THE EVOLUTION OF NEURAL PLASTICITY IN DIGITAL ORGANISMS

By

Leigh Sheneman

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science - Doctor Of Philosophy

2017

ABSTRACT

THE EVOLUTION OF NEURAL PLASTICITY IN DIGITAL ORGANISMS

By

Leigh Sheneman

Learning is a phenomenon that organisms throughout nature demonstrate and that machine learning aims to replicate. In nature, it is neural plasticity that allows an organism to integrate the outcomes of their past experiences into their selection of future actions. While neurobiology has identified some of the mechanisms used in this integration, how the process works is still a relatively unclear and highly researched topic in the cognitive science field. Meanwhile in the field of machine learning, researchers aim to create algorithms that are also able to learn from past experiences; this endeavor is complicated by the lack of understanding how this process takes place within natural organisms.

In this dissertation, I extend the Markov Brain framework [1, 2] which consists of evolvable networks of probabilistic and deterministic logic gates to include a novel gate type– feedback gates. Feedback gates use internally generated feedback to learn how to navigate a complex task by learning in the same manner a natural organism would. The evolutionary path the Markov Brains take to develop this ability provides insight into the evolution of learning. I show that the feedback gates allow Markov Brains to evolve the ability to learn how to navigate environments by relying solely on their experiences. In fact, the probabilistic logic tables of these gates adapt to the point where the an input almost always results in a single output, to the point of almost being deterministic. Further, I show that the mechanism the gates use to adapt their probability table is robust enough to allow the agents to successfully complete the task in novel environments. This ability to generalize to the environment means that the Markov Brains with feedback gates that emerge from evolution are learning autonomously; that is without external feedback. In the context of machine learning, this allows algorithms to be trained based solely on how they interact with the environment. Once a Markov Brain can generalize, it is able adapt to changing sets of

stimuli, i.e. reversal learn. Machines that are able to reversal learn are no longer limited to solving a single task. Lastly, I show that the neuro-correlate Φ is increased through neural plasticity using Markov Brains augmented with feedback gates. The measurement of Φ is based on Information Integration Theory[3, 4] and quantifies the agent's ability to integrate information.

For Gretchen, my parents and grandparents. Thank you for supporting me when I needed an ear and pushing me when I needed to have more faith in myself.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Many natural organisms rely on phenotypic plasticity, or the ability to adapt to changes in their environment, to survive and reproduce. Phenotypic plasticity can manifest in many forms from adjusting to environmental temperature changes to adapting to diet modifications [5, 6, 7]. Biologists observe organisms that display adaptation by using past experiences to shape future actions. This form of plasticity is often termed *neural plasticity* or *learning* and is an active area of research [8]. In fact, there are many competing schools of thought about what defines learning and how the brain executes the process [9, 10, 11, 8]. In this work, I aim to use the biological perspective to devise a computational model of neural plasticity capable of advancing knowledge in the fields of biology and artificial intelligence (AI). This model builds on the Markov Brains framework [1, 2] by adding a novel gate type– feedback gates. Here I will show that:

- Feedback gates effectively model neural plasticity in Markov Brains.
- Feedback gates allow the model to *generalize*.
- The neural correlate Φ through learning.

1.1 Natural Evolution and Adaptation

To model a system that exhibits processes similar to neural plasticity we must first understand some of the basic biological concepts involved. In *On the Origin of Species*, Charles Darwin presented his theory of evolution based on observations he made as he traveled the world from 1831-1836 [12]. He reasons that if a species were allowed to reproduce at its full potential, then its population size would exponentially increase, yet he observed that all the species he encountered remained relatively stable in size. He also noticed that natural resources are only produced in limited supply, yet the amount that is available in an environment remains relatively constant. Both these observations led Darwin to conclude that more individuals are born than there are resources to support them, so organisms must compete for the existing resources in order to survive. This struggle for survival allows both the population size and resource availability to remain stable. Further, when looking at a species' population, he also noted that no two individuals were identical and that the specific variances were passed down to a individual's offspring (i.e., inheritance). These variations can aid in an organisms ability to obtain resources, a process he termed natural selection. Natural selection allows beneficial mutations to be passed down from one generation to the next and eventually sweep the population; this is called evolution[1].

While evolution happens to the individuals within a population, the environment that a species inhabits can also change, and such change could even be facilitated by other organisms as well. Changes to a local environment can occur both between generations or over an individual's lifetime [14]. Additionally, an organism's response to the change can occur over an extended period (through evolution) or in a short time-span (phenotypic plasticity) and is often the result of a combination of environmental cues [15].

Phenotypic plasticity occurs within the lifetime of an organism; it allows the organism to adapt its characteristics to survive when faced with environmental changes. Some common environmental changes that trigger phenotypic plasticity are deviations in population density

[1]Complied from *Evolution: Making sense of life* [13]

or average temperature. Population density can also play a role in phenotypic plasticity. For example, female sexton beetles (*Nicrophorus orbicollis*) produce smaller offspring in low-density populations than in high-density populations [7]. Another environmental change that can trigger an adaption is when the temperature of the habitat increases or decreases. For example, the *Setophaga discolor* has an subadult plumage in the winter, an extreme body molt in the spring, and displays a adult plumage in the summer [5]. In the above examples phenotypic plasticity results in physical changes to the organisms that aid in their survival.

Not all forms of phenotypic plasticity manifest physically; neural plasticity can lead to adaptive behavioral changes. Neural plasticity is when the organism's brain can change its connections based on experience. For example, rats have evolved to use water as a food resource. However, if they drink water and then are exposed to radiation they will learn to avoid drinking water as it is associated with a poison. This is known as conditioned taste aversion [16], which uses past experiences to change future behaviors. As another example, when learning to navigate a maze, rats use place cells and grid cells to store a layout of their environment in their neural structure. Once this layout exists, they are able to identify their location in the environment [17, 18]. In both cases, the visual physical phenotype was unaffected, only the behavior changed. Additionally, the behavioral change was facilitated by storing information in the neural substrate (memory) of the organism while leaving its morphology unaffected.

Among Charles Darwin's observations in *On the Origin of Species* was that behavior was a heritable trait and therefore was subject to evolution [12]. We know that behavior is the product of updating neural mechanisms in the nervous system [19] which indicates that neural mechanisms themselves are inheritable traits. Inheritable traits are any characteristic that a parent passes down to its offspring through genes. The set of genes that the offspring receives is contained in its genome. The genome is subjected to mutations which can effect:

- the content of the genes.

- Gene regulation sites which are sites on the genome that control whether a gene is expressed or not.

- Epigenetic changes which turn gene expression on or off during an organism's lifetime without changing the underlying genome [20].

How these genetic changes translate to behavioral changes is a very complex process, and depends entirely on the organism and trait. Generally speaking, this could be due to genes that affect development and thus the structure of the brain, components in the brain that modify neural activity, or more complex interactions.

Both processes of learning and evolution are tightly coupled and can effect each other– known as the Baldwin effect [21] which, since its conception, has been integrated into the modern synthesis of evolution [22]. The Baldwin effect states that an organism has an increased chance to reproduce when its ability to survive in its environment is increased. With the increased ability to survive comes the increased chance to reproduce; this ability may be the result of an organism having a higher rate of learning a particular advantageous behavior than its peers. Upon reproduction the genes that aid in an organism's increased ability to learn the new behavior are then passed down to the next generation. After many generations the behavior becomes an instinct, or the inherent inclination to perform a complex behavior.

1.2 Lifetime Adaptation Using Natural Neural Processes

Often times when we discuss the behavior of learning we pair it with the term memory. To understand why, let us examine how biologists study an animal's ability to learn in a controlled setting such as a Radial-Arm Maze (RAM), as seen in Figure 1.1. The RAM has a central corridor with arms extending out. Each of the corridors is raised so that an animal traversing the maze can see multiple arms at once. The arms of the maze can contain food rewards that the animal must find to complete the task. In addition to being able to see

multiple arms at once, the RAM often contains landmarks for the animal to use as clues to its location; these allow it to efficiently navigate the maze. This task is designed to test the spatial learning and memory abilities of the animal, usually a rat [23].

