Investigating Modularity and Transparency within Bioinspired Connectionist Architectures using Genetic and Epigenetic Models

George Alexander Lacey, BSc

Department of Computer Science and Technology
University of Hull

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science

June 2022

Declaration

Some of the content within this thesis has been published by the author in the following articles:

- Turner, A. P., Lacey, G., Schoene, A., and Dethlefs, N. (2018). Evolutionary constraint in artificial gene regulatory networks. In *UK Workshop on Computational Intelligence*, pages 29–40. Springer
- Lacey, G., Schoene, A., Dethlefs, N., and Turner, A. (2019). Modularity within artificial gene regulatory networks. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 731–738. IEEE
- Lacey, G., Schoene, A., Dethlefs, N., and Turner, A. (2020). Improving the transparency of deep neural networks using artificial epigenetic molecules. In *12th International Joint Conference on Computational Intelligence*, pages 167–175. INSTICC

George Lacey

Abstract

Machine learning algorithms allow computers to deal with incomplete data in tasks such as speech recognition and object detection. Some machine learning algorithms take inspiration from biological systems due to useful properties such as robustness, allowing algorithms to be flexible and domain agnostic. This comes at a cost, resulting in difficulty when one attempts to understand the reasoning behind decisions. This is problematic when such models are applied in real-world situations where accountability, legality, and maintenance are of concern. Artificial gene regulatory networks (AGRNs) are a type of connectionist architecture inspired by gene regulatory mechanisms. AGRNs are of interest within this thesis due to their ability to solve tasks in chaotic dynamical systems despite their relatively small size.

The overarching aim of this work was to investigate the properties of connectionist architectures to improve the transparency of their execution. Initially, the evolutionary process and internal structure of AGRNs were investigated. Following this, the creation of an external control layer used to improve the transparency of execution of an external connectionist architecture was attempted.

When investigating the evolutionary process of AGRNs, pathways were found that when followed, produced more performant networks in a shorter time frame. Evidence that AGRNs are capable of performing well despite internal interference was found when investigating their modularity, where it was also discovered that they do not develop strict modularity consistently. A control layer inspired by epigenetics that selectively deactivates nodes in trained artificial neural networks (ANNs) was developed; the analysis of its behaviour provided an insight into the internal workings of the ANN.

Acknowledgements

First and foremost, I would like to thank Dr Alexander Turner for his unwavering guidance and support throughout the course of this journey. I would also like to thank Dr Nina Dethlefs and Dr Yongqiang Cheng for their support and encouragement; Phininder and Craig for their words of wisdom; Matthew for introducing me to new experiences; and my parents for being there when I needed them most.

Contents

List of Figures

List of Tables

Acronyms

ABN Artificial Biochemical Network.

AEL Artificial Epigenetic Layer.

AEM Artificial Epigenetic Molecule.

AEN Artificial Epigenetic Network.

AGRN Artificial Gene Regulatory Network.

AMN Artificial Metabolic Network.

AMP Adenosine Monophosphate.

ANN Artificial Neural Network.

cAMP Cyclic AMP.

CAP Catabolite Activator Protein.

CNN Convolutional Neural Network.

DNN Deep Neural Network.

GA Genetic Algorithm.

GRN Gene Regulatory Network.

LIME Local Interpretable Model-Agnostic Explanations.

LSTM Long Short-Term Memory.

mRNA Messenger RNA.

RBN	Random Boolean Network.
RNN	Recurrent Neural Network.
XAI	Explainable Artificial Intelligence.

Chapter 1

Introduction

Connectionism is an approach to studying cognition, it involves modelling the processes that occur in the brain as a network of simple, interconnecting processing units [Buckner and Garson, 2019]. The functionality of such networks results from the emergent behaviour of the connections between these units and their parameters. Artificial neural networks are the most pervasive implementation of connectionism; however, models inspired by biochemical phenomena such as gene regulation have been argued as being a type of connectionist architecture [Lones et al., 2013]. This is because artificial biochemical networks are similar to artificial neural networks in that they consist of nodes, the interactions and parameters of which determine the behaviour of the network as a whole. Much of the work in this thesis is based on artificial gene regulatory networks (AGRNs), a type of biochemical network that will be considered as a type of connectionist architecture.

The work described within this thesis is focused on exploring the potential and attempting to improve the transparency of connectionist architectures and has been split into three research questions. Answering the first question involves an investigation into how the evolution of an AGRN by a genetic algorithm may be constrained, and how knowledge of this can be used to improve the speed and performance of such evolution. Answering the second question involves an experiment that has been conducted to determine the degree to which AGRNs evolve to be modular in structure; this is of interest as the analysis of AGRNs that evolve to be modular in structure could provide insights into their optimisation. The modularity that is shown to occur in the second experiment motivated the application of a control layer inspired by epigenetics to an artificial neural network to improve the transparency of execution;

the experiments conducted on the epigenetic control layer constitute the answer to the third question.

Concepts relevant to the thesis will be introduced in this chapter, followed by the research aims and objectives. An outline of the thesis structure concludes this chapter.

1.1 Artificial Gene Regulatory Networks

Gene regulation is modelled usually for one of two reasons; the first is to gain a better understanding of how biological gene regulation operates [Kauffman, 1969, Lai et al., 2016], the second is to develop computational models that attempt to capture the useful properties found within biological gene regulation, such as robustness and adaptability [Taylor, 2004, Lones et al., 2010, Turner et al., 2013b]. The work described within this thesis is concerned with the latter.

Artificial gene regulatory networks (AGRNs) are computational models with a connectionist architecture; they differ from conventional artificial neural networks as they do not consist of layers, so the connectivity between nodes is more flexible during training. They differ also in terms of their regulatory function, which takes the place of the activation function found within the nodes of an ANN; the regulatory functions of an AGRN are parameterisable, allowing for varying behaviour across nodes. In derivative cases such as the artificial epigenetic network, connectivity may change dynamically during execution [Turner et al., 2013b]. AGRNs are typically trained by an evolutionary algorithm such as a genetic algorithm. The AGRNs used within this thesis operate in discrete time, and at each time step the input is read by the network and processed by its nodes for an output to be produced, which is used to enact change in an external system. The experiments in Chapters 7 and 8 involve AGRNs extensively, as their developmental and structural properties are investigated. Chapter 4 provides an overview of AGRNs.

1.2 Deep Learning

Deep learning is a type of machine learning that is capable of representation learning, based on the processes that occur in the brain that allow higher order biological organisms to learn. Rather than requiring input data to be specifically engineered, representation learning allows deep learning algorithms to automatically detect features in data, which allows the recognition of high-level features through multiple layers of abstraction.

Deep neural networks (DNNs) are an implementation of deep learning that consist of an artificial neural network with more than one hidden layer. With regard to their architecture, DNNs are comprised of multiple layers of artificial neurons, which are simple processing units that sum their inputs and process them using an activation function. There exist different types of DNNs, basic feedforward DNNs are often used to make predictions on numerical data; convolutional neural networks (CNNs) are often used for image detection and classification; recurrent neural networks (RNNs) are often used in the domain of speech recognition and natural language processing. The epigenetic control layer in Chapter 9 is applied to an external DNN in an attempt to improve the transparency of its execution. Section 3.5 provides a background in deep learning, as well as variations of DNNs and their uses.

1.3 Explainable Artificial Intelligence

Deep neural networks (DNNs) can be applied across many domains due to their abstract architecture, their functionality arises from the connectivity between nodes and their parameters. If one were to look at the parameters of a neural network, they would see matrices of real numbers, the relevance of which is not immediately apparent nor human readable. In order to calculate a forward pass of a DNN to determine its output, many calculations must be performed in parallel. Both of these factors have led to DNNs being labelled as 'black-box' systems as they are difficult to interpret.

The field of explainable artificial intelligence (XAI) attempts to address the issues faced when seeking answers pertaining to the decision making process behind AI systems. Artificial intelligence has applications in domains such as healthcare, climate prediction and self driving cars, where it is used to make decisions. Some implementations of AI may be inherently explainable in their design; however, for DNNs this is not the case. The output of a DNN can be easily determined, but the reasoning behind it cannot. This is problematic as the decision can have real world implications. Explainability is important in ensuring that decisions are being made for the correct reasons, and is invaluable when attempting to improve the performance of it. It is also useful when determining who is liable for errors that may have been caused by an AI system. AI has the potential to learn previously undiscovered trends in data, a degree of explainability could expose these trends and improve the understanding of the domain. The motivation behind the work described within this thesis is based on the improvement of explainability in connectionist architectures such as the AGRN and DNNs. The modularity of evolved AGRNs is investigated in Chapter 8 due to the potential for analysing networks at this level to provide a better insight into network execution. An epigenetic control layer is applied to a trained external connectionist architecture in Chapter 9 with the intention of improving its transparency of execution.

1.4 Evolutionary Algorithms

Evolutionary algorithms are a type of search heuristic that are used to 'evolve' a solution to an optimisation problem. They generally accomplish this by performing operations inspired by natural selection. Within this thesis, genetic algorithms will be used to evolve AGRNs and epigenetic control layers. Evolutionary algorithms are better suited than methods such as backpropagation at training AGRNs, as it is typical for the topology of an AGRN to change during the training phase. Also, the activation function of the nodes in an AGRN are parameterised and may change during training, introducing further complications. Some of the models in this thesis are trained using evolutionary algorithms. The AGRNs in Chapter 8 are evolved using a genetic algorithm variant, and the epigenetic control layer in Chapter 9 is evolved using a genetic algorithm. Chapter 5 describes evolutionary algorithms within the broader scope of optimisation.

Genetic algorithms evolve a population of members through a sequence of generations. Each member consists of a set of parameters that act as a potential solution to the problem, a fitness function measures how well a particular member performs relative to the others. The members of each generation are subject to a set of genetic operators with the intention of retaining useful traits that will persist into future generations. Recombination is a genetic operator that combines the traits of two parent members to produce two new 'children' members. Mutation is a simple genetic operator which makes (usually minor) random changes to members in the population with the intention of introducing new traits.

1.5 Hypothesis

The work described within this thesis has been primarily motivated by the importance of transparency when AI systems are used in the real world, particularly with regard to understanding the decision making process. The hypothesis is based on the following assertions:

- Artificial gene regulatory networks have been argued to be a type of connectionist architecture. Like artificial neural networks, their functionality arises from the interactions between their constituent nodes.
- Modularity is a characteristic found within the genomes of biological organisms, allowing for mutations to occur that do not disturb the whole system; additionally, epigenetics influences the transcription of an organism's DNA in response to changes to its internal and external environment, allowing for adaptation without changes to its DNA sequence. The robustness of organisms during evolution can be partially attributed to these properties.
- Control layers inspired by epigenetics have been developed as an extension of AGRNs, resulting in more performant networks. Additionally, analysis of the control layer behaviour has provided a degree of transparency of the underlying AGRN, providing a real world understanding of why certain decisions have been made.

Based on the assertions above, it is hypothesised that:

- There are predictable evolutionary and structural properties of artificial gene regulatory networks that can be exploited to improve the transparency of execution and produce more performant networks.
- A dynamic control layer inspired by epigenetics can be applied to a connectionist architecture, which can be analysed to provide a degree of transparency to the connectionist architecture in the form of a description of its internal workings.

1.6 Research Aims

The work described within this thesis is focussed on exploring the potential and attempting to improve the transparency of connectionist architectures. Three questions have been devised, from which the research aims will be derived and stated in this section.

> **Is it possible to constrain the evolution of an artificial gene regulatory network in order to produce a more performant network in less time?**

It is thought that the evolution of biological organisms is subject to constraint, which allows for rapid adaptation to their environment. To answer this question, the properties of AGRNs will be explored to determine whether there are predictable evolutionary pathways that can be exploited to produce better networks in a shorter time frame, or in other words, whether biological evolutionary constraint can be replicated in AGRNs. The ability to constrain AGRN evolution is of interest and has implications beyond producing better networks, as it is likely to indicate the importance of AGRN properties such as node connectivity, and how this affects the overall functionality of the network. Such developmental properties could be taken into account when developing techniques to obtain reasoning behind network behaviour.

> **To what end do artificial gene regulatory networks evolve to have a modular structure?**

Modularity is a property found within the genome of prokaryotic organisms in the form of operons, where functionally related genes are located together. To answer this question, the internal structure of evolved AGRNs will be investigated in order to determine whether they also exhibit modularity, and if so, whether this occurs consistently. The modularity of evolved AGRNs is of interest as it could provide insights into how AGRNs are operating on a higher level than observing individual nodes. This has the potential to improve the overall transparency of execution of AGRNs, as it may be possible to associate groups of nodes with specific network functionality. This would be of use as it could provide explanations of why a network is behaving in a certain way, or could be taken into account during the evolutionary process in order to produce better networks.

> **Is it possible to develop a control layer inspired by epigenetics to act externally on a trained connectionist architecture that when analysed, can provide information that may be used to improve the transparency of its execution?**

Control layers inspired by biological epigenetics have been incorporated into AGRNs and have been shown to capable of self-organisation, as well as providing a degree of transparency. To answer this question, a control layer inspired by epigenetics will be developed and applied to a trained external connectionist architecture in an attempt to improve the transparency of its execution. Artificial epigenetic networks have shown that an epigenetic control layer is capable of being analysed to improve the transparency of network execution [Turner and Dethlefs, 2017]; the purpose of this work is to ascertain whether it is possible to apply the self-organisational properties and robustness of an epigenetic control layer to an external connectionist architecture that has been trained separately. Improving the transparency of separately trained connectionist architectures is of interest as it would not require the AI system to be re-engineered or redeployed.

1.7 Research Objectives

The research objectives will be discussed in terms of scientific and technical contributions, each corresponding to one of the questions as previously stated. The work conducted to answer the research questions has been split into three chapters (7–9). Chapters 7 and 8 constitute an investigation into the developmental and structural properties of AGRNs. In Chapter 9, an external 'epigenetic layer' is applied to a trained deep neural network in order to improve its transparency by selectively and dynamically deactivating nodes from a hidden layer. The contributions made within this thesis have been summarised below, followed by an elaboration of contributions based on each of the research questions.

Scientific contributions:

Contribution 1 - Identification of AGRN evolutionary pathways that result in more performant networks in a faster time frame

Contribution 2 - Demonstration that AGRNs are capable of solving multiple independent sub-tasks concurrently

Contribution 3 - Demonstration of epigenetic control layer acting on a deep neural network

Contribution 4 - Analysis of deep neural network based on epigenetic control layer

Technical contributions:

Contribution 5 - Method to derive and apply evolutionary constraint rules on AGRNs

Contribution 6 - Development of AGRN with two-dimensional spatial representation

Contribution 7 - Method for analysing node interactions based on sub-tasks

Contribution 8 - Development of 'artificial epigenetic layer (AEL)' consisting of epigenetic molecules capable of acting on an external connectionist architecture

1.7.1 Understanding Evolutionary Constraint in Artificial Gene Regulatory Networks

Evolutionary constraints is a concept in biology that refers to the restrictions and biases that apply during the evolution of a biological organism [Hansen, 2015]. Within the genome of an organism, there exist genes that code for functions that are crucial for its survival. Redundancy has been shown to occur with such genes in the form of gene duplication [Li et al., 2010], so that disruption to these genes is less likely to be detrimental to survival. On the contrary, sections of DNA exist that are unlikely to result in significant changes to an organism. An 'evolutionary sweet spot' exists between these instances where mutations can cause beneficial changes to an organism without being detrimental to its survivability. It is likely that the evolution is constrained in such a way that mutations occur to parts of the genome that result in meaningful changes [Wollenberg Valero, 2020].

In Chapter 7, the evolutionary process of AGRNs will be investigated in order to determine whether it is possible to produce more performant networks in a shorter time frame by constraining it, similarly to how evolution is constrained in biology. The relationship between gene attributes and the performance impact of gene mutation will be observed, with the intention of discovering evolutionary pathways that produce better networks when followed (Contribution 1). Based on this, a set of rules will be devised with the intention of increasing the probability of mutations that result in desirable changes to the network. These rules will be used to guide the evolutionary process of another set of AGRNs (Contribution 5), so that impact on performance can be observed and it can be determined whether the application of such rules is beneficial.

1.7.2 Developing Modularity Within Artificial Gene Regulatory Networks

The transparency of AGRN execution has been researched regarding individual node behaviour and interactions between these nodes [Turner and Dethlefs, 2017]. The functionality of an AGRN is a result of the emergent behaviour of the interacting nodes, and it has been shown that AGRNs are capable of self-dividing their nodes into sub-networks of functionality.

In Chapter 8, an AGRN will be developed that is represented in two-dimensional space (Contribution 6); this will increase the flexibility of node connectivity over previous one-dimensional AGRNs as described in [Turner et al., 2013b]. The AGRNs will be designed to allow them to solve multiple independent tasks simultaneously (Contribution 2) to encourage the nodes to split into multiple sub-networks. The solving of multiple independent tasks will allow each node within the network to be associated with a particular task. When the network has solved each of the tasks to a satisfactory level, the structure of the network will be analysed to determine the degree to which the nodes have arranged themselves into distinct sub-networks corresponding with each task (Contribution 7), and whether they do so consistently and definitively.

1.7.3 Improving the Transparency of Deep Neural Networks Using an Artificial Epigenetic Layer

Applying a control layer inspired by biological epigenetics has been shown to improve the performance and transparency of AGRNs. The 'black-box' problem describes, in the context of artificial neural networks, that while the input and output of such networks can easily be determined, the internal processing that occurs cannot. This causes problems when trying to determine why a neural network made a particular decision.

In Chapter 9, a control layer inspired by epigenetics will be developed that is capable of selectively disabling parts of an external connectionist architecture (Contribution 8). This control layer will be applied to deep neural networks trained on different datasets, and will consist of multiple epigenetic molecules that selectively disable nodes from one hidden layer of the network based on the output of the preceding layer (Contribution 3). This will allow nodes that are not useful to be purged, as well as allowing nodes to be selectively switched based on the state of the network. The switching behaviour of the control layer will then be analysed in an attempt to correlate nodes within the ANN with certain classifications (Contribution 4).

1.8 Thesis Structure

This thesis is broken down into three sections in addition to this introduction, the sections are broken down further into chapters, summarised by Figure 1.1.

The background material and supporting literature constitute the first section. The first two background chapters provide a background in the fields that form the basis of the computational models used within the thesis. The computational models primarily used within this thesis take inspiration from the field of genetics, the relevant mechanisms and concepts of which are described in Chapter 2. The models used within this thesis are considered connectionist architectures; an introduction to this field and justifications for this classification are provided in Chapter 3.

Experimentation extensively involves artificial gene regulatory networks (AGRNs) and their derivatives, Chapter 4 provides a background to these concepts. The computational models described in Chapter 4 are trained using optimisation algorithms, the selections of which are discussed in Chapter 5. Explainable AI (XAI) is a field that has motivated much of the work in this thesis, an introduction to this field is provided in Chapter 6.

The second section consists of the contributions made in the thesis, broken down into three chapters, each chapter describes a separate experiment. Chapter 7 describes the experiments relating to evolutionary constraint in AGRNs, Chapter 8 describes the experiments related to the modularity of AGRNs, and Chapter 9 describes the epigenetic control layer experiments.

Conclusions of the experiment chapters are drawn in the final section (Chapter 10), and future work is discussed.

Chapter 2 The concepts and mechanisms of biological genetics and epigenetics relevant to the thesis are introduced

Chapter 3 Detailed explanations of connectionist architecture implementations are provided in the context of Deep Learning and AGRNs

Chapter 4 The development of the artificial gene regulatory network (AGRN) and it's purposes are described

Chapter 5 Optimisation techniques used to train AGRNs are described

Chapter 6 The importance of explainable artificial intelligence, and current attempts at achieving it are explored

Chapter 7 The methodology and results of experiments conducted on an AGRN in order to determine the degree of evolutionary constraint are described

Chapter 8 The methodology and results of experiments conducted on an AGRN in order to establish the degree to which it is modular are described

Chapter 9 The methodology and results of experiments conducted on deep learning models in an attempt to improve their transparency using AELs are described

Chapter 10 Conclusions are drawn based on the contributions made and their relevance to the overarching aim of the thesis, the potential for further work is also explored

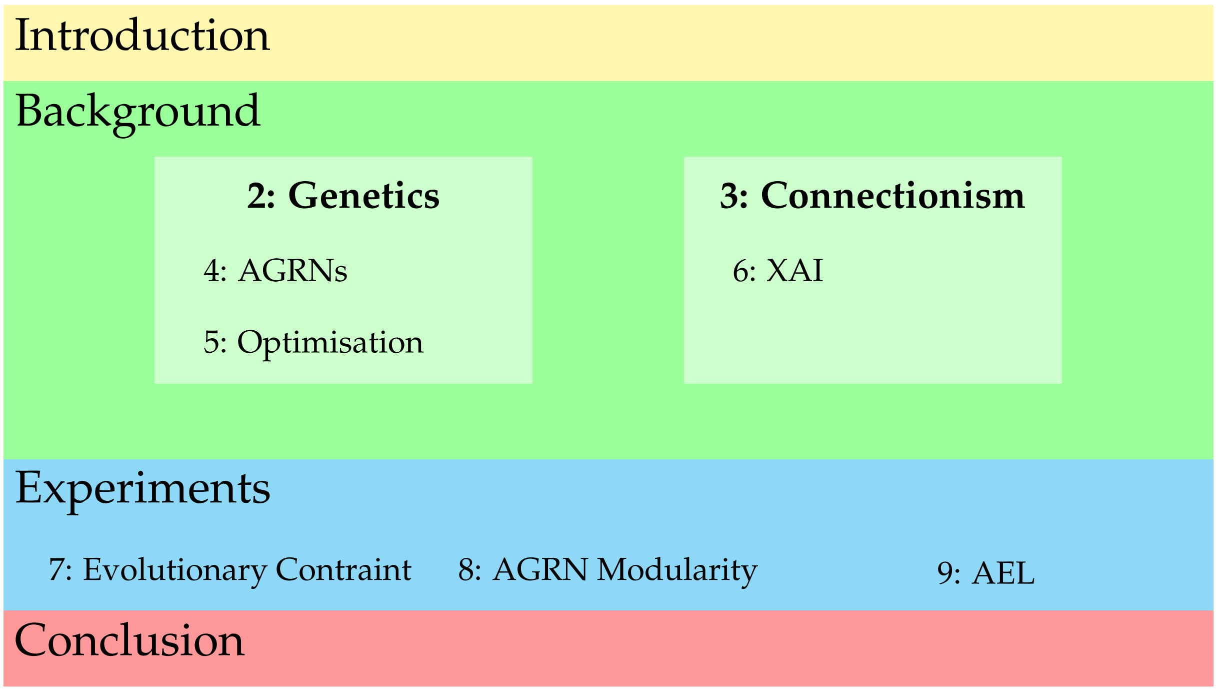


Figure 1.1: Chapter overview of thesis

Chapter 2

Genetics and Epigenetics

The computational models described within this thesis (see Chapter 4) are based on biological gene regulation, and take inspiration from different genetic and epigenetic phenomena. This chapter begins with a background in gene expression, as the processes and structures are fundamental to gene regulation. Gene regulation will then be explained separately for prokaryotic and eukaryotic organisms, as the computational models later described are based on regulatory mechanisms present in such organisms. Eukaryotic gene regulation involves epigenetic processes, which will also be described.

Genetics is the name given to the science of heredity [Griffiths et al., 2000], the process that causes traits to be passed from a parent organism to its offspring during reproduction. These traits result from the genetic code contained within sections of DNA known as genes in a process called gene expression, where gene products such as proteins are produced. The central dogma of molecular biology describes the flow of information that occurs during gene expression involving DNA, RNA, and the resulting protein [Crick, 1970], based on work by the same author on protein synthesis [Crick, 1958]. Gene regulation is a biological process that controls the expression of genes, and is necessary so that gene products are only produced when they are needed. Gene regulation is responsible for the variation of characteristics found within the different cells constituting higher order organisms [Latchman, 2005]. Epigenetics refers to heritable changes to the phenotype that cannot be explained by changes in the DNA sequence [Riggs et al., 1996].

2.1 Nucleic acids

Nucleic acids are a type of biological molecule that are involved in protein synthesis; they are large polymeric structures consisting of a sequence of monomeric nucleotides, which contain the base. In this section the structure and functionality of DNA will be described, followed by RNA.

2.1.1 DNA

Deoxyribonucleic acid (DNA) contains the genetic code that is essential for life [Hartwell et al., 2008]. All known cellular organisms contain DNA, which is used as instructions for the synthesis of proteins [Clancy, 2008a], a class of molecules that are responsible for most biological functionality. In eukaryotes, DNA can be found within the nucleus of most cells, an exception of this is the erythrocyte (red blood cell), which contains no nucleus or DNA. DNA is a polymer — a chain molecule consisting of smaller molecular units called monomers, in the case of DNA the monomers are nucleotides [Pray, 2008a], the sequence of which acts as instructions for synthesising proteins [Brown, 1998]. The complete set of an organisms DNA is referred to as its genome.

The structure of DNA consists of two complementary polynucleotide strands, which wrap around each other forming a double helix structure [Watson and Crick, 1953]. The polynucleotide structure of DNA strands, including the order of the sugar-phosphate backbone was first suggested in [Levene, 1919]. Each strand consists of a sequence of nucleotides, forming a polynucleotide chain; each nucleotide consists of a sugar, a nitrogenous base, and phosphate [Pray, 2008a]. The sugar and phosphate form the structure of the DNA, forming a sugar phosphate backbone which connects the nucleotides to form the chain. The genetic information is represented by the sequence of bases in the chain. Four different nucleotides are present in DNA differing only by the base, which may be Adenine, Thymine, Guanine, or Cytosine [Brown, 1998]. Adenine forms a pair with Thymine, and Guanine with Cytosine. The complementary base pairing of bases is shown by Figure 2.1. The two DNA strands are joined by the hydrogen bonds formed between the complementary base pairs, this is referred to as 'annealing'.

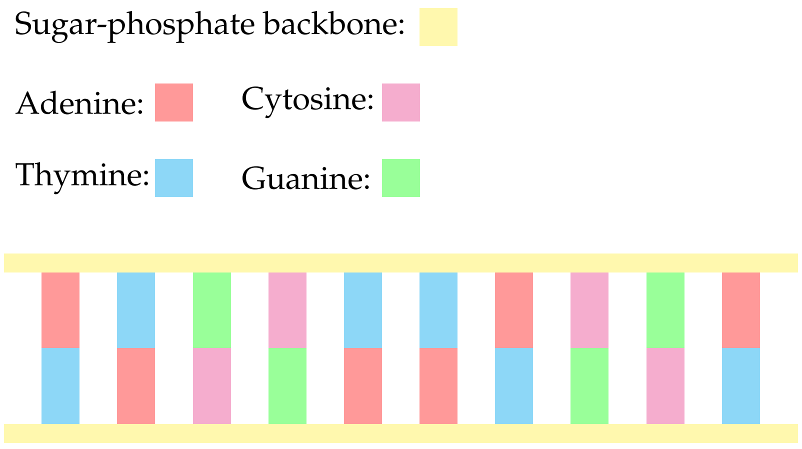


Figure 2.1: Complementary base pairing of DNA

The deoxyribose sugar of each nucleotide contains five carbon atoms, denoted 1′ to 5′. The 3′- and 5′-carbon atoms bond to the phosphate group to form the sugar phosphate backbone of a polynucleotide. At each end of the polynucleotide, one carbon will remain unbound with a phosphate group, either the 3′- or 5′-carbon. The carbon that is left unbound determines whether the end of the polynucleotide is the 3′ or 5′ end [Brown, 1998]. Complimentary DNA strands in the double helix structure of DNA run in opposite directions to each other. The direction of DNA becomes relevant during the process of DNA replication and during gene expression as some enzymes operate in only one direction.

Not all of an organism's genome is used to code for proteins and RNA (Section 2.1.3), there exist coding and non-coding sequences of DNA, where coding sequences are used to code for proteins [Brown, 1998]. The purpose of non-coding regions is not fully understood; some non-coding sequences are transcribed into functional products such as transfer RNA. They also act as transcriptional regulatory elements such as enhancers and silencers, which are sites that may be bound to in order to influence the rate of transcription [Ludwig, 2002, Maston et al., 2006].

2.1.2 DNA replication

DNA replication is a process that occurs prior to a cell dividing to produce a copy of the the cell's DNA [Pray, 2008b]. DNA replication is referred to as being 'semi-conservative' as each new DNA molecule consists of a strand from the parent molecule, and a newly synthesised strand [Filner, 1965]. DNA replication differs between prokaryotes and eukaryotes in that more DNA polymerases are involved during eukaryotic DNA replication, and there are multiple points of origin. Replication occurs in two directions within prokaryotes, but only one direction within eukaryotes. The process of DNA replication must be extremely accurate, as even a small margin of error would be multiplied through subsequent cell divisions.

DNA polymerase is the name given to the group of enzymes that synthesise new polynucleotide chains by joining nucleotides to each other [Loeb and Monnat Jr, 2008]. It only operates in the 5′→3′ direction. DNA polymerase recognises the specific base sequences of replication origins, where it will begin the replication process. Three types of DNA polymerase enzymes are found within the bacterium *E. coli*: DNA polymerase I, II, and III, each performing different functions such as DNA replication and repair.

The parent DNA strand is unwound at the replication origin, often consisting of many weaker adenine-thymine bonds. The region where the DNA has been unwound is referred to as the replication fork, this is where new nucleotides are added to the polynucleotide. A helicase enzyme separates the strands, single-strand binding (SSB) proteins attach to the DNA to prevent the strands from reattaching to each other. DNA polymerase replicates each strand in opposite directions; the leading strand is referred to as such because it is copied in the same direction that the helix is unwound. The lagging strand is copied in the opposite direction in segments known as ′okazaki fragments′ [Okazaki et al., 1968]; the leading strand is copied continuously. At a new replication fork, and enzyme called primase (and RNA polymerase) must create an RNA primer, which is a sequence of nucleotides complimentary to the DNA strand in order for DNA polymerase to begin replication [Alberts, 2003].

2.1.3 Ribosomes and RNA

RNA (ribonucleic acid) is a product of transcription (Section 2.2.2) [Clancy, 2008a]. Three major classes of RNA exist in prokaryotic and eukaryotic organisms. Ribosomal RNA (rRNA) and transfer RNA (tRNA) are end products of gene expression, whereas messenger RNA (mRNA) is translated to produce a polypeptide chain (Section 2.2.3). The base uracil takes the place of thymine in RNA [Clancy et al., 2008]. Some human diseases are caused by RNA viruses (viruses consisting of RNA) that infect a host organism and replicate within its cells.

The role of tRNA molecules during translation is to transport the correct amino acids so that they can be joined together to form a polypeptide chain [Kirchner and Ignatova, 2015]. A number of tRNA molecules exist, each recognising and binding to a specific amino acid. Once a tRNA molecule bonds with its corresponding amino acid, it is said to be 'charged' with it. Charged tRNA molecules recognise the codon specifying the amino acid that it is charged with, and positions itself on the ribosome so that the amino acid may be joined with the polypeptide. The anticodon loop of the tRNA contains nucleotides that form anticodons, which are used to form complementary base pairs with the codon on the mRNA. tRNA can recognise multiple codons in cases where multiple codons code for the same amino acid.

Ribosomes bind to mRNA during translation in order for a peptide chain to be formed [Clancy and Brown, 2008]. Ribosomes consist of ribosomal RNA (rRNA) bound to a protein and consist of large and small subunits. They provide sites for tRNA charged with the corresponding amino acid to bind with the mRNA sequence in order to form a peptide [Hoernes et al., 2018].

2.2 Gene expression

Gene expression is the process that results in the production of a gene product from a gene. Briefly, DNA is transcribed to produce mRNA and functional RNAs, and the resulting mRNA is translated; this involves the interaction of the functional RNAs to produce a polypeptide sequence that may form a protein either in itself, or in combination with other polypeptides. The 'central dogma of molecular biology' states that information flows from the DNA to the RNA and finally to a protein, it effectively states that the flow of information into a protein is a one-way process [Crick, 1958,Crick, 1970].

2.2.1 Proteins

Proteins are large molecules that perform many essential functions in organisms, such as the catalysis of biochemical reactions and cell signalling (e.g. insulin [Zhao and

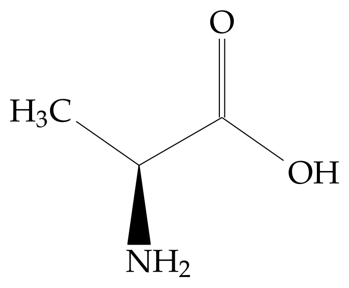


Figure 2.2: Skeletal formula of alanine, an amino acid with a $-CH_3$ side chain.

Alkon, 2001]). They consist of one or more polypeptide chains, which are polymers formed by chains of amino acid monomers. The process of gene expression results in a gene product, proteins being one prominent example. Proteins also play an important role during gene regulation; transcription factors are proteins that control the rate of transcription by binding to specific DNA sequences.

