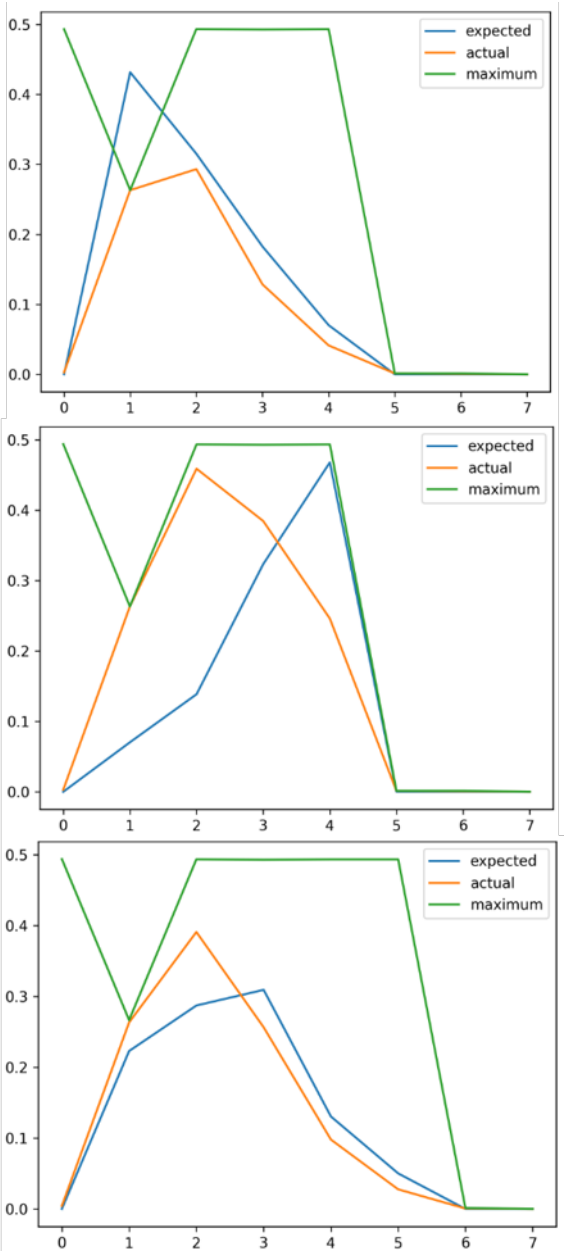