First, the animal is allowed to find all sources of food through exploration. By finding the food rewards, the animal develops a sense of what the given task is. The exploration also gives the animal a chance to develop an internal representation of the maze layout and location of landmarks. The animal stores this representation in long-term-memory (LTM). It is then removed from the maze for a period of time. During this period of time the food resources are replenished. Next, the animal is placed back in the maze and presented with the same task: find all the food. Ideally all the food would be found using the most efficient route. To accomplish this the animal navigates the maze and processes the cues within its field of vision. When these visual cues are combined with the maze layout stored in LTM, the animal should able determine where in the maze it is. As it begins to visit each arm, the animal can remember where it has visited since placement at the beginning of the trial; this information is stored in a type of short-term memory (STM) often referred to as working memory (WM). The working memory that the animal uses in the RAM will only last while the animal is actively thinking about the task and does not change the underlying neural substrate [24, 25].

The RAM gives biologists a platform to observe how memories are formed through the observation of learned behavior. This indirect observation is necessary because observing changes to the neural substrate of the animal is extremely difficult. This observation process makes memory and learning very nested terms that are at times indistinguishable. For simplicity, we can think of learning as the process of acquiring new knowledge while a memory is how the knowledge is expressed.

When neurobiologists seek to explain learning, they rely on the study of the molecular, cellular, and circuit mechanisms a brain uses when forming memories. We know that the functions that result in memory storage are interactive and are not the outcome of a se-

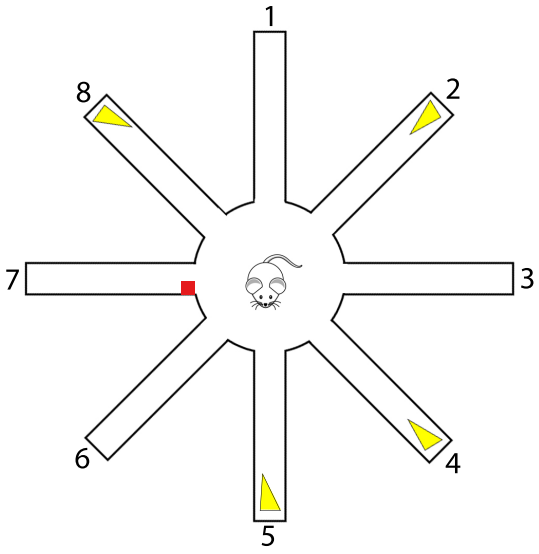

Figure 1.1: **Example radial-arm maze.** A radial-arm maze has identical arms that extend from a central chamber. The yellow food sources are placed at the end of each arm. The animal, represented by the mouse, is given the task of gathering all the food. The red block represents a visual cue the animal can use to aid in the navigation of the maze.

quential step by step process [8]. While researchers have identified some of these interactive processes, they have yet to determine the exact contribution each makes in forming and maintaining a memory [24].

A memory can be classified as either a STM or LTM based on the how the knowledge is stored. A STM lasts between 4-8 hours [24] and is formed by the continuous activation of neurons [26] in the hippocampus. However, once a STM is formed, the activation of neurons can stop and the information will still be retained [24, 8]. Reinforcement triggers the STM to be converted to LTM that is stored in the neocortex using a gene-expression-dependent transformation, also known as consolidation [27, 24, 8].

There are several theories of how the underlying neural networks process memories during consolidation. Among these are standard consolidation theory and multiple trace theory [9, 10, 11, 8]. The standard consolidation theory states that the transfer of memory from the hippocampus to the neocortex takes many years and after completion memories are only processed in the neocortex [28]. Multiple trace theory distinguishes between how episodic–

event based– and semantic– concept based–memories are stored. The theory proposes that while all LTMs are stored and retrieved from the neocortex, episodic memories are recalled due to triggers from the hippocampus. Sleep has also been shown to contribute to consolidation, though its exact role is unknown [27, 29, 24, 9, 8]. Adding information to an existing long-term memory, reconsolidation, is another area of interest in neurobiology. Suggested explanations for incorporating the new knowledge include the reconsolidation protocol, the semantic transformation model, and the schema modification model [24, 9].

We can look at one generation of a species and observe these basic neural mechanisms and theorize how they work together to achieve learning in an organism. However, as Theodosius Dobzhansky stated in his 1973 essay *Nothing in Biology Makes Sense Except in the Light of Evolution*, to fully understand biology we must integrate evolution into our studies [30]. Like all biological processes, evolution contributes to the ability to learn and to fully understand the ability to learn, we must examine many generations. However, given that lifespans of organisms vary, studying any generational changes to a species can be extremely difficult, especially when examining how learning evolves [31]. **These limitations can be circumvented by creating computational models that have clearly defined abstractions which map well to their natural counterparts and also incorporate evolution.**

Computational modeling is possible because the evolutionary process is substrate neutral. This means that evolution is a general process not reliant on the materials that are evolving [32]. Thus, evolution is not limited to natural organisms alone. By modeling the key components of evolution– inheritance, variation and natural selection– in a computation model we can form an experimental platform that contains evolution as opposed to simply simulating it [33]. In fact, we see huge advances coming from artificial life and digital evolution to understand Darwinian evolution [34].

1.3 Algorithmic Attempts to Learn

Before we examine computational models that incorporate the evolution of learning, it is important to understand the traditional computational methods that attempt to learn. Machine learning is a field of computer science that focuses on developing algorithms that learn. These learning algorithms are largely inspired by the learning seen in nature, but only have one goal: classification of data.

There have been many "learning algorithms" presented over the years and each have strengths and weakness. For example, Q-learning [35] uses an unsupervised learning environment to maximize delayed rewards in a neural network. Each possible action leads to a reward that affects whether that action is taken again. However, the values of the algorithm must converge on what the optimal action-to-reward policy is to prove useful, and this requires a very large, computationally expensive network [36, 37]. Another type of learning algorithm is back-propagation where the network typology is fixed. The algorithm is then trained using an initial set of data where correct or incorrect answers result in the weights of the network connections being updated [38]. The training stops when a high percentage of correct classifications is achieved, but this is not a trivial task [39, 40, 41]. If you are given the inputs and outputs of a network– a rarity in nature– the Baum-Welch algorithm allows you to uncover hidden internal functions, thus it is useful for pattern recognition problems [42, 43]. Multiplicative weights algorithm strengthens or weakens connections in a neural network– based on the consensus of a pool of experts– to reach a binary decision [44, 45]. All the experts start with an equal weight and the amount of influence it is given on the next decision is based on how many correct predictions the expert has made in the past. However, many times learning must occur when an organism is isolated from receiving advice from experts. To overcome the shortcomings of an individual method, Google recently combined a deep convolution neural network, which models the organization of the visual cortex to solve visual tasks, and a search tree to play the game of GO [46].

Artificial neural networks (ANNs) [27, 47, 48] are interconnected layers of nodes which

process inputs through hidden layers to determine the classification output. Each predefined function node is akin to the neurons of natural organisms and the connections between nodes represent synapses which are weighted. The systems are trained using a set of sample data where the correct outputs are known. The weights of the connections can be strengthened or weakened based on the final output being wrong. ANNs have been used to aid in facial recognition [49, 50] to predict the stock market [51] and to detect cancer [52, 53, 54].

While each of these methods of learning adds insight to the overall process, none of them are capable of completely modeling the phenomena of lifetime learning in natural organisms. As illustrated above, the plethora of computational approaches, which are all very specific, do not investigate the complexity of WM, biological learning, and the evolution thereof at the same time. These approaches provide external feedback of the "right" and "wrong" responses to stimuli, while natural organisms must first evolve internal mechanisms that detect feedback and then integrate them into their cognitive behavior. For example, the human eye had to first evolve so we can obtain visual feedback to our actions. We then had to evolve the mechanisms to react to this feedback.

1.4 Evolutionary Computing Harnessing the Power of Evolution

Now that we are equipped with a basic understanding of how natural organisms learn and how machine learning attempts to model learning over evolutionary time, we now turn our focus to modeling the ability to gain knowledge within a lifetime of an AI system. Since neural plasticity cannot be directly studied, we are forced to use observable behavior to examine how learning and memory work. The observation of behavior can be easily performed using computational models.