The general structure of an amino acid consists of a hydrogen atom, an amine group ($-NH_2$), and a carboxyl group ($-COOH$) bonded to a central carbon atom referred to as the α-carbon. The side chain of an amino acid (also referred to as the R group) is bonded to the α-carbon at the remaining position and is the only part that differs amongst amino acids. For example, the side chain of alanine is $-CH_3$ (Figure 2.2). Peptide bonds between the amine group of one amino acid and the carboxyl group of another, resulting in a peptide chain [Blanco and Blanco, 2017]. A peptide bond formed between a glycine and alanine amino acid is illustrated in Figure 2.3. Proteins are formed when a large number of amino acids form a polypeptide chain. A polypeptide in itself may form a protein, or multiple polypeptides may form the subunits of a protein. Bonds formed between the different components of multiple amino acids are what allow higher order protein structures to form. The variable length and combination of side chains allow proteins with vastly different functionalities to form, which result in the different types of cells in higher order organisms.

Enzymes are biological molecules that act as catalysts during biochemical reactions. Catalysts act to increase the rate of a chemical reaction without themselves being consumed by the reaction. Most enzymes are proteins, ribozymes are an example of enzymes that are not proteins, and are made of RNA. Most metabolic reactions involve enzymes as they would occur too slowly without them [Berg et al., 2002]. DNA and RNA polymerases are examples of enzymes that are used during DNA replication and

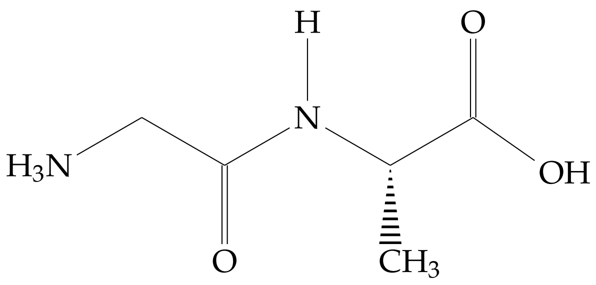


Figure 2.3: Peptide bond formed between a glycine and alanine amino acid.

gene expression to construct polynucleotides. Enzymes such as lactase and maltase break down lactose and maltose into glucose.

2.2.2 Transcription

The first stage of gene expression, transcription, is the process that results in the production of an RNA transcript of a gene's DNA sequence by the enzyme RNA polymerase [Clancy, 2008a]; the RNA transcript will later be used to produce an amino acid sequence during translation (Section 2.2.3). Ideally, genes will only be transcribed when they are required by the organism in order to conserve energy, the selectivity of transcription is a factor that makes this possible. Transcription involves RNA polymerase, an enzyme that creates RNA molecules based on a DNA template; the process can be broken down into the initiation, elongation and termination stages. The process of transcription occurring in eukaryotic organisms will be summarised based on the description in [Klug et al., 2019].

During initiation, RNA polymerase binds to the promoter region of the DNA, the ability of RNA polymerase to recognise and bind to the promoter region determines the selectivity of transcription. The promoter regions of genes are similar, making it easier for RNA polymerase to recognise them. After binding to both strands of the DNA, RNA polymerase begins to separate the two strands to expose the nucleotides of the template strand.

During elongation, RNA polymerase unwinds the two DNA strands as it moves along the template strand, constructing the RNA transcript by attaching ribonucleotides complementary (see Section 2.1.1) to the template strand, with nucleotides containing Uracil pairing with Adenine nucleotides, as Uracil replaces Thymine in RNA. At any one time, a limited number of base pairs are unwound forming a 'transcription bubble'. Within the transcription bubble the RNA transcript remains bound to the DNA, and eventually dissociates allowing the DNA strands to reattach.

The final stage of transcription is called termination, where RNA polymerase detaches from the DNA strands after transcribing the terminator region. In some cases, another protein 'rho' is required for RNA polymerase to recognise the terminator region.

The result of transcription is an RNA transcript, the sequence of which is complementary to the gene found on the template DNA strand. This completed transcript is called messenger RNA (mRNA), and is ready to be translated immediately within prokaryotes. Additional processing must occur before translation may begin within eukaryotes.

2.2.3 Translation

Translation is a process that results in the production of a polypeptide chain based on the sequence of nucleotides on an mRNA transcript, this occurs at the site of ribosomes — structures consisting of ribosomal RNA and proteins. During translation, a sequence of amino acids is formed to produce a polypeptide chain, one or more of which produce a protein. The process will be described based on translation in prokaryotes, based on the description in [Clancy and Brown, 2008]. The stages of translation are identical to those of transcription: initiation, elongation, and termination. The mRNA is read three nucleotides at a time, referred to as a codon.

Codons consist of a sequence three bases, which allows for a total of 64 unique combinations. 61 codons code for amino acids and the remaining three are 'stop codons', indicating the end of the sequence [Grantham et al., 1980]. The codon for the amino acid methionine (AUG) also indicated the start of a sequence and is known as the 'initiation codon'. This differs from some prokaryotic organisms that may use other sequences. Amongst the 61 codons that code for amino acids, there are redundancies where multiple codons code for the same amino acid.

During initiation, the small ribosomal subunit binds to the mRNA at a section before the initiation codon, and will traverse the chain until the initiation codon is reached. Once reached, tRNA (transfer RNA) charged with methionine will form an 'initiation complex' with the mRNA. The initiation complex is required for elongation to begin.

During elongation, the mRNA is traversed by the ribosome a single codon at a time, exposing the 'next' codon at the A site. Once the initiation complex has formed, the methionine occupies the 'P site' of the ribosome. tRNA molecules charged with amino acids will enter the 'A site' of the ribosome, facilitated by elongation factor proteins to ensure that only tRNA molecules charged with the correct amino acid may enter the site. A peptide bond will be formed between the molecules in the P and A sites, and the tRNA in the P site will release its amino acid and is moved to the E site. The tRNA charged with the newly bonded amino acid moves from the A to the P site; the ribosome will then traverse the mRNA transcript by a codon, ready for tRNA charged with the matching amino acid to enter the A site. This process repeats to form a polypeptide chain.

Once a stop codon enters the A site, a release factor will enter the site instead of tRNA causing synthesis to stop. This involves the addition of a water molecule to the end of the polypeptide chain.

2.3 Differences in Prokaryotes and Eukaryotes

The computational models discussed and utilised within this thesis take inspiration from gene regulation. Different models are based on different regulatory concepts, which vary amongst prokaryotic and eukaryotic organisms. This can be attributed to the differences in the structure, and the process of gene expression, which will be described in this section.

Prokaryotes are unicellular organisms that lack membrane-bound organelles such as mitochondria and nuclei. The DNA of a prokaryote is situated within the nucleoid—a structure similar to a nucleus that is not surrounded by a cellular membrane. Eukaryotes can be distinguished from prokaryotes by the presence of a nucleus situated within the nuclear membrane. Eukaryotes are typically more complex than prokaryotes, and contain more membrane-bound organelles other than the nucleus; most eukaryotes are multicellular; however, unicellular eukaryotes exist such as algae and amoeba [Fuerst, 2010]. Table 2.1 summarises the differences between prokaryotic and eukaryotic organisms.

Prokaryotes	Eukaryotes
Unicellular	Mostly Multicellular
No membrane bound Organelles	Has membrane bound Organelles
DNA situated in Nucleoid	DNA situated in Nucleus
No Chromatin	DNA coiled around Chromatin

Table 2.1: Summary of eukaryote and prokaryote differences

2.3.1 Cell Structure Differences

Differences in cell structure between prokaryotic and eukaryotic organisms are described in [Fuerst, 2010], a summary of such differences will be provided.

A plasma membrane encapsulates the cytoplasm of all prokaryotic organisms to separate the inside of the cell from the external environment. The plasma membrane of most prokaryotes consists of two layers of lipids, known as a phospholipid bilayer. The phospholipid bilayer is selectively permeable, this ensures that important proteins and ions do not diffuse out of the cell, while still allowing certain molecules to move through it. The cytoplasm of prokaryotes is the area within the plasma membrane mostly consisting of a liquid substance known as cytosol. Most cellular reactions occur within the cytoplasm, including transcription and translation. The DNA of prokaryotes is situated within the cytoplasm in a region called the nucleoid, it floats freely and is not bound by structures such as chromatin.

A prominent difference in the cells of eukaryotes is the presence of a nucleus. Exceptions to this exist such as mature erythrocytes (red blood cells), which do not contain a nucleus, nor any other membrane-bound organelles. Nuclei are surrounded by a nuclear envelope consisting of an inner and outer phospholipid bilayer; this protects the DNA of the cell, which is usually situated within the nucleus. Additional membrane-bound organelles exist within eukaryotic cells, such as the mitochondria that create adenosine triphosphate (ATP), which provides the chemical energy needed for processes such as muscle contraction. The endoplasmic reticulum also differentiates eukaryotes from prokaryotes, a structure that transports and folds proteins. DNA does not float freely within the nucleus of eukaryotic cells, as it is packaged into chromatin, a structure that facilitates gene regulation and prevents damage to the underlying DNA.

2.3.2 Differences in Gene Expression

The differences of gene expression are described in [Klug et al., 2019], some of these differences will be summarised. The DNA of prokaryotic organisms floats freely within the cytoplasm and is not bound by chromatin, this allows transcription and translation to occur concurrently. During transcription RNA polymerase binds to promoter regions within the DNA to produce RNA, which is then translated into proteins. A single RNA polymerase is responsible for the transcription of DNA in prokaryotic organisms. The DNA of eukaryotes is stored within the nucleus surrounded by a nuclear membrane, this keeps the contents separate from the rest of the cell. Transcription occurs in the nucleus and the resulting RNA is transported outside of the nucleus to the cytoplasm, where translation occurs. Transcription of eukaryotic DNA involves multiple RNA polymerases, which serve different functions.

2.4 Gene Regulation

The processes involved in gene regulation are described in [Latchman, 2005], and will be summarised in this section. Gene regulation ensures that the correct genes are expressed in the correct quantities and at the right times. This process is essential in multicellular organisms, where there are different cell types that perform different functions. The differing functionality amongst cells is a result of the proteins and functional RNAs they contain; however, the DNA contained within most types of cells is the same, meaning that the process of producing functional RNA and proteins from the DNA must be controlled. Gene regulation results from interactions that occur at multiple stages during gene expression, primarily during transcription.

Gene expression is conducted differently by prokaryotic and eukaryotic organisms, as a result of this, there are differences in how it is regulated. Transcription and translation occurs simultaneously within prokaryotes because the DNA is not contained within a nucleus, and is situated within the cytoplasm of the organism. Eukaryotic gene expression is a more complicated process, where DNA is transcribed to RNA within the nucleus of the cells, the resulting RNA is then transported to the cytoplasm where translation occurs. Gene regulation in both prokaryotes and eukaryotes will be described within this section.

2.4.1 Gene Regulation in Prokaryotes

The rate at which DNA is transcribed is the primary method of gene regulation in prokaryotic organisms [Seshasayee et al., 2011]. Activators, repressors and inducers are regulatory molecules that interact with DNA and RNA polymerase in order to influence transcription, they are examples of proteins. Most activator molecules increase the rate of transcription by binding to the activator-binding sites of DNA. In prokaryotes, genes exist with promoter regions that have a low affinity for RNA polymerase, requiring activators to assist in binding. In some cases, activators are required for transcription to occur at all. Some activators have an allosteric site, to which molecules can bind in order to enable or inhibit the activator. Repressors are proteins that decrease the rate of transcription, generally by binding to operator regions of DNA to prevent RNA polymerase from transcribing it. Repressors exist that must bind to a corepressor in order to function. The functionality of activators and repressors can be influenced by repressors. Some inducers bind to activators in order to increase their affinity for DNA to allow them to bind more easily, thus increasing the rate of transcription. Inducers may also increase the rate of transcription by binding to repressor molecules, this prevents them from binding to operator regions of DNA. Gene regulation in prokaryotes involves operons, sections of DNA that allow a group of genes to be transcribed together.

Operons

Operons are sequences of DNA that allow for groups of structural genes, often related in functionality, to be transcribed together under the control of a single promoter [Ralston, 2008]. RNA polymerase recognises the promoter and will initiate transcription upon binding.

Regulator genes are genes existing separately to the operon, but are involved in the regulation of transcription. If a regulator protein binds to the operator site of an operon, transcription will be prevented as the binding of RNA polymerase is blocked. This is because the operator overlaps with the promoter. 'Negative inducible' and 'negative repressible' are types of operon.

In the case of negative inducible operons, a regulator protein will readily bind to the operator site, thus preventing transcription. Therefore, for transcription to occur the binding of the regulator protein must be prevented. A small molecule known as an inducer must bind to the regulator protein, this causes a structural change that results in its inability to bind to the operator of the operon, thus permitting transcription.

In the case of a negative repressible operon, transcription occurs under normal conditions as the regulator protein does not bind to the operator site by itself. A corepressor molecule must bind to the regulator protein, so that the regulator protein may bind to the operator causing transcription to stop.

The *lac* Operon

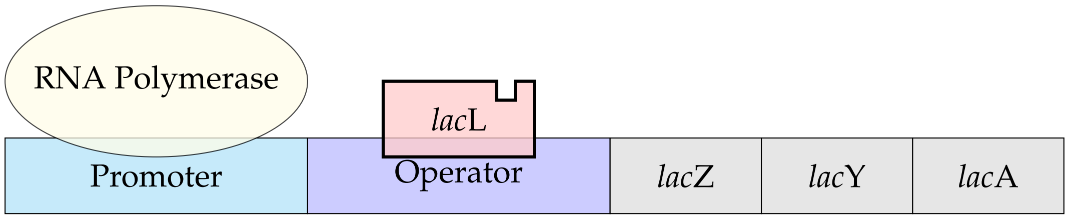
The *lac* Operon can be found within the bacterium *E. coli* (*Escherichia coli*). Both glucose and lactose can be used by e-coli as an energy source; however, glucose is preferable as lactose must be hydrolysed by β-galactosidase before it can be utilised, which is less efficient. The *lac* operon controls the transcription of the genes required for the transport and metabolism of lactose, ensuring that they are only transcribed when lactose is present and glucose is not [Jacob and Monod, 1961,Griffiths, 1999].

The *lac* repressor (*lac*L) binds to the operator site (Figure 2.4a) in the absence of lactose, thus preventing transcription. Allolactose (a molecule produced during the metabolism of lactose) acts as an inducer to *lac*L, preventing it from binding to the operator site (Figure 2.4b).

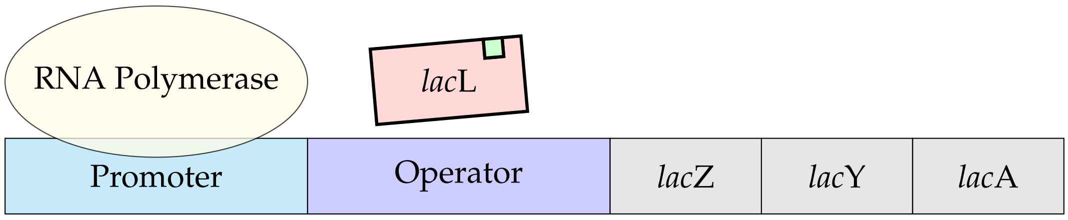
Catabolite repression is a mechanism that reduces the transcription of the *lac* operon when glucose is present. The CAP (catabolite activator protein) binding site exists upstream of the *lac* operon promoter, which CAP will bind to in the presence of a molecule called cAMP (cyclic adenosine monophosphate). In the presence of cAMP the CAP-cAMP complex is formed, which binds to the CAP binding site as shown in Figure 2.4c. The presence of glucose decreases cAMP levels as glucose acts as an inhibitor to adenylate cyclase, a catalyst involved in the production of cAMP. This mechanism causes any available glucose to be used as a fuel source before lactose is utilised.

2.4.2 Gene Regulation in Eukaryotes

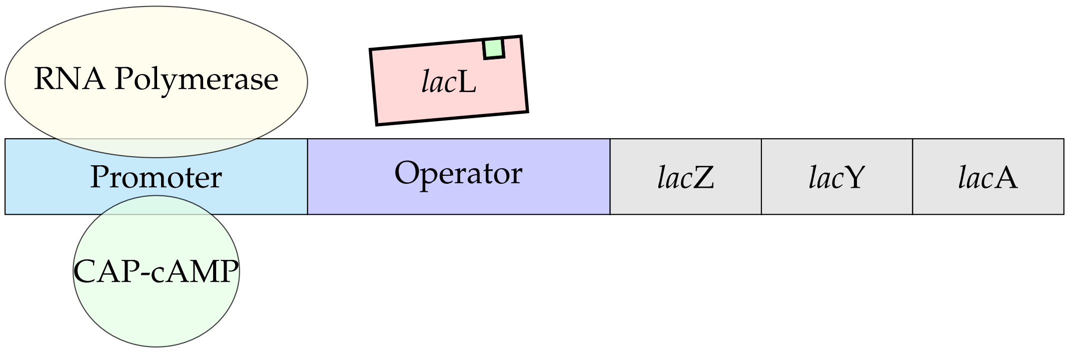
Prokaryotic gene regulation primarily occurs during transcription; in eukaryotes, gene regulation is more complicated and occurs during multiple stages. Similarly to prokaryotic organisms, gene regulation may occur when the RNA is transcribed;



(a) The *lac* repressor blocks transcription when lactose is not present.



(b) Allolactose binds to the repressor (*lac*L) in the presence of lactose, preventing it from binding to the operator.



(c) The CAP-cAMP complex binds to the promoter in the absence of glucose.

Figure 2.4: Three stages of the *lac* operon

however, eukaryotic transcription occurs within the nucleus [Latchman, 2005]. Gene regulation in eukaryotes also occurs when the resulting RNA is transported to the cytoplasm, called the 'post-transcriptional level'. It may also occur during and after the translation of RNA, called the translational and post-translational level respectively. In eukaryotic organisms, DNA is coiled around nucleosomes, and must be uncoiled for transcription factors to bind to the DNA to allow transcription to occur. Gene regulation may occur during this process, which is known as the 'epigenetic level'. The following section will describe some of the processes that occur during gene regulation at the epigenetic level.

2.5 Epigenetics

The term 'epigenetics' has a different definition in two different scientific fields. The different definitions are thought to have arisen from two origins. The adjective 'epigenetic' was originally used to refer to the noun 'epigenesis' [Haig, 2004], the theory that an embryo develops by gradual stages of differentiation from an unstructured mass such as an egg cell or seed. During the 19th century, inheritance and development were considered to be the same problem. Towards the middle of the 20th century, scientists began to realise that they could be studied together [Holliday, 2006]. The term 'epigenetics' was used by Waddington to refer to all of the interactions that occur between genes and their external environment that result in changes to the phenotype of an organism [Waddington et al., 1939,Müller and Olsson, 2003].

The precise definition of epigenetics is subject to debate. A modern definition of epigenetics used by molecular biologists is given by Riggs as: "the study of mitotically and/or meiotically heritable changes in gene function that cannot be explained by changes in DNA sequence" [Riggs et al., 1996,Haig, 2004]. This is the definition of epigenetics that will be used in this thesis. Put simply, epigenetics is the study of heritable changes in an organisms traits that do not result from changes to the organisms DNA sequence. The environment of an organism may influence its epigenetics to allow the organism to adapt to it. Epigenetic mechanisms often do not act in isolation.

2.5.1 DNA Methylation

DNA methylation is an epigenetic mechanism primarily occurring when a methyl group (CH_3) is added to the cytosine base to form 5-methylcytosine. Methylation of cytosine in this way causes gene expression to be repressed by preventing transcription factors from binding to DNA, or by recruiting proteins that cause the repression [Moore et al., 2013]. Methylation acts as a method of silencing parts of the genome in some eukaryotic organisms as the occurrence of it correlates with noncoding and repetitive DNA content [Phillips et al., 2008,Suzuki and Bird, 2008]. Methylation of the adenine base has also been observed [Hattman, 2005,Heithoff et al., 1999]. DNA methylation is thought to influence cognition in areas such as memory formation and learning, and it has been associated with neurological and developmental disorders such as Alzheimer's [Kramer, 2013] and Rett syndrome [Amir et al., 1999].

In mammals, methylation occurs frequently at CpG islands, which are regions of DNA with a high frequency of CpG sites — regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide. The methylation of multiple promoter CpG sites results in the long term silencing of genes by preventing the initiation of transcription [Jones and Takai, 2001]. DNA methylation is thought to have a role in X chromosome inactivation [Plath et al., 2003], a process that occurs in mammalian females where one of their X chromosomes is silenced, this is essential as extra chromosomes can be lethal and result in death at the embryonic stage. Methylation has been shown to be important in maintaining the stability of the genome, as the loss of ability to do so can result in disease and the progression of cancer [Baylin, 2005].

2.5.2 CpG islands

A CpG site is a region of DNA where a cytosine nucleotide follows a guanine nucleotide along the $5' \rightarrow 3'$ direction. The cytosines of CpG sites are frequently methylated in mammalian genomes. Regions of DNA containing a disproportionately high frequencies of CpG sites are referred to as CpG islands, and in mammalian genomes are frequently found in close proximity with promoter regions. Despite the frequency of CpG sites, CpG islands mostly lack methylation [Deaton and Bird, 2011]. The methylation of CpG sites is associated with transcriptional suppression, and can cause the stable silencing of genes [Hsieh, 1994].

2.5.3 Chromatin

In its raw form, DNA is too long to fit inside a cell's nucleus so it must be condensed. In eukaryotes, DNA is folded into a series of units to form a highly condensed structure called chromatin. 146 base pairs of DNA wrap around a group of histone proteins known as a histone octamer to form a nucleosome [Annunziato, 2008], a section of the base pairs acts as a chain linked to the next nucleosome. Heterochromatin and euchromatin are two forms of chromatin that differ in their densities. Heterochromatin is a highly condensed form of chromatin, because of this it is associated with a low occurrence of transcription as the DNA is less accessible; conversely, euchromatin is a less condensed form of chromatin, it is associated with a high occurrence of transcription as the DNA is more accessible. The function of chromatin extends beyond being a structure that condenses DNA, it acts as an indexing system that controls the transcription of the DNA within it.

2.5.4 Chromatin Remodelling

Chromatin exists in different states that vary in density, determining how readily the packaged DNA may be transcribed. Chromatin remodelling is the process that facilitates access to DNA by altering the structure and positioning of the nucleosomes that constitute chromatin, this is achieved by converting it from a condensed state to a more accessible, less condensed state. Transcription is a fundamental process in which DNA is transcribed into RNA by RNA polymerase enzymes, to be later translated into functional gene products. In Eukaryotes, RNA polymerase and transcription factors must be granted access to the packaged DNA in order for transcription to occur, this is facilitated by chromatin remodelling [Phillips and Shaw, 2008].

Chromatin remodelers are protein complexes that catalyse the reactions that result in alterations to chromatin such as nucleosome sliding, whereby the connection between DNA and the histone proteins are temporarily disrupted in order to make the DNA more accessible for transcription [Narlikar et al., 2002, Becker, 2002]. Histone modifying complexes are a type of chromatin remodeler that catalyse reactions resulting in the modification of histones, the structures around which DNA is wound to eventually form chromatin. These modifications primarily involve modifications to the histone tails, which in some cases change the affinity between the histones to the DNA. Acetylation, methylation, phosphorylation and ubiquitination, in addition to their inverse counterpart reactions—e.g. demethylation—are examples of histone modifications, discussed further in Section 2.5.5. A second class of chromatin remodelling complexes exist that can be distinguished by their ATPase subunit [Vignali et al., 2000]. This subunit allows the complexes to utilise energy released from the hydrolysis of ATP to bring about changes to the structure of chromatin via mechanisms such as nucleosome sliding and the introduction of histone variants.

Chromatin remodelers are involved in histone variant exchange, a process that results in changes to the structure of octamers by introducing variants of histones H2A and H3 [Jin et al., 2005]. H2A.Z is a variant of histone H2A, the absence of which has been shown to cause genomic instability in mice by disrupting chromosome segregation, indicating its the importance of its role in this process [Rangasamy et al., 2004]. The importance of histone variant exchange is further demonstrated by experiments performed on the fruitfly *Drosophila melanogaster*, which indicates the importance of histone variants during development and the formation of heterochromatin [Swaminathan et al., 2005,Clarkson et al., 1999].

The INO80 class of chromatin remodelers are found within the human and yeast genomes, they have been associated with DNA repair, DNA replication and transcriptional regulation [Poli et al., 2017]. In humans, an INO80 complex was observed to interact with the transcription factor YY1, resulting in the activation of transcription [Cai et al., 2007]. INO80 complexes have also been associated with transcriptional repression in response to stress induced transcription, it is thought to achieve this by restoring nucleosomes associated with promoter regions, resulting in the limitation of high levels of transcription in yeast [Klopf et al., 2017].

2.5.5 Histone Modifications

Nucleosomes are the subunits of chromatin, consisting of a histone octamer—a structure containing 8 histone proteins, around which 147 base pairs of DNA wrap. There are 4 types of core histone proteins, referred to as H2A, H2B, H3, and H4, all having similar structures consisting of a histone fold and histone tail. Histone tails are protruding structures that interact with the DNA by forming connections with its sugar-phosphate backbone, this increases the positional stability of the DNA [Iwasaki et al., 2013].

Post-translational modifications such as acetylation, methylation, phosphorylation and ubiquination occur to histones resulting in structural changes [Peterson and Laniel, 2004]. These alterations are known as histone modifications, epigenetic mechanisms that bring about the changes to gene expression by altering the structure of chromatin, or by recruiting remodelling enzymes that reposition nucleosomes [Bannister and Kouzarides, 2011]. These modifications usually occur to the tail domain of the histone, providing a site to which other proteins can bind. It is hypothesised that modifications to the tail domains of histones constitute a type of encoding that is interpreted by other regulatory molecules, referred to as the 'histone code' [Strahl and Allis, 2000].

Acetylation is a type of histone modification involving the addition of an acetyl group ($COCH_3$); it is involved in processes such as nucleosome assembly, where the pattern of acetylation influences the ordering of the chromatin structure by histone chaperones [Shahbazian and Grunstein, 2007]. Histone acetylation and deacetylation are thought to operate in two ways. Acetylation decreases the affinity of histone tails for DNA causing the remodelling of chromatin into euchromatin, this makes it more accessible to regulatory molecules that cause transcription to occur; conversely, deacetylation reverses this effect causing the DNA to wrap more tightly around the histones to form heterochromatin, making the DNA less accessible to transcription factors, thus decreasing gene expression levels. Acetylation and deacetylation are also thought to influence transcription by indirectly altering the chromatin structure via interactions with activator and repressor proteins [Struhl, 1998].

Similarly to DNA, histones may also be methylated involving the addition of a methyl group, occurring at the lysine and arginine residues located on the histone tail. A class of enzymes called protein arginine methyltransferases (PRMTs) catalyse some of the reactions that result in histone methylation, as well as the methylation of transcription factors [Zhang and Reinberg, 2001]. Interactions between histones and DNA methylation have been observed that suggest a type of bidirectional relationship in which the two influence each other [Cedar and Bergman, 2009].

The occurrence of histone modifications influencing DNA methylation is apparent with Dnmt3L, a regulatory protein that has been shown to interact with histones. The co-expression of Dnmt3L with Dnmt3a, a DNA methyltransferase enzyme, has been associated with an increasing rate of methylation by Dnmt3a [Ooi et al., 2007]. 'Ubiquitin-like, containing PHD and RING finger domains, 1' (UHRF1, also known as NP95 and ICBP90) is a gene that codes for a type of ubiquitin ligase, a protein that transfers ubiquitin to a substrate. It is thought that UHRF1 and DNMT1 play an important role in ensuring that DNA methylation is propagated from parent cells during the S phase of DNA replication [Bostick et al., 2007].

2.5.6 Gene Silencing

Gene silencing is a term used to refer to the effects of epigenetic mechanisms that result in the deactivation of genes [Filipowicz and Paszkowski, 2013]. Histone modifications such as acetylation alter the properties of histones in order to bring about changes in gene expression by changing the chromatin structure or by recruiting other regulatory molecules. Histone modifications are also thought to have an influence on DNA methylation in some organisms. It is thought that DNA methylation and histone modifications interact, potentially bidirectionally in such a way that causes genes to be silenced. There are different theories for the specific mechanisms and interactions that occur to cause this effect, there is evidence suggesting that either DNA methylation or histone deacetylation may be the cause of gene silencing [Vaissière et al., 2008].

The process of gene silencing is thought to play a role in the formation and progression of cancerous tumours. Promoter regions of mammalian DNA are often located in close proximity to CpG islands, which in normal circumstances are usually unmethylated, allowing transcription to occur. The abnormal silencing of CpG islands caused by methylation has been found to occur frequently in tumours, and many genes associated with tumour suppression have been observed as being methylated in certain types of cancer [Jones and Baylin, 2002,Baylin and Herman, 2000].

2.5.7 Genomic Imprinting

Mammalian cells contain two sets of chromosomes, one inherited from the mother, and the other from the father, which are expressed equally in most cases [Barlow and Bartolomei, 2014]. Genomic imprinting is an epigenetic process involving DNA and histone methylation that influences the expression of genes based on the parental origin [Reik and Walter, 2001]. An experiment performed on mice eggs demonstrated that the survival depended on whether a particular gene was inherited maternally or paternally [Johnson, 1974]; if the gene was inherited maternally, death would occur. Genomic imprinting is also thought to be involved with genetic disorders in humans, such as Prader-Willi syndrome that has been observed to arise based on the parental origin of the gene mutation [Reik, 1989].

2.5.8 Advantages and Importance of Epigenetic Mechanisms

Within this section, the existence of various epigenetic structures and mechanisms has been described. Eukaryotic organisms differ from prokaryotes in that generally speaking, they are more complex as a result of them possessing far more complex genomes, as well as more complex regulatory mechanisms that take place during the various stages of gene expression. A summary will now be provided in the form of a discussion of the specific advantages of epigenetic phenomena.

DNA Packaging and Indexing

The complexity of higher order eukaryotes is a consequence of their complex genome, coded by a single strand of DNA that is far too long to fit within the nuclei of eukaryotic cells. Chromatin is a structure that heavily condenses DNA, allowing it to fit within the nucleus of eukaryotic cells. Prokaryotic DNA floats freely within the cytoplasm in an uncondensed state allowing RNA polymerase and transcription factors to bind to it in order to enable transcription, this is not the case with Eukaryotes as the condensed chromatin structure restricts access to the DNA [Phillips and Shaw, 2008]. Chromatin remodelers are protein complexes that facilitate regulated access to the DNA by other regulatory molecules, so that gene expression may occur in a regulated fashion.

Regulation of Stem Cell Differentiation

Despite the different types of cells found within multi-cellular organisms, the DNA code contained within them is identical. Cellular differentiation the process that causes unspecialised stem cells to undergo changes that result in more specialised cells such as skin cells and muscle fibre cells, this process is known as specialisation. Gene expression and repression are two principal mechanisms that influence the resulting cell's shape and function [Ralston and Shaw, 2008], allowing for a wide range of cells with varying characteristics and functions to be produced. Epigenetic mechanisms have been shown to be involved during the regulation of stem cell differentiation. DNA methylation has been observed as being involved in multiple stages of embryonic development and gestation; histone acetylation is known to be involved during the development of neurons [Juliandi et al., 2010].

Adaptation

Epigenetic mechanisms are capable of causing long term reversible changes to the phenotype without altering the DNA sequence, allowing for the adaptation of an organism to its environment without involving DNA mutation, which must occur to the gametes in order to be propagated to offspring. Epigenetics has been shown to allow for a degree of phenotypic plasticity in response to environmental factors [Duncan et al., 2014]. An example of this in humans is the influence of maternal depression on the methylation of Nr3c1, which was associated with an increased stress response, indicating that the mental state of the maternal parent also affected that of the child [Oberlander et al., 2008]. Phenotyping plasticity is beneficial within the scope of adaption as it allows the physiology of an organisms to change in response to environmental factors.

2.6 Summary

In this chapter, the fundamental concepts and mechanisms of gene expression have been introduced, followed by the higher-level processes that regulate it. These fields inspired artificial gene regulatory networks (AGRNs), the models used within the experiments described within this thesis. AGRNs are described in more detail in Chapter 4.

The experiments described in chapters 7 and 8 involve an investigation into the evolutionary process and developmental properties of AGRNs, computational models that take inspiration from biological gene regulatory mechanisms. In Chapter 9, a control layer is created that is capable of dynamically switching parts of external connectionist architectures, this takes inspiration from the processes of epigenetic mechanisms such as gene silencing.

Chapter 3

Connectionism

Cognitive science is the study of the mind and the process that occur within it [Thagard, 2005]. Cognition refers to the processes and structures that facilitate thinking, involving the acquisition of knowledge, decision making, interpretation of information gathered from the senses, and the formation of memory. Cognitive science is an interdisciplinary field involving neuroscience, psychology, linguistics, philosophy, and computer science. The human mind can be described as a complex system capable of receiving, storing, transforming and transmitting information [Stillings et al., 1995].