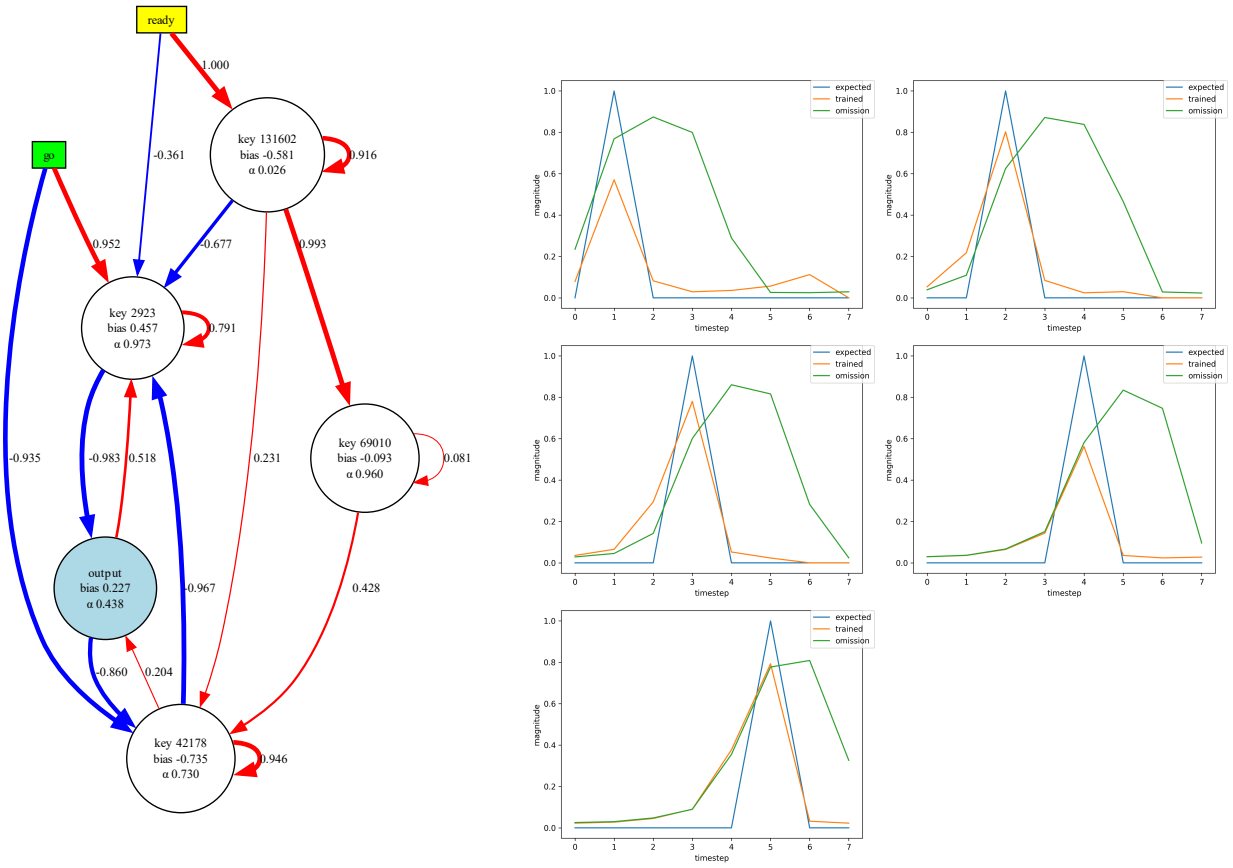