Evolutionary computing (EC) mimics the evolution found in biological systems using computational resources, a process Ingo Rechenberg coined as evolutionary strategies [55]. By using evolutionary strategies, we can randomize a population of digital organisms, rep-

resented as binary genomes, then apply the inheritance, variation and natural selection to produced increasingly better performing agents. There are many subfields of EC that are used to solve different classes of problems. In a genetic algorithm (GA), which was introduced by John Holland in 1975 [56], the genomes represent possible solutions to a given problem, and evolution is used to produce increasingly optimal solutions. National Aeronautics and Space Administration (NASA) has used this to solve a number of complex problems: GAs led to evolved antennas to handle unusual radiation patterns [57] as well as handle scheduling for NASA Deep Space Network's satellites [58]. Another sub-field, genetic programming (GP), aims to create or optimize computer programs by providing the algorithm high-level criteria of what the program needs to accomplish [59]. Recent research has used GP to increase network security by adapting to attacks [60], build new mechanisms into existing software [61], and to evolve virtual chess agents who win against champion chess players [62].

Building upon the ANNs mention in Section 1.3, NeuroEvolution of Augmenting Topologies (NEAT) [63] add the ability to evolve the connections between the nodes of an ANN and their weights. The evolution of these connections are based on the same principles found in nature– inheritance, variation, and natural selection. Researchers have used NEAT to automatically select the best features for an algorithm to use in machine learning [64], to evolve neural networks that can play video games [65, 66, 67], and to search for novel solutions to problems to avoid local maxima of the fitness function [68].

While EC is an engineering endeavor inspired by biology it does not aim to model natural organisms. For example, it has been used to optimize laser fields [69], to optimize welded beam design [70], or to produce levels in video games [71] – all important problems which have nothing to do with the way natural organisms learn. However, here my aim is to understand the natural evolution of intelligence, more specifically learning, thus the computational model I use needs to closely model biology. To achieve this we need to mimic natural feedback which occurs **internally**, not as a function of external verification. At the same time, we don't want nor need to model every aspect of biology. A computational model

such as molecular dynamics might be more accurate in respect to nature, but it would also be too complex to understand the essential components. It is therefore necessary to create a model that contains the right level of abstraction. Here, I focus on the principles of internally generated feedback, and the idea that the neural substrate has to change over both the course of evolution and during the lifetime of the organism. Further, the mechanisms that change the neural substrate during the lifetime of the organism are also controlled again by evolution.

Now that we are equipped with a basic understanding of the theory of natural evolution, how learning processes work and the history of EC, we can explore the evolution of learning. To facilitate this I will use Markov Brains. In the following sections I will describe the components of my computational model.

1.5 Experimental System

1.5.1 Markov Brains

To investigate the evolution of learning I will use Markov Brains (MBs) [1, 72, 73, 74, 75, 76, 77, 78, 79], a specific form of artificial neural networks, to explore neural mechanisms while optimizing outcomes using EC. MBs are computational logic units (gates) that are genetically coded and subject to evolution. Often when we speak of MBs, we refer to agents or animats, both of which are simply the embodiment [80] of MBs that allows their sensors and actuators to control an agent.

Each MB is represented by a sequence of numbers that serves as the genome that can be subdivided into number sequences that encode computation units. This process is similar to how genes encode proteins. These genes determine the number, type and starting values of all the gates, as well as how they connected to each other, sensors and actuators. Additionally, the MBS contain hidden states that are only used internally, i.e. they cannot be directly accessed by elements of the agent's environment.

At the core of every MB is a state buffer which serves as a storage unit for all input, hidden, and output states (nodes). Typically, the nodes in the buffer are referred to using

indexes that start with 0. The genome encodes how logic gates read and write to this buffer. The starting values for the buffer are initiated to 0 at the birth of the organism. However, the environment sets the values for the inputs to be written into the same buffer; we can think of this as time step t. These new values are then processed as inputs to all the Brain's gates which then update the state buffer with their outputs. We refer to the new set of values as the state buffer at time step $t + 1$.

To fully understand the transformation that takes place between t and $t + 1$, we must explore how each gate processes information. Every gate has a set number of inputs, i, and outputs, o, and each input has a chance to select any output. It is important to note that, at least theoretically, any input should be able to result in any output. However, how each gate does this mapping between inputs and outputs can evolve. To control when a specific input results in a given output the gates rely on what can be thought of as a probability table whose size is $2^i x 2^o$. Each cell within the table represents the percentage of time a given input will call a certain output.

MBs contain either deterministic gates, probabilistic gates, or a combination of the two. In deterministic gates every input correlates to a specific output, while the inputs of probabilistic gates use a stochastic model to determine output. Typically, MBs contain a series of gates which are encoded numbers that are akin to the nucleotides of a natural organism. To decode the genome into the series of gates a process similar gene transcription in DNA is used. Specifically, each gate type has a unique start codon (two nucleotides) that when encountered during MB genome decoding signals that the next section of the genome contains information to initiate the gate. More specifically, there are four nucleotides, or individual numbers in the genome, directly following the start codon: the number of inputs, the number of outputs, the states used for input, and the states used for output. However, each of these numbers are not directly placed in the genome, but instead undergo encoding processes.

To encode the nucleotide that specify the number of inputs and outputs Equation 1.1,

where N is the number of input or outputs, l is the value of the nucleotide and N_{max} is the maximum number of inputs or outputs. Current implementations use a $N_{max} = 4$. When $N < 4$ the genome contains non-coding nucleotide for all placements between N and N_{max}. The nucleotide for each individual input or output indicates which node in the state buffer it connects to. Equation 1.2 is used to determine each nucleotide x.

$$N = \left\lfloor \frac{l}{\frac{255}{N_{max}}} \right\rfloor \tag{1.1}$$

$$x = \left\lfloor \frac{l * N}{255} - 0.5 \right\rceil \tag{1.2}$$

As discussed in Section 1.1, evolution occurs to individuals in a population over many generations. Therefore, a single MB cannot evolve in isolation but rather must evolve with other agents. When creating an evolutionary model of MBs, we first randomly generate the genomes for all organisms in the population. Generally all genomes have the same starting length and are seeded with the same number of start codons. These agents then perform a task and receive a fitness value based on their performance level. The agents are then ranked by fitness and a selection process is implemented. Generally this results in the better performing agents being given more offspring than those who perform poorly.

Each MB is decoded at the beginning of every generation since the genomes are subjected to evolution and thus can vary each time they reproduce. There are three mutation types allowed during the inheritance process: point mutations, insertion, or deletions. Point mutations occur when the value of an individual nucleotide is changed and typically have a 0.3% chance of occurrence at each sites.

In the MB framework, there are upper and lower limits to the length of the genomes to prevent them from becoming extremely large or small. Genomes only have a chance to experience insertion or deletion once per generation. Therefore, the chance of occurrence of these events are higher than the chance of a point mutation occurring. Genomes with less