Connectionism is a theory of information processing within the field of cognitive science that models the mind as a network of interacting neurones, known as an artificial neural network [Medler, 1998]. Within this thesis, connectionist architectures are of great interest. Evolutionary pathways of AGRNs are explored in Chapter 7, and the behaviour of nodes within AGRNs is explored in Chapter 8, AGRNs can be considered as biochemical connectionist architectures. In Chapter 9, a control layer is applied to a pre-trained connectionist architecture in an attempt to improve its transparency. This chapter will first briefly explain the relevant biological concepts, followed by their computational analogues, artificial neural networks, and how they operate. The case for classifying artificial biochemical networks as connectionist architectures is also discussed.

3.1 The Biological Neural Network

Artificial neural networks are inspired by the neurons that interconnect and interact in order to produce the functionality of animal brains. The central nervous system

(CNS) co-ordinates most of the functionality within the body and mind of most multicellular animals by processing information from the peripheral nervous system [Daly et al., 2012]. The brain and spinal cord make up the two major components of the CNS, which act together to co-ordinate cognition, movement, senses and emotion [Tuladhar et al., 2015]. The CNS is made up of nerve cells, or neurons, as well as glial cells, neurons transmit information while glial cells protect the neurons and perform auxiliary functions.

3.1.1 The Neuron and Neural Circuits

Neurons are cells that communicate with each other, as well as with muscle and gland cells to transmit information. There are three types of neurons. Sensory neurons interpret chemical and physical inputs to detect touch, sound, light, smells and taste. Motor neurons connect to muscles, organs and glands to enact control over muscles to cause movement. Interneurons transmit signals between sensory and motor neurons, forming neural circuits. Synapses allow neurons to pass electrical and chemical signals to other neurons or other target cells [Südhof, 2018].

Structurally, neurons consist of dendrites, an axon, and a soma (the cell body). The dendrites branch from the soma and receive signals from other neurones, which they may then process and transfer to the soma [Shah, 2014]. The axon is also attached to the soma and conducts the action potential when the neuron fires. The neuron decides to fire based on a number of incoming inhibitory and excitatory signals received by the dendrites. Alone, neurons perform very simple functionality, they receive inputs, process them in some way, and may or may not fire. Neural circuits are formed by many neurons that act together to perform specific functions related to sensation, perception and behaviour. Synapses are the junctions between neurons (and other target cells) that allow neurons to communicate [Purves et al., 2004]. A neural network is formed by many neural circuits that act together to produce higher order functionality.

3.2 The Artificial Neural Network

Artificial neural networks (ANNs) are connectionist architectures consisting of multiple layers arranged in a hierarchy. They are inspired by biological neural networks

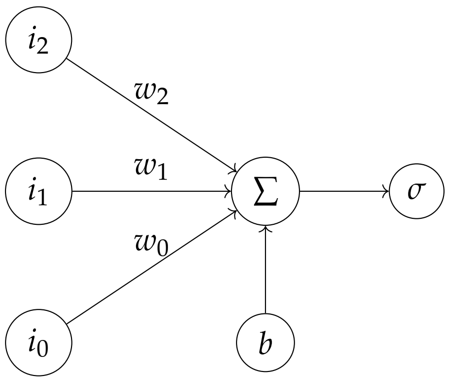


Figure 3.1: Illustration of a neural network node with three inputs. i represents node inputs, and w their weights, b is the bias term, and σ is the activation function, a sigmoid in this case.

that form the functionality of animal brains, consisting of interconnecting neurons that interact via electrochemical signals. Each successive layer of an ANN combines attributes from the preceding layer in order to recognise higher order features. The layers of ANNs contain processing units referred to as nodes, which are analogous to neurons found within biological neural networks. The functionality of the ANN arises as an emergent property of the interactions between its constituent nodes. The structure of ANNs is described in [Watt et al., 2020] and will be summarised in this section. Deep neural networks are a type of artificial neural network that are capable of automatic feature recognition and have more than one hidden layer.

3.2.1 The Artificial Neuron

Nodes within artificial neural networks consist of three major components and have input connections sourced from other nodes, or in the case of the first hidden layer, the input data. The first component of a node is its weights, a vector of real numbers, each one representing an input. When a node is executed, its inputs are multiplied by the corresponding weights and summed together. This weighted sum is added to the second component of the node, the bias term, a scalar. The weighted sum plus the bias term acts as the input to the third component of the node, the activation function. An activation function determines the output of a node, and must be continuous and differentiable for the neural network it is part of to be capable of solving non-trivial problems. This is illustrated by Figure 3.1.

3.3 Forward Propagation of an Artificial Neural Network

Forward propagation is the process by which a sample from a dataset, represented as an input vector is fed into and then processed by the network. The activations of each layer are calculated based on the previous layer's outputs, or in the case of the first hidden layer, the input vector. This occurs starting with the first hidden layer, then the second hidden layer, followed by the output layer (assuming a neural network with two hidden layers). The processes involved in forward propagation are described in [Watt et al., 2020] and will be summarised in the following text.

3.3.1 Calculating the Activation of a Node

The equation for calculating the activation of a single node is shown by Equation 3.1. Each node has associated with it a set of weights, one for each input node from the previous layer, as well as a bias term. The activations (outputs) of the nodes from the previous layer a are multiplied by their corresponding weight w and summed, the bias term b is added to this sum. This is referred to as the weighted sum, and is processed by the node's activation function, in this case a sigmoid. The subscript numbers represent the index of the node in the preceding layer that the weight or activation corresponds with.

$$\sigma(w_1 a_1 + w_2 a_2 + \cdots + w_n a_n + b) \tag{3.1}$$

3.3.2 Calculating the Activations of a Layer

Non-trivial neural networks consist of many parameters, meaning that it would be extremely inefficient to calculate the activations of each node individually. The rigid, static structure of ANNs results in a set of parameters that can be conveniently represented as matrices; therefore, the activations of the nodes in the same layer can be calculated together. The notation becomes more complicated in this case, $w^l_{j,k}$ is an example of the notation that will be used within this chapter to refer to nodes and their parameters. The superscript l indicates the layer of the node to which the parameter or activation refers. The subscript j and k are used to refer to the index of the target and origin nodes of the weight respectively, where the connection is made from the origin to the target node.

Equation 3.2 shows how the activations of the first hidden layer of the ANN are calculated. The first matrix represents the weights of the layer, each row contains the weights that connect the node at that index in the layer, to all of the nodes of the previous layer, each column representing the index of the nodes in the previous layer. The layer in question has *k* nodes, and the previous layer *j* nodes. This matrix is multiplied by the second vector, containing the activations of the previous layer; the result is a vector containing the weighted sums of each node. This is added to the second vector, which contains the bias term of each node. The activations are finally calculated by processing the resulting vector using an activation function, in this case a sigmoid function. This process is repeated recursively with each successive layer until the activations of the output layer have been calculated, from which the network prediction can be determined.

$$\mathbf{a}^1 = \sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,k} \\ w_{1,0} & w_{1,1} & \dots & w_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j,0} & w_{j,1} & \dots & w_{j,k} \end{bmatrix} \begin{bmatrix} a_0^0 \\ a_1^0 \\ \vdots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right) \tag{3.2}$$

The equation can be simplified by representing the matrices and vectors containing the activation functions and parameters as variables, as shown by Equation 3.3, and further simplified by representing the weighted sum (including the bias term addition) as a single variable, as shown by Equation 3.4. This notation will be used when describing backpropagation calculations in the next section.

$$\mathbf{a}^1 = \sigma(\mathbf{W}\mathbf{a}^0 + \mathbf{b}) \tag{3.3}$$

$$\mathbf{a}^1 = \sigma(\mathbf{z}) \tag{3.4}$$

3.4 Artificial Neural Network Training

A neural network is initialised with random parameters once its architecture has been determined. In order to produce any meaningful functionality, these parameters must be changed in a logical way. The gradual changing of these parameters occurs during training, an iterative process that happens over a number of time steps, or epochs. Neural networks may be trained using supervised or unsupervised learning, within this work, supervised learning will be used as it is most suited to classification tasks. Supervised learning involves the training of a network using a training dataset [Cunningham et al., 2008]. Each sample of a training dataset consists of a set of attributes and a label. The labels describes what the sample refers to, in an image dataset consisting of pictures of animals, the label might refer to the name of the particular animal. The output of the neural network is compared to the label to determine its accuracy.

3.4.1 Loss Functions

When training a neural network, the goal is to minimise the loss function. The loss function of a neural network determines how accurate the network output is when compared to the expected output derived from the label of the training set, this is determined by calculating the difference. Some examples of loss functions are given in [Wang et al., 2022]. Mean square error is an example of a loss function that calculates the average difference between the actual and predicted values, and is applicable to regression tasks. This is shown by Equation 3.5, where the y refers to the actual value, and $\tilde{y}$ the predicted value. The difference is squared in order to ensure positive values, and to penalise predictions that are far away from the actual value more.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{3.5}$$

3.4.2 Gradient Descent

Gradient descent is an algorithm used to optimise ANNs by minimising the loss function [Ruder, 2016]. The aim is iteratively tweak the parameters of the network to decrease the loss function. When training, the network is initialised with random parameters that are highly unlikely to produce the desired functionality of the network. The loss is calculated by running the training data through the network to obtain the predicted values, which are compared to the actual values by the loss function. The slope of the loss function is calculated, which indicates the direction in which the parameters must be tweaked in order to decrease the loss, by making the network more accurate. The parameters are tweaked via backpropagation.

3.4.3 Backpropagation

Backpropagation [Rumelhart et al., 1986] is used when training ANNs using gradient descent. It is used to adjust the parameters of the network based on the result of the loss function when an incorrect prediction is made. The parameters are adjusted based on the gradient of the loss function with respect to the parameters. The gradient with respect to a parameter describes the direction that it must change in order to minimise the output of the loss function, and the magnitude describes how sensitive the cost function is to that particular parameter. The gradients of each successive layer when traversing the network backwards are calculated using the chain rule, which is a method for calculating the derivatives of composite functions.

Because back propagation occurs backwards, it makes sense to first think about the parameters of the output layer. Backpropagation will be described with regard to a network with two hidden layers attempting a multi-classification problem. It is more efficient to make predictions on batches of training data samples so that the changes to the network can be taken as an average, but for simplicity backpropagation will be described as if it is being applied for a single incorrect network prediction at a time. Nodes can be thought of as their output, a scalar calculated from the weighted sum processed by an activation function; therefore, positive nodes refer to those that have a positive output and vice versa.

The network prediction when attempting a multi-classification problem is indicated by which of the output layer neurons has the highest output, determined by the activation function applied to the weighted sum of its inputs (the outputs of the second hidden layer). The parameters must be altered in a way that causes the output neuron representing the correct class to become more positive, and the output of the rest of the output neurons to become more positive proportional to the magnitudes of the parameters. To enact the desired changes, the weights of the correct output neuron connecting to positive nodes from the second hidden layer are increased, weights connecting to negative nodes are decreased. The opposite process is applied to the incorrect output nodes, instead decreasing the weights connecting to positive nodes from the previous layer, and increasing the weights connecting to negative nodes.

Enacting changes on the weights of the output layer is a relatively simple process. In order to determine the changes that must be made to previous layers, their impact on all successive layers must be taken into account. The parameters must be adjusted so that the outputs of the layer eventually cause the network to be more accurate when making predictions. The influence of a node in the second hidden layer on all of the nodes in the output layer is taken into account. For every node in the output layer that needs to increase in order to improve the accuracy of the network, the parameters of nodes in the second hidden layer that are connected with positive weights are increased, and those with negative weight decreased. The opposite is true for parameters in the second hidden layer that are connected with negative weights. This is applied inversely for every node in the output layer that needs to decrease in order for the accuracy of the network to improve. The changes are applied proportionally based on the magnitude of the influence on the output layer, by taking the magnitude of the second layer weights and the influence of the second layer weights on the output layer. This concept is applied recursively backwards to the previous layers, so that their influence on the eventual output of the network is taken into account in order to bring about the necessary changes.

3.4.4 Sensitivity of the Cost Function with Respect to a Weight

The chain rule is applied in order to calculate the derivative of the cost function with respect to a particular network parameter, so that it can be adjusted accordingly when the network makes an incorrect prediction. The chain rule will first be described as if it is being applied to adjust the weight of a single node in the output layer that is connected only to a single node in the previous layer, in a network consisting of two hidden layers. The output layer is at index 3. The derivative of the cost function C_0 must be calculated with respect to the output node's weight w^3 (Equation 3.6, in simpler terms, this refers to the effect on the output of the cost function when the weight is changed.

$$\frac{\partial C_0}{\partial w^{(L)}} \tag{3.6}$$

Changes to the weight parameter cause a change to the weighted sum, which causes a change in the activation of the node, eventually causing a change to the cost function. Therefore the derivative can be calculated by multiplying the derivatives of the weighted sum z with respect to the weight, the activation a with respect to the weighted sum, and the cost function with respect to the activation function, as shown by Equation 3.7. This equation calculates the effects of changing the weight to the cost function, or the cost function's sensitivity to that particular weight parameter.

$$\frac{\partial C_0}{\partial w^{(3)}} = \frac{\partial z^{(3)}}{\partial w^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial C_0}{\partial a^{(3)}} \tag{3.7}$$

The derivative of the weighted sum with respect to the weight is equal to the activation of the node in the previous layer, as shown by Equation 3.8.

$$\frac{\partial z^{(3)}}{\partial w^{(3)}} = a^{(2)} \tag{3.8}$$

The derivative of the node activation with respect to the weighted sum is calculated by taking the derivative of the sigmoid function with respect to the weighted sum, as shown by Equation 3.9.

$$\frac{\partial a^{(3)}}{\partial z^{(3)}} = a^{(3)} = \sigma'(z^{(3)}) \tag{3.9}$$

The derivative of the cost function with respect to the activation of the node is calculated by multiplying the error of the network by 2, as shown by Equation 3.10, where y refers to the desired output of the network.

$$\frac{\partial C_0}{\partial a^{(3)}} = 2(a^{(3)} - y) \tag{3.10}$$

The full equation for calculating the sensitivity of the cost function to the weight is shown by Equation 3.11. To calculate the derivative of the cost function, the average sensitivity across all training samples must be calculated.

$$\frac{\partial C_0}{\partial w^{(3)}} = a^{(2)}\sigma'(z^{(3)})2(a^{(3)} - y) \tag{3.11}$$

3.4.5 Sensitivity of the Cost Function with Respect to a Bias Term

The sensitivity of the cost function to the bias term of a node is calculated similarly to that of a weight, as shown by Equation 3.12.

$$\frac{\partial C_0}{\partial b^{(3)}} = \frac{\partial z^{(3)}}{\partial b^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} \frac{\partial C_0}{\partial a^{(3)}} \tag{3.12}$$

The derivative of the weighted sum with respect to the bias term is equal to one, resulting in the full calculation shown by Equation 3.13.

$$\frac{\partial C_0}{\partial w^{(3)}} = 1\sigma'(z^{(3)})2(a^{(3)} - y) \tag{3.13}$$

3.4.6 Sensitivity of the Cost Function with Respect to the Activation of a Node

When updating the parameters of a particular node, the overall effect it has on the output of the network must be calculated so that it's parameters can be changed in proportion to said effect. The equation for calculating the derivative of the cost function with respect to the activation of a node in the second hidden layer is shown by Equation 3.14. This is similar to the previous examples; however, in this case the network may have multiple nodes in each layer, the example reflects this. In this case, the result of the derivatives for each of the nodes in the output layer are summed, as the influence of the node on all of the output layer nodes must be taken into account.

$$\frac{\partial C_0}{\partial a_k^{(2)}} = \sum_{j=0}^{n_3-1} \frac{\partial z_j^{(3)}}{\partial a_k^{(2)}} \frac{\partial a_j^{(3)}}{\partial z_j^{(3)}} \frac{\partial C_0}{\partial a_j^{(3)}} \tag{3.14}$$

3.5 Deep Learning

Deep learning is a machine learning technique that is capable of recognising high level features from unstructured or unlabelled data such as images, sounds, and sensor output. This concept is termed 'representation learning' and removes the need for manual feature engineering, which is necessary for some machine learning techniques. A deep neural network is a type of artificial neural network that has more than one hidden layer.

The highly abstract nature of deep learning architectures allows them to be applied across many domains. Deep convolutional neural networks are capable of classifying images [He et al., 2016,Krizhevsky et al., 2012] and detecting objects [Liu et al., 2019] in datasets such as imagenet, which contains thousands of categories [Deng et al., 2009]. Recurrent neural networks have found their use in the field of linguistics, where they are used for tasks such as machine translation [Luong et al., 2015,Edunov et al., 2018]. Deep learning is of interest in the industry of self driving cars, where has been used to detect obstacles [Ramos et al., 2017], predict car steering angles [Maqueda et al., 2018], and detect traffic light states [Kulkarni et al., 2018].

3.5.1 Deep Neural Networks

Conventionally speaking, a deep neural network (DNN) refers to one that consists of more than one hidden layer. More hidden layers are associated with an increase in accuracy, as they encourage higher order functionality to occur based on the output from previous hidden layers. This section will describe different DNN architectures and how they are used.

Feedforward Neural Networks

Feedforward networks are the simplest type of neural network, they are constructed of multiple sequential layers, where each layer acts as an input to the next [Svozil et al., 1997]. Information flows in one direction, as opposed to recurrent neural networks where connections may form a cycle. The term 'feedforward' refers to the process that occurs when the network executes. For any prediction the network makes, the input data is stored in the input layer as a vector of real numbers. The values of the nodes in the first hidden layer are determined by matrix multiplication on the input vector by the weights of the first hidden layer. This process is repeated for each layer until the output layer is reached, the value (or values) contained by the output layer determine the output of the network, such as a predicted class.

Recurrent Neural Networks

Traditional feedforward neural networks receive an input vector with a fixed size, and output a vector with a fixed size; however, this is insufficient when dealing with data of indefinite length. Recurrent neural networks (RNNs) are designed to receive a series of inputs in order to process them in such a way that previous inputs are remembered [Lipton et al., 2015]. Hidden state vectors allow an RNN to remember previous input data, so that it can be taken into account. The ability to take previous inputs into account is essential when context is important, as is the case when processing sentences and speech. As an input sequence of indefinite length is processed by an RNN, an output vector also of indeterminate length is produced.

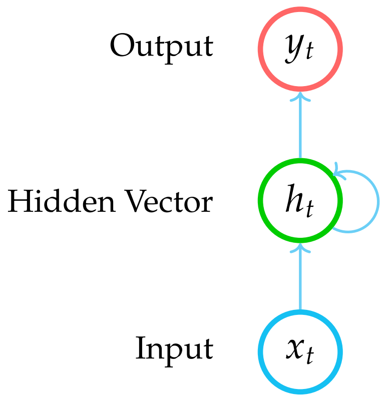


Figure 3.2: Illustration of a traditional RNN.

The architecture of a traditional RNN is shown by Figure 3.2. The RNN processes an input sequence over a series of time steps. The input is represented by x_t, at a particular time step this could be a word in a sentence, or a single frame of a video. At each time step, the hidden vector h is updated, allowing information from previous time steps to accumulate. An 'unrolled' version of the RNN is shown by Figure 3.3, this shows how the network executes over 3 time steps.

Different types of RNN configurations exist that differ in the number of inputs they process and the outputs that they produce. Many to one RNNs process a sequence

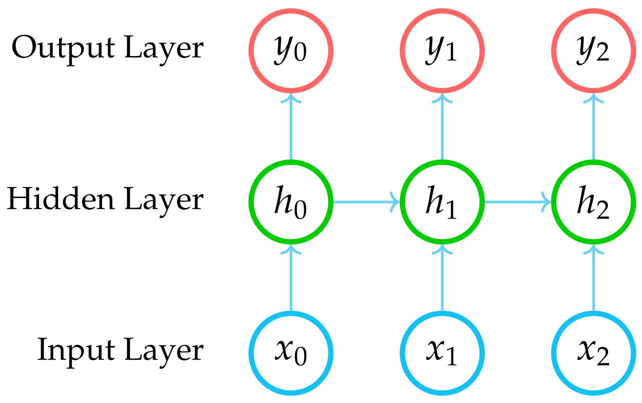


Figure 3.3: Unrolled version of RNN in Figure 3.3 with a processing sequence of length 3.

and produce a single output, a network of this configuration could be used to classify natural language. Many to many RNNs produce an output sequence after processing an input sequence, this configuration could be used to translate text into a different language. One to many RNNs also exist, producing an output sequence based on a single input. A one to one configuration, where a single output is produced from a single input, would not be considered an RNN as there would be no temporal processing involved.

Long Short-Term Memory

Traditional recurrent neural networks struggle to learn patterns in long sequences. They have been shown to become increasingly ineffective as the temporal distance between relevant data increases, when trained with gradient descent [Bengio et al., 1994]. Long short-term memory networks (LSTMs) are a class of RNNs that are capable of processing sequences of data and taking context into account. They were designed to be capable of retaining information over a long period of time and have been shown to be capable of bridging long time lags [Hochreiter and Schmidhuber, 1997]. LSTMs have a similar structure to conventional RNNs in that they have an input, a hidden state vector, and an output; however, the way in which information flows differs significantly.

The heart of an LSTM is its cell state, retaining information from each time state to the next. Modifications to the cell state are conducted via three gates consisting of a sigmoid function and a multiplication operation. The boolean output of these gates controls whether or not information should be let through. The first gate determines

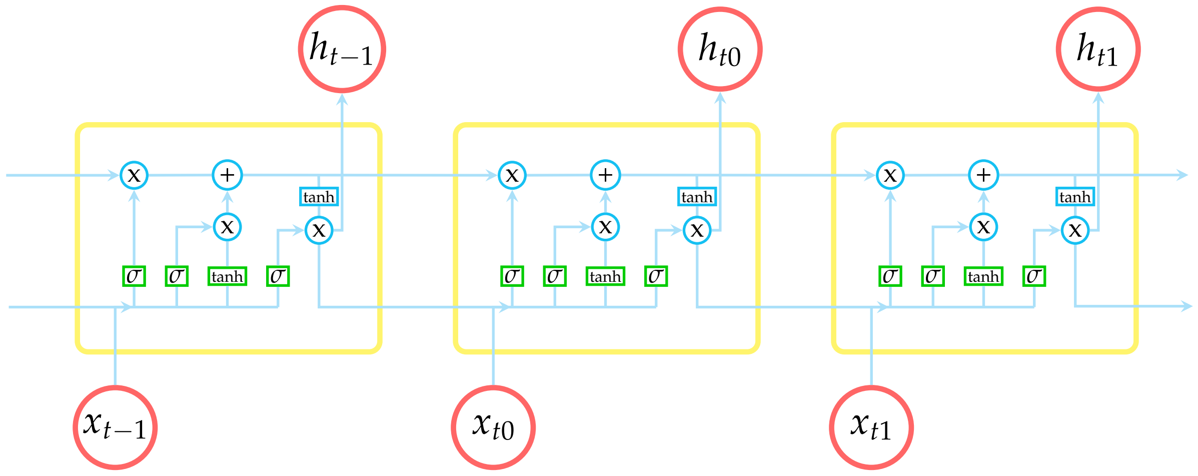


Figure 3.4: An unrolled LSTM.

whether or not the information from the previous time step should be discarded. The second gate along with a tanh function determines which new information should be stored. The final gate determines which information from the cell state should be output to the next time step.

Hopfield Networks

Hopfield networks [Hopfield, 1982] are a primitive example of RNNs capable of reconstructing incomplete data; they consist of a vector of symmetrically interconnected nodes, each having a binary state of 1 or -1. Equation 3.15 shows an example of a Hopfield network with 4 nodes, vector **v** represents the state of each node, and matrix **W** their weights.

$$\mathbf{v} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_{aa} & w_{ab} & w_{ac} & w_{ad} \\ w_{ba} & w_{bb} & w_{bc} & w_{bd} \\ w_{ca} & w_{cb} & w_{cc} & w_{cd} \\ w_{da} & w_{db} & w_{dc} & w_{dd} \end{bmatrix} \tag{3.15}$$

The weights are initialised by multiplying the states of the respective nodes, nodes do not connect to themselves, the resulting weights are shown by Equation 3.16. The state of each node can be calculated from this weight matrix by taking all of the connected nodes and multiplying their values by their respective weights. A positive value results in the node's state being equal to one, and a negative value negative one. A corrupted version of $\mathbf{v}$ with a point bit flip can now be passed back through the network to obtain the original.

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & -1 & 1 \\ 1 & 0 & -1 & 1 \\ -1 & -1 & 0 & -1 \\ 1 & 1 & -1 & 0 \end{bmatrix} \tag{3.16}$$

Training Methods

Deep neural networks must be trained in order to be useful, involving iterative changes to the network's parameters. This process will vary based on the nature of the problem that needs to be solved, as well as the data that is available. Training methods can generally be categorised as being supervised [Caruana and Niculescu-Mizil, 2006], unsupervised [Barlow, 1989] or based on reinforcement [Kaelbling et al., 1996].

Supervised learning methods require a dataset consisting of attributes and labels. During training, the network reads each set of attributes and predicts the output, which is then compared to the label in order to determine whether the model is correct or not. If the network prediction is correct, its state isn't changed; however, if the prediction is incorrect the parameters will be altered with the intention of improving its accuracy. A trained model can be trained, validated, and tested on subsets of the dataset in order to reduce the likelihood of over-fitting the training data. A drawback to supervised learning is that the dataset must be labelled prior to training, and the quality of the model produced will depend on the quality of the dataset and its labels. Supervised learning tends to be used for classification and regression tasks.

Unsupervised learning is a technique used to find patterns in unlabelled datasets. A simple example of this is k-Means clustering, where a dataset is split into a specified number of clusters, based on their euclidean distance to a centroid defined for each cluster. Generative models are a more complicated example of unsupervised learning, they attempt to generate instances of data similar to existing data, they are often used to generate images based on existing images from a dataset. Generative models are a component of generative adversarial networks (GANs), along with a discriminator that is trained to distinguish real data from generated data. They are trained in succession of one another in order to improve the quality of generated data.

Reinforcement learning models are somewhat similar to supervised learning models in that the output of the model is evaluated; however, in the case of reinforcement learning, the model is evaluated using the concept of a reward, which is increased when the model performs better. The reward is used by a training algorithm in order to improve it in some way. Reinforcement learning models are often used in cases where the model is an agent acting within an environment, attempting to complete a particular task. A typical example of a reinforcement learning algorithm is a genetic algorithm (GA), that consists of a population of individuals. Each individual is essentially a set of parameters that forms a potential solution to a problem. A GA operates by advancing the population through a set of time steps, or generations. At each generation, the parameters of individuals may be combined and mutated (randomly changed). The aptitude of an individual is measured by its fitness function, which is used by the GA so that it can favour better individuals to improve the probability of them passing their parameters onto the next generation of the population. This is analogous to the concept of natural selection in biology, where fitter members of a population are more likely to survive and pass their traits to their offspring, causing the population to adapt to their environment.

3.6 Biochemical Networks as Connectionist Architectures

Complex cellular function can be explained by biochemical networks, which describe many related interactions that occur involving enzymes, metabolites, and other biological molecules. Biochemical networks fall into three categories, which are metabolic, regulatory, and signalling [Price and Shmulevich, 2007]. Models of biochemical networks, referred to as artificial biochemical networks, have been shown to be capable of exhibiting complex computational behaviour, which allowed them to be used to control chaotic dynamics in dynamical systems [Lones et al., 2010].

In [Lones et al., 2013], it is argued that biochemical networks can contribute to connectionist knowledge, this chapter will summarise the reasoning behind this. The paper expresses that at a nodal level, neurons in the brain are homogeneous and are quite simple; whereas biochemical networks consist of many diverse biochemicals which interact combinatorially.

Conventional ANNs consist of fixed layers of nodes, and the nodes of a layer tend to have the same activation function. The organisation of nodes within an ABN may change during training, and there is potential for much more diversity in terms of the activation functions of nodes; activation functions may also be parameterised, allowing for more flexible characteristics [Turner et al., 2013b]. This suggests that ABNs are capable of more complex behaviour with relatively small networks.

The connections between nodes within an ABN are a consequence of the training process. They are not predetermined and are capable of changing based on the desired functionality of the network, and are referred to as being 'decoupled'. This is in contrast with conventional ANNs, where the connections between nodes are defined before training, and are not changed beyond this point. The decoupled nature of ABNs is advantageous as it allows for the network structure to be formed dynamically according to the task it is solving, allowing for a more efficient network.

3.7 Summary

In this chapter, an overview of connectionist architectures and the biological concepts that inspired them has been provided. Conventionally speaking, connectionist architectures refer to artificial neural networks (ANNs) and their derivatives; however, it has been argued that artificial gene regulatory networks (AGRNs) are a type of connectionist architecture due to their topology and inner workings, which are in this chapter referred to in the broader context of biochemical networks. AGRNs are the subject of Chapter 4, where an overview of the different types and how they operate is provided. Deep learning has also been introduced in this chapter, a field concerned with producing machine learning models capable of representation learning.

The capabilities of the machine learning models used within this thesis come at a cost, their inner workings and the reasoning behind their decisions are not self-evident. This issue underpins the reasoning behind the work conducted within this thesis and is described in more detail in Chapter 6, specifically Section 6.1. AGRNs are the subject of the experiments described in chapters 7 and 8. The experimentation described in Chapter 9 involves a control layer acting on a deep neural network.

Chapter 4

Modelling Gene Regulatory Networks

The artificial gene regulatory network (AGRN) is an example of a gene regulation model that is used within the experiments described later in this thesis. See Chapter 2 for an overview of the biology relevant to gene regulation models. Models of gene regulation take many forms; the AGRN is a computational model that is just one example. This chapter will describe the different types of gene regulation models and how they are applicable in different situations to justify the use of AGRNs; a brief history of how the AGRN developed will also be provided.

Models of gene regulation are usually created to serve one of two purposes: to learn more about the dynamics of biological gene regulation by simulating them [Kauffman, 1969, Keedwell et al., 2002, Kauffman et al., 2003, Cheng et al., 2013, Lai et al., 2016], or to capture the useful properties exhibited by the biological systems and apply them to computational problems [Taylor, 2004, White et al., 2005, Lones et al., 2010, Trefzer et al., 2010, Turner et al., 2012, Turner et al., 2013b, Turner et al., 2017]. Different classes of networks exist to serve these purposes, operating at differing levels of abstraction, complexity, and accuracy [De Jong, 2002, van Riel, 2006, Karlebach and Shamir, 2008]. Models must be engineered to support their purpose. For example, biological correctness would be important when designing a model to predict the effects of changing nutrient concentrations within a particular regulation process, but would not be when designing a model to control a chaotic dynamical task. The gene regulatory network models utilised within this thesis are modelled in order to solve computational problems.

A recent review of AGRNs discusses the different modelling and simulation methods [Cussat-Blanc et al., 2019]. There exist models that operate in continuous time [Gillespie, 1976, McAdams and Arkin, 1997, Chen et al., 2004, Li et al., 2008] and discrete time [Kauffman, 1969, Lones et al., 2010, Turner et al., 2013b], as well as continuous space [Lones et al., 2010, Li et al., 2008] and discrete space [Kauffman, 1969, Gillespie, 1976]. Models existing in discrete time operate in a series of time steps, these are of interest when dealing with computational problems as the networks must receive input from an environment in order to process it and enact control over the environment in order to solve the problem. Clearly defined time steps permit this process to occur reliably. Accurate mathematical models may use continuous time for biological accuracy as it allows for interactions to be studied at a small time resolution [Polynikis et al., 2009]. Early models operated in discrete space due to the simplicity of analysis and (presumably) lower cost in terms of computation [Dellaert and Beer, 1994, Harvey and Bossomaier, 1997]. Recent models often operate in continuous space as it allows for the accurate representation of natural systems.

4.1 Ordinary differential equations

A differential equation measures the change to a dependant variable with respect to either one or more independent variables. An ordinary differential equation (ODE) is a differential equation with only one independent variable. In simple terms, they measure the rate of change of one variable as another is changed, such as the change in the rate of synthesis of a protein when subject to changing nutrient concentrations. ODEs can provide highly detailed information about gene regulatory network dynamics when high quality data on kinetic parameters is available [Karlebach and Shamir, 2008].

4.2 Dynamical systems

Dynamical systems are of interest throughout the work described within this thesis. One of the major strengths of artificial gene regulatory networks is their ability to solve tasks within chaotic dynamical systems despite their relatively simple structure, and AGRNs themselves are dynamical systems. Generally speaking, a dynamical system consists of a state that changes over time according to a mathematical function, referred to as an 'evolution rule'. The system's state is represented as a set of variables that describe specific aspects of it, such as the velocity of an object. There exist different types of dynamical systems; discrete time dynamical systems can be defined as a map, where X is a non-empty set [Barreira and Valls, 2019]:

$$f : X \to X$$

Discrete models have a state space, and continuous models a phase space, which is the set of all the possible states resulting from all of the possible combinations of state variables. Models of most natural dynamical systems are subject to abstraction, as it is infeasible to quantify and account for every influencing factor. The evolution rule (or state transition rule) determines the state that will follow from another.