Figure A.4: Priliminary network utilizing Hebbian learning and corresponding network output for three distributions: sinking linear, rising linear, and binomial

Appendix B

Detailed results of Network B



(A) (B)

Figure B.1: **Network B.** (A) Topology and weights of Network B, with a fitness of 0.977. (B) Network output for each foreperiod trial of network B, in order.

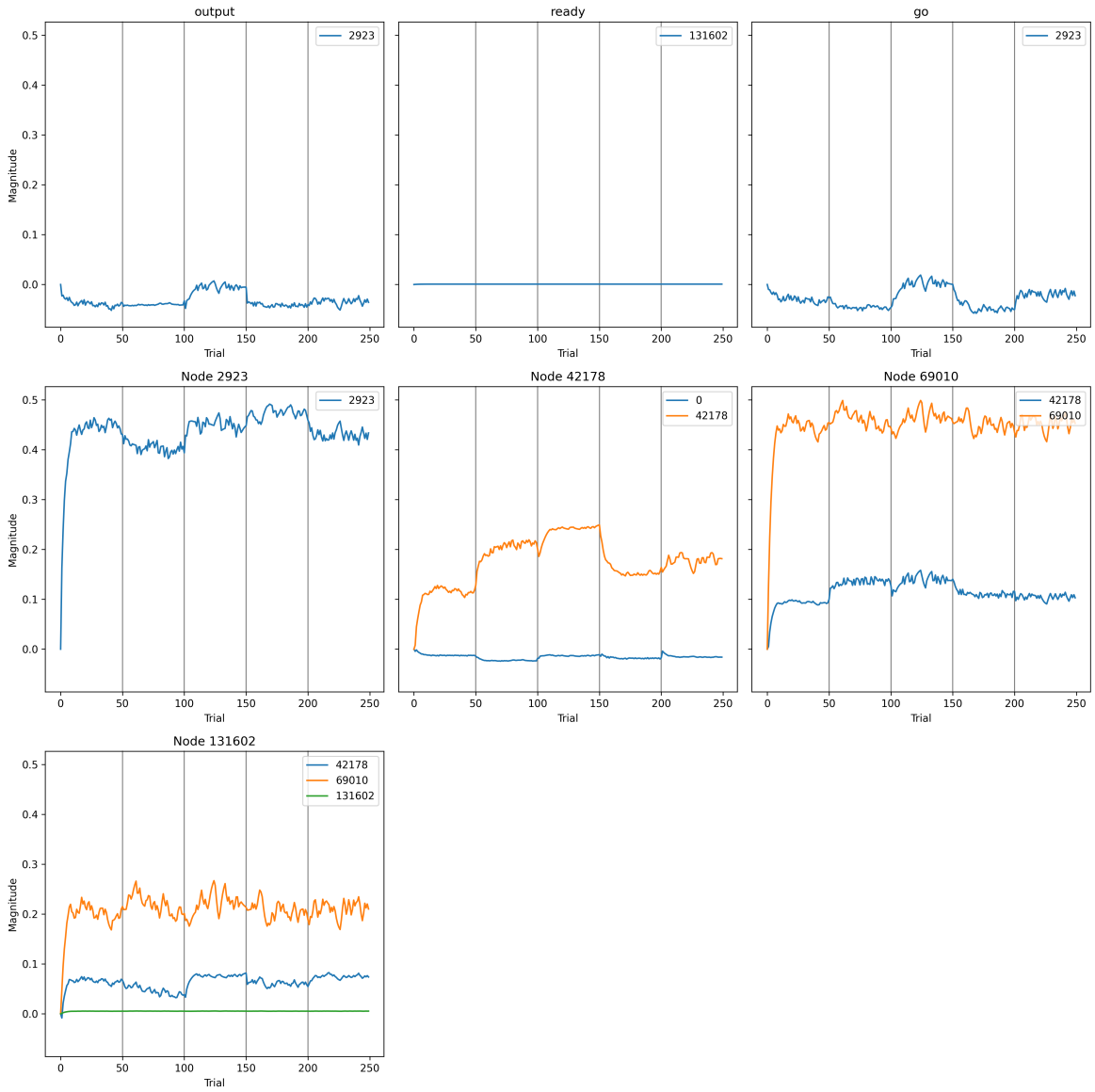


Figure B.2: Hebbian weight change for network B. Each graph indicates the change in Hebbian weight of the outgoing connections from the node in the graph title to the node indicated by the plot color. Only positive connections are included and values are scaled based on the input node's Hebbian magnitude factor. Vertical bars indicate the start of a new foreperiod block, which are in the order [3, 1, 5, 2, 4].