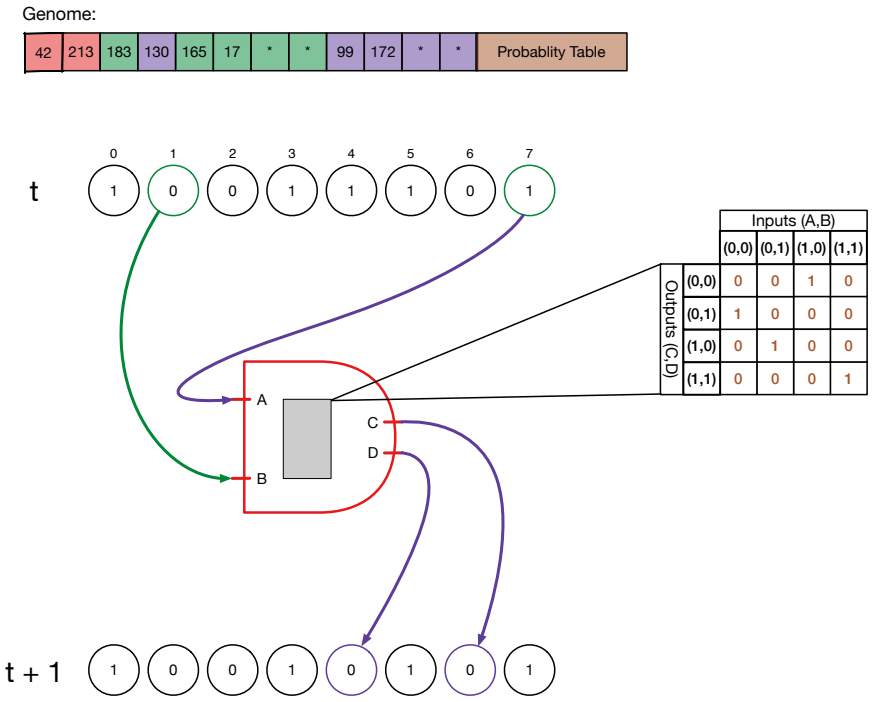

Figure 1.2: **Genetic encoding of a deterministic gate.** The genome is a sequence of nucleotides that encode the structure of all gates and their connections to the nodes of the state buffer. Each node in the state buffer is numbered sequentially with a starting index of 0. In the sample genome the numbers in the red boxes represent the starting codons of a deterministic gate; 43 followed by 213. All green boxes are responsible for encoding the number of inputs and the nodes they connect to. Purple boxes encode the number of outputs and the nodes they are connected to. While the remaining nucleotides determine the values in the gate's probability table. The gate diagram is a visualization of the gate the genome encodes.

than $20,000$ nucleotides have a 2% chance to experience an insertion event. The insertion process mimics a transposition event in genetics in that it takes a section of between 256 and 512 nucleotides which are copied and then inserted into a random location. Genomes that have more than $1,000$ nucleotides have a 2% chance of deletion of a random section of between 256 and 512 nucleotides.

The number of generations that the evolutionary process takes to converge on a solution varies based on both the type of gates that the MBs are allowed to used and the complexity of the task. Based on prior experiments, we know that deterministic gates usually converge in fewer generations than when probabilistic gates are used.

None of the MB implementations to date have supported lifetime adaptation that change the MB structure because it is set at birth, yet they have been shown extremely useful at solving tasks [1, 72, 73, 74, 75, 76, 77, 78, 79]. In this work, I will present a novel gate type that enables me to implement and investigate neural plasticity in digital organisms.

1.5.2 Feedback Gates

The key feature of neural plasticity in natural organisms is that the organism is able to update their neural structure based on their past experiences. While there have been other evolvable neural networks that model neural plasticity by updating the topology and/or weights of the networks during their lifetime [81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91], they fail to consider how natural organisms receive feedback. In nature, the process of learning begins with a sensorial percept that must than be interpreted to produce positive or negative feedback. For example, a small child may hear the phrase "good job" after putting away their toys, but until they are able to internalize the praise they must learn that this is positive feedback. Here I present a novel MB gate type – *feedback gates*– that are capable of updating their probability gates within generations, i.e. their lifetime, and thus model neural plasticity.

Feedback gates are very similar to probabilistic gates in that their internal probability

table defines the likelihood a given input results in a specific output at any given update. More specifically, we use a probability matrix P that has a size of $2^i x 2^o$. Each cell of the matrix is represented by P_{io} which contains the probability that input i results in output o. By definition of a probability, we know that the sum of all o for a i must be 1.0. This is shown in Equation 1.3 where O is the maximum number of possible outputs of each gate.

$$1.0 = \sum_{o=0}^{O} P_{io} \tag{1.3}$$

The gates output at any given update is the result of probability P_{io}, thus if that input-output mapping proves beneficial then the probability it used again should increase. By the same token, if the action was detrimental then the probability that it occurs again should decrease. However, we know that the sum over all probabilities for each input must sum to 1.0. This means that when P_{io} is adjusted, then all other probabilities must also adjust. To ensure that 1.3 remains true, when any probability changes in row P_i the entire row is then normalized.

Feedback gates add two input channels, one for positive feedback and one for negative feedback. These feedback channels are present in all feedback gates they are not functional until they are connected to nodes in the MB's t state buffer. These connections are a product of evolution.

Since the connections link any node in the t state buffer, the source of the feedback can be a sensor or a hidden node. Regardless of the type of node the channels read from, 0 the gate acts like a standard probability gate when it is set to . However, when set to 1, feedback is used to trigger an update to the gate's probability table using the process described above.

In the most basic conditions, when feedback is triggered the P_{io} of the last action taken is changed by a random number in the range $[0, \delta]$. If the input adjustment comes from the positive input channel the value is increased; when negative feedback is called for the value in decreased. The value of P_{io} has a limited range of 0.01 to 0.99.

Because MBs are networks of gates it is possible that the signal a feedback gate generates

connects into other gates. Therefore it might take several updates for the effects of the feedback gates signal to effect the $t+1$ state buffer (See Figure 1.3). By the same token, there may also be a delay in how the environment reacts to the agent's actions, which would in turn cause a delay in when new sensorial inputs are received. Thus, if P_{io} is updated based solely on the action taken on the last update the feedback provided to the gate would be insufficient.

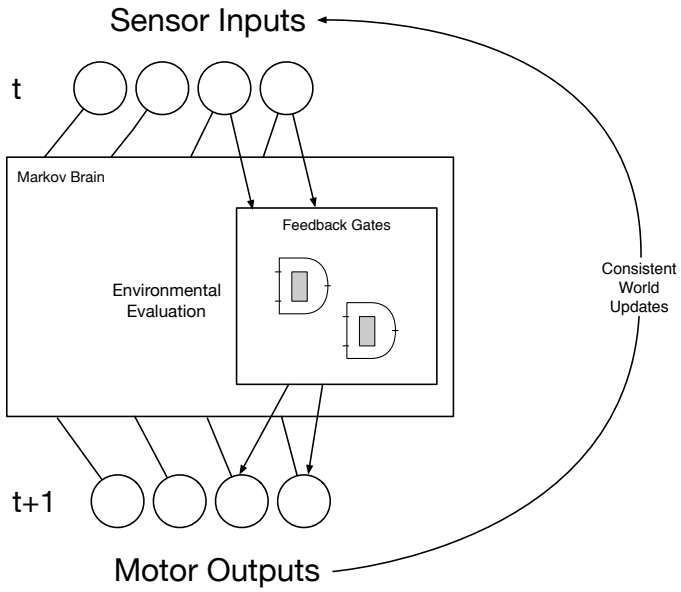

Figure 1.3: **How feedback gates interact with their environments.** Feedback gates receive input through the t state buffer and control the actions of the Markov Brain through the $t+1$ state buffer. The feedback gates are part of a bigger network of gates that update the $t+1$ state buffer. The world only provides information about itself through changes to the t state buffer and the feedback gate must interpret those signals internally. This figure is adapted from Figure 2.1 .

Since the action that was disadvantageous or advantageous could have occurred many updates back, the gate needs a way ensure the responsible action is adjusted. Therefore, a memory queue was added to the gates that stores a list of the probabilities responsible for the past D actions. As shown in Figure 1.4, both the positive and negative feedback channels have queues encoded in their genome. Much like the number of inputs and outputs, these channels have a $D_{max} = 4$ of queues. Each D in the queue represents the magnitude of the change to the probability table using Equation 1.4. The elements of the queue are processed

sequentially from oldest to youngest, and after each step of the sequence the probability matrix becomes normalized again (See Figure 1.5 for an example). Because the forces are encoded in the genome they are subjected to evolution.