Some dynamical systems exhibit chaotic behaviour [Lorenz, 1963, Sprott, 1994, Li and Chen, 2004]. Chaos refers to the complexity and apparent randomness of a system's behaviour [Hilborn et al., 2000], resulting in difficulty when predicting the behavioural future [Sharifi et al., 1990, Guo et al., 2000] and controlling such systems [Hunt, 1991, Singer et al., 1991, Lu et al., 2004]. Chaotic systems are extremely sensitive to their initial conditions, as small differences in early conditions can eventually result in dramatic behaviour changes, this is referred to colloquially as the 'butterfly effect' [Persson and Wagner, 1996]. Despite chaotic dynamical systems being deterministic, they are difficult to predict due to the complexity of behaviour that they exhibit. Chaotic dynamical systems are of particular interest when evolving AGRNs, as they are difficult to solve mathematically, demonstrating the capability of AGRNs.

An attractor is a set of states that a dynamical system settles towards over time [Boeing, 2016]. There are different types of attractors that may be comprised of a single point, a set of points or a curve. Attractors in natural dynamical systems arise when forces that were causing the system to change dissipate, causing the system to move towards its natural state [Sprott et al., 2013]. There exist different types of attractors, a fixed point attractor arises when a system reaches a state that maps directly onto itself. Strange attractors result from chaotic dynamical systems [Lorenz, 1963, Grassberger and Procaccia, 1983]. Random boolean networks (RBNs) reach a limit cycle attractor, where the state of the system falls into stable cycles [Kauffman, 1969].

4.3 Random Boolean Network

The random boolean network (RBN) was created as an early attempt to model the interactions between genes. The purpose of this was to explore the hypothesis that the metabolic stability found in multicellular organisms can result from randomly formed regulatory mechanisms, rather than 'precise programming' as a result of evolution [Kauffman, 1969]. This was attempted by modelling the interactions that occur between genes in biological organisms in order to establish whether such a model would behave with stability.

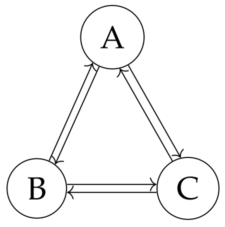
Genes within RBNs are represented as boolean devices, at any point a gene's state may be 1 or 0, analogous to the transcription of a gene being turned on or off in biological cells. The state of a gene is determined by its set of inputs, a fixed set of genes also existing within the network. The network executes across a series of time steps, at each step every gene will calculate its state based on its boolean function and the state of each of its inputs. The state of each gene at $T + 1$ is dependant on the states of its input genes at T. A network is initialised with a set of genes (of cardinality N), each with an assigned boolean function and a set of inputs (of cardinality K), neither of which will change after being assigned. An example of a fully connected RBN is illustrated by Figure 4.1.

The state of the RBN can be visualised as an array of the states of its constituent genes. The network is completely deterministic as a result of the fixed inputs and boolean function of each gene, meaning that for any state, the state following it will be the same every time. Inevitably, the RBN will reach the same state twice resulting in a cycle, the length of which is determined by the number of different states encountered during the cycle. By executing the network with all of its possible states, the number of potential cycles can be determined. Analysis of these cycles indicated that the relationship between the duration of cycles and the number of genes within the network was similar to that of the relationship between the approximate number of genes in various organisms and their cell replication times.

A criticism of the original RBN is that the synchronous updating of node states is not necessarily an accurate representation of how genes interact in nature. Synchronous in this sense referring to the fact that the states of all of the nodes are updated in the same order every time, resulting in the deterministic nature of the system. The asynchronous random boolean network (ARBN) is a modified version of the RBN where the states of each node are updated in a random order. It was found that the indeterministic nature of ARBNs meant that cyclic attractors were not reached [Harvey and Bossomaier, 1997]; the random order of updating nodes means that more than one state can result from a particular state. Loose attractors were observed where

A = B OR C
B = A AND C
C = A XOR B

(a) Transition rules



(b) Node connectivity

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

(c) RBN state t=0

a	b	c
0	0	0
1	0	0
1	0	1
1	0	1
0	0	1
1	1	1
1	0	0
1	1	0

(d) RBN state t=1

Figure 4.1: An example of a fully connected RBN with 3 nodes, each with 2 inputs (*N*=3, *K*=2). Tables c and d show the state of the network at 2 successive time steps.

the network cycled indefinitely through a subset of its states where each state in the subset, and every possible successive state was a part of that set.

4.4 Continuous Models

Random boolean networks are limited in that representation within the network is restricted to discrete values. When networks are applied to tasks in the real world, discrete representation becomes problematic as it is not possible to represent concepts such as movement through time and space as a set of discrete values without discretization, which reduces the accuracy of the data [Vijesh et al., 2013]. To address this limitation, the state of the network may be represented as a set of continuous values consisting of real numbers within a specified range [Lones et al., 2010]. The boolean functions that define the state transition rules of the network of RBNs are no longer applicable as they are unable to deal with continuous inputs. Both the regulatory function and connections between genes define the state transition rules of the network.

The artificial gene regulatory network (AGRN), also referred to as an 'artificial genetic network (AGN)' [Lones et al., 2010] brought the principles defined by the RBN into continuous space. Instead of representing the state of genes as boolean values, they are represented as real numbers referred to as an expression level. The expression level of a gene is determined by the output of a parametrised sigmoid function (Equations 4.2 and 4.1), the input being the weighted sum of the gene's inputs. Similarly to the RBN, the expression level of a gene at time $T+1$ is dependant on the state of its inputs at time T, this ensures that expression updating is fully predictable. Like the RBN, the AGRN operates in discrete time. The AGRN may be described as a tuple $\langle G, L, \mathsf{In}, \mathsf{Out}\rangle$, where:

G is a set of indexed genes $\{n_0 \dots n_{|N|} : n_i = \langle e_i, I_i, W_i\rangle\}$ where:

$e_i \in \mathbb{R}$ is the expression level of the gene

$I_i \subseteq G$ is the set of inputs used by the gene

W_i is a set of weights, where $0 \leq w_i \leq 1$, $|W_i| = |I_i|$

L is a set of initial expression levels, where $|L_N| = |N|$

$\mathsf{In} \subset G$ is the set of genes used as external inputs

$\mathsf{Out} \subset G$ is the set of genes used as external outputs

$$f(x) = (1 + e^{-sx-b})^{-1} \tag{4.1}$$

$$x = \sum_{j=1}^{n} i_j w_j \tag{4.2}$$

The AGRN operates in a series of discrete time steps as described in Algorithm 1. The expression levels of each gene are initialised randomly before the first iteration. When an AGRN is set to solve a task, the task is defined as a set of variables that describe the state of the task's environment, changes to the task's state can be influenced by an action (or multiple actions). At each time step, the expression levels of the input genes are set by the task state variables, then the expression levels of the genes in the network except those of the input genes are updated according to their inputs and regulatory function. The task actions can then be read from the external output genes in order to bring about a change in the task's environment.

Algorithm 1: Execute AGRN on a task

```
1:  for $x = 1 \rightarrow$ Number of iterations do
2:      if $x$ is 1 then
3:          Initialise genes randomly
4:      end if

5:      Scale task state variables to 0,1
6:      Map task state variables to input genes

7:      for $x$ to $numberofnodes$ do
8:          Initialise genes randomly
9:      end for

10:     Scale outputs to task range
11:     Map output genes to task actions
12: end for
```

In the AGRN, a sigmoid (Figure 4.2) is used as the regulatory function; it is a bounded function, so it can take a sum of any quantity of inputs, and produces an output value between a specific range. This is advantageous as a gene can take any number of input genes, so a variety of highly interconnected architectures may

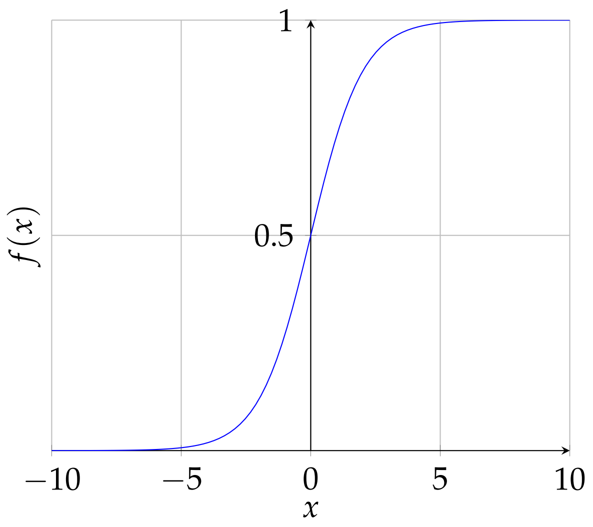


Figure 4.2: The sigmoid is often used as a regulatory function. Any real number may be used as an input to produce an output between the range 0–1.

form as opposed to the predefined number of inputs of RBNs. The sigmoid function is parameterised, allowing its behaviour to be tweaked as it is trained. The slope parameter s controls the gradient of the sigmoid to be changed, allowing it to function similarly to a step function. The offset parameter b controls the position of the sigmoid along the x axis.

4.5 Incorporation of Epigenetic Functionality

The artificial epigenetic network (AEN) is an extension of the artificial gene regulatory network involving the implementation of functionality inspired by biological epigenetic phenomena such as chromatin remodelling and DNA methylation.

In [Turner et al., 2012], continuous models of AGRNs were extended with the implementation of 'epigenetic frames'. Inspired by chromatin modification and DNA methylation, the epigenetic frames allowed the network to assign different genes to different task during evolution, so that the active genes could change during execution depending on the task they were solving. Each frame consists of a set of boolean values, each corresponding to a gene within network. The values within each frame determine whether a gene is active or not during a particular objective, whilst the network is solving a task. The networks were tasked with traversing a chaotic system in two directions, each direction corresponding with a task. Comparison between networks with epigenetic frames and networks without indicated that the networks with performed better than those without, indicating potential for the implementation of network functionality inspired by epigenetics. The epigenetic frames also provided additional information on the execution of the network, in some cases, not all of the inputs were required to solve a particular task, which was clearly indicated by the frames [Turner et al., 2013a].

The epigenetic frame functionality was limited in that they had to be activated at pre-specified points during execution in order to be useful. The idea of incorporating epigenetic inspired functionality was developed to develop a more realistic epigenetic control layer [Turner et al., 2013b]. The epigenetic layer in this case consisted of a set of 'epigenetic molecules', which form connections with genes in order to selectively deactivate them. The connections between genes and epigenetic molecules are derived based on their positions in a virtual one-dimensional reference space (see Figure 4.3). Each gene exists as a line segment with a centre point referred to as its 'identifier', if a gene's centre point is positioned within the space of another node, the former node acts as an input to the latter, influencing its expression level. Epigenetic molecules operate similarly to genes, but do not form expression levels themselves, instead using the value derived from its inputs to activate or deactivate genes in the network. If a gene's identifier is positioned within the space of an epigenetic molecule, the gene is activated or deactivated based on the value produced by the epigenetic molecule. This is advantageous when compared to the epigenetic frames as the objective of the network does not need to be explicitly broken down into tasks before the AEN is trained, so that the epigenetic frames can be created. Instead, the network may position the epigenetic molecules during training in order to partition the task itself, resulting in two major benefits; the first is that the network may be tasked with achieving unfamiliar objectives as they have the ability to break the objective down into tasks. Related to the first benefit, the partitioning may provide additional insight to the objective being solved as the active nodes at any point can easily be determined by analysing the epigenetic molecules. The epigenetic molecules also allow for a degree of manual external control; each molecule could be controlled by a switch instead of its input genes.

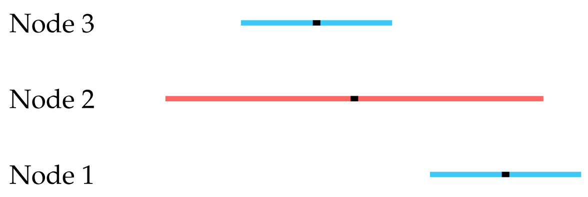


Figure 4.3: Illustration of small virtual reference space AGRN. Nodes 1 and 3 are inputs to node 2; node 2 is an input to node 3.

4.6 Summary

This chapter has introduced models that are based on biological gene regulation, see Chapter 2 for an overview of the underlying biological concepts. The history that led to the development of the artificial gene regulatory network (AGRN) has been described by introducing models such as the random boolean network and more advanced continuous models. The incorporation of epigenetic functionality in such models has also been described, as this has been shown to improve the performance and transparency of execution. AGRNs must be trained in order to exhibit useful functionality; Chapter 5 provides an overview of different optimisation techniques including evolutionary algorithms, which are used to train some of the AGRNs in this thesis. See Chapter 3 for an explanation of why AGRNs can be considered connectionist architecture.

AGRNs and their derivatives are used extensively during the experiments described within this thesis due to useful properties such as robustness. In Chapter 7, the developmental properties of AGRNs are investigated in an attempt to find predictable evolutionary pathways that result in more performant networks. An investigation into the structural properties of evolved AGRNs is performed in Chapter 8 to determine whether modularity occurs consistently enough; the purpose of this is to determine whether it is feasible to improve the transparency of execution of such models based on this knowledge. A control layer inspired by epigenetic mechanisms is developed in Chapter 9; it operates very similarly to AGRNs, but instead operates on an external connectionist architecture in an attempt to improve its transparency. AGRNs are a type of artificial intelligence, Chapter 6 explains the importance of transparency by providing a background in explainable artificial intelligence.

Chapter 5

Optimisation

The models used within the experiments described later in this thesis must be trained in order to solve the task that is being attempted. This training constitutes an optimisation problem, as it involves searching for the model parameters that would ideally allow the task to be solved in the best possible way. In this chapter, optimisation techniques will be discussed that could potentially be used to train, or in some cases evolve the AGRN and epigenetic models.

Broadly speaking, optimisation is the process of making the best out of a situation or thing. The use of optimisation within this thesis refers to the process of attempting to find the best possible solution to an optimisation problem. Mathematical optimisation consists primarily of an objective function, a set of variables, and a set of constraints. In this case, the objective is to either minimise or maximise the output of the objective function by changing the variables, which may be subject to constraints. A typical example of a continuous optimisation problem is given in [Coello Coello et al., 2020], and has been included below:

Minimise $f(x) := [f_1(x), f_2(x), \ldots, f_k(x)]$

Subject to:

$g_i(x) \leq 0 \quad i = 1, \ldots, m$

$h_i(x) = 0 \quad i = 1, \ldots, p$

In this example, x is a vector over which the objective function f must be minimised, $g_i(x) \leq 0$ are the inequality constraints, and $h_i(x) = 0$ are the equality constraints.

Training a neural network is an optimisation problem, where the objective function is often the difference between the predicted and desired output, which must be minimised by altering the node parameters. Similarly, the evolution of an artificial gene regulatory network (AGRN) by a genetic algorithm is an optimisation problem, where the performance of the AGRN, determined by a fitness function, must be maximised by changing the gene parameters.

5.1 Optimisation Techniques

The optimisation techniques described within this section belong to a broad category of algorithms termed 'heuristics'. Within the field of computer science, a heuristic refers to a technique of approximating the solution to a problem. They are often used when it would be unreasonable to reach an exact solution due to time or resource constraints. Many categories of heuristics exist, each making different trade-offs in order to reach an approximate solution in a reasonable time frame. Complex problems and those that are not well understood are ideal candidates for heuristic algorithms.

Local search algorithms are a type of relatively simple heuristic algorithms that involve iteratively traversing the search space of a problem by exploring neighbouring solutions, starting with a randomly chosen candidate solution [Pirlot, 1996]. Local search algorithms are suitable for problems where a 'neighbourhood relation' can be defined of the candidate solutions, meaning that for a given candidate solution, there exist neighbouring candidate solutions that exhibit a minor change in characteristics. The hill climbing algorithm is an example of a local search.

5.1.1 Hill Climbing Algorithm

The hill climbing algorithm (Algorithm 2) is a local search optimisation technique that is used to iteratively improve upon an arbitrary solution to a problem by making changes to it. The operation of a hill climbing algorithm will be described with respect to a problem that requires a solution consisting of a number of continuous values. The starting solution is created by randomly generating a set of continuous numbers, the performance of which is then determined. During each iteration, a small change is made to one of the values by adding or subtracting a number from it. If the change results in a better solution the change is retained, otherwise the solution is reverted to its previous state. This continues until the solution is deemed acceptable or a time constraint has been reached.

Algorithm 2: Hill Climbing

```
1:  $S \leftarrow$ INIT() {Generate initial state}
2:  $T \leftarrow S$ {Copy initial state}
3:  $B \leftarrow$ EVALUATE(S) {Store initial best performance}

4:  repeat
5:    ALTER(S) {Traverse search space}
6:    $C \leftarrow$ EVALUATE(S)

7:    if $C > B$ then
8:      $T \leftarrow S$ {Replace with improved state}
9:      $B \leftarrow C$ {Update best performance}
10:   else
11:     $S \leftarrow T$ {Restore state to that before alteration}
12:   end if
13: until $B >$ Goal performance OR Time constraint reached
```

Benefits

The greatest benefit of the hill climbing algorithm is its simplicity; only a single state needs to be stored along with the change made, so that it can be reverted. In addition, the implementation is simple so it can be applied to many problems assuming they can be represented appropriately. The simplicity of the algorithm results in much faster execution, meaning that it is possible for it to reach a better solution than more complex solutions when the time available to execute is limited, as more iterations can be performed.

Challenges

Due to the nature of hill climbing, it is possible to reach a local maximum where the current state of the solution is better than its neighbouring states, but there exists other non-neighbouring states that would provide a better solution. This can be mitigated with backtracking by storing the previous states of the solution so that their unexplored neighbouring states can be explored.

The search space of the problem may contain flat areas, where all neighbouring states result in a solution that performs no better than the current. This essentially 'traps' the solution as there is no way in which it can improve as it is repeatedly reverted to the same initial state. This can be mitigated by making changes that differ in magnitude, allowing single changes of greater magnitude to explore more of the search space.

Ridges may be present in the search space that are not aligned with an axis, meaning that two or more parameters may need to be changed in order to ascend it. Only a single value may be changed at a time during hill climbing, resulting in difficulty when ascending a non-axis-aligned direction as the solution will have to slowly ascend each of the axes at each iteration by changing each of the respective parameters at once.

5.1.2 Evolutionary Algorithms

Evolutionary algorithms [Bäck and Schwefel, 1993] are a type of optimisation algorithm used to 'evolve' a solution to a problem. Genetic operators inspired by processes such as reproduction and mutation are applied to a 'population' of individuals that each represent a potential solution to a problem. Evolutionary algorithms are applicable to many types of problems, as the implementation is loosely coupled to the problem itself. There are different types of evolutionary algorithms that differ in their implementations and solution representation.

Genetic programming [Koza and Poli, 2005] is a type of evolutionary algorithm where solutions are modelled as computer programs, often in the form of a tree structure. Each leaf node contains an operand, and each internal node an operator. The structure is iteratively changed using genetic operators until an adequate program is created. Genetic programming is typically used to solve computational problems. Neuroevolution is similar to genetic programming, but is used to generate an artificial neural network. The solutions are modelled as aspects of the ANN, such as the parameters and topology. Different neuroevolution techniques exists that evolve different aspects of the ANN.

The experiments described in chapters 8 and 9 utilise genetic algorithms to evolve AGRNs and an epigenetic control layer, a technique where solutions are typically represented as strings of numbers. Genetic algorithms are applicable as the AGRN and control layer consist of nodes containing different types of parameters, that can be encoded into the strings. Genetic programming and neuroevolution are not utilised within this thesis, but have been defined to mitigate the confusion between them and genetic algorithms.

5.1.3 Genetic Algorithms

Genetic algorithms are a type of heuristic that are used to find solutions for search and optimisation problems [Holland, 1992]. They are inspired by the process of natural selection in evolution, where traits that increase an organism's probability of survival are more likely to be passed onto its offspring, thus increasing the prevalence of desirable traits.

Genetic algorithms (GAs) are typically used to evolve AGRNS [Taylor, 2004, Turner et al., 2013b] due to their flexibility. The topology of AGRNs can change during training, and whilst there are fewer nodes, nodes are more complicated that nodes found in other connectionist architectures such as artificial neural networks (ANNs) which are trained by backpropagation. Many of the advantages of AGRNs over ANNs would be lost if they were to be trained via backpropagation, such as the ability for the topology of the network to change as it evolves, as the inputs to a node are not fixed. Backpropagation is limited to changing weights and biases based on the error of the network, the nodes within AGRNs contain multiple parameters including those relating to the regulatory function. Genetic algorithms can change an arbitrary number of parameters of varying data types, making them suitable for training (or evolving) AGRNs.

Genetic algorithms evolve a population of potential solutions through a series of time steps referred to as generations. The population of a genetic algorithm is a set of initially randomised potential solutions to a problem, each member of the population is compared using a fitness function to measure its performance at solving the problem. Members are defined as a set of parameters, which the GA may randomly change or swap amongst members. Operations will be performed on the population in order to propagate members and parameters that are the most likely to improve the fitness of the population into successive generations. A selection algorithm utilises the fitness function to prioritise members of the population that are more successful at solving the objective.

Genetic Operators

In order for the population to advance and improve, genetic operators are applied at each population. The operators are usually applied to a subset of the population, usually chosen either by a fitness function or at random. Ideally, genetic operators are applied in such a way that the population slowly improves over time.

Mutation is essential during biological evolution as it results in genetic variation. Mutation occurs when the DNA of an organism is changed, which may result in a new trait; if the trait is beneficial to the survival of an organism, the organism is more likely to pass the trait on to its offspring. When the mutation operator is applied during the execution of a GA, a percentage of the population is selected at random. The chosen members will be subject to random, often small parameter changes that may result in new behaviour. If the parameter change improves the member, this will be reflected in its fitness function.

Recombination is the process that causes an organism to inherit a combination of traits derived from both parents. In diploid eukaryotic organisms, recombination occurs during meiosis causes gametes to contain both maternal and paternal genetic information [Clancy, 2008b]; this results in genetic diversity that facilitates the adaptation of a population to its potentially changing environment. The recombination operator acts to combine traits between members of the population, the intention being to produce members that have a combination of useful traits. The operator involves selecting two members of the population to act as parents; the parents are copied, and some of the parameters are swapped in order to produce two offspring members. The offspring members will form a part of the next generation. A selection algorithm is used to select the parents based on their fitness function, to increase the probability of retaining beneficial traits.

Elitism is an operator that selects a subset of the population with the highest fitness function and copies them directly to the next generation. This ensures that the best performing members are not lost due to random chance. Elitism is applied before other operators to prevent changes to the fittest members, such as those resulting from mutation that could be detrimental to the performance of the organism.

Algorithm 3: Evolving an AGRN with a GA

```
 1: $P \leftarrow \{\}$ {Initialise empty initial population}

 2: for $x = 1 \rightarrow$ Population size do
 3:     $P \leftarrow P \cup$ Randomly initialised AGRN
 4: end for

 5: for $y = 1 \rightarrow$ Number of generations do
 6:     for all $p \in P$ do
 7:         EVALUATE(p)
 8:     end for

 9:     $Q \leftarrow \{\}$ {Initialise empty child population}
10:     $Q \leftarrow Q \cup$ Elite members
11:     repeat
12:         if recombine then
13:             $Q \leftarrow Q \cup$ Recombine(P)
14:         else
15:             $Q \leftarrow Q \cup$ TOURNAMENT_SELECT(P)
16:         end if
17:     until $|Q|$ is Population size
18:     $P \leftarrow Q$

19:     Mutate population
20: end for
```

Multi-objective Genetic Algorithms

The genetic algorithms described so far have been concerned with optimising a single fitness function. In some cases, it may be desirable to optimise multiple functions simultaneously, for more complex problems requiring solutions that consider a range of criteria. Conventional genetic algorithms consider the population member with the highest fitness function to be the most performant, whereas multi-objective genetic algorithms must account for multiple objectives simultaneously [Murata et al., 1995]. Some genetic algorithms rank their population members based on the concept of domination. A member is considered to dominate another based on two criteria, it must achieve a better performance in at least one objective, and perform no worse in any of the other objectives.

When evolving a solution to a multi-objective problem, it is desirable to maintain a population of diverse members, rather than focussing solely on the best member. A more diverse population improves the chances of beneficial characteristics developing, resulting in more performant members. For a multi-objective problem, there exists a set of nondominated solutions called the pareto optimal set, where no improvement can be made towards solving one objective that doesn't compromise another. The pareto-optimal set forms the pareto front, from which solutions with varying compromises can be chosen in order to solve the objectives. The NSGA-II algorithm [Deb et al., 2002] is a multi-objective selection function that has been used to evolve artificial gene regulatory networks in previous work [Turner et al., 2013b].

Benefits

Genetic algorithms are applicable to complex combinatorial problems that require solutions consisting of different types of variables. It is only necessary to understand the problem to a degree where the solution can be defined as a set of variables. The implementation of genetic algorithms is relatively simple, requiring little mathematical knowledge or a deep understanding of the problem.

The recombination operator allows traits from two members to be combined, and mutation allows for small random changes to result from the population. This allows a lot of the search space to be covered throughout execution, rather than relying on small changes to single parameters as is the case with other optimisation techniques.

This also mitigates the issue of being stuck at a local optima as is the case with some other optimisation techniques.

Genetic algorithms may be executed in parallel as the evaluation of the fitness functions of members are calculated without respect to each other. This results in efficient execution even with high population sizes, if the computational resources are available.

Challenges

Genetic algorithms allow a wide range of the search space to be explored, but do not guarantee an optimal solution. Genetic algorithms are applied to combinatorial problems where derivatives are not available for precise calculations. As the complexity of the problem increases, the number of parameters that must be changed increases and it becomes less likely that the optimal solution will be found.

The ability of the genetic algorithm to solve a problem relies on the fitness function and the constraints applied to the potential solutions. If the fitness function is not representative of the ideal solution, the resulting solution will not be optimal. The constraints applied to solutions must be carefully chosen so that invalid solutions are not created if avoidable, and so that all potential solutions may be represented.

The parameters of a genetic algorithm, such as the population size, crossover probability, and mutation rate influence how the genetic algorithm performs. The selection of the parameters influence how the genetic algorithm operates in order to evolve a solution, directly affecting how the population members are initialised, altered, and carried through to the next generation. The parameters must be chosen carefully in order to ensure that a solution is iteratively evolved by retaining beneficial characteristics, and discarding those that are not useful. The selection of genetic algorithm parameters could be considered an optimisation problem in itself, as there is no definitive rule on what constitutes ideal parameters.

5.2 Chosen Optimisation Techniques

In this chapter, a multitude of optimisation techniques have been described and critiqued due to their applicability of training gene regulatory network models. The experiments described in the chapters that follow make use of some of these optimisation techniques in order to produce functional artificial gene regulatory networks. The optimisation techniques used within the experiments described within this thesis will now be described briefly.

An optimisation technique similar to the hill climbing algorithm is used in the experiments of Chapter 7. The purpose of the experiments is to observe the evolutionary process of the AGRNs, rather than trying to achieve the best performance. More specifically, the implications of mutating specific genes is to be observed, and for this reason, it is essential that small changes are made at each time step of training. The optimisation technique chosen involves mutating 3 genes at a time, and then evaluating the network in order to determine whether the changes were beneficial or not. The mutations will be reversed if the mutations are not beneficial.

The experiments described in Chapter 8 involve the training of an artificial gene regulatory network to solve multiple independent tasks concurrently. Based on the requirement that multiple tasks are solved, it may seem logical that a multi-objective genetic algorithm is used; however, the purpose of experimentation is to observe whether the AGRNs are capable of evolving modularity in their internal structure, rather than attempting to obtain the best possible performance. The ability for AGRNs to develop modularity with a single objective genetic algorithm will be investigated, to demonstrate whether or not they are capable. In addition, the genetic algorithm used will employ only the mutation and elitism operators, and not the crossover operator. The crossover operator will not be used as it is likely to disrupt the topology of the AGRNs, as the connections between nodes are determined based on their proximity to each other, and their positions are represented by parameters optimised by the genetic algorithm.

In Chapter 9, a control layer inspired by epigenetics that operates similarly to an AGRN will be used in an attempt to improve the transparency of a trained artificial neural network (ANN). In these experiments, the connections between nodes are defined discretely, rather than being based on proximity. For this reason, the crossover operator is not likely to be detrimental to the evolutionary process, and may in fact be beneficial as it could allow useful functionality to be shared between population members. There is only a single objective, which is to remove as many nodes from the ANN as possible without reducing its performance. For these reasons, a single objective genetic algorithm with crossover, mutation, and elitism operators will be used to evolve the control layer.

Chapter 6

Explainable AI

Transparency of execution is referred to throughout this thesis; it refers to the degree to which AI behaviour can be understood by an observer. Much of the motivation behind the experimentation conducted within this thesis can be attributed to the desire to improve the transparency of machine learning models such as artificial gene regulatory networks (Chapter 4), and deep neural networks (Chapter 3). Explainable artificial intelligence (XAI) is a field that is related to this cause, which will be described within this chapter. The necessity of the field of XAI will be described, followed by an overview of some existing XAI methods.

The pervasiveness of artificial intelligence in society is clear. Users of modern smartphones have access to virtual assistants that interpret natural language in order to perform tasks such as web searches and calendar organisation [Kepuska and Bohouta, 2018, Garcia and Lopez, 2018]. Search engines may utilise artificial intelligence in order to determine the relevance of content when queried by a user [Dupret and Liao, 2010, Ye et al., 2016]. Self driving cars utilise optical data [Bojarski et al., 2016] and data from sensors such as Lidar [Hecht, 2018] to navigate urban environments. As AI becomes more prevalent, the importance of being able to analyse such systems becomes more and more important. The field of explainable AI, shortened to 'XAI', attempts to address the problems that occur when an AI system must be inspected.

6.1 The Black Box Nature of Deep Learning Networks

Deep neural networks (see Section 3.5) are applicable to many domains due to their abstract nature. Their functionality arises from the interactions between nodes that

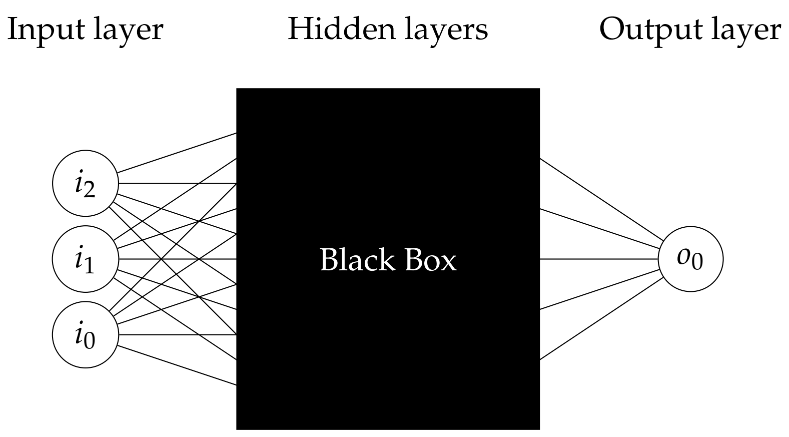


Figure 6.1: Illustration of an artificial neural network as a black box system.

are represented by weights and a bias, the totality of which make up the parameters of the DNN. The parameters of a DNN are essentially matrices of real numbers that correspond with the nodes within each layer of it. Nodes interact using their activation functions; they are very general, taking a sum of the outputs of other nodes within the neural network and process them to produce a single output. The behaviour of the network is determined almost wholly by its parameters and the configuration of its layers, including the number of nodes within them, the connections they make, and their activation functions. See Section 3.4 for an explanation of how neural networks are trained.

Neural networks take a set of inputs, and process them using layers of neurons. They consist of multiple layers of neurons. The input data is processed by the hidden layers, and then the output layer that is observed in order to obtain the decision that the network is made. Neural networks are described as being 'black box' systems as the processing that occurs within the hidden layers is difficult to understanding by simply observing it, this is illustrated in Figure 6.1.

6.2 Why is XAI important

Deep learning models are used to make decisions, a simple example of this is an image classifier that attempts to recognise the features of an image in order to determine what category it fits into. The possibility of error in this case may not seem particularly consequential per se, but consider the application of it in the context of a self-driving car, where it may be used to detect pedestrians. Incorrect classification in this case could result in the system not recognising the pedestrian and not applying the relevant safety procedures. Neural networks do not inherently provide a method of showing why their decisions are made, which is problematic for reasons that will now be discussed.

6.2.1 Obtaining Reasoning Behind Decisions and Justification

The context of the input data is not understood by a deep learning model; therefore, it is possible that the model may conflate features of the dataset during training. Within datasets, particularly those with limited sample size, it is possible that there exist misleading trends that do not accurately describe the domain [Suresh and Guttag, 2019]. This becomes troublesome when the model is deployed, as the network may operate based on these trends and produce incorrect output. An example of this occurrence was found when a neural network was used to predict the risk of pneumonia in patients, where it was found that the network considered the fact that patients with asthma had a lower risk of contracting it [Caruana et al., 2015, Adadi and Berrada, 2018], when in reality this is not the case.

The ability to assert that the network was making this assumption was important here, and would be if the network were to be used by a healthcare professional to aid with diagnosis, as they would be able to identify the fallacy and potentially disregard the model. Because the model does not understand the context behind of the dataset, as it has no prior domain knowledge, it is crucial that the reasoning behind its decisions are apparent.