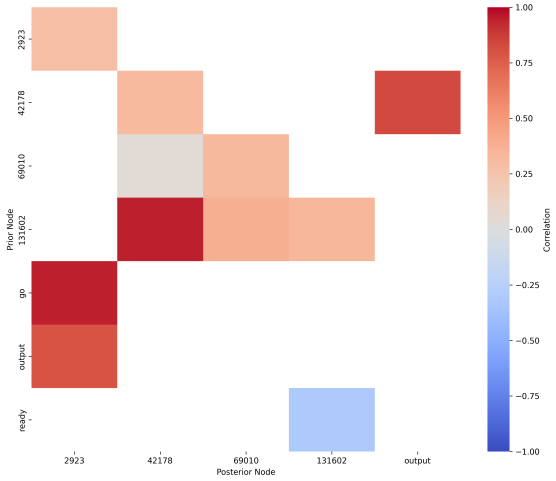
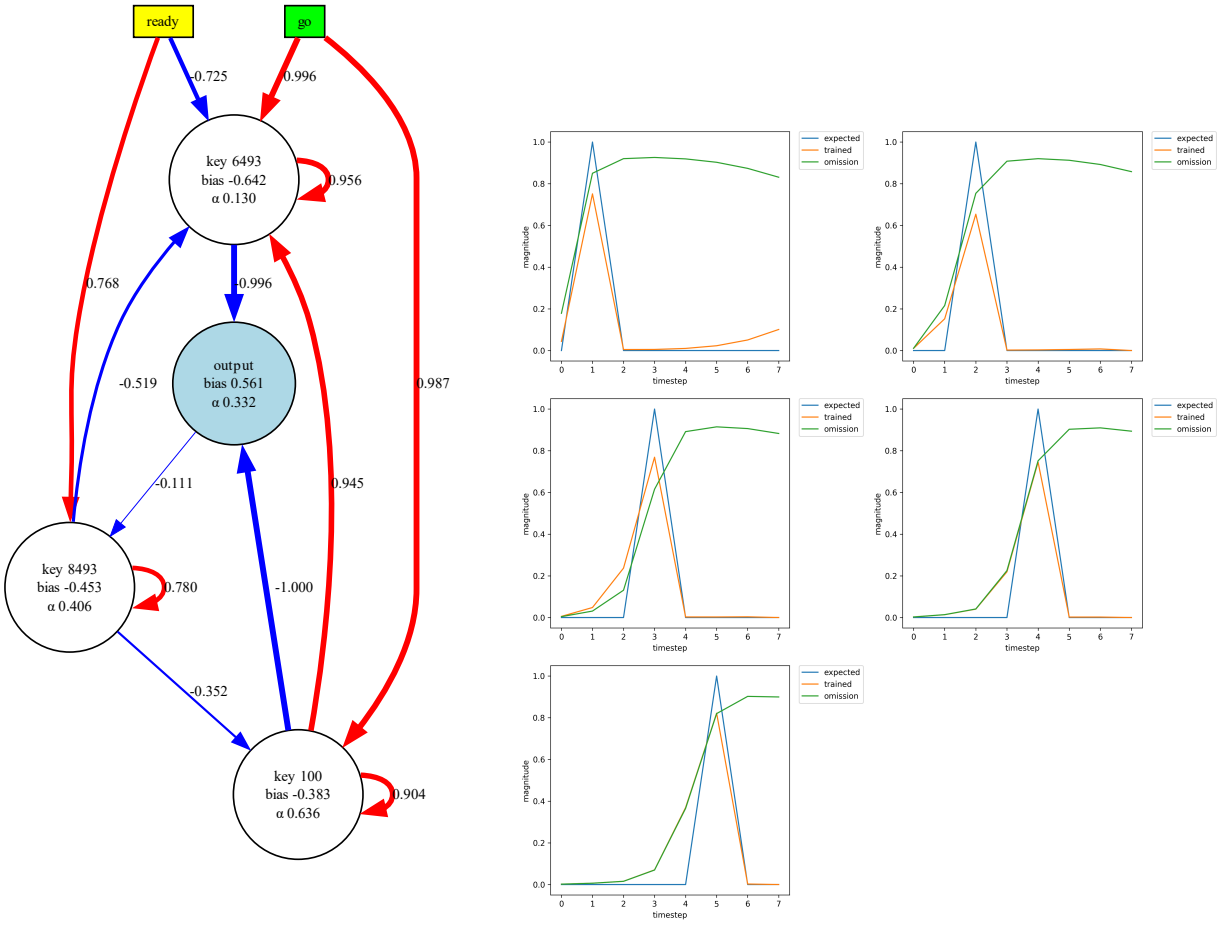


Figure B.3: **Correlation of Hebbian weights and foreperiod.** Correlation is calculated by taking the average Hebbian weight of the last 40 trials of each block for each connection and comparing those averages to the foreperiod.

Appendix C

Detailed results of Network C



(A) (B)

Figure C.1: **Network C.** (A) Topology and weights of Network C, with a fitness of 0.982. (B) Network output for each foreperiod trial of Network C, in order.