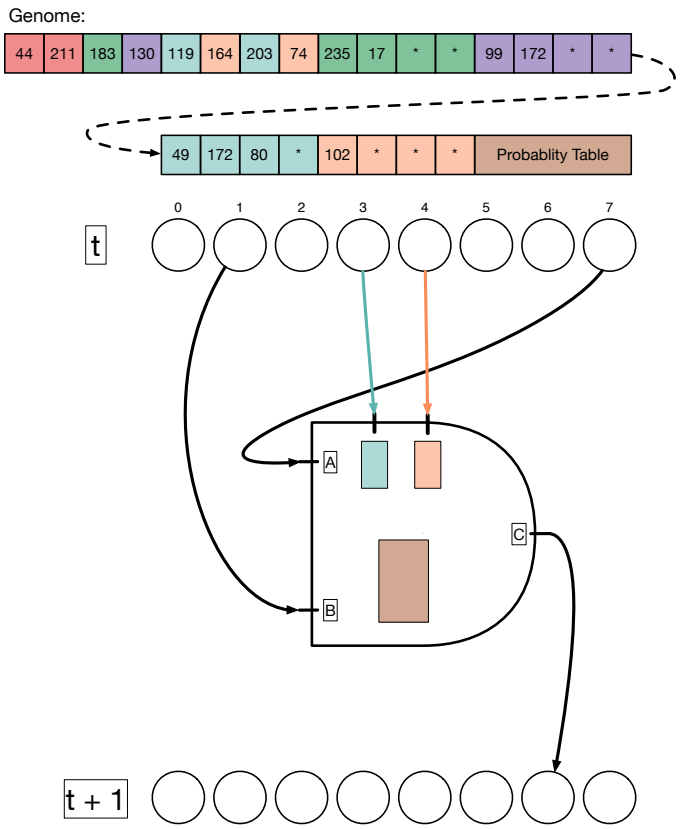

Figure 1.4: **Genetic encoding of a feedback gate.** The start codons (44 and 211) are shown in red, followed by the number of inputs in green and number of outputs in purple. The next two numbers are encode the number of the node the positive (teal) and negative (peach) feedback channels. After that the length of the each channels queues are given. Each queue has a maximum length of 4, thus all four spots are always present in the genome. The encoding of the probability table in brown completes the gate encoding.

$$F = \frac{l+1}{256} \tag{1.4}$$

Before the advent of feedback gate, MBs relied on hidden nodes of the state buffer to store a "memory". In order to maintain this information through multiple time points the node containing the memory cannot change. When we compare this to natural organisms this "memory" is akin to WM. On the other hand, feedback gates add the ability to store

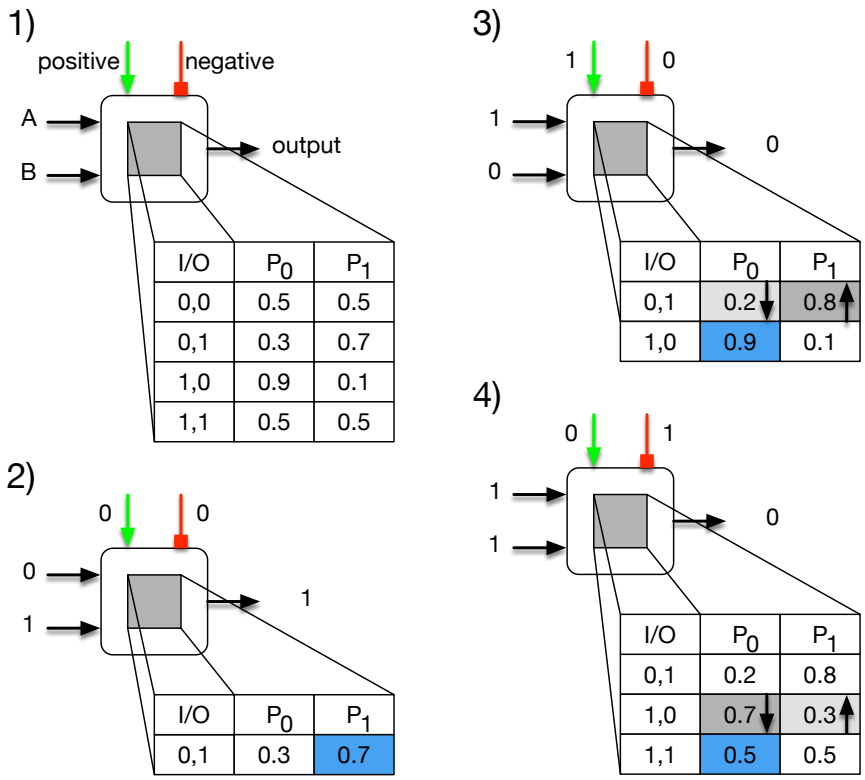

Figure 1.5: **Update process of feedback gates.** The positive (green) and negative (red) channels detect when feedback is given. In this specific example, I assume that at the first update the gate received an input of 0 for A and 1 for B. The probability table then returns an output of 1 (See Panel 2). In the probability table I marked the value that mapped the input 0,1 to the output 1 in blue. Technically, all probabilities in the row for that input are responsible, but here we call only the probability in blue the responsible one since it was used. It should be higher if that mapping was advantageous, and lower if the decision of the gate was had negative consequences. In Panel 3 the gate now receives a positive feedback signal, and no negative signal otherwise. Consequently, the formerly marked blue probability needs to be increased indicated by the upward facing arrow in the probability table (0.7 becomes 0.8). Because each row in the table is summing to 1.0 in order to ensure that all entries remain probabilities, I need to re-normalize the row, and thus the other values need to be decreased (light gray probability with an arrow facing down. After feedback is computed, new inputs need to be processed. Here the gate receives a 1 for input A and a 0 for input B. This leads to a new output of 0, which identifies a new probability in the probability table, highlighted in blue (See Panel 3 blue highlight). At the next update (See Panel 4), the gate now receives a new negative feedback, but no positive one. This means that the probability remembered (blue) was too high and needs to be decreased. Similarly, due to normalization, the other values in that row need to be increased, indicated by the upwards facing arrow in the light gray box. Again, after feedback was applied, the gate still needs to map the newly received input (1,1) to an output, and remembers the responsible probability (blue). This demonstration is also shown in Chapter 2.

memory "physically" within their logic table. This method of storage allows the MB to store information throughout the lifetime of the MB, regardless of the the numbers contained in the state buffer. The updates to the probability tables of feedback gates are an abstraction of changes to neural substrate that we observe in nature.

1.5.3 Experimental Setup

Many organisms in nature have to learn how to interact with the world over their lifetime. As humans we do not know how to walk at birth, rather over a series of trial and error learn how to control our muscles to achieve the task over time. I will show that feedback gates use their ability to update their probability tables to learn to accomplish a complex task. This task is the basis of all projects presented in the remaining chapters.

Agents are presented with a complex navigation task where, in order to optimally reach their goal, they must learn to correlate their cognitive states to their movements. The environment they are placed in is a $64x64$ cell lattice where a cell is randomly selected to serve as the goal the agent must reach. To prevent the agent from escaping the lattice, a 1 cell boundary is placed around the perimeter. At this point the lattice is empty with boundaries if the agent were asked to navigate to the goal the task would be somewhat trivial. However, if obstacles are added the task becomes much harder, so non-passable walls are placed in $\frac{1}{7}$ of the cells within the boundaries of the lattice. At each remaining cell, Dijkstra's path [92] to the goal is computed so the optimal path is known, and an indicator is placed in the cell. In cases in which two neighboring cells have the same optimally, one of these is chosen at random as the next closest to the goal. We can now think of the cell indicators as arrows that give the agent directions to reach the goal in the shortest number of cells.

The agent is then placed on a random, empty cell whose path to the goal is 32 cells long facing a random direction (north, west, south, or east). The agent's sensor nodes are able to detect the arrow provided by the cell they are placed on. The agent perceives the action indicated by the arrow, depending on the direction the agent is facing. For example, when

the arrow points west the agent will detect that it should move forward only when facing west. Input to the agent is provided via four binary sensors that indicate how to reach the next cell in its Dijkstra's path.

The agent can move through the lattice by turning to the left or right 90 degrees or by moving forward. In order to achieve optimal performance the agent would need to follow the arrows directions by turning the proper way and moving forward. If the agent was able to directly control its agents based simply on decoding the given arrow the navigation task would be quite trivial. Instead the agent is tasked with choosing four intermediate options (A,B,C,or D) at any given update, as seen in Figure 1.6. At the agent's birth these four intermediate options are mapped to the four possible actions: move forward, turn left, turn right or do nothing. There are now 24 combinations that the option-to-action maps that the agent could be placed in. It must use its experiences to determine the mapping of its current environment. This additional learning process is required to reach the goal in an optimal number of steps and thus increases the task complexity. Since an agent is not given direct feedback (as explained in Section 1.5.2) its actions about further intricacy arises which require the agent to evolve a mechanism to discern its current mapping.