6.2.2 Maintenance and Development

Despite the abstract and general nature of deep neural networks, their architecture must be configured via the definition of the number of layers, the number of nodes within each layer, the activation functions of each node, and node connectivity. There exist methods for defining the architecture, such as the use of evolutionary algorithms [Benardos and Vosniakos, 2007] and reinforcement learning [Baker et al., 2016], but the ability to understand how an existing neural network is currently functioning could provide insight on how to improve the architecture in order to produce a successor.

During the life cycle of an AI system, particularly soon after it has been deployed, it is possible that situations will arise that it was not adequately trained for. In these cases it is desirable to be able to identify the weakness in the system, so that it can be rectified.

6.2.3 Providing New Insights

The capabilities of modern hardware permit AI systems to be trained with large volumes of data [Russom et al., 2011]; this data can be automatically processed to develop a generalised understanding of what it is representing. Such AI systems may surpass the performance of existing methods in a particular field by learning trends that have not yet been discovered. In such a case, it would be useful to be able to analyse the AI system in order to identify how the trend was found.

6.2.4 Compliance to Legislation and Accountability

As artificial intelligence is becoming more prevalent, it is inevitable that undesirable situations will occur where it is possible that the AI system is to blame. These situations may cause damage to property or bring harm to people, in which case it is likely that the people involved will want to know what is at fault, and it must be possible to analyse the AI system to determine whether it was in error. This is necessary in situations where legal action or prosecutions may be made against the person or party responsible for the incident. XAI benefits the designers of a legitimate AI system as it allows them to validate its decision making process, and benefits society as it ensures that AI systems can be held accountable when necessary to do so, and promotes the development of safe AI.

6.3 Existing Methods of XAI

The pervasiveness of AI has prompted many attempts at improving the understanding of how such systems operate. Such attempts exhibit different characteristics, such as the scope of the explainability and the type of interactions they have with the AI systems that they are attempting to explain. XAI methods may operate at a local or global scope, globally scoped XAI attempts to analyse the operation of the entire model, whereas locally scoped XAI analyses a specific decision or operation that the model has performed. There exist methods that have been designed for specific models, and those that are more general and are capable of analysing a multitude of models.

6.3.1 Neural Network Pruning

Pruning an artificial neural network involves removing nodes or node connections with the intention of decreasing the overall size of it. This has the potential to reduce the computation time and the space in memory that the network occupies. A simple way of achieving this is to assume that parameters of low magnitude are less useful than those with higher magnitudes, and prune the weights with the lowest magnitudes.

The idea of pruning weights based on their magnitude was compared to a method where parameters were instead pruned based on their saliency by a technique called 'optimal brain damage'. The saliency of a parameter refers to how much it influences the objective function when changed, and is calculated with a method similar to backpropagation. When pruning networks with this method, it was found to be capable of reducing the size of them by a factor of four, and the network performance was less negatively affected than when pruning based on magnitude [LeCun et al., 1990]. This suggests that the assumption of parameters with lower magnitudes being less useful is not correct in all applications.

Convolutional neural networks have been pruned by removing filters that were determined to be the least useful [Li et al., 2016]. The paper describes methods of pruning the networks by considering a single layer at a time, and the entire network. To prune filters in a single layer, they were ranked based on their importance by calculating the sum of their absolute weights. The least important filters were pruned along with their feature maps, and replacement kernel matrices were constructed. The network may then be retrained. The paper also describes methods for pruning filters by taking already pruned filters into account. The methods described resulted in CNNs that were approximately 30% smaller with minor reductions in accuracy.

Another example of pruning the connections of neural networks is given by the three-step method described in [Han et al., 2015]. The network is initially trained in order to determine which connections are important based on the magnitude of their weighting, the unimportant connections are then pruned and the network is retrained. If a node has zero input, or zero output connections, it is removed from the network as it does not influence the output. The three step process is repeated iteratively, without the reinitialisation of parameters to maintain the integrity of learned features. This technique was used to reduce the number of connections in convolutional neural networks such as AlexNet by a factor of 9–13.

6.3.2 Lottery Ticket Hypothesis

The 'lottery ticket hypothesis' states that a randomly initialised ANN contains a subnetwork that can be retrained to match the accuracy of the original network [Frankle and Carbin, 2018]. The paper proposes a method for discovering such subnetworks by training an ANN, pruning the parameters, and retraining the remaining parameters after resetting them to their initial values. This process is repeated iteratively. When the pruned network parameters were randomly re-initialised, it was found that they trained more slowly and were less accurate to when they were set to their original values, suggesting that their initial weights are important.

6.3.3 Sensitivity Analysis and Layer-wise Relevance Propagation

Sensitivity analysis (SA) involves the observation of how the uncertainty of an independent variable influences the uncertainty of the entire system [Saltelli et al., 2004]. In simple terms, it is the observation of how changes to the parameters of a system affect the output of it. It is a very general method that is often applied to complex systems where it is infeasible to analyse the system as a whole. This can be applied to neural networks by evaluating the gradient for a specific data point, the output of the network is then observed as the network inputs are changed [Montavon et al., 2018]. The inputs that have the most affect on the output when changed are considered to be the most important in this case. This technique was used on a convolutional neural network in order to determine the importance of the different architectural components of it [Zhang and Wallace, 2015].

Layer-wise relevance propagation (LRP) is a technique somewhat similar to SA in that network inputs are ranked based on their importance for a single network decision; however, LRP uses local redistribution rules to propagate the network prediction backwards, in a process similar to backpropagation [Bach et al., 2015, Montavon et al., 2019]. SA and LRP were compared when applied to convolutional neural networks that classified images and text documents in [Samek et al., 2017]. With the image classification task it was found that heat maps produced by LRP indicated more significant features in the images, and the heat maps generated from SA were much more noisy. When perturbation analysis was applied to the inputs ranked by both of the methods, it was found that perturbing the inputs indicated by LRP resulted in a quicker decrease in network performance. When applied to the text document classification, LRP was capable of detecting words that contradicted the class that was predicted, whereas SA was not capable of this, and only indicated the relevance of words that contributed towards the class prediction.

DeepLIFT (deep learning important features) is a technique that is used to determine the importance of each of the inputs of a neural network for a specific data point. It improved on LRP by following the chain rule like gradients, allowing it to distribute the net relevance to sections of the images in the MNIST dataset [Shrikumar et al., 2017].

6.3.4 LIME

Local Interpretable Model-Agnostic Explanations (LIME) [Ribeiro et al., 2016] is a technique where specific data points are perturbed and fed back into a machine learning model. The resulting outputs of the model are then observed in order to understand how each perturbation influences the model's decision of that particular data point. For example, in an image classifier, certain groups of pixels could be hidden in order to determine the parts of the image that the model is basing its decision on [Palatnik de Sousa et al., 2019].

6.3.5 Ablation

Ablative brain surgery, or lesioning, is a technique used to selectively destroy tissue within the brain [Franzini et al., 2019]. The effects of this are permanent and causes lesions. It has been used for the treatment of cluster headaches [Narouze et al., 2009] and neurological conditions such as Parkinson disease [Jankovic, 2001]. Experimental ablation is a process conducted on animals to learn about the functionality of the brain.

During experimental ablation, a part of the brain is lesioned and the behaviour of the animal is observed for changes. If the behaviour of the animal changes consistently when a particular part of the brain is lesioned, it is likely that the lesioned part of the brain is responsible in some way for that behaviour. Experimental ablation has been used to correlate specific rat behaviour with hemispheres of their brains [Garbanati et al., 1983], and to discover the importance of the forebrain for hue discrimination in fish [Bernstein, 1961].

Experimental ablation inspired an attempt at improving the transparency of a convolutional neural network tasked with moving a metal loop across a wire without touching it [Lillian et al., 2018]. The networks were trained to navigate 24 difference wire paths, the second to last layer of each network was split into 10 groups, allowing the average function of the nodes to be derived by later comparing the outputs of the network as each group is ablated. The behaviour of each network was monitored with respect to two specific movements to establish a baseline, which was compared to the behaviour of the network when each of the 10 groups was ablated by setting the parameters to zero. The outputs of each of the ablated networks were grouped using k-means clustering across different wire paths, and it was found that 6 clusters best represented the data, suggesting that there are 6 different ablation groups. Ablation in ANNs has also been used to measure the reliance of a network on a particular node or feature map by measuring the change in performance in response to ablation [Morcos et al., 2018, Amjad et al., 2018].

6.4 Improving the Transparency of AENs

The concept of transparency is relevant in the context of AENs from two points of view, as previously discussed the ability of an AEN to break an objective down into subtasks can provide insight into the dynamics of the system one is working with. Transparency is also relevant in the context of improving the AEN itself. AENs provide approximate solutions to problems of varying accuracy, to improve the accuracy of an AEN it must be possible to analyse it in order to determine why cases of incorrect output occur.

The work described in [Turner and Dethlefs, 2017] introduces methods for analysing networks with respect to the activity of epigenetic molecules causing topological changes that influence the output of the network. A simple algorithm for purging networks of nodes that do not provide useful functionality is also introduced (Algorithm 4), where the network is simulated with each node removed at once, if the network does not decrease in performance the node is permanently removed from the network. An AEN is tasked with processing sensor data to detect the state of a person's eyes. When the network makes an incorrect decision, the network is stripped down and analysed specifically at the time steps at which the error occurred. The activity of an epigenetic molecule that links a particular input gene to a gene whose expression correlates with the incorrect decision is identified as the cause of the error. With additional work there is potential for the generalisation of such a method in order to provide real time analysis of network decisions.

Algorithm 4: Purge network of redundant nodes

```
1:  $t \leftarrow$ Threshold
2:  $b \leftarrow$ Simulate network {Base performance}
3:  $N \leftarrow$ Network nodes

4:  for all $n \in N$ do
5:      $N \leftarrow N \setminus \{n\}$ {Remove node from network}
6:      $p \leftarrow$ Simulate network {Performance without node}
7:      if $p < b - t$ then {Performance decreases}
8:          $N \leftarrow N \cup \{n\}$ {Replace node}
9:      end if
10: end for
```

6.5 Summary

This chapter has provided an overview of the field of explainable artificial intelligence (XAI). Justifications have been provided of the importance of XAI when AI systems are used in the real world. The black-box problem has been described, which is applicable to artificial gene regulatory (AGRNs) networks (see Chapter 4) and artificial neural networks (ANNs) (see Chapter 3). Examples of existing XAI solutions have been described that can be applied to AGRNs or ANNs. AGRNs and their derivatives, as well as ANNs are the subject of the experiments described later in this thesis.

The overarching aim of the work conducted within this thesis is to improve the transparency of execution of connectionist architectures. Chapters 7 and 8 describe experiments that involve an investigation into the developmental and structural properties of AGRNs in the hope that such knowledge can be exploited to improve the transparency of execution and performance of such models. In Chapter 9, an attempt at improving the transparency of execution of a trained ANN is made by applying an external control layer.

Chapter 7

Constrained Evolution

7.1 Understanding Evolutionary Constraint in Artificial Gene Regulatory Networks

Evolutionary constraint is a phrase used to describe restrictions and biases that apply to the evolutionary pathway of a biological organism [Hansen, 2015]. Organisms must be capable of adapting to their environments in order to survive, this is achieved by natural selection, where organisms possessing characteristics that enable them to thrive in their environment are more likely to survive and pass their characteristics onto their offspring [Hendry, 2005]. Some traits are essential for their survival, whilst others are less important.

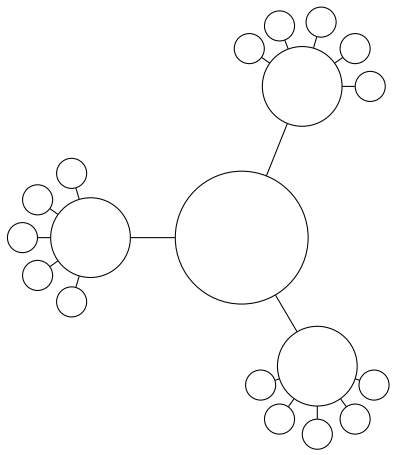


Figure 7.1: Diagram of gene interactions modelled as a hub.

Figure 7.1 illustrates how a set of genes may interact with each other. The centre node connects directly to three outer nodes, which in turn directly connect to many other nodes. The centre node can be thought of as a gene that facilitates interactions between the three outer nodes, each facilitating interactions with their respective set of outermost nodes. A mutation to the centre node is likely to alter the functionality of the hub as a whole, which could be detrimental to the overall functionality of it. Mutations to the three outer nodes are likely to alter the functionality of the hub to a lesser degree, as they connect only to their set of the outermost nodes and the hub. Mutations to the outermost nodes are likely to cause much smaller alterations in functionality, as they have only a single connection each. There is a sweet spot where mutations can occur that will produce beneficial changes without disrupting the function of the rest of the network.

The experiments within this chapter have been conducted in order to ascertain whether the evolution of an AGRN with regard to gene connectivity is subject to constraint similar to that in biological organisms. The knowledge of such constraints could then be used to improve the performance of resulting AGRNs by subjecting the evolutionary process to rules based on this. The purpose of this experimentation is to provide insights into how the understanding evolutionary constraint can be applied to the evolution of AGRNs, and how this can be used in order to improve the efficacy of such evolution.

7.2 Evolving AGRNs with Constraint

The experiments in this section involve the evolution of a set of AGRNs that will be analysed in order to determine whether the evolution of such networks are subject to evolutionary constraint. The AGRNs will be evolved to solve two separate tasks, so that the evolutionary pathway can be examined concerning the mutation of genes, and their connections. A set of rules will be generated based on the first set of experiments with the aim of determining the genes, which when mutated, are most likely to produce beneficial changes based on the number of their connections. Another set of AGRNs will then be evolved, this time with rules applied in order to determine whether the application of such constraints is beneficial.

7.2.1 Evolution of the AGRNs

The focus of experimentation is to observe how often a gene mutation results in a positive outcome with respect to the number of connections it has. To achieve this, a single AGRN will be slowly evolved using an iterative process involving mutations to the AGRN parameters. Genetic algorithms are typically used to evolve AGRNs as they are capable of producing AGRNs with a wide range of attributes; however, the purpose of the experiment is to observe the evolutionary process of an AGRN, for which a simpler algorithm is more suitable.

The algorithm (Algorithm 5) chosen to evolve the AGRNs begins with a randomly initialised AGRN; the fitness of the initial network is then calculated based on the amount of times the task is solved. Then the network is cloned and three genes are selected for mutation. A random parameter of each gene will be selected and mutated to a random value subject to range and type constraints. Further details on how the genes were mutated can be found in Section 7.2.5. The parameters of a gene pertain to its connectivity, regulatory function and weight, and are shown by Table 7.1, the 'Input' parameter consists of a boolean array with a length of 20, each boolean value represents a different gene. Both tasks are evaluated based on how many times they solve the tasks; the criteria for this is different based on each task, and is described in more detail in sections 7.2.3 and 7.2.4. If the performance of the mutated AGRN is better than the initial network, the mutated network replaces it. This process is repeated for a number of iterations. The algorithm chosen is similar to the hill-climbing algorithm, the primary difference being that the AGRN is randomly mutated in an attempt to reach the optimum solution, rather than tweaking a parameter (or set of parameters) to climb a hill.

Table 7.1: The variables held within each gene of an AGRN.

Name	Type	Range
Expression	Real	[0,1]
Weight	Real	[0,1]
Slope	Real	[0,20]
Sigmoid offset	Real	[0,1]
Input	Bool Array	[0 | 1,0 | 1,0 | 1. . .]

Algorithm 5: Evolving an AGRN using Iterative Mutation

```
1: $N \leftarrow INITIALISE\_AGRN()$ {Randomly initialise an AGRN}
2: $P = EVALUATE(N)$ {Obtain initial fitness}

3: for $i = 1 \rightarrow$ Number of Epochs do
4:     $M \leftarrow N$ {Clone AGRN}
5:     MUTATE(M)
6:     $F = EVALUATE(M)$
7:     if $F > P$ then
8:         $N \leftarrow M$
9:         $P \leftarrow F$
10:    end if
11: end for
```

7.2.2 Tasks

The evolutionary process of the AGRNs will be recorded as they are used to solve the tasks. Two sets of AGRNs will be evolved, one for each task; the AGRNs in the sets will act as repeat experiments. The parameters of the tasks will be changed each time they are solved by the network to ensure that the evolutionary process will occur continuously until sufficient data has been acquired. The AGRNs will have to adapt according to the changing evolutionary pressure, so that the tasks can be solved as their parameters change. The performance of the AGRNs is measured based on the amount of times the task is solved after its attributes are changed. The fitness of an AGRN is calculated as the number of times is successfully completes its task as a raw number. The AGRNs solving both tasks contain 20 genes including any that are used for mapping task variables.

7.2.3 Coupled Inverted Pendulum Task

The coupled inverted pendulum task consists of series of carts on a one-dimensional track, attached to each of the carts is a pendulum that may rotate about the point at which it is attached [Hamann et al., 2011]. This task has been chosen due to the chaotic nature of the double pendulum's dynamics, requiring the AGRNs to develop complex behaviour in order to control and stabilise the movement of the pendulums. The carts must be controlled so that each of the pendulums swing upwards, at which point they must be stabilised so that they remain balanced. Each of the carts is attached to its adjacent carts via a chain that constrains the distance between the carts. Each cart may move independently from the others, but may not pull on another cart via the chain, move into another cart, or touch the walls of the environment. For the purpose of this experiment, the task has been simplified to include a single cart and its pendulum.

The state of the task's environment is described by a set of observation variables described in Table 7.2, corresponding to virtual sensors that detect the angle and velocity of the pendulum as well as the proximity of the carts to the closest object. Each cart is controlled by two action variables described in Table 7.3, which correspond with acceleration in each direction. The observation variables are mapped from measurements taken from the virtual sensors, and the action variables are mapped to values for the cart actuators to apply lateral force on the carts. The observation variables are each mapped to an input gene of the AGRN, and the action variables to the output genes. The AGRNs used to solve this task each contain 20 genes.

Table 7.2: Observations of the coupled inverted pendulum task

Name	Type	Range
Pendulum angle 1	Real	[0, 127]
Pendulum angle 2	Real	[0, 127]
Pendulum angle 3	Real	[0, 127]
Pendulum angle 4	Real	[0, 127]
Proximity 1	Real	[0, 127]
Proximity 2	Real	[0, 127]
Cart Velocity 1	Real	[0, 127]
Cart Velocity 2	Real	[0, 127]
Angular Velocity 1	Real	[0, 127]
Angular Velocity 2	Real	[0, 127]

The starting conditions of the task will be randomly initialised for every evaluation in both sets of experiments to ensure that the AGRNs are evolving to solve them generally, and not relying on specific starting conditions. In order for the task to operate for a sufficient duration, once the task is considered solved (i.e. the pendulum is stabilised in an upright position) the gravity of the environment will be changed. This will be achieved by randomly changing the gravitational constant of the environment to a value in the range 7–12, with an initial value of 9.81. This alters the dynamics of

Table 7.3: Actions of the coupled inverted pendulum task

Name	Type	Range
Cart 1 Left	Real	[0, 127]
Cart 1 Right	Real	[0, 127]
Cart 2 Left	Real	[0, 127]
Cart 2 Right	Real	[0, 127]

the task in a way that requires the AGRN to continuously adapt, but is unlikely to cause vast changes to the parameters of it. A successful AGRN will evolve with the robustness capable of solving the task in a changing environment. The AGRN will operate for a million time steps in total.

7.2.4 Pattern Generation Task

The pattern generation task requires that an AGRN generates a sequence of ten boolean values in a specific order. A target sequence of boolean values is randomly generated, then the AGRN is executed over ten time steps outputting a value every time. If the boolean values generated by the AGRN match the target sequence, the task is considered solved. The AGRN is supplied with no input whatsoever as it is not required. A single gene within the AGRN is designated for the boolean value output to be generated from. At each time step, if the designated gene's expression level is above 0.5 this is considered as true, otherwise it is considered as false.

In order for the task to operate for the required duration, a new target sequence will be randomly generated when the AGRN successfully generates the current target sequence. With each target sequence an AGRN must be developed that is capable of generating a sequence of boolean values. The sequence generated by the AGRN is a result of a set of gene parameters and node connections that cause the dynamic of the network to produce the correct behaviour from the initial state of the network. The correct behaviour will result in the output gene expressing values that map to the target sequence over ten time steps. The AGRN will operate for 10 million time steps overall.

7.2.5 Experimentation

Two sets of experiments were conducted in order to determine the degree of evolutionary constraint the AGRNs were subject to. During the first set of experiments, a set of AGRNs for each task were evolved using the iterative hill climbing algorithm. The evolutionary pathways of the AGRNs were then analysed to determine whether the number of connections a gene had correlated with the probability of it improving the performance of the network when mutated, forming the basis of the second experiment, where the rules were used to constrain the evolution of the AGRNs. For each set of experiments, two sets of AGRNs were evolved to solve each of the tasks. Fifty AGRNs were evolved to solve the coupled inverted pendulum task, and one hundred were evolved to solve the pattern generation task as it requires fewer computational resources to run.

Random mutation

The AGRNs were evolved using the iterative mutation algorithm (Algorithm 5). At each iteration, three random genes were randomly selected. For each of the three genes, one of the parameters was randomly chosen to be mutated. The parameters were mutated by changing them to a random value in their respective ranges (see Table 7.1). Three genes were mutated at once to allow for enough change to occur to the network to make meaningful difference without changing the operation of the network entirely. During the first set of experiments, the frequency at which each gene was mutated and subsequently resulted in an improvement to the network was recorded. This was then compared to the number of connections to other genes, the results of which are displayed by Figure 7.2 for the coupled inverted pendulum task, and Figure 7.3 for the pattern task. For both tasks the means of the most successful gene mutations are highest for genes connected to two others, genes with an increasing number of connections have a decreasing mean number of successful mutations. The median number of successful mutations in genes with a single connection is higher than with two for the pendulum task.

It is possible that a high number of successful mutations with a particular number of connections could be caused by the fact that there are more of these genes in the network, the previous results do not take this into account. The results were recalculated by dividing the successful number of mutations by the number of nodes

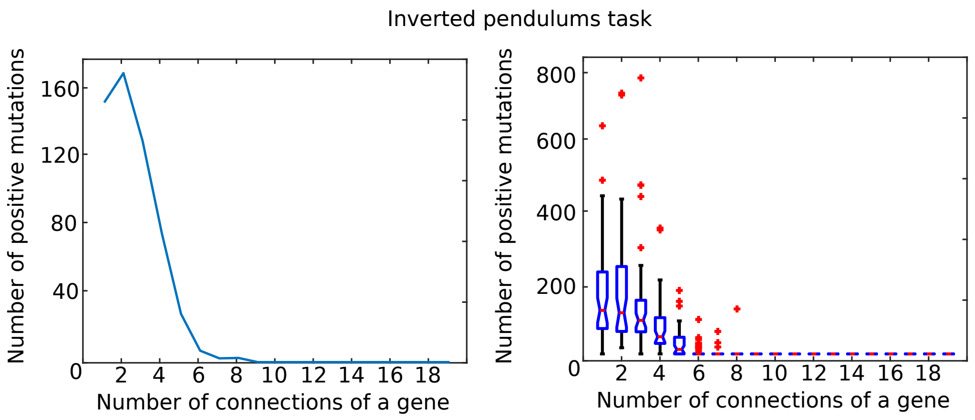


Figure 7.2: Raw data for the AGRN solving the inverted pendulum task. (Left: Mean, Right: Distributions)

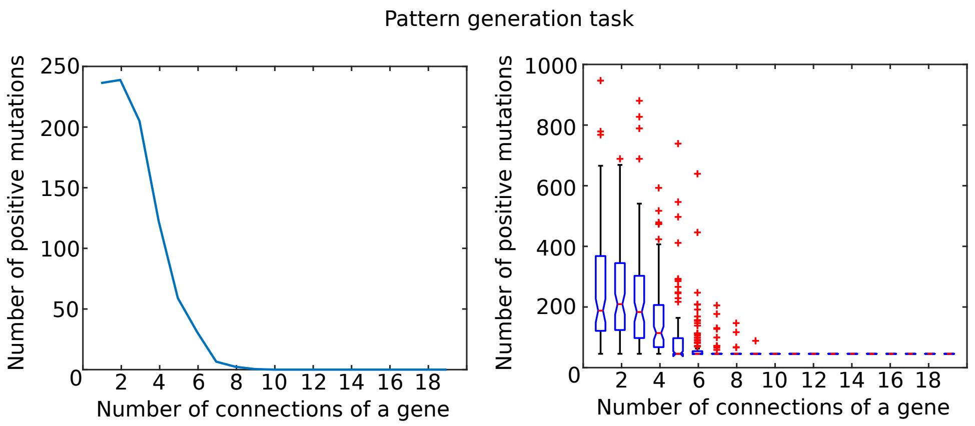


Figure 7.3: Raw data for the AGRN solving the pattern task. (Left: Mean, Right: Distributions)

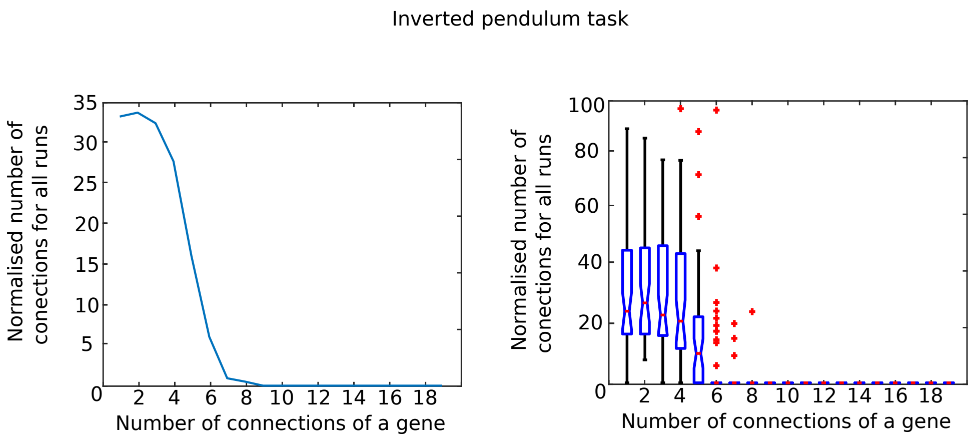


Figure 7.4: Data from the pendulum task transformed by dividing the number of successful mutations by the number of nodes in the network that have that many number of connections. (Left: Mean, Right: Distributions)

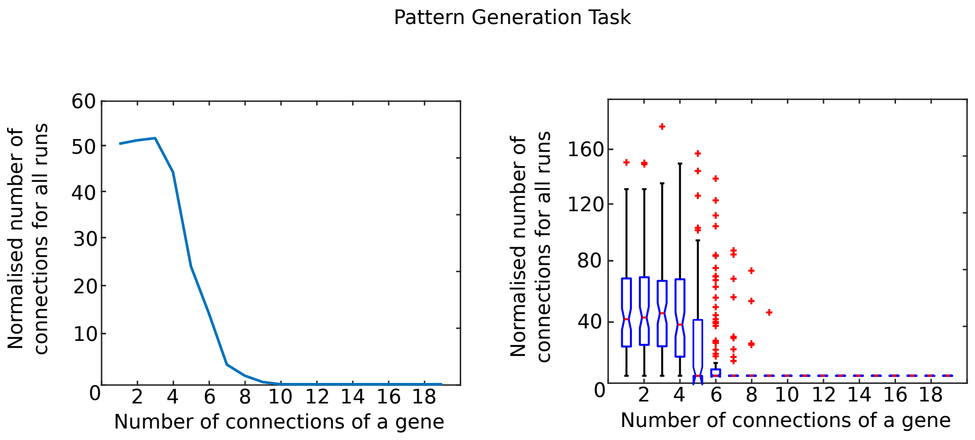


Figure 7.5: Data from the pattern task transformed by dividing the number of successful mutations by the number of nodes in the network that have that many number of connections. (Left: Mean, Right: Distributions)

with n connections in the network, in order to normalise them. The results are shown in Figure 7.4 for the pendulum task, and Figure 7.5 for the pattern task.

The results obtained from the first set of experiments show that genes with certain numbers of connections are more likely to result in mutations that increase the performance of the AGRN, which indicates that the evolution of AGRNs is constrained to some degree. Using this knowledge, it is possible to predict how an AGRN will evolve with respect to the mutations of genes based on the amount of their connections.

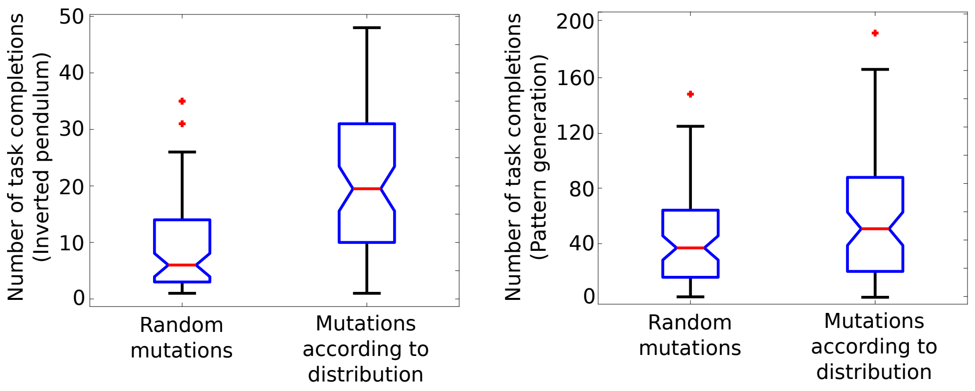


Figure 7.6: Comparison of AGRN performance when trained with random mutations and constrained mutations.

Constrained Mutation

The evolutionary process from the first experiment was analysed to determine the frequency at which genes were mutated, in order to generate rules that indicate the likelihood of a gene mutation resulting in an improvement to the performance of the AGRNs. The second set of experiments were conducted using the rules generated to constrain the mutation of the AGRNs, so that the AGRNs from both sets can be compared.

Figure 7.6 compares the performance of the AGRNs evolved in both sets of experiments. The performance of the AGRNs were measured based on how many times they were capable of solving their assigned task. The AGRNs were trained with random mutations in the first set of experiments, and a second set were trained with constraints based on the distributions derived from the first set. The performance of AGRNs solving both tasks generally increased when the mutations were constrained based on how many connections the genes have, indicating that it is a constructive strategy. The results for both tasks were shown to be statistically significant based on the Wilcoxon rank-sum test (the significance of the pendulum task was 2.037e$-$5, and the pattern generation task 0.0422). The increase in performance indicates that the proposed constraints produced more performant networks when applied, indicating the possibility of a predictable evolutionary pathway that when followed, results in more performant AGRNs.

7.2.6 Summary

Two sets of experiments were conducted, each involving the evolution of a set of AGRNs to solve each of the two tasks. Each time the tasks were considered solved, the attributes of them were changed so that the AGRNs would have to continuously evolve in order to solve them. The AGRNs were evolved using an iterative mutation algorithm that focussed on a single network at a time so that the evolutionary process could be analysed. The mutations to genes that resulted in positive network changes were recorded based on the number of connections of the respective genes. It was found that beneficial mutations occurred in genes with a certain number of connections more than others, which formed the basis of the constraints applied to the second set of experiments. When the constraints were applied, it was found that the AGRNs generally evolved to be more performant than the initial set. This indicated that there are specific evolutionary pathways that result in better AGRNs.

Chapter 8

Modularity Within Spatial Artificial Gene Regulatory Networks

Genes related in function are often situated closely to each other in an organism's genome, this is evident in operons, where multiple, often related, structural genes are situated adjacent to each other in the genome [Osbourn and Field, 2009]. Gene clustering occurs when multiple genes coding for similar gene products are situated within a few thousand base pairs of each other. In this chapter, experimentation will be conducted in order to ascertain the degree to which modularity occurs within AGRNs with regard to their internal structure, similar to modularity that has been observed in biology. A series of AGRNs will be evolved to solve three independent tasks, so that the resulting organisation of nodes can be compared based on their interactions with each of the tasks, in order to improve the understanding of AGRN behaviour.

8.1 Control tasks

AGRNs are capable of solving tasks in dynamical systems with chaotic elements [Lones et al., 2010, Turner et al., 2013b]. Such tasks have been used to show the capability of AGRNs, as it is infeasible to adequately control such systems using conventional approaches, such as procedural programming. The experiment described within this chapter involves an AGRN that has been evolved to solve three independent control tasks simultaneously, which will be described within this section.

An environment exists for each task consisting of two sets of variables. The observation variables describe the state of the environment; the action variables are used by an agent to control certain aspects of the environment. A reward is returned by each of the environments that determines how successfully the task is being solved, the aim being to maximise the reward. The environment operates over a series of time steps; the actions of the environment may be changed at each time step, and the observations are updated.

8.1.1 Inverted Pendulum Cart Task

The pendulum cart task consists of a single inverted pendulum attached to a cart that may be moved left or right (Figure 8.1). The inverted pendulum is attached to the cart, and may rotate around this point. The aim of the task is to balance the pendulum, so that it does not rotate too far away from an upright position. A failure condition is met if the cart moves too far to the left or right, or the pendulum swings too far in either direction, and the environment is terminated. The environment is also terminated if the task executes for 500 time steps.