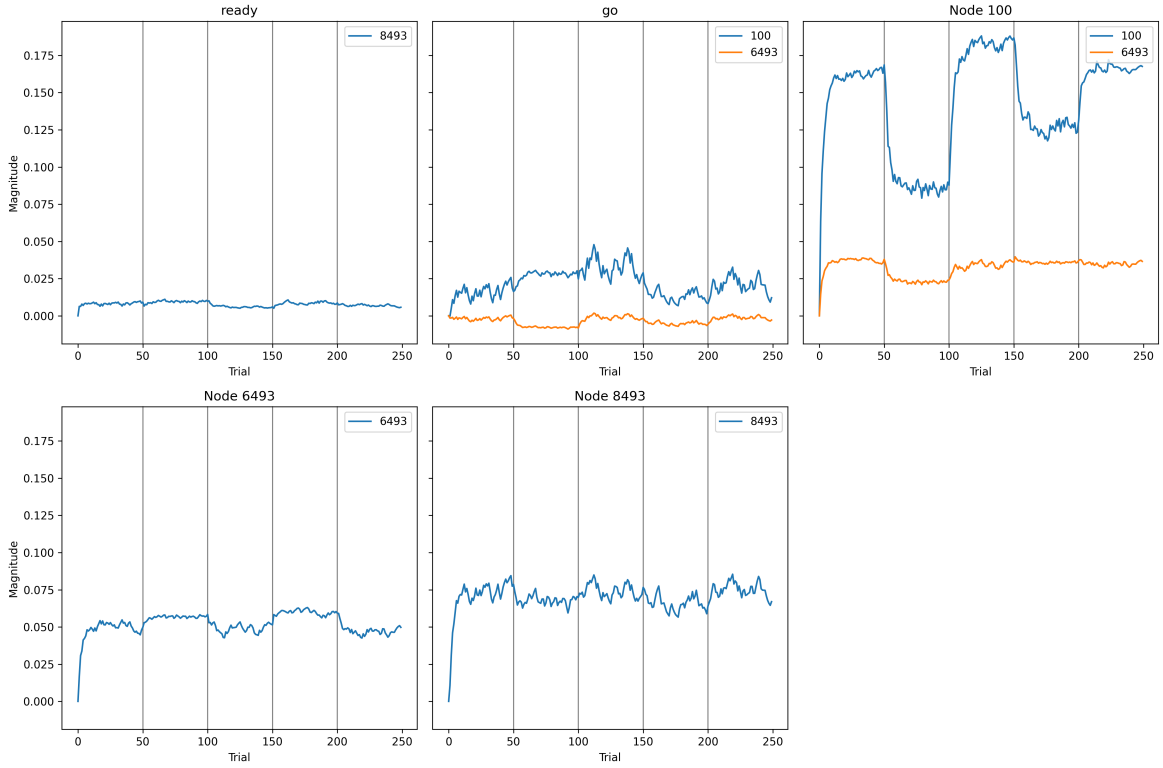


Figure C.2: Hebbian weight change for network C. Each graph indicates the change in Hebbian weight of the outgoing connections from the node in the graph title to the node indicated by the plot color. Only positive connections are included and values are scaled based on the input node's Hebbian magnitude factor. Vertical bars indicate the start of a new foreperiod block, which are in the order [3, 1, 5, 2, 4].