1.6 Contributions

Now that the experimental components– the MBs, feedback gates, and the task used– of the system I used have been explained, the following four sections describe the experiments I performed to answer:

- Do feedback gates evolve the ability to learn?
- Can feedback gates allow for generalization? If so, does generalization lead to reversal learning?
- How does learning influence information integration?

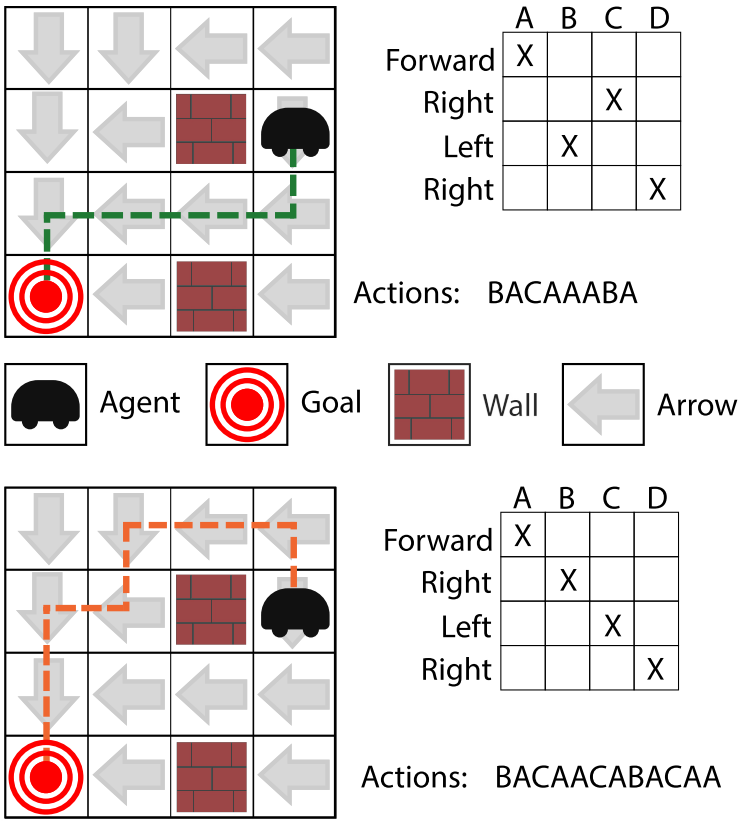

Figure 1.6: **Navigation task overview.** In the top panel an agent (black vehicle) navigates to the goal (red bulls-eye) by correctly following the gray arrows. In this panel the agent has learned the correct action-to-behavior mapping and reaches the goal in minimal steps by selecting actions BACAAABA. The bottom shows the path the agent may take if the action-to-behavior mapping was changed. In this mapping the agent takes extra steps to reach the goal by executing actions BACAACABACAA. (This figure appears in Chapter 1.6.2 and is a derivative figure from 4.1.

1.6.1 Do Feedback Gates Evolve the Ability to Learn?

As stated in Sections 1.3, there have been many attempts to develop learning algorithms, however none of these successfully investigate the intricacies of biological learning. There has also been research that focuses on modeling evolution to solve problems, as discussed in 1.4, but none specifically understand lifetime plasticity. In Chapter 2, I prove that MBs with feedback gates are a tool capable of doing just that: studying the natural evolution of intelligence.

One key to modeling natural learning is to limit the feedback agents receive to internal

feedback. This requires agents to evolve cognitive mechanisms to interpret how their actions affect their position in the environment. Here I evolve 300 populations of 100 agents for $500,000$ generations in the environment described in Section 1.5.3. Each generation the agents are given 512 updates to reach the goal as many times as possible in each of the 24 environments. Their performance on each map is used to drive the evolutionary process. At the conclusion of the evolutionary replications I then analyze each replicate's line of descent.

Using the numbers obtained in the analysis phase, I compare the average performance when feedback is applied versus when it is not applied over evolutionary time. I found that when agents are given to the ability to use feedback they reach the goal significantly more than when they are not. Agents who were able to complete the task using only advanced heuristics emerged during evolution, though these agents lacked optimal performance. Additionally, I explored if the probability table of the feedback gates as well as the amount of mutual information gained during the agents lifetime. At the end of the agents lifetime, feedback gates converge on an optimal probability table that mimics a probability table of a deterministic gate. Along these lines, a gain in mutual information is correlated with increased performance. These results prove that feedback gates allow MBs to model the evolution of neural plasticity.

1.6.2 Can Feedback Gates Allow for Generalization?

In Section 1.6.1 I showed that MBs with feedback gates are capable of solving the navigation task by changing their probability tables during their lifetime to become almost deterministic using feedback from their current environment. However, it is possible that agents are capable of learning because they were shown all possible action-to-behavior mappings during evolution. Given that feedback gates adjust their probability tables based on the environment they are placed in, it is likely that they posses the ability to *generalize*. A generalist agent is able to find the relation between different stimuli seen in previous experiments and solve any given task set in front of it despite having never seen that exact

task before. In the context of human learning this process is akin to using context clues to aid in task completion. However, in machine learning the generalization is often limited by *overfitting* and *underfitting* of the learning algorithm [93], concepts we will explore in more detail in Chapter 3. Additionally, in the chapter I proposed a set of experiments to identify:

- Are agents are able to navigate novel mappings?
- If so, what are the minimum number of maps required during evolution to facilitate generalization?

To test if MBs with feedback gates are able to generalize, I again make use of the environment described in Section 1.5.3. Instead of showing agents all 24 combinations of action-to-behavior mappings, I form subsets of the mappings that show agents between 1 and 24 mappings during evolution. For each of the 24 subsets I evolve 100 instances of evolution with a population size of 100 agents. Each replicate was evolved for $500,000$ generations and fitness was based on the average number of goals reached in 512 updates during each generation. Further analysis was run on the line of descent of all replicates where the agents were shown all 24 mappings.

During analysis I compared the the average agent performance on maps they have seen during evolution to their performance on maps they have not seen in each of the 24 subsets of mappings. I found that when agents are presented enough experiences, aka mappings, they are able to generalize. Further there are a clear number of experiences required to develop the skill. I also analyzed the mean delta mutual information of the agents both across subsets and versus performance. These tests show the amount of mutual information agents gain within their lifetime is influenced by these experiences which is reflected their performance across all mappings. Further, I discovered that 3 strategies to solving the navigation task exist: memorization, advance heuristics, and learning. The emergence of each strategy is heavily influenced by the number of mappings seen during evolution. These findings suggest that there is a lower bound of experiences that a natural organism must have before they

are able to generalize.

1.6.3 Does Generalization Lead to Reversal Learning?

In addition to the ability of MBs with feedback gates to generalize, Chapter 1.6.2 explores if agents can *reversal learn*. Reversal learning is when organisms adapt to one set of stimuli and then the stimuli set is changed which forces them to learn the new set to survive. The change in stimuli requires organisms to first recognize a difference in outcomes and then adjust their behavior to the new queues. Since MBs are able to generalize to new environments, the question now becomes: Are they able to reversal learn?

Additional analysis steps were performed on the same agents evolved in Section 1.6.2 to test if they were able to reversal learn. For each subset of mappings seen in evolution, I allow agents to explore a mapping for 512 updates and then switch the action-behavior-mapping which they explore for an additional 512 updates. This test demonstrates that reversal learning does occur, but the performance suffers in the second mapping agents see. These findings indicate some degree of reversal learning is a by-product of the evolution of generalization.

1.6.4 How Does Learning influence Information Integration?

The neuro-correlate Φ is used to quantify the amount of information integration a cognitive system performs, and will be explained in 4. Since the introduction of information integration theory [94, 95, 96, 97] it has been shown that integrated information increases over the course of Darwinian evolution [1]. Further, Joshi et. al. [74] established that the complexity of the task to be solved defines a lower bound of Φ and Albantakis et.al. [77] that more complex environments require an ever increasing amount of Φ. Given that feedback gates change their probability tables within their lifetime we can ask:

- Does learning result in an increase or decrease in Φ?
- Does *Phi* increase or decrease within the lifetime of an agent?