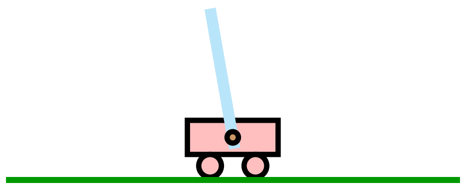


Figure 8.1: Illustration of the pendulum cart task.

The task is controlled using a single action variable shown in Table 8.1 that causes the cart to move left or right. Four observation variables shown in Table 8.2 describe the state of the environment with regard to the cart position and velocity, and the angle of the pole and velocity of the outward end. Reward is provided by the environment at each time step, until the environment is terminated.

Table 8.1: Actions of the pendulum cart task

Name	Type	Values
Cart control	Discrete	0 (Left), 1 (Right)

Table 8.2: Observations of the pendulum cart task

Name	Type	Min	Max
Cart position	Real	0	1
Cart velocity	Real	-10	10
Pole angle	Real	0	360
Pole velocity	Real	-20	-20

8.1.2 Double Pendulum Task

The double pendulum task consists of a single double pendulum that pivots around a fixed point (Figure 8.2). Torque may be applied to the second joint of the pendulum in order to enact rotation. The aim of the task is to maneuver the double pendulum so that it swings up and reaches a pre-defined height. The environment is terminated if the task is not completed in a fixed number of time steps.

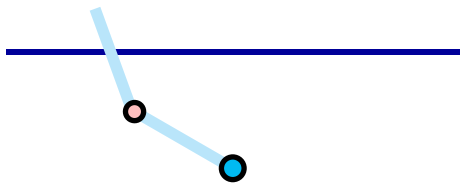


Figure 8.2: Illustration of the double pendulum task.

A single action variable controls the torque applied to the second joint of the double pendulum, shown by Table 8.3. Six observation variables describe the velocity and positions of each joint, shown by Table 8.4. The reward provided by the environment decreases over time, to encourage the task to be solved as quickly as possible.

Table 8.3: Actions of the double pendulum task

Name	Type	Values
Torque	Discrete	-1 (Negative torque), 0 (No torque), 1 (Positive torque)

Table 8.4: Observations of double pendulum task

Name	Type	Min	Max
Cosine of first joint angle	Real	-1	1
Sine of first joint angle	Real	-1	1
Cosine of second joint angle	Real	-1	1
Sine of second joint angle	Real	-1	1
First joint velocity	Real	-4π	4π
Second joint velocity	Real	-9π	9π

8.1.3 Mountain Car Task

The mountain car task consists of a car positioned in a valley between two mountains (Figure 8.3). The aim of the task is to reach the goal post on the right mountain; however, the car is not powerful enough to reach the goal post by driving directly up the mountain. The car must be controlled in such a way that it gains momentum by driving partway up each mountain in turn.

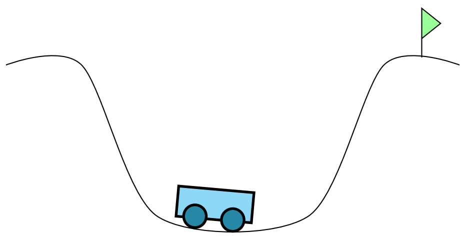


Figure 8.3: Illustration of the mountain car task.

The car is controlled by a single action variable (Table 8.5) that causes the car to accelerate to the left, right, or not at all. Two observation variables (Table 8.6) describe the position and velocity of the car. More reward is provided when the task is completed in fewer time steps.

Table 8.5: Actions of mountain car task

Name	Type	Values
Torque	Discrete	0 (Move left), 1 (Don't move), 2 (Move right)

Table 8.6: Observations of mountain car task

Name	Type	Min	Max
Position of car	Real	-1.2	0.6
Velocity of car	Real	-0.07	0.07

8.2 Representing Node Connections

In addition to the parameters of each gene's regulatory function, the behaviour of an AGRN results from the configuration of its genes, which interconnect to produce emergent behaviour. Node connections have been represented using different methods. Within the experiments discussed in this chapter, nodes have been represented in a 2-dimensional reference space, based on the indirect reference space described in [Turner et al., 2013b]. The methods of representing node connections will be discussed briefly, following a justification for the method that was chosen for the experiment in this chapter.

8.2.1 Discrete Connections

Connections may be represented using discrete boolean values, as they are in the random boolean network [Kauffman, 1969]. In a network where it is possible for each node to be connected to any number of other nodes; node connections could be represented in a 2-dimensional array of size $n \times n$, where n is the number of nodes in the network. Figure 8.4 shows an example of discrete node connections in a 5 node network, where nodes may not act as inputs to themselves.

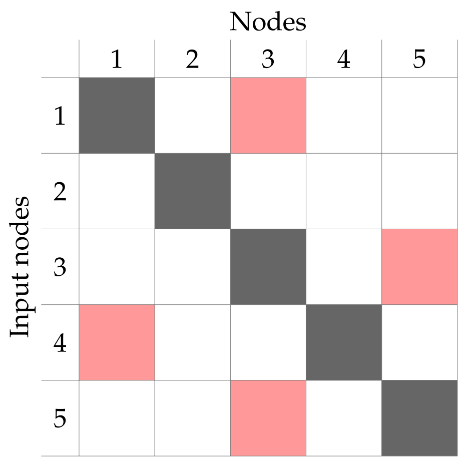


Figure 8.4: Illustration of an AGRN with discrete connections. Red squares represent connections, and in this network, nodes cannot be inputs to themselves.

8.2.2 Indirect Reference Space

AGRNs may be represented using an 'indirect reference space' [Turner et al., 2013b], where each node exists as a line segment. Prior to this, representing gene connections indirectly has been shown to be successful [Reil, 1999, Lones et al., 2010]. Connections between genes are formed based on their proximity to each other, and if a gene's centre point is situated within the space of another gene, it acts as an input to the latter gene and influences its expression level. Using this method provides space for genes to slowly move around and interact, introducing constraints that prevent connections from changing sporadically, causing significant changes to the dynamics of the AGRN as with discrete representation. Figure 8.5 shows an example of an indirect space AGRN consisting of 3 nodes.

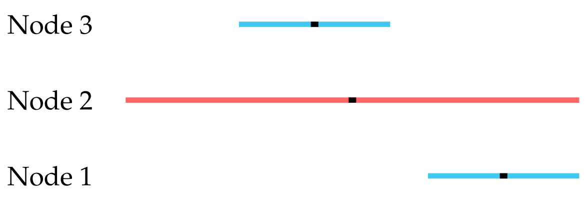


Figure 8.5: Illustration of small indirect reference space AGRN. Nodes 1 and 3 are inputs to node 2 Node 2 is an input to node 3.

8.2.3 Two-dimensional Reference Space

The AGRNs used within the experiment described within this chapter use a technique based on the indirect reference space. For this experiment, a novel technique for representing AGRNs in two-dimensional reference space has been developed where each node is a rectangle with a centre point. Similarly to the one-dimensional reference space, node connections are formed when a node's centre point is situated within the space of another node. The difference being that there is an additional dimension, which allows for more complex connection configurations. This method was used in order to encourage nodes to arrange themselves into groups that are easy to identify via observation. Figure 8.6 illustrated a connection formed between two nodes in a network represented in two-dimensional space.

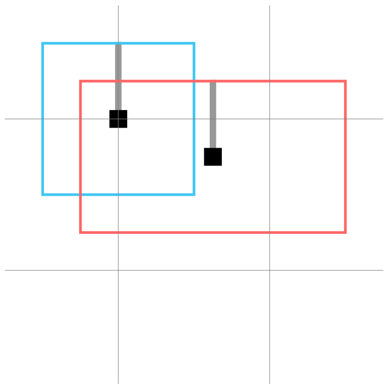


Figure 8.6: Illustration of two nodes that have formed a connection. The blue node acts as an input to the red node, due to the position of its centre point.

8.3 Evolution Using a Genetic Algorithm

AGRNs consist of many nodes, each with a parameterised activation function. The connections between nodes often change as the AGRN is trained. Gradient descent and backpropagation are techniques used to train artificial neural networks; however, they are not applicable to AGRNs as they rely on a well structured, static graph. Genetic algorithms are applicable to complex combinatorial problems consisting of variables of different types, for this reason a genetic algorithm will be used to train the AGRNs in this experiment.

8.3.1 Mutation Only Genetic Algorithm

The behaviour from an AGRN results partially from the configuration of the connections between nodes, which are determined during evolution. The recombination operator causes attributes from two members of the population to be combined. The AGRNs used within this experiment define their connectivity based on the proximity of nodes, which is determined by each node's position and size attributes. This means that during recombination, it is possible for the proximity of nodes to be arbitrarily swapped between networks without any regard for context. This is problematic as the networks may have formed useful connections that would be broken during recombination. In addition, the purpose of the experiment is not to produce the most performant AGRNs, but to analyse their dynamics. For these reasons, recombination will not be applied during the execution of the GA, and it will be referred to as a 'mutation only genetic algorithm'.

The AGRNs consist of 18 genes, each with a set of regulatory function parameters, as well as a position and size. The 18 genes consist of those mapped to task observations (12 in total), those mapped to task actions (3 in total), leaving 3 'processing' genes, which are not mapped at all, the genes mapped to each of the tasks' observations and actions are shown in Table 8.7. The parameters of a single gene and their valid ranges have been summarised in Table 8.8. The set of genes and their parameters constitute a member in the context of the genetic algorithm. The fitness function of a member is defined as the AGRN's success at solving all three of the tasks.

Table 8.7: Nodes mapped to each task, nodes 15, 16, and 17 do not belong to any task, and are referred to as processing nodes.

Task	Observation Genes	Action Gene
1	0, 1, 2, 3	12
2	4, 5, 6, 7, 8, 9	13
3	10, 11	14

During the execution of the GA, the random initialisation and mutation of parameters is subject to constraints ensuring that they are set to values within the range and correct type. The mutation of gene positions and sizes are additionally constrained to favour smaller changes, to encourage genes to slowly move around similarly to how evolution generally occurs as a series of gradual changes over time, so that connections are formed in a controlled, iterative process. This reduces the probability of genes sporadically changing position and size, which reduces the probability of clusters forming.

Table 8.8: The variables held within each gene of an AGRN.

Name	Type	Range
Expression	Real	[0,1]
Weight	Real	[0,1]
Slope	Real	[0,20]
Sigmoid offset	Real	[0,1]
Position X	Real	[0,1]
Position Y	Real	[0,1]
Size X	Real	[0,0.5]
Size Y	Real	[0,0.5]

A mutation only GA with a population size of 1000 is used to evolve the AGRNs. Evolution is terminated when the AGRN reaches an overall performance of 990, as at this point it is likely that all three tasks have been solved. Earlier experiments indicated that once the performance had reached 990, each of the three tasks were likely to be solved. The fitness of each member is calculated as the average of ten repeated runs, meaning that the AGRN will attempt to solve all three tasks 10 times in total; each task will be initialised with random starting conditions every time. Elitism is applied at each generation, where the 10 best performing members will be copied to the next generation without mutation, this ensures that the best members are not lost due to random chance. Tournament selection is used to choose the members that will be copied to the next generation. The execution of the GA is shown by Algorithm 6.

8.4 Gene Purging

When analysing the AGRNs, it is desirable for there to be as few genes as possible. The AGRNs consist of 18 genes; however, it may occur that not all of the genes are required for the AGRN to operate. For this reason the AGRNs will be purged before analysis, involving the removal of a gene at a time, and evaluating the performance

Algorithm 6: Execute mutation only genetic algorithm

```
 1: $P \leftarrow \{\}$ {Initialise empty initial population}

 2: for $x = 1 \rightarrow$ Population size do
 3:     $P \leftarrow P \cup$ Randomly initialised AGRN
 4: end for

 5: for $y = 1 \rightarrow$ Number of generations do
 6:     for all $p \in P$ do
 7:         EVALUATE(p)
 8:     end for

 9:     $Q \leftarrow \{\}$ {Initialise empty child population}
10:     $Q \leftarrow Q \cup$ 10 Elite members
11:     repeat
12:         $Q \leftarrow Q \cup$ TOURNAMENT_SELECT(P)
13:     until $|Q|$ is Population size
14:     $P \leftarrow Q$

15:     Mutate population
16: end for
```

of the network to observe the change in performance. If the performance does not decrease, it will be removed from the AGRN. Algorithm 7 describes this process.

Algorithm 7: Purge Genes from AGRN

```
 1: $t \leftarrow$ Threshold
 2: $b \leftarrow$ Simulate network {Base performance}
 3: $N \leftarrow$ Network genes

 4: for all $n \in N$ do
 5:     $N \leftarrow N \setminus \{n\}$ {Remove gene from network}
 6:     $p \leftarrow$ Simulate network {Performance without gene}
 7:     if $p < b - t$ then {Performance decreases}
 8:         $N \leftarrow N \cup \{n\}$ {Replace gene}
 9:     end if
10: end for
```

8.5 AGRN Execution

Each of the AGRNs will solve the three control tasks simultaneously, meaning that at each time step the observation variables are mapped from all three tasks, processed, and the action variables are mapped back to the corresponding tasks. The observation variables of the tasks will be mapped onto AGRN genes, acting as inputs to the AGRN. Each of the tasks has a single action parameter, which will be mapped from three AGRN genes designated as the output genes. The observation parameters consist of real numbers, and are transformed to the range 0–1 before they are mapped onto the input genes. The numbers are copied to their respective genes within the AGRN at the start of each time step, replacing the expression levels. The action variables consist of a single integer, which is taken from the respective genes in the AGRN by taking the expression level and converting it to an integer. This is achieved by splitting the expression level range 0–1 into n equal ranges, and assigning each to an integer value, where n is the quantity of valid integers accepted by the task in question. Algorithm 8 describes the execution of an AGRN across a number of time steps.

Algorithm 8: AGRN execution

```
1:  for $x = 1 \rightarrow$ Time steps do
2:      if $x$ is 1 then
3:          Initialise nodes randomly
4:      end if

5:      Scale task observations to 0,1
6:      Map task observations to input nodes

7:      for $x$ to $numberofnodes$ do
8:          Initialise nodes randomly
9:      end for

10:     Scale outputs to task range
11:     Map output nodes to task actions
12: end for
```

8.6 Analysis of the AGRNs

The genes of each AGRN will be analysed based on their relation to each task. A gene will be referred to as a 'task gene' if it is mapped from a task observation, or if it maps to a task action. Nodes that are not task genes will be referred to as 'processing genes'. In total, 50 AGRNs were trained, each acting as a repeat experiment. Task genes will be compared to the actual connection configuration resulting from the evolution of the AGRN, in order to determine how the genes are ordered. The output genes of the AGRN are mapped onto the tasks actions, which are used to control the environments of each task. Therefore, if a gene acts as an input to an output gene, it influences the completion of the task. Genes may also influence an output node indirectly, by influencing a node that influences an output gene. Connections in this way may form chains of more than two genes, where each gene eventually influences an output gene.

The connectivity of genes in each of the AGRNs has been analysed based on their association with each of the tasks. The genes were split into three sets, the sets containing genes that influence each of the output genes. The three sets were individually split into three more groups, based on their relationship to the corresponding output gene. The genes mapped from the observation variables of the same task, observation and action genes of other tasks, and processing genes. It is undesirable for an observation gene of one task to influence the action gene of another, as the state of another task's environment is irrelevant. Genes fitting this criteria will be referred to as 'interfering nodes'.

The percentage of non-interfering genes separated by task in each of the 50 networks has been shown in Figure 8.7, and the averages in Figure 8.8. Non-interfering genes are genes acting as inputs for a particular task that have been mapped from observations of the same task, or are processing genes. These results show that the genes have organised themselves based on the task that is being solved to some degree, particularly for task 1 and task 2. Task 3 appears to be less organised, as there are fewer non-interfering nodes. This is because, generally speaking, the nodes influencing the genes controlling the tasks mostly belong to the genes mapped from the observations of the same task, indicating that the AGRNs have evolved in such a way to minimise interference.

There were fewer non-interfering genes associated for the mountain car task (task 3), this could be due to the fact that task 3 is easier to solve due to its simpler dynamics,

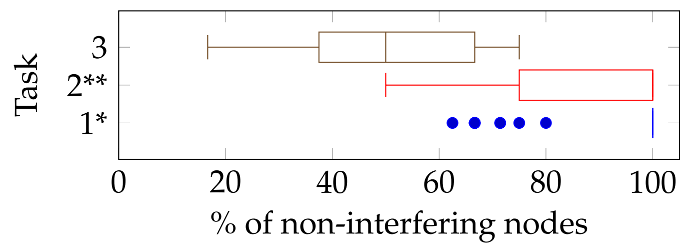


Figure 8.7: Box plot of the percentage of non-interfering genes separated by task for the AGRNs. *All values are equal to 1 except outliers. **Median and upper quartile are equal to 1.

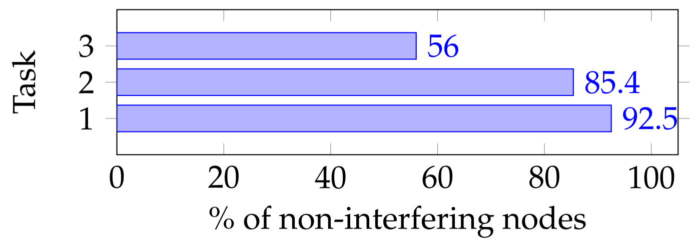


Figure 8.8: Bar plot of the average percentage of interfering genes separated by task.

requiring less processing to solve. The task is controlled using relatively simple mechanics as the car only has to be moved backwards and forwards based on it's position on the horizontal axis. The other tasks have more complex dynamics. Task 1, the inverted pendulum cart task had far less interference on average. This task requires that the angle and velocity of the pendulum are monitored so that the cart can be moved in either direction to first balance, and then stabilise it. The position of the cart must also be monitored so that it does not move too far in either direction. The double pendulum task (task 2) also has complex dynamics as it requires that the angles and velocities of both pendulums are monitored, so that the second pendulum can be moved in a way that causes it to swing upwards to reach the required height. Task 2 also has fewer interfering genes, which could suggest that interference is detrimental to tasks with more complicated dynamics.

The inverted pendulum cart task reward is calculated based on how long the pendulum remains balanced, meaning that it is possible for the maximum reward to be provided. The rewards of the mountain car and double pendulum tasks are calculated differently, and are provided based on how long it takes for the task to be completed, meaning that the task will often be solved without achieving the maximum reward. This could mean that evolution favours the inverted pendulum cart task, as it can provide more reward, which could be the cause of fewer interfering nodes.

The genes of an AGRN with no interference are shown in Figure 8.9, and the connections in Table 8.9. Each coloured rectangle represents a gene within the network, its centre point shown by the inner solid grey rectangle. The purple rectangles represent processing genes, the blue, red and green rectangles represent genes belonging to tasks one, two and three respectively. The AGRN is said to have no interference because the genes that act as input to each of the action genes are either observation genes belonging to the same task, or processing nodes that do not take input from any of the other tasks.

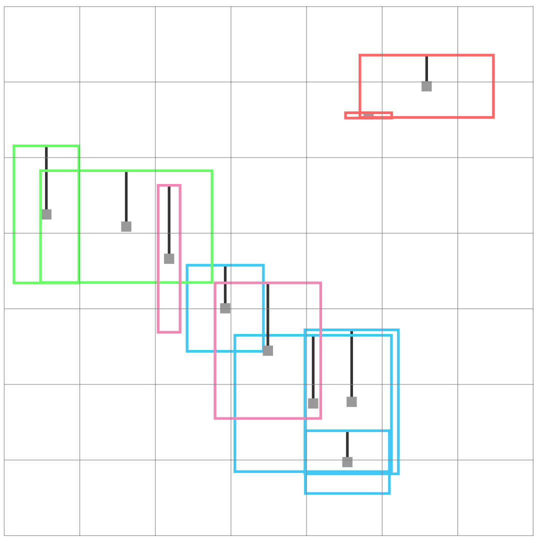


Figure 8.9: Layout of network with no interference. The purple rectangles represent processing genes; the blue, red and green rectangles represent genes belonging to tasks one, two and three respectively.

Table 8.9: Display of gene connections for network in Figure 8.9.

Task	Task genes	Processing genes	Interfering genes
1	0, 2, 3	16	none
2	8, 9	none	none
3	11	15, 17	none

Figure 8.10 shows the genes of an AGRN with a high amount of interference among tasks, Table 8.10 shows the connections to each of the action genes. All of

the actions nodes have at least one input gene that belongs to another task, and the genes belonging to each task do not arrange themselves into separate groups. This is opposed to the AGRN with no interference, where the genes do arrange themselves into completely separate groups.

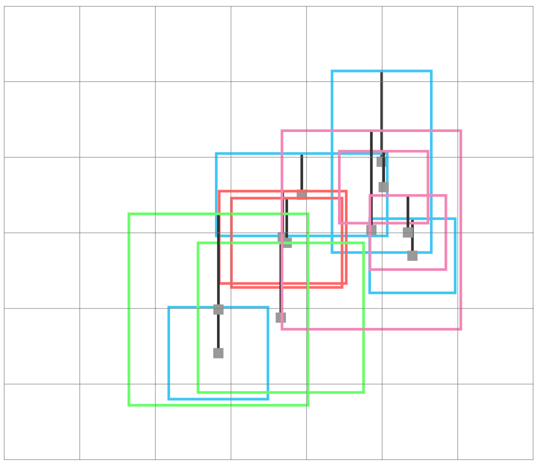


Figure 8.10: Layout of network with one of the highest rates of interference. The purple rectangles represent processing genes; the blue, red and green rectangles represent genes belonging to tasks one, two and three respectively.

Table 8.10: Display of gene connections for network in Figure 8.10.

Task	Task nodes	Processing nodes	Other task nodes
1	2, 3	15, 16, 17	9, 10, 13
2	9	none	10
3	10, 11	none	0, 9, 13

Despite the two networks (Figures 8.10 and 8.9) arranging themselves completely differently, they both perform very similarly. The first AGRN would be much easier to analyse with regard to the gene clusters that form, as they are easier to identify. The second network would be much more difficult to analyse with regard to gene clusters, but it is more likely that AGRNs solving more complex tasks that can be broken down into multiple related sub-tasks would evolve in this manner, as the sub-tasks are related to each other, unlike the tasks in this experiment.

The average percentage of non-interfering nodes has been plotted against the overall performance of the AGRNs tested in Figure 8.11. There exist performant networks with both a high and low number of non-interfering nodes, showing that interference is not a good indicator of network performance.

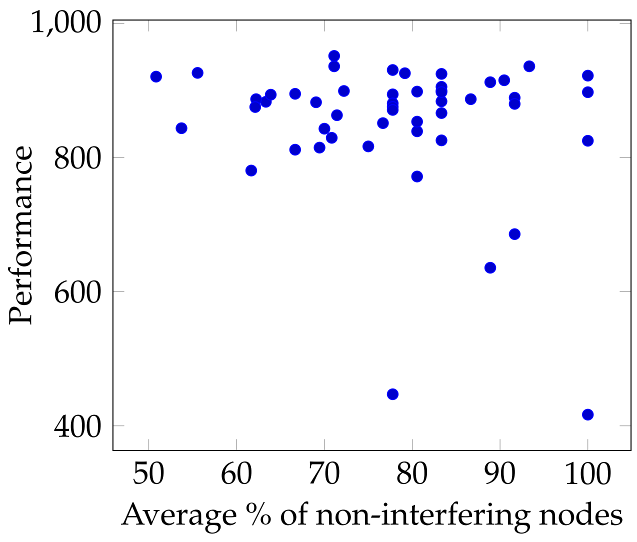


Figure 8.11: Scatter plot of performance compared with interference.

8.7 Summary

Multiple AGRNs were evolved to solve three independent tasks so that the arrangement of genes could be observed with regard to the task that each gene is associated with, to see whether the AGRN develops modularity. This was in an attempt to mimic an AGRN that was tasked to solve a complex task that could be broken down into three sub-tasks, that could in theory form three clusters of genes that solve each of the tasks.

It was shown that the AGRNs evolved with a range of modularity, and in many cases interference did occur, which generally differed according to the task that was being solved. The more complex tasks tended to show less interference, suggesting that the structure of the AGRNs depended on the complexity of the tasks in some way.

Some networks did evolve with clear separation between genes belonging to the different tasks, but interference did occur to some degree in most cases. It was also found that the performance of networks was not directly influenced by the amount of interference, probably due to the robustness of AGRNs. The main implication in this work being that networks have little incentive to evolve to be modular in structure, making analysis more difficult.

Chapter 9

Improving the Transparency of Deep Neural Networks Using an Epigenetic Control Layer

Artificial neural networks (ANNs) are known to operate as 'black-boxes' [Adadi and Berrada, 2018], meaning that the internal structure does not provide an explanation for the decisions being made. A prominent example of this is in healthcare, where it is important to be able to discern whether a prediction is made due to causation, rather than an invalid correlation. When ANNs are deployed to solve real-world problems, it is important to be able to obtain reasoning behind the decisions being made. From a maintenance perspective, it is likely that the ANN will need to be improved over time as weaknesses in its training are discovered. If a system involving an ANN causes a problem resulting in harm to a person or an organisation, it must be possible to determine whether the ANN was the cause so that the right people can be held accountable. Biases must also be detectable in order to prevent discrimination caused by ANNs.

A control layer inspired by epigenetics has been developed in an attempt to improve the transparency of a trained ANN, which dynamically disables nodes from the second hidden layer of the network based on the output from the previous layer. This method shows which nodes are not essential for the operation of the ANN, as well as allowing for the correlation of node switching with aspects of the ANN execution, to show which nodes are used at a particular point in time. Existing methods of improving the transparency of ANNs will be described in this chapter, followed by a description of the implementation of the control layer and a demonstration of its practical uses.

Mechanisms inspired by epigenetics have been used to improve the transparency of artificial gene regulatory networks by observing the switching of the nodes within the network [Turner and Dethlefs, 2017]. In this experiment, an artificial epigenetic layer (AEL) is applied to a trained artificial neural network to determine whether it is possible to improve the transparency of an external connectionist architecture. The artificial epigenetic layer, consisting of artificial epigenetic molecules has been developed to act between two layers of an ANN and disable nodes in the second layer based on the output of the first.

The AEL that has been developed is capable of performing two functions. Pruning is a method of reducing the number of nodes or parameters in an artificial neural network with the intention of reducing the number of computational resources required to execute it. The AEL is capable of identifying nodes that may be pruned by switching them off 100% of the time, this information can then be used to prune the ANN. In addition to pruning, the AEL may dynamically switch nodes off based on the state of the ANN with respect to a particular input vector. This has the potential to improve the transparency of the ANN as it is possible to observe the switching behaviour of the AEL with respect to the certain aspects of ANN execution, such as the predicted class or the input itself. This allows one to observe which nodes within the ANN are active at a particular point in time, to identify which nodes are useful in certain circumstances.

9.1 Datasets

Two datasets have been used during the experiments, they will be described briefly followed by an explanation of how they have been split.

9.2 Wall Following Robot

The wall following robot dataset consists of 5456 sensor readings collected from a robot traversing a room in an anti-clockwise direction. Readings were collected at a rate of 9 per second by 24 ultrasound sensors positioned along the circumference of the robot, each attribute corresponds with a particular sensor. The direction of the robot at each time step was also collected, constituting the label of the dataset. The robot was recorded as moving forward, turning slightly left, slightly right, and sharply right. A simplified illustration is shown by Figure 9.1.

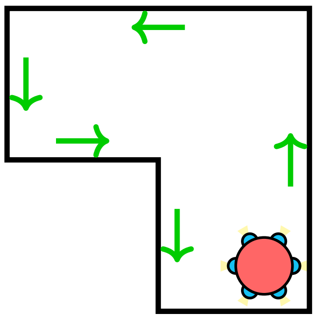


Figure 9.1: Simplified illustration of the wall following robot that collected the ultrasound data. Only six sensors are shown.

9.3 Cardiotocography

Cardiotocography (CTG) is a method used during pregnancy to monitor the well-being of the foetus by recording the fetal heartbeat and uterine contractions. The CTG dataset used consists of features generated from 2126 readings. With each reading, the foetus is classed as being normal (77.8%), suspect (13.9%), or pathologic (8.3%). The classification indicates how healthy the foetus is, and what actions need to be taken (summarised in Table 9.1).

Category	Definition
Normal	No problematic readings
Suspect	Readings may indicate a problem
Pathologic	Readings are likely to indicate a problem

Table 9.1: Classification criteria based on CTG readings

9.4 Dataset Splitting

Conventional artificial neural network training requires that the a dataset be split into subsets for training and evaluating the network, and often a third set used for validation during training. Within this experiment, the dataset will be split into three subsets A, B and C for the training, validation and testing of the neural network respectively. Sets B and C will be used for the evolution and evaluation of the artificial epigenetic layers (AELs) respectively. Set C will be unseen by both the ANN and the AEL for fair evaluation. This is shown in Figure 9.2.

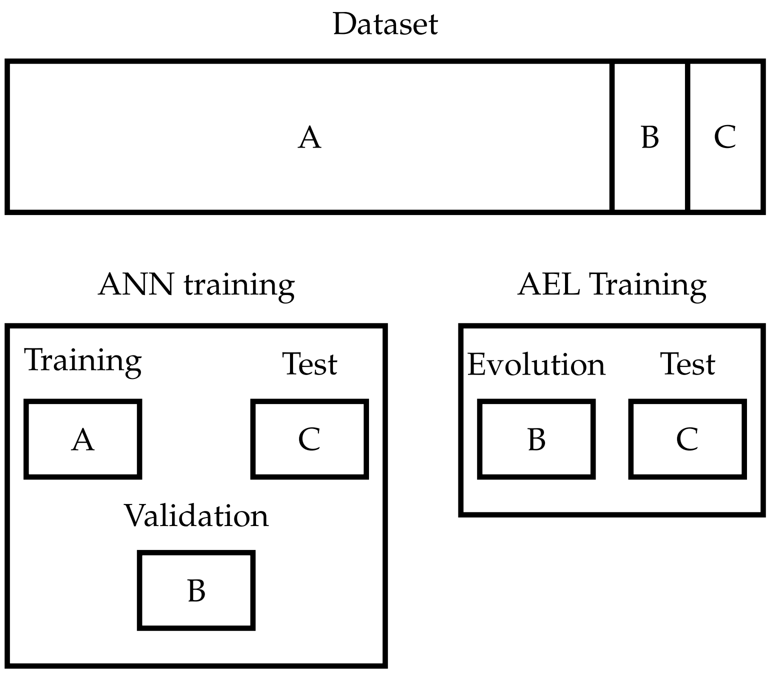


Figure 9.2: Splitting of the datasets

9.5 The Artificial Epigenetic Molecule

The artificial epigenetic molecule (AEM) is a processing unit that processes a number of inputs to produce a single value, which is used to determine whether an AEM is active or inactive. The inputs consist of continuous values, which are processed by a regulatory function to produce a single continuous value. Originally, AEMs were used within artificial gene regulatory networks and were found to improve the performance [Turner et al., 2013b] and improved the transparency of the networks [Turner and Dethlefs, 2017]. They function similarly to genes within the network, but selectively activate/deactivate other genes based on their relative positions.

The AEM operates by first reading its inputs, in this case from the nodes in the first hidden layer of the ANN, each represented by a single real number. The inputs are summed (Equation 9.2) and then processed by the AEM's regulatory function, which in this case is a parameterised sigmoid function (Equation 9.1). The parameterisation of the sigmoid function allows for flexibility in how it acts; the slope parameter s alters the gradient of the function, allowing it to act more like a step function; the b parameter allows the function to be offset along the x axis. The sigmoid function maps the sum of the AEMs inputs to a single value between 0 and 1. This value is used to determine whether the AEM is active or not, which will bring about a change in the system that it is acting upon.

$$f(x) = (1 + e^{-sx-b})^{-1} \tag{9.1}$$

$$x = \sum_{j=1}^{n} i_j \tag{9.2}$$

9.6 The Artificial Epigenetic Layer

The artificial epigenetic layer (AEL) is a dynamic control layer that acts between two layers of a connectionist architecture, consisting of one or more artificial epigenetic molecules (AEMs). The AEL receives input from the former layer, and selectively switches part of the latter layer based on the state of the former layer. This allows for changes to be brought about dynamically, based on the input of the network.

The AEL has been designed to perform two tasks, purging and switching. Purging is the process of removing redundant nodes from the network, that are not essential for the network to perform adequately. Nodes that are always disabled when the network is evaluated on a dataset can be considered purged as they are never active. This provides a basic level of transparency, and the information could be used to construct a more efficient ANN with the nodes completely removed, which has the potential to reduce the computational resources required for execution. The AEL is capable of dynamically switching nodes on and off based on its inputs, resulting in behaviour that is dependant on the state of the ANN. The switching behaviour can be analysed over time and compared to other aspects of the network, such as the input data and predicted class in order to determine which nodes within the network are useful with regard to these aspects.

The work within this section utilises an AEL that act externally on a connectionist architecture that has already been trained. The AEL will operate between the first and second hidden layer of an ANN when used to make predictions on two datasets. The behaviour of the AEL will then be analysed to determine which nodes are purged, and how nodes are switched based on the class prediction.