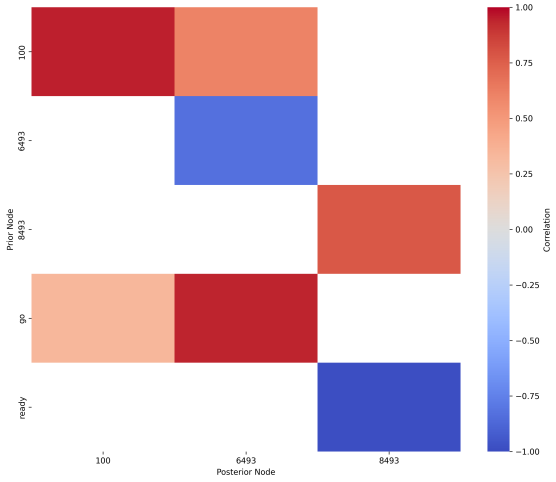
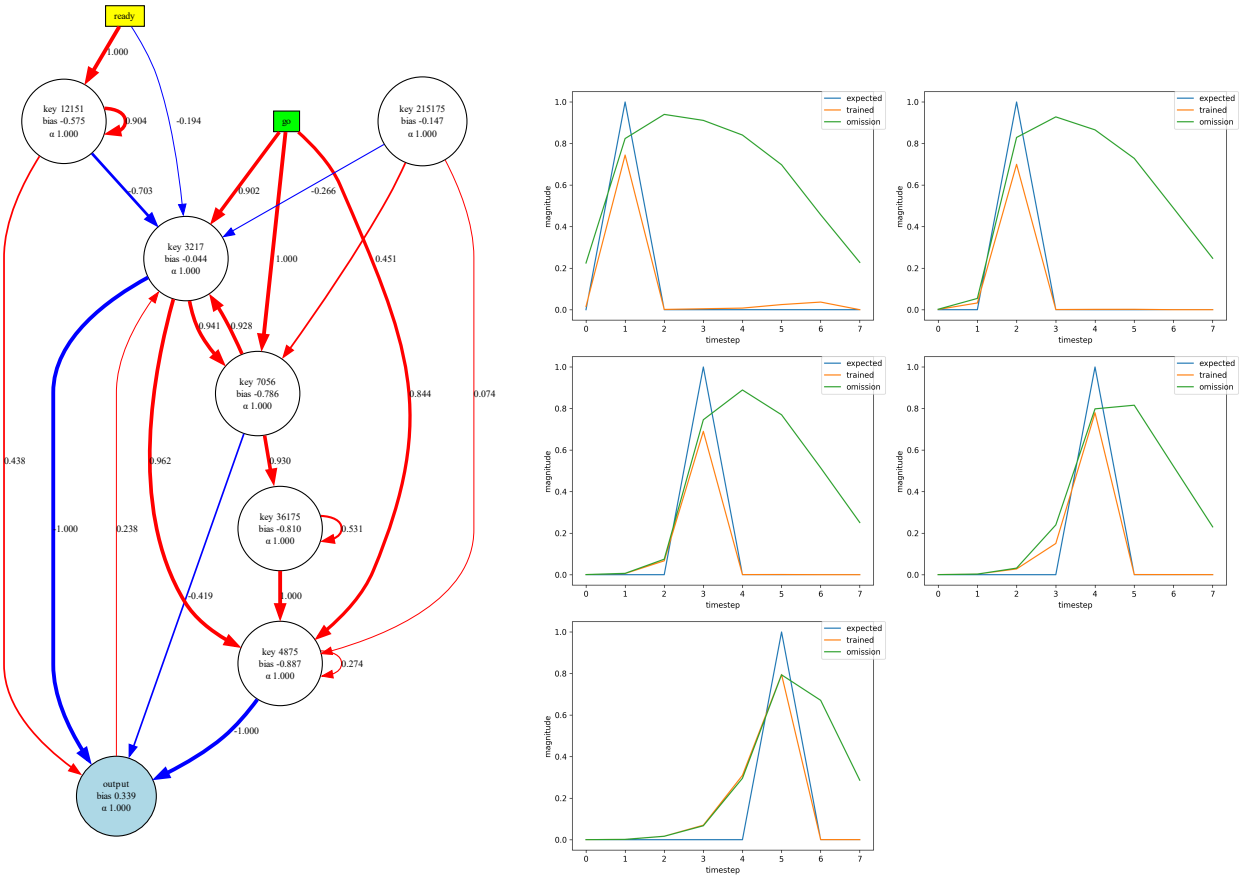


Figure C.3: **Correlation of Hebbian weights and foreperiod.** Correlation is calculated by taking the average Hebbian weight of the last 40 trials of each block for each connection and comparing those averages to the foreperiod.