- Is performance correlated to the change in Φ?

To this end, I used agents [2] from Section 1.6.1 who were top performers on the navigation. First I used the line of descent of these agents to compare the measurement of Φ when feedback was allowed and when it was not. The comparison showed that learning increases the value of Φ over evolutionary time. I then subdivided the agents' lifetimes and computed Φ at 5 major milestones. I find that Φ increases over the course of the lifetime. To determine if performance is correlated to Φ, I examined $\delta_{performance}$ vs $\delta\Phi$ and found that the increase Φ correlates to a increase in performance. This means that Φ increases not only over evolutionary time but also when learning.

[2] Observe that in the context of phi agents are called animats. This is purely traditional, both terms are synonymous.

Chapter 2

Evolving Autonomous Learning in Simple Cognitive Networks

2.1 Introduction

Natural organisms not only have to fit their environment, but also have to adapt to changes in their environment which means that they have to be plastic. While plasticity occurs in many forms, here we focus on *neural plasticity* which we define as an organisms ability to use "experiences" to improve later decisions and behavior. Being able to solve a T-maze repetitively, remembering where food located, avoiding places where predators have been spotted, and even learning another language are all cognitive abilities that require an organism to have neural plasticity. We will show how this neural plasticity can evolve in a computational model system. Neural plasticity allows natural organisms to learn due to reinforcement of their behavior [98]. However, learning is tied to specific neural mechanisms-working memory (WM), short-term memory (STM) and long-term memory (LTM). While learning was initially perceived as a new "factor" in evolution [21], potentially even independent, it has since been well integrated into the *Modern Synthesis of Evolution* [22]. Evolution and learning can have a positive effect on each other [99, 100], however, this is not necessarily always the case [101]. This has several implications: Evolution begin with organisms that could not adapt during their lifetime, which means that they had no neural plasticity. The

only feedback that the evolutionary process receives is differential birth and death. As a consequence, learning will only evolve if it can increase the number of viable offspring, and it can only do so if there is a signal that predictably indicates a fitness advantage [102].

Organisms receive many signals from their environment which have to be filtered and interpreted. Irrelevant signals should be ignored while others require adaptive responses. This can be done through instincts or reflexes in cases where fixed responses are necessary. In other cases information has to be stored and integrated in order to inform later decisions, which requires memory and learning. To distinguish between actions that lead to advantageous results and those that are disadvantageous organisms need positive or negative feedback. However, none of the signals organisms receive are inherently "good" or "bad"; even a signal as simple as food requires interpretation. The consumption of food has to trigger positive feedback within the organism in order to function as a reward. The machinery that triggers the feedback is an evolved mechanism and is often adaptive to the environment. If food were a global positive feedback signal, it would reinforce indiscriminate food consumption. Organisms would not be able to avoid food or store it for later, but instead eat constantly.

Another important detail we have to consider is the difference between learning and memory. While memory is information about the past, learning is the process that takes a sensorial percept and, typically by reinforcement, retains that information for later use. Specifically, sensor information is stored in working memory (WM) [25, 24]. Imagine this as the flurry of action potentials that go through the brain defining its current state. Information that a living organism needs to store for a moment is believed to reside in STM [24, 8], but how information transforms from WM to STM is not fully understood [8, 25, 10]. Natural systems use their LTM if they want to keep information for longer. Presumably, information from STM becomes reinforced and thus forms LTM, this is sometimes referred to as consolidation [24, 8, 27]. The reinforcement process takes time and therefore is less immediate than STM. In addition, memories can be episodic or semantic [9, 8, 24] and can

later be retrieved to influence current decisions. While information in the working memory can be used to influence decisions, it does not change the cognitive substrate. However, long term potentiation (or other neural processes) use this information to change the neural substrate by presumably forming or modifying connections.

In summary, if we want to model the evolution of learning in natural organisms properly, we need to take the following statements seriously:

- evolution happens over generations while learning happens during the lifetime of an organism
- evolution is based on differential birth and death (selection) and learning evolved to increase the number of viable offspring and/or to avoid death
- organisms do not receive an objective "positive" or "negative" signal, but instead evolved mechanisms to sense and interpret the world so that they can tell what actions were positive and which ones were not
- memory is information about the past and can be transient
- information in the WM does not change the cognitive machinery, while learning changes the substrate to retain information for longer, which turns transient into permanent information

2.1.1 Machine Learning

Computer science and engineering are typically not concerned with biological accuracy but more with scalability, speed, and required resources. Therefore, the field of machine learning is much more of a conglomerate of different methods which straddle the distinct concepts we laid out above. Machine learning includes methods such as data mining, clustering, classification, and evolutionary computation [103]. Typically, these methods try to find a solution to a specific problem. If we provide an explicit reference or example class we

refer to it as supervised learning since the answer is known and the fitness function quantifizes the difference between the provided solution and the ones the machine generates. For unsupervised learning we provide a fitness function that measures how well a machine or algorithm performs without the need to know the solution in advance. Genetic algorithms (GAs), which are a form of evolutionary search, work in supervised or unsupervised contexts, whereas learning algorithms are typically supervised. A special class are learning to learn algorithms, which improve their learning ability while adapting to a problem [104] but do not necessarily apply evolutionary principles.

Genetic algorithms clearly optimize from one generation to the other, while learning algorithms on the other hand could be understood as lifetime learning. Q-learning [35] optimizes a Markov Decision Processes by changing probabilities when a reward is applied. Typically, delayed rewards are a problem, that deep-Q learning and memory replay try to overcome [105]. Artificial neural networks can be trained by using back propagation [41, 39, 40], the Baum-Welch algorithm [42, 43], or gradient decent [106, 107] which happens episodically, but on an individual level and not to a population that experiences generations. Multiplicative weights algorithm strengthens or weakens connections in a neural network-based on the consensus of a pool of experts [45, 44], again on an individual level.

At the same time, memory and learning are often treated interchangeably. Recurrent artificial neural networks can store information in their recurrent nodes (or layer) without changing their weights, which would be analogous to WM. Similarly, the system we use, Markov Brains (MB), can form representations about their environment and stores this information in hidden states (WM), again transiently without changing its computational structure [72] (For a general explanation of Markov Brains see `https://arxiv.org/abs/1709.05601` [2]). Changes to the weights of an ANN, the probabilities of a Markov process, or the probabilities of a partially observable Markov decision process (POMDP) [108] reflect better learning, since those changes are not transient, and change all future computations executed by the system.

We also find a wide range of evolvable neural network systems [109, 63, 110] (among many others) which change from generation to generation and allow for memory to form by using recurrent connections. Alternatively, other forms of evolving systems interact and use additional forms of memory [61, 89]. In order to evolve and learn, other systems allow the topology and/or weights of the neural network to change during evolution while also allowing weight changes during their lifetime [81, 82, 83, 84, 85, 90, 86, 87, 88, 89, 91]. Presenting objective feedback to adapt these systems during their lifetime allowed their performance to improve. As a consequence, the machinery that interprets the environment to create feedback was of no concern, but, as stated above, natural organisms also need to evolve that machinery to learn. We think it is quite possible to change these systems to not rely only on external feedback. Instead, they themselves could create the feedback signal as part of their output. However, none of the systems mentioned above is an evolvable MB (for a comparison see Figure 2.1 A vs. B).