9.7 Artificial Neural Network Architecture

Two neural networks have been trained on the wall following robot and CTG datasets. The neural networks contain two hidden layers, the first contains 24 nodes, and the second 16. Both hidden layers use the ReLU activation function, the output layer uses softmax. The first hidden layer has a dropout rate of 20%, and the second 10% in order to reduce overfitting.

9.8 Evolution of the Artificial Epigenetic Layer

Genetic algorithms have been used to evolve AGRNs due to their dynamic topology during training, as well as the different types of parameters that must be adjusted. The AEMs that constitute the AEL are similar to the nodes within an AGRN, as they form connections and have a parameterised activation function. Therefore, a genetic algorithm will be used to evolve the AEL.

The AELs sit between the two hidden layers of the artificial neural network; nodes from the first layer constitute the inputs of the AEMs, which are processed using a regulatory function to determine whether the AEMs are active. If an AEM is active, it will deactivate its output nodes, taken from the second layer of the ANN. Each AEL contains 16 AEMs, as the neural networks both have 16 neurones in their second hidden layer. This allows the AEL to individually control each of the nodes, but does not guarantee that it will as AEMs may connect to the same nodes. Each AEM within the AEL is represented as a set of integers for the inputs and outputs, as well as a set of continuous numbers for the regulatory function parameters. The quantity of inputs and outputs have been constrained. The number of inputs has been fixed to three to simplify analysis, as a single AEM will only be controlled by a small number of nodes. A single AEM will not connect to the same input node more than once. The number of outputs has been limited to one, so that only one node is affected by a single AEM. The parameters of an AEM have been summarised in Table 9.2, a set of these parameters constitutes the parameters of an AEL.

Table 9.2: Parameters of a single AEM

Parameter	Type	Description
Sigmoid Offset	Real	[-1, 1]
Sigmoid Slope	Real	[0, 20]
Input 1	Int	[0, 23]
Input 2	Int	[0, 23]
Input 3	Int	[0, 23]
Output	Int	[0, 15]

A genetic algorithm with a population size of 512 has been used to evolve the AELs during the experiments, the process is shown by Algorithm 9. The selection of the population size was chosen as the result of a trade off of being high to allow for diversity within the population, but limited to reduce the time taken to execute the GA. The mutation rate was set to 20%, meaning that approximately 20% of the parameters of the AELs will be changed at each generation; from a technical standpoint, each parameter has a 20% chance of being mutated. The mutation rate was set at this percentage to encourage new traits into the population, it was not set higher as this could cause the GA to act more as a random search. The crossover rate was set to 50% to encourage traits to be exchanged between members. The mutation operator was implemented by simply iterating through the parameters of every molecule, and each parameter had a 20% chance of being mutated. The crossover operator was implemented similarly to this; when two of the population members had been selected, the parameters of all the molecules were iterated and there was a 50% chance of a parameter swap occurring.

The tournament selection algorithm with an arena size of 16 was used to choose the parents for crossover. Eight 'elite' members were copied directly to the next generation to ensure that the best members were not lost due to random chance. The GAs execute for 300 generations, but terminate if the best member has not improved for 10 generations. Due to the large population size, if an improvement is not made for 10 generations it is unlikely that any more significant improvements will be made.

Algorithm 9: Evolving an AEL with a GA

```
1:  $P \leftarrow \{\}$ {Initialise empty initial population}

2:  for $x = 1 \rightarrow$ Population size do
3:      $P \leftarrow P \cup$ Randomly initialised AEL
4:  end for

5:  for $y = 1 \rightarrow$ Number of generations do
6:      for all $p \in P$ do
7:          EVALUATE(p)
8:      end for

9:      $Q \leftarrow \{\}$ {Initialise empty child population}
10:     $Q \leftarrow Q \cup$ Elite members
11:     repeat
12:         if recombine then
13:             $Q \leftarrow Q \cup$ Recombine(P)
14:         else
15:             $Q \leftarrow Q \cup$ TOURNAMENT_SELECT(P)
16:         end if
17:     until $|Q|$ is Population size
18:     $P \leftarrow Q$

19:     Mutate population
20: end for
```

The fitness function of a GA determines how well each member performs the task. In this case the intention is to purge as many nodes as possible, so the fitness function calculates the average number of weights zeroed over time as a percentage. It is undesirable to reduce the performance of the network at the expense of improving transparency; therefore, if the application of the AEL causes the performance of the neural network to decrease, the fitness function returns zero.

9.9 Results

Five individual AELs were evolved for the ANN tasked with classifying the wall following robot dataset, then five more individual AELs were trained on the CTG

ANN. The effect of the AELs on the performance of the network will be discussed, followed by an analysis of the AEL behaviour with regard to purged nodes, and those that are selectively switched.

9.9.1 Decrease in ANN Performance with Test Dataset

The fitness function of each member was set to return zero during training if the performance of the neural network dropped with the application of the AELs. Despite this, the performance still decreases when the data from the test set is used. The drop in performance when the networks were evaluated with the test set on the wall following robot ANN are shown in Figure 9.3, and the CTG dataset in Figure 9.4. The 'None' column represents the ANN with no AEL applied. The greatest performance drop was 7%, with an approximate range of 0.5%-7%. Any performance drop is undesirable, but the effects of applying the AELs was not significant enough to cause a problem.

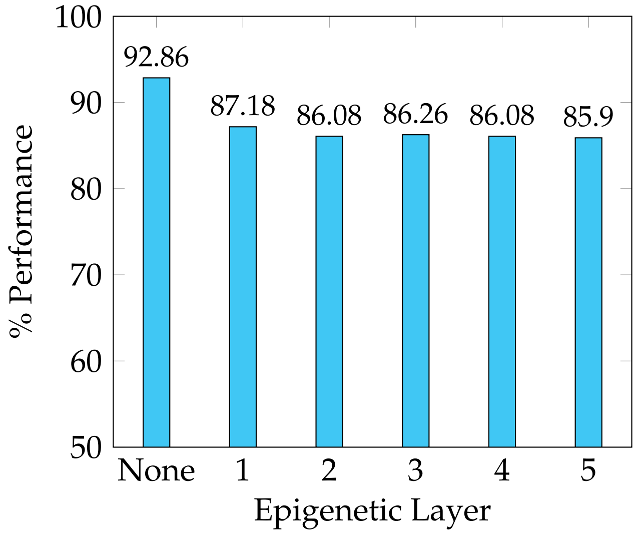


Figure 9.3: Performance of the wall following robot neural network with each of the 5 repeat AELs applied compared to the network with none of the AELs applied.

The decrease in performance with the test set when the AELs are applied is likely due to the limited size of the subset of data used to evolve the AELs. The largest subset was used to train the ANNs, the performance of the ANNs was a priority as the purpose of the experiment was to determine whether the AELs are capable of acting on trained connectionist architectures. If the ANNs were insufficiently trained, the experiments would not be representative of applying the AELs in a useful scenario.

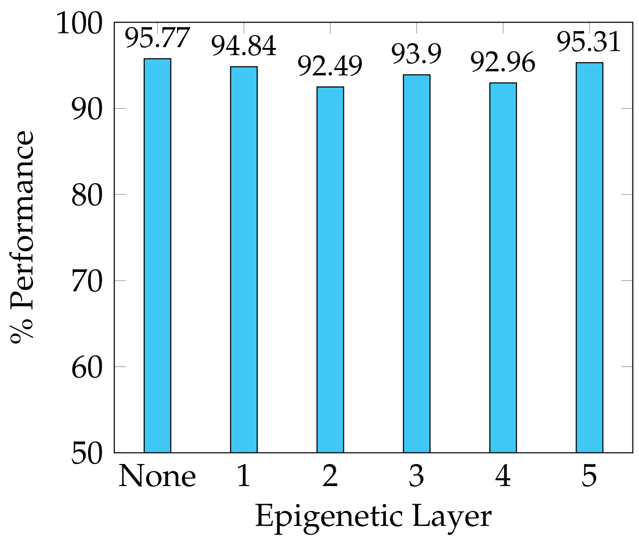


Figure 9.4: Performance of the CTG neural network with each of the 5 repeat AELs applied compared to the network with none of the AELs applied.

For this reason, and due to time constraints with regard to GA execution, a smaller subset of data was used to evolve the AELs.

9.9.2 Purged Nodes

If an ANN node is deactivated by an AEM 100% of the time, it has effectively been removed from the network and can be considered as being 'purged'. The nodes in the second hidden layer of the ANNs purged by each of the five repeat AELs are shown in Table 9.3 and Table 9.4. The number of nodes purged indicates the potential for reducing the size of each neural network.

Table 9.3: Nodes in the wall following robot ANN purged by each of the AELs.

AEL	Purged nodes
1	4, 5, 6, 9, 11, 13, 14
2	4, 5, 8, 9, 11, 14
3	7, 8, 11, 12, 13, 14, 15
4	5, 6, 10, 14, 15
5	3, 5, 8, 10, 11, 14, 15

Table 9.4: Nodes in the CTG ANN purged by each of the AELs.

AEL	Purged nodes
1	1, 10, 13, 14, 15
2	2, 6
3	1, 8, 9, 10, 11, 13
4	0, 5, 10, 13, 14
5	0, 2, 5, 6, 10, 13

The frequency of each node purged by an AEL in the wall following robot ANN is shown in Figure 9.5. If many AELs purge the same node, it suggests that the node does not perform a function that is crucial for the neural network to perform. Node 14 is purged by all five AELs, suggesting that it does not provide functionality that is useful to the ANN. Nodes 5 and 11 are purged by four of the AELs, again suggesting that they do not provide crucial functionality. The frequency of each node purged by an AEL in the CTG ANN is shown in Figure 9.6. Nodes 10 and 13 are purged by 4 of the AELs, again suggesting that they do not perform useful functionality in the network.

Many nodes in both ANNs are purged by only a few of the AELs. This could suggest that there was potential for the AELs to improve if they were evolved for more generations. It could also suggest that there are multiple groups of nodes that act together to perform a particular function within the ANN, and that the different AELs were disabling different redundant groups. The experiments were terminated if there was no improvement after 10 generations, this would need to be extended to investigate whether further improvement would be possible. In the case that the AELs are disabling different redundant groups, more repeat experiments could help to make these more obvious. Using a lower arena size for the tournament selection algorithm may also be beneficial, as it would encourage the population to converge more slowly.

9.9.3 Nodes Selectively Switched

AELs are capable of dynamically switching nodes off based on the output of the first hidden layer. This behaviour has been compared to the class predicted by each network, by calculating the percentage time that particular nodes were switched off

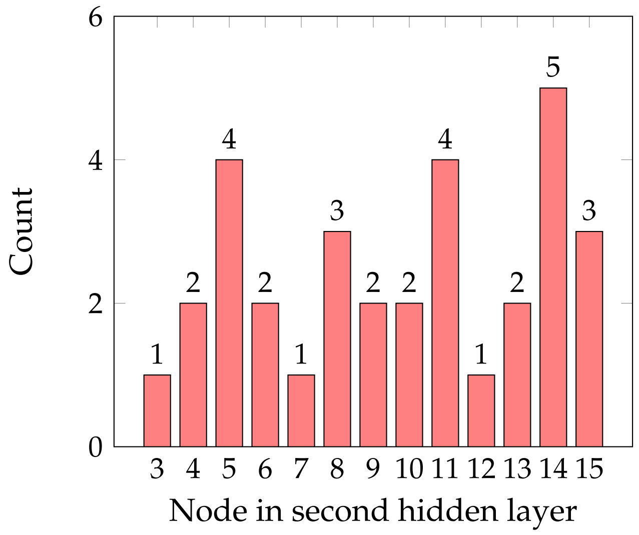


Figure 9.5: This bar graph shows the amount of times a wall following robot DNN node from the second hidden layer has been purged by an AEL. The occurrence of a node is likely to indicate its importance in the network.

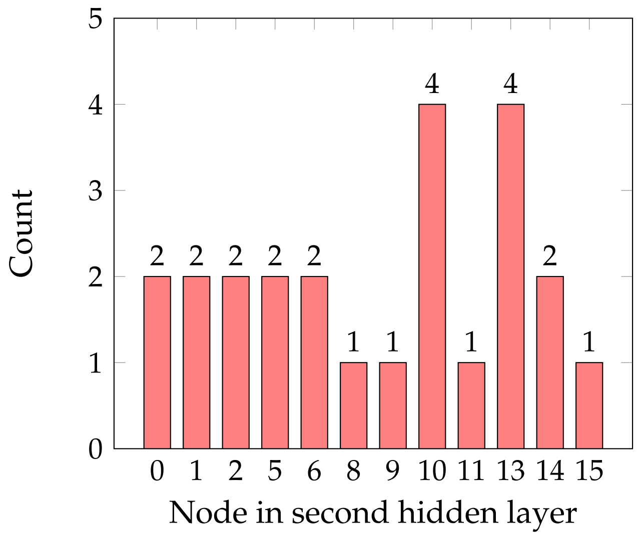


Figure 9.6: This bar graph shows the amount of times a Cardiotocography DNN node from the second hidden layer has been purged by an AEL. The occurrence of a node is likely to indicate its importance in the network.

based on the class that the ANN was predicting at the time. AEMS controlling nodes that are always active/inactive have not been included as they do not display the switching behaviour.

The activity of nodes in the wall following robot ANN that were switched by the first AEL were compared with the class that the network was predicting at the time is shown in Table 9.5. The AEM controlling node 15 is disabling it most of the time when the network is predicting a slight right turn, indicating that it is not particularly useful when the network is predicting that class. None of the AEMs disable nodes when the network is predicting a slight left turn, indicating that they are useful when predicting that class.

Table 9.6 shows the activity of nodes in the CTG ANN controlled by the fifth AEL with respect to the prediction of the network. All 3 nodes are highly active when the network is predicting a pathologic case, indicating that their functionality contributes to this decision. They are active approximately 50% of the time when the network is predicting a normal state, indicating that they are sometimes not useful, but this does not correlate with the prediction the network is making. Nodes 1 and 15 are highly inactive when the network is predicting a suspect case, indicating that they may not be particularly useful.

Table 9.5: Nodes deactivated based on wall following robot network prediction

Node	**Time inactive**			
	forward	sharp right	slight left	slight right
2	48.3%	18.8%	0%	29.3%
10	0.4%	29.3%	0%	51.2%
15	12.8%	8.2%	0%	92.7%

9.10 Summary

An artificial epigenetic layer has been designed to act between the two hidden layers of an ANN. The AEL receives input from the first hidden layer, processes it, and switches nodes from the second layer on and off dynamically. The functionality of the

Table 9.6: Nodes deactivated based on CTG network prediction

Node	**Time inactive**		
	normal	suspect	pathologic
1	53.0%	91.2%	15.4%
14	54.8%	35.3%	0%
15	53.0%	70.6%	7.7%

AEL arises from its constituent AEMs, each consisting of a regulatory function that performs the processing of the AEL. The AEL performs two functions that can be used to improve the transparency of the ANN it is acting upon.

Purging is the process of effectively removing nodes from an ANN, providing a level of transparency by showing which nodes are unimportant with regard to the performance of the network. This has the additional benefit of allowing smaller networks with fewer parameters to be constructed that require less computational resources, and can therefore run on simpler architectures.

The AELs have been shown to dynamically switch nodes based on the output of the nodes in the first hidden layer. This behaviour has been analysed with regard to each network prediction to suggest the nodes that were important when the predictions are being made. This has the potential to be further developed in order to determine the reasoning behind the predictions made by the network.

Chapter 10

Summary and Conclusions

This chapter summarises the work reported in this thesis and the rationale behind it. The overarching conclusions and contributions made will be discussed, followed by the limitations and areas where further work could be conducted.

10.1 Work Conducted and Rationale

Inspiration has been taken from biological systems to produce computational models such as artificial neural networks, artificial gene regulatory networks, and genetic algorithms. The motivation behind this stems from the recognition of the capabilities of their biological counterparts, and the desire to apply such capabilities to solving computational problems.

Typical procedural programming as found within conventional computer science is applicable to problems that are well understood and can be solved by a series of structured mathematical and logical operations. This becomes ineffective when dealing with more complex real world problems involving uncertainty and variability, such as object detection in images, and speech recognition from audio. Complex biological systems such as the brain and gene regulation have developed over millions of years due to evolution, and allow biological organisms to perform such tasks, which are crucial for their survival. The network of neurones found within the brain allows for co-ordination and the processing of information acquired by the senses in order for an organism to interact with its environment. Gene regulation ensures that genes are expressed at the right times to allow for the adaptation of an organism to a changing environment.

The artificial epigenetic network was designed as an improvement of the artificial gene regulatory network with the addition of a control layer inspired by epigenetic mechanisms. The control layer was shown to improve the performance of the networks by encouraging task decomposition by allowing genes to be selectively deactivated based on the state of the task being solved. The epigenetic control layer is typically evolved as part of the rest of the network, developing its behaviour alongside the regular genes in the network. Another proven benefit of the epigenetic layer is its ability to provide a degree of transparency to the execution of the AGRN. The epigenetic molecules constituting the layer are capable of dynamically deactivating genes within the AGRN, this behaviour can be analysed to correlate the deactivation of genes with the data that has been processed in order to deduce the reasoning behind the AGRN's actions.

One of the objectives of the research described within this thesis was to investigate the modularity of connectionist architectures. The functionality of such architectures is a product of the connectivity between nodes and their parameters. Previous research has involved the investigation of AGRN behaviour by analysing the behaviour of particular genes, but does not investigate the behaviour of groups of genes acting together. The experiments conducted on the modularity of AGRNs was intended to extend this research by investigating the occurrence of modules in the form of sub-networks, the intention being to determine whether it is feasible to reliably detect the formation of such sub-networks so that their behaviour can be analysed.

The abstract nature of connectionist architectures means that they are capable of performing tasks in many domains, provided that the input data is transformed to a suitable format. A major drawback of this is that the actions and decisions of such systems are not inherently explainable due to their structure. The black-box nature of such systems, conventionally referring to artificial neural networks, refers to the fact that observing their internal state does not provide any sort of explanation as to why they are functioning in a particular way.

10.2 Conclusions

The purpose of the experimentation described within this thesis was to establish whether a greater understanding of the modularity that occurs in connectionist architectures would allow for more effective models to be created, and based on this, whether a control layer inspired by epigenetics could be applied to a trained connectionist architecture in order to improve its transparency of execution.

10.2.1 Existence of More Effective Evolutionary Pathways in AGRNs

By constraining the evolution of artificial gene regulatory networks (AGRNs) based on the number of connections each gene has, it was shown to improve the performance of the resulting models. This provided evidence for the existence of evolutionary pathways that, when followed, produce more performant AGRNs in a shorter time frame.

10.2.2 Modularity within AGRNs

The experimentation conducted in Chapter 8 demonstrated that AGRNs are capable of solving multiple tasks with differing dynamics concurrently. In order to accomplish this, the AGRNs evolved with a degree of modularity pertaining to their internal organisation, and the relative positioning of genes assigned to each of the tasks. Internal modules developed in the form of sub-networks, each consisting of a set of genes.

10.2.3 It is Unlikely for AGRNs to Evolve with Clear Separation Between Modules

AGRNs are capable of evolving with modularity in the form of allocating different nodes to different tasks/sub-tasks; however, the experiments conducted in Chapter 8 showed that AGRNs evolve with much variation to their internal organisation. Although it is clear in most cases that there is modularity, there is often a degree of crossover between modules meaning that it is difficult to reliably detect sections of the network without manual analysis.

10.2.4 There is no Clear Indication of an Incentive for AGRNs to Evolve with Clear Modularity

When the modularity of AGRNs was investigated, the degree of interference between modules was observed. Interference in this case refers to interactions between one or more genes that are working to solve different tasks. When the performance of the AGRNs was compared to the interference occurring within them, it was found that there was no significant correlation. Performant networks evolved with varied interference, in this case meaning that there is no clear incentive for networks to evolve with clear separation between modules.

10.2.5 Application of Artificial Epigenetic Layer to Trained Connectionist Architecture

A dynamic control layer inspired by epigenetics was applied to the hidden layers of a trained artificial neural network (ANN) as described in Chapter 9. The control layer was shown to be capable of producing two types of functionality. Pruning involves the removal of parameters/nodes within ANNs to reduce their size, producing a smaller ANN that requires fewer computational resources to execute. The control layer was capable of identifying nodes that have the potential to be pruned by deactivating them regardless of the network state when used to make predictions on a dataset. The control layer was also shown to be capable of selectively deactivating parts of the ANN based on its state at that point in time. This information is useful in determining how the ANN functions with regard to the activity of particular nodes, as it identifies parts of the network that are not useful for making particular predictions. The ability to correlate network functionality with particular nodes has the potential to be used when solving the issue of AGRNs not evolving with clear modularity, as discussed previously.

10.3 Contributions

The following contributions have been made by the work described by this thesis:

- Evolutionary constraint based on the connectivity of genes has been applied to the evolution of artificial gene regulatory networks by a genetic algorithm. This demonstrated that there are certain evolutionary paths that produce more performant networks.
- The understanding that artificial gene regulatory networks evolve with varying degrees of modularity when trained to solve multiple independent tasks.
- Further evidence that artificial gene regulatory networks are capable of performing well even when there is interference amongst genes that are performing unrelated tasks.
- A control layer inspired by epigenetics has been developed that is capable of acting on a trained connectionist architecture. The control layer can be evolved independently, and is capable of dynamically disabling parts of the connectionist architecture based on its state.
- Analysis of control layer to correlate sections of the connectionist architecture with specific network functionality; in addition, information gathered from the analysis has the potential to be used to improve the computational efficiency of the connectionist architecture via pruning.

10.4 Validation of Hypothesis

The hypothesis of the thesis was broken down into two parts, the first part was stated as follows:

There are predictable evolutionary and structural properties of artificial gene regulatory networks that can be exploited to improve the transparency of execution and produce more performant networks.

The experiments conducted in chapters 7 and 8 produced two conclusions. The first, is that a predictable evolutionary pathway of AGRNs exists that results in more performant networks in a shorter timescale, and that the evolutionary process can be guided in this direction in order to take advantage of such networks. The second conclusion is that whilst AGRNs are capable of evolving a modular internal structure, the inherent robustness of the networks means that they can evolve with much variability to their modularity, resulting in difficulty when attempting to understand and manipulate it.

The second part of the hypothesis shifted focus toward the application of an epigenetic analogue to a trained connectionist architecture, and was stated as follows:

A dynamic control layer inspired by epigenetics can be applied to a connectionist architecture, which can be analysed to provide a degree of transparency to the connectionist architecture in the form of a description of its internal workings.

A control layer inspired by epigenetic mechanisms was constructed to act on a trained artificial neural network. The control layer was shown to be capable of dynamically disabling nodes within the ANN based on its state. The analysis of the behaviour of the epigenetic layer was shown to prune nodes and switch them on and off as the data processed by the ANN changed. Analysis of the switching behaviour of the ANN allowed for correlations to be made between the deactivation of nodes and the prediction the network was making at that point in time. Indeed, this information can be used to determine the parts of the ANN that are responsible for certain functionality.

10.5 Discussion and Areas for Further Research

The conclusions drawn from the research conducted and the contributions made have been stated; what follows is a further discussion on the implications, limitations, and directions in which the research could be taken in order to develop it.

10.5.1 Evolutionary Pathways for AGRNs

The experiments in Chapter 7 were conducted in order to investigate the possibility of constraining the evolution of AGRNs in order to produce more performant networks. It was predicted that the connectivity of genes influenced the probability of a successful mutation, meaning one that results in a more performant network. AGRNs were iteratively evolved to gather data based on this, which was used to generate rules that were applied to the evolution of a second set of AGRNs in order to constraint it. Generally speaking, it was found that mutations to genes with two or three connections were more beneficial. The performance of the latter set of AGRNs was found to be an improvement on those that were not constrained, suggesting that the theorised evolutionary pathway is beneficial to follow. The experiment involved the evolution of AGRNs to solve two different tasks, a control task, and a pattern recognition task. In both cases the constraint was shown to be useful.

Constraining the evolution of AGRNs in this way was shown to produce more performant AGRNs in a shorter timescale, requiring fewer computational resources and less time in order for evolution to take place. When AGRNs are evolved, a high population size may be used in order to explore as much of the search space as possible. In this case it is desirable to reduce the duration of evolution in order to minimise the resources used. Constraining the evolution of AGRNs in this way has the potential to be applied in order to achieve this.

The AGRNs were evolved to solve a pendulum and pattern recognition task, demonstrating the applicability to AGRNs operating within environments of different dynamics; however, further investigation involving AGRNs solving tasks in more environments would support the generalisability of this technique. In this case, it was beneficial to encourage mutations to genes with two or three connections; however, it may be the case that this differs in AGRNs solving tasks with other dynamics, or AGRNs that are greater in size. In order for the constraint to be applied, the rules had to be generated based on data gathered from the evolution of a set of AGRNs. In practice, this would negate the time saved by applying the constraint. Perhaps a general rule could be devised and applied to all AGRNs, or a conditional rule where the constraint is applied differently depending on the size of the network. It may also be feasible to generate the rules based on a smaller set of AGRNs that would take less time to evolve.

One of the major limitations was that the AGRNs were only trained to solve two different tasks. This demonstrated that the technique could be used on AGRNs solving tasks with two different dynamics; however, this is only a starting point and it would be beneficial to demonstrate this technique with AGRNs solving more tasks in different domains with differing dynamics. For example, an AGRN solving a signal processing task could be tested.

10.5.2 Detecting Modularity in AGRNs

Based on the prevalence of modularity found in biological organisms, such as the existence of operons found within the genomes of prokaryotic organisms, the experiments in Chapter 8 were conducted in order to explore the extent to which modularity occurs during the evolution of AGRNs. A set of AGRNs were evolved to solve three completely independent tasks concurrently, so that the internal organisation of the genes could be studied in order to observe whether sub-networks of genes were formed, and if so, to what extent. The AGRNs were trained to solve completely independent tasks so that the genes directly interacting with each environment could be easily identified, as well as interference between genes of different tasks. The AGRNs were represented as a virtual indirect reference space, the gene connections were determined based on their position in this space, and their proximity with other genes. The positions of genes were encouraged to slowly move around in this space and to slowly change shape, to allow for small, incremental changes to allow for sub-networks of genes to form.

The internal structures of the evolved AGRNs were analysed by identifying the genes responsible for solving each task, and observing the degree to which they formed distinct sub-networks, as well as the interference that occurred between the sub-networks. The internal organisation of nodes varied amongst the evolved AGRNs, some evolved with a clear separation between genes solving each of the tasks; however, most evolved with at least some interference between them, as genes acting to solve different tasks formed connections. This is most likely attributed to the robustness of AGRNs; that is, that they are capable of functioning despite internal and external perturbations.

The purpose of conducting the experiments in Chapter 8 was to determine whether AGRNs are capable of evolving to organise their genes into clear sub-networks that can be easily detected. The detection of such sub-networks could allow for the improvement of AGRN performance by taking advantage of such knowledge; for example, by devising a technique to preserve such sub-networks so that they are not broken up during evolution. Based on the variability of the evolved AGRNs, it was determined that it would not be feasible to reliably and autonomously detect the sub-networks formed, due to the interference that often occurred.

The AGRNs were represented in a two-dimensional reference space, each gene existing as a rectangle with a variable position, length and width. A connection from one gene was made to another if the former gene's centre point existed within the space of the latter. This meant that if many genes were grouped closely together, it was likely that many connections would form amongst these genes. Future work could involve refining this mechanism to make it more restrictive; for example, by defining specific areas on each gene where connections could be made, similar to the activation sites of enzymes. This is likely to reduce the number of connections made between genes, as well as encouraging connections to be made in a slower and more controlled manner. This would increase the complexity of the AGRN, meaning that it may take longer to evolve and execute; so care would have to be taken to ensure that the implementation is efficient.

The AGRNs were trained to solve three independent tasks with the intention of causing the network to internally organise itself based on these tasks. The tasks were chosen due to their differing dynamics; however, they were all control tasks. This is a limitation of the experiment as it only demonstrates the applicability of the method in this domain. It would be beneficial to perform the same experiment on AGRNs solving tasks in other domains, such as pattern recognition and signal processing. Future work could involve this, with an emphasis on training AGRNs to solve multiple tasks in different domains simultaneously. This may also have the benefit of encouraging the AGRNs to self-organise more than they would when solving tasks in the same domain.

10.5.3 Application of an Epigenetic Control Layer

The experiments described in Chapter 9 were conducted in order to determine whether is was possible to apply a control layer inspired by epigenetic mechanisms to a trained artificial neural network (ANN). The intention being for the control layer to provide information about the execution of the ANN that could be used to improve the transparency of its execution. The artificial epigenetic layer (AEL) was designed to act between two layers of an ANN, the first layer acting as the input to the AEL and the second as the output. The AEL consisted of many artificial epigenetic molecules (AEMs), consisting of a regulatory function and a state. AEMs can be activate or inactive when activated they effectively remove their output nodes from the ANN. Their state is determined by processing the outputs of the nodes from the first layer of the ANN using their regulatory function. Each time the ANN processed an input vector, the AEMs would execute. During experimentation, the AELs acted between the first and second hidden layer of two ANNs, trained to classify two different datasets.

Two sets of AELs were evolved by a genetic algorithm to act on each of the ANNs. The AELs were capable of producing two types of behaviour. The switching behaviour of the AELs was analysed to determine the percentage of time that the AEL deactivated a node from the ANN. In some cases, an AEL would deactivate a node 100% of the time as it classified the dataset. In this case, the node was considered as being purged, as it was effectively removed from the ANN. The dynamic switching behaviour of the AELs allowed nodes from the ANN to be deactivated and reactivated over time. This behaviour was correlated with the prediction that the network was making in order to determine the nodes that were not used based on the activity of the ANN.

Purging has been used in previous work in order to reduce the size of a neural network in order to reduce the computational resources required for it to execute. This is beneficial as it has the potential to reduce the energy consumption of the machine that it is running on, and can allow machines with less powerful architectures to execute the network, such as mobile and potentially embedded devices. The purging behaviour of the AEL is capable of indicating nodes that could be removed from the ANN in order to reduce its size. Analysis of the switching behaviour has been used to compare the structure of the ANN to its functionality. This information has the potential to be used to provide a degree of transparency in the form of indicating the nodes responsible for particular network actions,

The evolution of the AELs required a multi-core computer to operate over a period of days. During each epoch of the genetic algorithm, the ANNs had to be evaluated on a dataset with an AEL applied. The application of the AELs increased the duration of ANN evaluation, as the state of the AEMs had to be processed. For this reason, the size of the datasets had to be limited for the experiments to be conducted in a reasonable time frame. The AEL would benefit from further developments to improve the efficiency of execution, particularly during training. This could be accomplished by redesigning the AELs to operate at a lower level computationally, as they currently act by extracting the weights of the ANN in order to process them, when it may be possible to interact with the network directly by implementing them as a layer compatible with a neural network framework.

When improving the transparency of connectionist architectures, it is contradictory to achieve this at the expense of complexity. It could be argued that an epigenetic control layer is not inherently transparent, as it consists of nodes that are similar to those found within the ANN it is operating on. For this reason, a compromise was carefully established between the intelligibility and functionality of the AEL. Each epigenetic molecule within the AEL had two discrete states. When active, an AEM would deactivate a single node from the second hidden layer. This allowed the AEMs to be associated with a node directly, meaning that its effect on the ANN was clear. Each AEM consisted of a regulatory function, a sigmoid with a slope and offset parameter. Despite this simplicity, the AEL was capable of producing dynamic behaviour. Any developments to the AEL would have to be made with careful consideration concerning its complexity, as it would be paradoxical to attempt to improve the transparency of a connectionist architecture using a technique that is in itself unintelligible.

The scope of the experimentation was to determine whether it was possible for an epigenetic layer to be used in this particular circumstance, for this reason it was only applied to the ANN between the first and second hidden layer. To gain a more comprehensive understanding of the ANN, it would be beneficial to apply the epigenetic layer to other layers in the network. For example, the application of the layer between the input and first layer could allow for deductions to be made about the effects of the input data.

Bibliography

[Adadi and Berrada, 2018] Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160.

[Alberts, 2003] Alberts, B. (2003). Dna replication and recombination. *Nature*, 421(6921):431–435.

[Amir et al., 1999] Amir, R. E., Van den Veyver, I. B., Wan, M., Tran, C. Q., Francke, U., and Zoghbi, H. Y. (1999). Rett syndrome is caused by mutations in x-linked mecp2, encoding methyl-cpg-binding protein 2. *Nature genetics*, 23(2):185–188.

[Amjad et al., 2018] Amjad, R. A., Liu, K., and Geiger, B. C. (2018). Understanding individual neuron importance using information theory. *arXiv preprint arXiv:1804.06679*.

[Annunziato, 2008] Annunziato, A. (2008). Dna packaging: nucleosomes and chromatin. *Nature Education*, 1(1):26.

[Bach et al., 2015] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7).