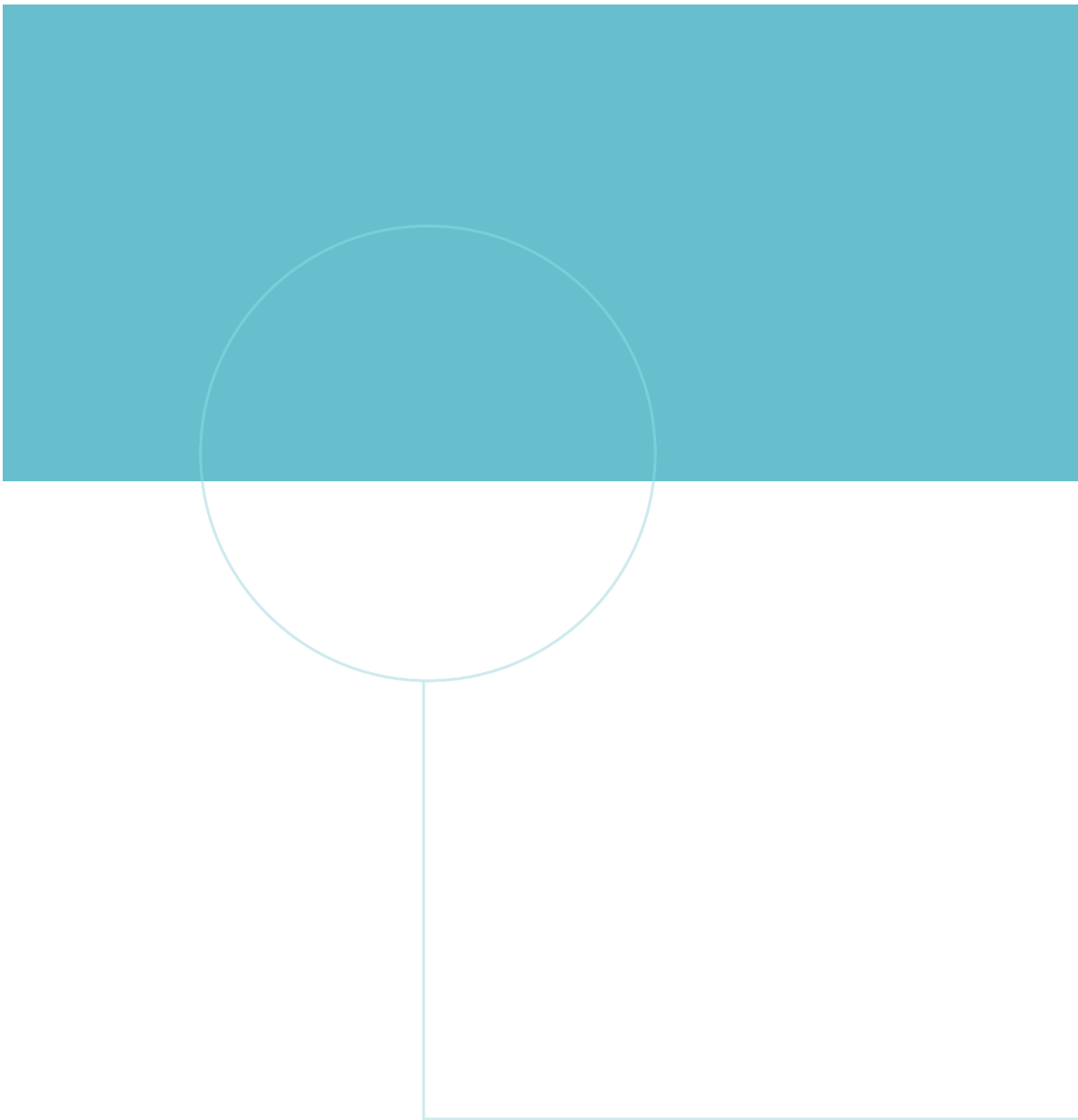
Appendix D

Results of Network D



(A) (B)

Figure D.1: **Network D.** (A) Topology and weights of Network D, with a fitness of 0.986. (B) Network output for each foreperiod trial of Network D, in order.



NTNU
Norwegian University of
Science and Technology

東京大学
THE UNIVERSITY OF TOKYO