[Bäck and Schwefel, 1993] Bäck, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

[Baker et al., 2016] Baker, B., Gupta, O., Naik, N., and Raskar, R. (2016). Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

[Bannister and Kouzarides, 2011] Bannister, A. J. and Kouzarides, T. (2011). Regulation of chromatin by histone modifications. *Cell research*, 21(3):381–395.

[Barlow and Bartolomei, 2014] Barlow, D. P. and Bartolomei, M. S. (2014). Genomic imprinting in mammals. *Cold Spring Harbor perspectives in biology*, 6(2):a018382.

[Barlow, 1989] Barlow, H. B. (1989). Unsupervised learning. *Neural computation*, 1(3):295–311.

[Barreira and Valls, 2019] Barreira, L. and Valls, C. (2019). *Dynamical systems by example*. Springer.

[Baylin, 2005] Baylin, S. B. (2005). Dna methylation and gene silencing in cancer. *Nature clinical practice Oncology*, 2(1):S4–S11.

[Baylin and Herman, 2000] Baylin, S. B. and Herman, J. G. (2000). Dna hypermethylation in tumorigenesis: epigenetics joins genetics. *Trends in genetics*, 16(4):168–174.

[Becker, 2002] Becker, P. B. (2002). Nucleosome sliding: facts and fiction. *The EMBO journal*, 21(18):4749–4753.

[Benardos and Vosniakos, 2007] Benardos, P. and Vosniakos, G.-C. (2007). Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3):365–382.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Berg et al., 2002] Berg, J. M., Tymoczko, J. L., and Stryer, L. (2002). Biochemistry, ; w. h.

[Bernstein, 1961] Bernstein, J. J. (1961). Loss of hue discrimination in forebrain-ablated fish. *Experimental Neurology*, 3:1—17.

[Blanco and Blanco, 2017] Blanco, A. and Blanco, G. (2017). Proteins. In Blanco, A. and Blanco, G., editors, *Medical Biochemistry*, pages 21–71. Academic Press.

[Boeing, 2016] Boeing, G. (2016). Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. *Systems*, 4(4):37.

[Bojarski et al., 2016] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[Bostick et al., 2007] Bostick, M., Kim, J. K., Estève, P.-O., Clark, A., Pradhan, S., and Jacobsen, S. E. (2007). Uhrf1 plays a role in maintaining dna methylation in mammalian cells. *Science*, 317(5845):1760–1764.

[Brown, 1998] Brown, T. A. (1998). *Genetics: a molecular approach*. BIOS Scientific/Garland Science, 3 edition.

[Buckner and Garson, 2019] Buckner, C. and Garson, J. (2019). Connectionism. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2019 edition.

[Cai et al., 2007] Cai, Y., Jin, J., Yao, T., Gottschalk, A. J., Swanson, S. K., Wu, S., Shi, Y., Washburn, M. P., Florens, L., Conaway, R. C., et al. (2007). Yy1 functions with ino80 to activate transcription. *Nature structural & molecular biology*, 14(9):872–874.

[Caruana et al., 2015] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730.

[Caruana and Niculescu-Mizil, 2006] Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168.

[Cedar and Bergman, 2009] Cedar, H. and Bergman, Y. (2009). Linking dna methylation and histone modification: patterns and paradigms. *Nature Reviews Genetics*, 10(5):295–304.

[Chen et al., 2004] Chen, K. C., Calzone, L., Csikasz-Nagy, A., Cross, F. R., Novak, B., and Tyson, J. J. (2004). Integrative analysis of cell cycle control in budding yeast. *Molecular biology of the cell*, 15(8):3841–3862.

[Cheng et al., 2013] Cheng, X., Sun, M., and Socolar, J. E. (2013). Autonomous boolean modelling of developmental gene regulatory networks. *Journal of the Royal Society Interface*, 10(78):20120574.

[Clancy, 2008a] Clancy, S. (2008a). Dna transcription. *Nature education*, 1(1):41.

[Clancy, 2008b] Clancy, S. (2008b). Genetic recombination. *Nature education*, 1(1):40.

[Clancy and Brown, 2008] Clancy, S. and Brown, W. (2008). Translation: Dna to mrna to protein. *Nature Education*, 1(1):101.

[Clancy et al., 2008] Clancy, S. et al. (2008). Chemical structure of rna. *Nature Education*, 1(1):223.

[Clarkson et al., 1999] Clarkson, M. J., Wells, J. R., Gibson, F., Saint, R., and Tremethick, D. J. (1999). Regions of variant histone his2avd required for drosophila development. *Nature*, 399(6737):694–697.

[Coello Coello et al., 2020] Coello Coello, C. A., González Brambila, S., Figueroa Gamboa, J., Castillo Tapia, M. G., and Hernández Gómez, R. (2020). Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. *Complex & Intelligent Systems*, 6(2):221–236.

[Crick, 1958] Crick, F. (1958). On protein synthesis. *Symposia of the Society for Experimental Biology*, 12:138–163.

[Crick, 1970] Crick, F. (1970). Central dogma of molecular biology. *Nature*, 227:561–563.

[Cunningham et al., 2008] Cunningham, P., Cord, M., and Delany, S. J. (2008). Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer.

[Cussat-Blanc et al., 2019] Cussat-Blanc, S., Harrington, K., and Banzhaf, W. (2019). Artificial gene regulatory networks—a review. *Artificial life*, 24(4):296–328.

[Daly et al., 2012] Daly, B., Eichen, D., Bailer, B., Brown, R., and Buchanan, C. (2012). Central nervous system. In Ramachandran, V., editor, *Encyclopedia of Human Behavior (Second Edition)*, pages 454 – 459. Academic Press, San Diego, second edition edition.

[De Jong, 2002] De Jong, H. (2002). Modeling and simulation of genetic regulatory systems: a literature review. *Journal of computational biology*, 9(1):67–103.

[Deaton and Bird, 2011] Deaton, A. M. and Bird, A. (2011). Cpg islands and the regulation of transcription. *Genes & development*, 25(10):1010–1022.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[Dellaert and Beer, 1994] Dellaert, F. and Beer, R. D. (1994). Toward an evolvable model of development for autonomous agent synthesis. In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 246–257. MIT press.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on* *computer vision and pattern recognition*, pages 248–255. Ieee.

[Duncan et al., 2014] Duncan, E. J., Gluckman, P. D., and Dearden, P. K. (2014). Epigenetics, plasticity, and evolution: how do we link epigenetic change to phenotype? *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution*, 322(4):208–220.

[Dupret and Liao, 2010] Dupret, G. and Liao, C. (2010). A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 181–190.

[Edunov et al., 2018] Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

[Filipowicz and Paszkowski, 2013] Filipowicz, W. and Paszkowski, J. (2013). Gene silencing. In Maloy, S. and Hughes, K., editors, *Brenner's Encyclopedia of Genetics (Second Edition)*, pages 221–222. Academic Press, San Diego, second edition edition.

[Filner, 1965] Filner, P. (1965). Semi-conservative replication of dna in a higher plant cell. *Experimental cell research*, 39(1):33–39.

[Frankle and Carbin, 2018] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

[Franzini et al., 2019] Franzini, A., Moosa, S., Servello, D., Small, I., DiMeco, F., Xu, Z., Elias, W. J., Franzini, A., and Prada, F. (2019). Ablative brain surgery: an overview. *International Journal of Hyperthermia*, 36(2):64–80.

[Fuerst, 2010] Fuerst, J. A. (2010). Beyond prokaryotes and eukaryotes: planctomycetes and cell organization. *Nature Education*, 3(9, Article No. 44):Online.

[Garbanati et al., 1983] Garbanati, J. A., Sherman, G. F., Rosen, G. D., Hofmann, M., Yutzey, D. A., and Denenberg, V. H. (1983). Handling in infancy, brain laterality and muricide in rats. *Behavioural Brain Research*, 7(3):351–359.

[Garcia and Lopez, 2018] Garcia, M. P. and Lopez, S. S. (2018). Building trust between users and telecommunications data driven virtual assistants. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 628–637. Springer.

[Gillespie, 1976] Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics*, 22(4):403–434.

[Grantham et al., 1980] Grantham, R., Gautier, C., Gouy, M., Mercier, R., and Pave, A. (1980). Codon catalog usage and the genome hypothesis. *Nucleic acids research*, 8(1):197–197.

[Grassberger and Procaccia, 1983] Grassberger, P. and Procaccia, I. (1983). Characterization of strange attractors. *Physical review letters*, 50(5):346.

[Griffiths, 1999] Griffiths, A. J. (1999). *Modern genetic analysis*. W.H. Freeman.

[Griffiths et al., 2000] Griffiths, A. J., Doebley, J., Peichel, C., and Wassarman, D. A. (2000). *An Introduction to Genetic Analysis*. W.H. Freeman, 7th edition.

[Guo et al., 2000] Guo, S.-M., San Shieh, L., Chen, G., and Lin, C.-F. (2000). Effective chaotic orbit tracker: a prediction-based digital redesign approach. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(11):1557–1570.

[Haig, 2004] Haig, D. (2004). The (dual) origin of epigenetics. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 69, pages 67–70. Cold Spring Harbor Laboratory Press.

[Hamann et al., 2011] Hamann, H., Schmickl, T., and Crailsheim, K. (2011). Coupled inverted pendulums: a benchmark for evolving decentral controllers in modular robotics. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 195–202.

[Han et al., 2015] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143.

[Hansen, 2015] Hansen, T. F. (2015). Evolutionary constraints.

[Hartwell et al., 2008] Hartwell, L., Hood, L., Goldberg, M., Reynolds, A., Silver, L. M., and Veres, R. (2008). *Genetics from genes to genomes*. McGraw-Hill, 3 edition.

[Harvey and Bossomaier, 1997] Harvey, I. and Bossomaier, T. (1997). Time out of joint: Attractors in asynchronous random boolean networks. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 67–75. MIT Press, Cambridge.

[Hattman, 2005] Hattman, S. (2005). Dna-[adenine] methylation in lower eukaryotes.

Biochemistry (Moscow), 70(5):550–558.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hecht, 2018] Hecht, J. (2018). Lidar for self-driving cars. *Optics and Photonics News*, 29(1):26–33.

[Heithoff et al., 1999] Heithoff, D. M., Sinsheimer, R. L., Low, D. A., and Mahan, M. J. (1999). An essential role for dna adenine methylation in bacterial virulence. *Science*, 284(5416):967–970.

[Hendry, 2005] Hendry, A. P. (2005). The power of natural selection. *Nature*, 433(7027):694–695.

[Hilborn et al., 2000] Hilborn, R. C. et al. (2000). *Chaos and nonlinear dynamics: an introduction for scientists and engineers*. Oxford University Press on Demand.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hoernes et al., 2018] Hoernes, T. P., Faserl, K., Juen, M. A., Kremser, J., Gasser, C., Fuchs, E., Shi, X., Siewert, A., Lindner, H., Kreutz, C., et al. (2018). Translation of non-standard codon nucleotides reveals minimal requirements for codon-anticodon interactions. *Nature communications*, 9(1):4865.

[Holland, 1992] Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

[Holliday, 2006] Holliday, R. (2006). Epigenetics: a historical overview. *Epigenetics*, 1(2):76–80.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[Hsieh, 1994] Hsieh, C.-L. (1994). Dependence of transcriptional repression on cpg methylation density. *Molecular and cellular biology*, 14(8):5487.

[Hunt, 1991] Hunt, E. R. (1991). Stabilizing high-period orbits in a chaotic system: The diode resonator. *Physical Review Letters*, 67(15):1953.

[Iwasaki et al., 2013] Iwasaki, W., Miya, Y., Horikoshi, N., Osakabe, A., Taguchi, H., Tachiwana, H., Shibata, T., Kagawa, W., and Kurumizaka, H. (2013). Contribution of histone n-terminal tails to the structure and stability of nucleosomes. *FEBS open bio*, 3:363–369.

[Jacob and Monod, 1961] Jacob, F. and Monod, J. (1961). On the regulation of gene activity. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 26, pages 193–211. Cold Spring Harbor Laboratory Press.

[Jankovic, 2001] Jankovic, J. (2001). Surgery for parkinson disease and other movement disorders: benefits and limitations of ablation, stimulation, restoration, and radiation. *Archives of neurology*, 58(12):1970–1972.

[Jin et al., 2005] Jin, J., Cai, Y., Li, B., Conaway, R. C., Workman, J. L., Conaway, J. W., and Kusch, T. (2005). In and out: histone variant exchange in chromatin. *Trends in biochemical sciences*, 30(12):680–687.

[Johnson, 1974] Johnson, D. (1974). Hairpin-tail: a case of post-reductional gene action in the mouse egg? *Genetics*, 76(4):795–805.

[Jones and Baylin, 2002] Jones, P. A. and Baylin, S. B. (2002). The fundamental role of epigenetic events in cancer. *Nature reviews genetics*, 3(6):415–428.

[Jones and Takai, 2001] Jones, P. A. and Takai, D. (2001). The role of dna methylation in mammalian epigenetics. *Science*, 293(5532):1068–1070.

[Juliandi et al., 2010] Juliandi, B., Abematsu, M., and Nakashima, K. (2010). Epigenetic regulation in neural stem cell differentiation. *Development, growth & differentiation*, 52(6):493–504.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[Karlebach and Shamir, 2008] Karlebach, G. and Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770.

[Kauffman, 1969] Kauffman, S. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437 – 467.

[Kauffman et al., 2003] Kauffman, S., Peterson, C., Samuelsson, B., and Troein, C. (2003). Random boolean network models and the yeast transcriptional network.

Proceedings of the National Academy of Sciences, 100(25):14796–14799.

[Keedwell et al., 2002] Keedwell, E., Narayanan, A., and Savic, D. (2002). Modelling gene regulatory data using artificial neural networks. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 1, pages 183–188. IEEE.

[Kepuska and Bohouta, 2018] Kepuska, V. and Bohouta, G. (2018). Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 99–103. IEEE.

[Kirchner and Ignatova, 2015] Kirchner, S. and Ignatova, Z. (2015). Emerging roles of trna in adaptive translation, signalling dynamics and disease. *Nature Reviews Genetics*, 16(2):98–112.

[Klopf et al., 2017] Klopf, E., Schmidt, H. A., Clauder-Münster, S., Steinmetz, L. M., and Schüller, C. (2017). Ino80 represses osmostress induced gene expression by resetting promoter proximal nucleosomes. *Nucleic acids research*, 45(7):3752–3766.

[Klug et al., 2019] Klug, W. S., Cummings, M. R., A, C., Palladino, M. A., and Killian, D. J. (2019). *Concepts of Genetics*. Pearson Education, 12 edition.

[Koza and Poli, 2005] Koza, J. R. and Poli, R. (2005). Genetic programming. In *Search methodologies*, pages 127–164. Springer.

[Kramer, 2013] Kramer, J. M. (2013). Epigenetic regulation of memory: implications in human cognitive disorders. *Biomolecular concepts*, 4(1):1–12.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kulkarni et al., 2018] Kulkarni, R., Dhavalikar, S., and Bangar, S. (2018). Traffic light detection and recognition for self driving cars using deep learning. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–4. IEEE.

[Lacey et al., 2019] Lacey, G., Schoene, A., Dethlefs, N., and Turner, A. (2019). Modularity within artificial gene regulatory networks. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 731–738. IEEE.

[Lacey et al., 2020] Lacey, G., Schoene, A., Dethlefs, N., and Turner, A. (2020). Improving the transparency of deep neural networks using artificial epigenetic molecules. In *12th International Joint Conference on Computational Intelligence*, pages 167–175. INSTICC.

[Lai et al., 2016] Lai, X., Wolkenhauer, O., and Vera, J. (2016). Understanding microRNA-mediated gene regulatory networks through mathematical modelling. *Nucleic Acids Research*, 44(13):6019–6035.

[Latchman, 2005] Latchman, D. (2005). *Gene regulation: a eukaryotic perspective*. Taylor & Francis, 5th edition.

[LeCun et al., 1990] LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605.

[Levene, 1919] Levene, P. (1919). The structure of yeast nucleic acid. *Studies from the Rockefeller Institute for Medical Research*, 30:221.

[Li and Chen, 2004] Li, C. and Chen, G. (2004). Chaos in the fractional order chen system and its control. *Chaos, Solitons & Fractals*, 22(3):549–554.

[Li et al., 2016] Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. (2016). Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

[Li et al., 2010] Li, J., Yuan, Z., and Zhang, Z. (2010). The cellular robustness by genetic redundancy in budding yeast. *PLoS Genet*, 6(11):e1001187.

[Li et al., 2008] Li, S., Brazhnik, P., Sobral, B., and Tyson, J. J. (2008). A quantitative study of the division cycle of caulobacter crescentus stalked cells. *PLOS Computational Biology*, 4(1):1–19.

[Lillian et al., 2018] Lillian, P. E., Meyes, R., and Meisen, T. (2018). Ablation of a robot's brain: Neural networks under a knife. *arXiv preprint arXiv:1812.05687*.

[Lipton et al., 2015] Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

[Liu et al., 2019] Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z., and Ling, H. (2019). Cbnet: A novel composite backbone network architecture for object detection. *arXiv preprint arXiv:1909.03625*.

[Loeb and Monnat Jr, 2008] Loeb, L. A. and Monnat Jr, R. J. (2008). Dna polymerases and human disease. *Nature Reviews Genetics*, 9(8):594.

[Lones et al., 2013] Lones, M. A., Turner, A. P., Fuente, L. A., Stepney, S., Caves, L. S., and Tyrrell, A. M. (2013). Biochemical connectionism. *Natural Computing*, 12(4):453–472.

[Lones et al., 2010] Lones, M. A., Tyrrell, A. M., Stepney, S., and Caves, L. S. (2010). Controlling complex dynamics with artificial biochemical networks. In Esparcia-Alcázar, A. I., Ekárt, A., Silva, S., Dignum, S., and Uyar, A. Ş., editors, *Genetic Programming*, pages 159–170, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Lorenz, 1963] Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141.

[Lu et al., 2004] Lu, J., Wu, X., Han, X., and Lü, J. (2004). Adaptive feedback synchronization of a unified chaotic system. *Physics Letters A*, 329(4-5):327–333.

[Ludwig, 2002] Ludwig, M. Z. (2002). Functional evolution of noncoding dna. *Current Opinion in Genetics & Development*, 12(6):634 – 639.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Maqueda et al., 2018] Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427.

[Maston et al., 2006] Maston, G. A., Evans, S. K., and Green, M. R. (2006). Transcriptional regulatory elements in the human genome. *Annu. Rev. Genomics Hum. Genet.*, 7:29–59.

[McAdams and Arkin, 1997] McAdams, H. H. and Arkin, A. (1997). Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences*, 94(3):814–819.

[Medler, 1998] Medler, D. A. (1998). A brief history of connectionism. *Neural computing surveys*, 1(1):61–101.

[Montavon et al., 2019] Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. (2019). Layer-wise relevance propagation: an overview. In *Explainable*

AI: Interpreting, Explaining and Visualizing Deep Learning, pages 193–209. Springer.

[Montavon et al., 2018] Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

[Moore et al., 2013] Moore, L. D., Le, T., and Fan, G. (2013). DNA methylation and its basic function. *Neuropsychopharmacology*, 38(1):23–38.

[Morcos et al., 2018] Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., and Botvinick, M. (2018). On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*.

[Müller and Olsson, 2003] Müller, G. B. and Olsson, L. (2003). Epigenesis and epigenetics. *Keywords and concepts in evolutionary developmental biology*, 114.

[Murata et al., 1995] Murata, T., Ishibuchi, H., et al. (1995). Moga: multi-objective genetic algorithms. In *IEEE international conference on evolutionary computation*, volume 1, pages 289–294. IEEE Piscataway, NJ, USA.

[Narlikar et al., 2002] Narlikar, G. J., Fan, H.-Y., and Kingston, R. E. (2002). Cooperation between complexes that regulate chromatin structure and transcription. *Cell*, 108(4):475–487.

[Narouze et al., 2009] Narouze, S., Kapural, L., Casanova, J., and Mekhail, N. (2009). Sphenopalatine ganglion radiofrequency ablation for the management of chronic cluster headache. *Headache: The Journal of Head and Face Pain*, 49(4):571–577.

[Oberlander et al., 2008] Oberlander, T. F., Weinberg, J., Papsdorf, M., Grunau, R., Misri, S., and Devlin, A. M. (2008). Prenatal exposure to maternal depression, neonatal methylation of human glucocorticoid receptor gene (nr3c1) and infant cortisol stress responses. *Epigenetics*, 3(2):97–106.

[Okazaki et al., 1968] Okazaki, R., Okazaki, T., Sakabe, K., Sugimoto, K., and Sugino, A. (1968). Mechanism of dna chain growth. i. possible discontinuity and unusual secondary structure of newly synthesized chains. *Proceedings of the National Academy of Sciences of the United States of America*, 59(2):598.

[Ooi et al., 2007] Ooi, S. K., Qiu, C., Bernstein, E., Li, K., Jia, D., Yang, Z., Erdjument-Bromage, H., Tempst, P., Lin, S.-P., Allis, C. D., et al. (2007). DNMT3L connects unmethylated lysine 4 of histone H3 to de novo methylation of DNA. *Nature*, 448(7154):714–717.

[Osbourn and Field, 2009] Osbourn, A. E. and Field, B. (2009). Operons. *Cellular and Molecular Life Sciences*, 66(23):3755–3775.

[Palatnik de Sousa et al., 2019] Palatnik de Sousa, I., Maria Bernardes Rebuzzi Vellasco, M., and Costa da Silva, E. (2019). Local interpretable model-agnostic explanations for classification of lymph node metastases. *Sensors*, 19(13):2969.

[Persson and Wagner, 1996] Persson, P. and Wagner, C. (1996). General principles of chaotic dynamics. *Cardiovascular Research*, 31(3):332–341.

[Peterson and Laniel, 2004] Peterson, C. L. and Laniel, M.-A. (2004). Histones and histone modifications. *Current Biology*, 14(14):R546–R551.

[Phillips et al., 2008] Phillips, T. et al. (2008). The role of methylation in gene expression. *Nature Education*, 1(1):116.

[Phillips and Shaw, 2008] Phillips, T. and Shaw, K. (2008). Chromatin remodeling in eukaryotes. *Nature Education*, 1(1):209.

[Pirlot, 1996] Pirlot, M. (1996). General local search methods. *European journal of operational research*, 92(3):493–511.

[Plath et al., 2003] Plath, K., Fang, J., Mlynarczyk-Evans, S. K., Cao, R., Worringer, K. A., Wang, H., Cecile, C., Otte, A. P., Panning, B., and Zhang, Y. (2003). Role of histone h3 lysine 27 methylation in x inactivation. *Science*, 300(5616):131–135.

[Poli et al., 2017] Poli, J., Gasser, S. M., and Papamichos-Chronakis, M. (2017). The ino80 remodeller in transcription, replication and repair. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1731):20160290.

[Polynikis et al., 2009] Polynikis, A., Hogan, S., and di Bernardo, M. (2009). Comparing different ode modelling approaches for gene regulatory networks. *Journal of theoretical biology*, 261(4):511–530.

[Pray, 2008a] Pray, L. (2008a). Discovery of dna structure and function: Watson and crick. *Nature Education*, 1(1):100.

[Pray, 2008b] Pray, L. (2008b). Major molecular events of dna replication. *Nature Education*, 1(1):99.

[Price and Shmulevich, 2007] Price, N. D. and Shmulevich, I. (2007). Biochemical and statistical network models for systems biology. *Current opinion in biotechnology*, 18(4):365–370.

[Purves et al., 2004] Purves, D., Augustine, G., Fitzpatrick, D., Hall, W., LaMantia, A., McNamara, J., and Williams, S. (2004). *Studying the Nervous Systems of Humans and Other Animals*, chapter 1, pages 1–28. Sinauer Associates, Inc., 3 edition.

[Ralston, 2008] Ralston, A. (2008). Operons and prokaryotic gene regulation. *Nature Education*.

[Ralston and Shaw, 2008] Ralston, A. and Shaw, K. (2008). Gene expression regulates cell differentiation. *Nat Educ*, 1(1):127–131.

[Ramos et al., 2017] Ramos, S., Gehrig, S., Pinggera, P., Franke, U., and Rother, C. (2017). Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1025–1032. IEEE.

[Rangasamy et al., 2004] Rangasamy, D., Greaves, I., and Tremethick, D. J. (2004). Rna interference demonstrates a novel role for h2a. z in chromosome segregation. *Nature structural & molecular biology*, 11(7):650–655.

[Reik, 1989] Reik, W. (1989). Genomic imprinting and genetic disorders in man. *Trends in Genetics*, 5:332–336.

[Reik and Walter, 2001] Reik, W. and Walter, J. (2001). Genomic imprinting: parental influence on the genome. *Nature Reviews Genetics*, 2(1):21–32.

[Reil, 1999] Reil, T. (1999). Dynamics of gene expression in an artificial genome—implications for biological and artificial ontogeny. In *European Conference on Artificial Life*, pages 457–466. Springer.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

[Riggs et al., 1996] Riggs, A. D., Martienssen, R. A., and Russo, V. E. (1996). *Introduction*, volume 32, page 1. Cold Spring Harbor Laboratory Press, New York.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Russom et al., 2011] Russom, P. et al. (2011). Big data analytics. *TDWI best practices report, fourth quarter*, 19(4):1–34.

[Saltelli et al., 2004] Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). *Sensitivity analysis in practice: a guide to assessing scientific models*, volume 1. Wiley Online Library.

[Samek et al., 2017] Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

[Seshasayee et al., 2011] Seshasayee, A. S. N., Sivaraman, K., and Luscombe, N. M. (2011). An overview of prokaryotic transcription factors. *A handbook of transcription factors*, pages 7–23.

[Shah, 2014] Shah, M. (2014). Dendrites. In Aminoff, M. J. and Daroff, R. B., editors, *Encyclopedia of the Neurological Sciences (Second Edition)*, page 970. Academic Press, Oxford, second edition edition.

[Shahbazian and Grunstein, 2007] Shahbazian, M. D. and Grunstein, M. (2007). Functions of site-specific histone acetylation and deacetylation. *Annual review of biochemistry*, 76.

[Sharifi et al., 1990] Sharifi, M., Georgakakos, K., and Rodriguez-Iturbe, I. (1990). Evidence of deterministic chaos in the pulse of storm rainfall. *Journal of the Atmospheric Sciences*, 47(7):888–893.

[Shrikumar et al., 2017] Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

[Singer et al., 1991] Singer, J., Wang, Y., and Bau, H. H. (1991). Controlling a chaotic system. *Physical Review Letters*, 66(9):1123.

[Sprott, 1994] Sprott, J. C. (1994). Some simple chaotic flows. *Physical review E*, 50(2):R647.

[Sprott et al., 2013] Sprott, J. C., Wang, X., and Chen, G. (2013). Coexistence of point, periodic and strange attractors. *International Journal of Bifurcation and Chaos*, 23(05):1350093.

[Stillings et al., 1995] Stillings, N. A., Chase, C. H., Feinstein, M. H., and Garfield, J. L. (1995). *Cognitive science: An introduction*. MIT press.

[Strahl and Allis, 2000] Strahl, B. D. and Allis, C. D. (2000). The language of covalent histone modifications. *Nature*, 403(6765):41–45.

[Struhl, 1998] Struhl, K. (1998). Histone acetylation and transcriptional regulatory mechanisms. *Genes & development*, 12(5):599–606.

[Südhof, 2018] Südhof, T. C. (2018). Towards an understanding of synapse formation. *Neuron*, 100(2):276–293.

[Suresh and Guttag, 2019] Suresh, H. and Guttag, J. V. (2019). A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002*, 2.

[Suzuki and Bird, 2008] Suzuki, M. M. and Bird, A. (2008). Dna methylation landscapes: provocative insights from epigenomics. *Nature Reviews Genetics*, 9(6):465.

[Svozil et al., 1997] Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

[Swaminathan et al., 2005] Swaminathan, J., Baxter, E. M., and Corces, V. G. (2005). The role of histone h2av variant replacement and histone h4 acetylation in the establishment of drosophila heterochromatin. *Genes & Development*, 19(1):65–76.

[Taylor, 2004] Taylor, T. (2004). A genetic regulatory network-inspired real-time controller for a group of underwater robots. In *Intelligent Autonomous Systems*, volume 8, pages 403–412.

[Thagard, 2005] Thagard, P. (2005). *Mind: Introduction to cognitive science*, volume 17. MIT press Cambridge, MA.

[Trefzer et al., 2010] Trefzer, M. A., Kuyucu, T., Miller, J. F., and Tyrrell, A. M. (2010). Evolution and analysis of a robot controller based on a gene regulatory network. In *International Conference on Evolvable Systems*, pages 61–72. Springer.

[Tuladhar et al., 2015] Tuladhar, A., Mitrousis, N., Führmann, T., and Shoichet, M. S. (2015). Chapter 30 - central nervous system. In Atala, A. and Allickson, J. G., editors, *Translational Regenerative Medicine*, pages 415 – 435. Academic Press, Boston.

[Turner and Dethlefs, 2017] Turner, A. and Dethlefs, N. (2017). Transparency of execution using epigenetic networks. In *ECAL 2017, the Fourteenth European Conference on Artificial Life*, pages 404–411. MIT Press.

[Turner et al., 2017] Turner, A. P., Caves, L. S. D., Stepney, S., Tyrrell, A. M., and Lones, M. A. (2017). Artificial epigenetic networks: Automatic decomposition of dynamical control tasks using topological self-modification. *IEEE Transactions on Neural Networks and Learning Systems*, 28(1):218–230.

[Turner et al., 2018] Turner, A. P., Lacey, G., Schoene, A., and Dethlefs, N. (2018). Evolutionary constraint in artificial gene regulatory networks. In *UK Workshop on Computational Intelligence*, pages 29–40. Springer.

[Turner et al., 2012] Turner, A. P., Lones, M. A., Fuente, L. A., Stepney, S., Caves, L. S., and Tyrrell, A. M. (2012). Using artificial epigenetic regulatory networks to control complex tasks within chaotic systems. In Lones, M. A., Smith, S. L., Teichmann, S., Naef, F., Walker, J. A., and Trefzer, M. A., editors, *Information Processing in Cells and Tissues*, pages 1–11, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Turner et al., 2013a] Turner, A. P., Lones, M. A., Fuente, L. A., Stepney, S., Caves, L. S., and Tyrrell, A. M. (2013a). The incorporation of epigenetics in artificial gene regulatory networks. *Biosystems*, 112(2):56 – 62. Selected papers from the 9th International Conference on Information Processing in Cells and Tissues.

[Turner et al., 2013b] Turner, A. P., Lones, M. A., Fuente, L. A., Stepney, S., Caves, L. S. D., and Tyrrell, A. (2013b). The artificial epigenetic network. In *2013 IEEE International Conference on Evolvable Systems (ICES)*, pages 66–72.

[Vaissière et al., 2008] Vaissière, T., Sawan, C., and Herceg, Z. (2008). Epigenetic interplay between histone modifications and dna methylation in gene silencing. *Mutation Research/Reviews in Mutation Research*, 659(1-2):40–48.

[van Riel, 2006] van Riel, N. A. (2006). Dynamic modelling and analysis of biochemical networks: mechanism-based models and model-based experiments. *Briefings in bioinformatics*, 7(4):364–374.

[Vignali et al., 2000] Vignali, M., Hassan, A. H., Neely, K. E., and Workman, J. L. (2000). Atp-dependent chromatin-remodeling complexes. *Molecular and cellular biology*, 20(6):1899–1910.

[Vijesh et al., 2013] Vijesh, N., Chakrabarti, S. K., and Sreekumar, J. (2013). Modeling of gene regulatory networks: a review. *Journal of Biomedical Science and Engineering*, 6(02):223.

[Waddington et al., 1939] Waddington, C. H. et al. (1939). An introduction to modern genetics. *An introduction to modern genetics.*

[Wang et al., 2022] Wang, Q., Ma, Y., Zhao, K., and Tian, Y. (2022). A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212.

[Watson and Crick, 1953] Watson, J. D. and Crick, F. H. C. (1953). Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738.

[Watt et al., 2020] Watt, J., Borhani, R., and Katsaggelos, A. K. (2020). *Machine learning refined: Foundations, algorithms, and applications*. Cambridge University Press.

[White et al., 2005] White, P., Zykov, V., Bongard, J. C., and Lipson, H. (2005). Three dimensional stochastic reconfiguration of modular robots. In *Robotics: Science and Systems*, pages 161–168. Cambridge.

[Wollenberg Valero, 2020] Wollenberg Valero, K. C. (2020). Aligning functional network constraint to evolutionary outcomes. *BMC Evolutionary Biology*, 20:1–14.

[Ye et al., 2016] Ye, X., Li, J., Qi, Z., and He, X. (2016). Enhancing retrieval and ranking performance for media search engine by deep learning. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 1174–1180. IEEE.

[Zhang and Reinberg, 2001] Zhang, Y. and Reinberg, D. (2001). Transcription regulation by histone methylation: interplay between different covalent modifications of the core histone tails. *Genes & development*, 15(18):2343–2360.

[Zhang and Wallace, 2015] Zhang, Y. and Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

[Zhao and Alkon, 2001] Zhao, W.-Q. and Alkon, D. L. (2001). Role of insulin and insulin receptor in learning and memory. *Molecular and Cellular Endocrinology*, 177(1):125–134.