In our approach we use MBs [1], which are networks of deterministic and probabilistic logic gates, encoded in such a way that Darwinian evolution can easily improve them. MBs have been proven to be a useful tool to study animal behavior [75, 76], neural correlates [74, 72, 77],evolutionary dynamics [111], decision making [78, 112], and can even be used as a machine learning tool [73, 113]. One can think of these MBs as artificial neural networks (ANN) [47] with an arbitrary topology that uses Boolean logic instead of logistic functions. Through sensors these networks receive information about their environment as zeros or ones, perform computations and typically act upon their environment through their outputs. We commonly refer to MBs that are embodied and through that embodiment [80] interact with their environment as agents (others use the term animat which is synonymous). MBs use hidden states to store information about the past similar to recurrent nodes in an ANN. The state of these hidden nodes has to be actively maintained which makes the information volatile. The information in the hidden states can be used to perform computations and functions as memory. This form of memory resembles WM or STM more than LTM due

its volatile nature. In the past, the entire structure of a MB would be encoded by the genome and would not change over the lifetime of the agent. Here we introduce what we call feedback gates, which allow MBs to use internal feedback to store information by changing their probabilistic logic gates (see Methods for a detailed description of feedback gates). Like other systems these updates do not change the topological structure of the node network but rather the probabilities within the gates; similar to how learning in ANN is achieved through weight changes. However, feedback is not an objective signal coming from the environment but must be generated as part of the evolved controller. The feedback gates only receive internally generated feedback to change their behavior. This linkage between inputs, evaluation of the environment to generate feedback, how feedback gates receive this information, and how everything controls the actions of the agent evolves over time (see Figure 2.1 B).

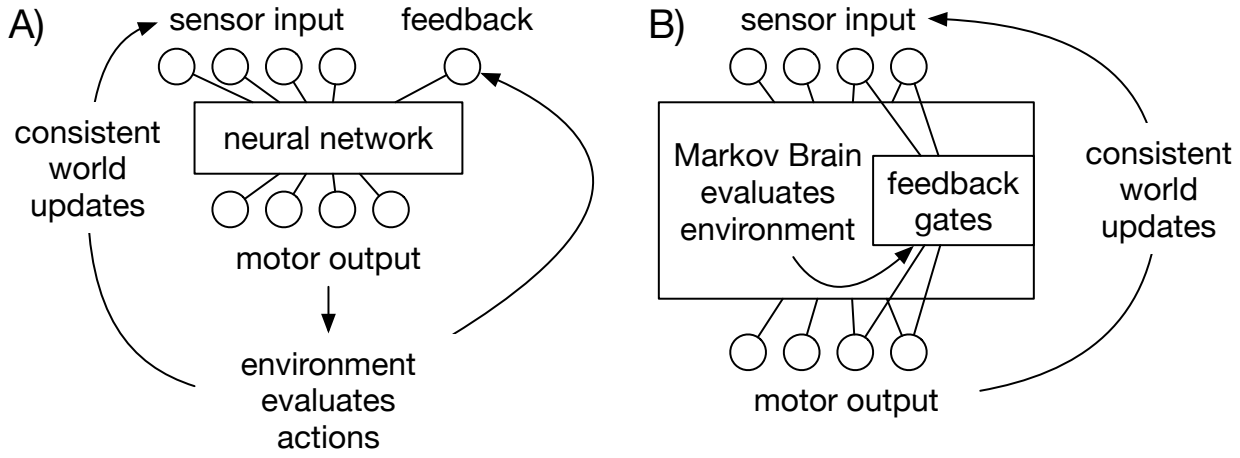

Figure 2.1: **Comparison of two different approaches to feedback generation.** Traditional methods have the environment evaluate the actions of a neural network (or similar system) and then generate feedback which is provided as an external signal, similar to adding another input (panel A). Our approach (panel B) integrates both the feedback generation and the feedback gates into the Markov Brain. This way, the world provides consistent information about itself which can be evaluated so that feedback generation does not need to be offloaded to the environment, but becomes part of the entire machine. The entire connectivity as well as feedback gates are integrated as part of what needs to evolve.

Here we introduce feedback gates that allow MBs to change their internal structure

which is akin to learning and forming long-term memories during the lifetime of an organism. The closest similarity can be found in Q-learning or deep-Q learning, which changes the probabilities within a Markov Decision process even if rewards are delayed [103] (for a detailed explanation of the feedback gate function see Materials and Methods as well as supplementary information Figure 2.7). We will show that feedback gates function as expected and allow agents to evolve the ability to decipher sensor inputs so that they can autonomously learn to navigate a complex environment .

2.2 Results

Natural organisms have to learn many things over their lifetime including how to control their body. Here the environment used to evolve agents resembles this problem. Agents have to learn to use their body in order to navigate properly. The environment is a 2D lattice (64x64 tile wide) where a single tile is randomly selected as the goal an agent must reach. The lattice is surrounded by a wall so agents cannot escape the boundary, and $\frac{1}{7}$ of the lattice is filled with additional walls to make navigation harder. From the goal the Dijkstra's path is computed so that each tile in the lattice can now indicate which of its neighboring tiles is the next closest to the goal. In cases where two neighbor tiles might have the same distance, one of these tiles is randomly selected as the next closest. For ease of illustration we can now say that a tile has an arrow pointing towards the tile that should be visited next to reach the goal in the shortest number of tiles.

The agent, controlled by a MB, is randomly placed on a tile that is 32 tiles away from the goal and facing in a random direction (north, west, south, or east). Agents can see the arrow of the tile they are standing on. The direction indicated by the tile is relative to the that of the agent, so that a tile indicating north will only be perceived as a forward facing arrow if the agent also faces north. The agent has four binary sensors that are used to indicate which relative direction the agent should go to reach the goal.

The agent can move over the lattice by either turning 90 degrees to the left or right, or by

moving forward. So far, in order to navigate perfectly, the agent would simply need to move forward when seeing a forward facing arrow, or turn accordingly. Instead of allowing the agent to directly pick a movement, it can choose one of four intermediate options (A,B,C,or D) at any given update. At the birth of an agent, these four possible options are mapped to the four possible actions: move forward, turn left, turn right, do nothing. As a result, the complexity of the task increases when the agent has to learn which of the 24 possible option-to-action maps currently applies to navigate the environment properly. The agent is not given any direct feedback about its actions; a mechanism must evolve to discern the current mapping which is rather difficult.

In prior experiments [1, 72, 77, 73, 76, 74, 78, 111, 75], MBs were made from deterministic or probabilistic logic gates that use a logic table to determine the output given a particular input. Deterministic gates have one possible output for each input, while probabilistic gates use a linear vector of probabilities to determine the likelihood for any of the possible outputs to occur. To enable agents to form LTM and learn during their lifetime we introduce a new type of gate: feedback gate. These gates are different from other probabilistic gates, in that they can change their probability distribution during their lifetime based on feedback (for a detailed description, see below). This allows for permanent changes which are akin to LTM. While MBs could already retain information by using hidden states, now they can also change "physically". MBs must evolve to integrate these new gates into their network of other gates and find a way to supply feedback appropriately.

2.2.1 Feedback Gate Usage

To test if the newly introduced feedback gates help evolution and increase performance, we compare three different evolutionary experimental conditions. Agents were evolved over 500,000 generations that could use only deterministic logic gates, only probabilistic logic gates, or all three types of gates–deterministic, probabilistic, and feedback gates to solve the task.

When analyzing the line of descent (LOD; see materials and methods), we find a strong difference in performance across the three evolutionary conditions (see supplementary information Figure 2.8). None of the 300 agents that were evolved using only probabilistic logic gates were capable of reaching the goal in any of the 24 mappings. The agents that were allowed to use only deterministic logic gates failed to reach the goal in 225 of the 300 experiments, but the remaining 75 agents never reached the goal more than five times. Agents allowed to use all gates including feedback gates only failed to reach the goal in 75 experiments and in the remaining 225 experiments they reached the goal on average 5 times; the best performer reaching the goal 9 times on average. A Wilcoxon rank sum test [114] comparing the final distributions of performances for each experimental condition showed that we can reject the hypothesis that they were drawn from the same distribution (with the lowest p-value smaller than 10^{-20}). This shows that agents that were allowed to use feedback gates outperform all other conditions by far.

It is not entirely surprising to us that agents using only probabilistic gates struggle in this task; because agents using probabilistic gates generally evolve slower, which might explain the effect. However, to our surprise, we found a couple of agents who were only allowed to use deterministic gates evolved to solve the task at least a couple of times. In the group that could use all three types of gates, we found 3 agents that could reach the goal on average 5 times using only probabilistic gates, as opposed to the 9 times the top agent with feedback gates could reach the goal on average. This shows two things: the task can be solved using only the gate inputs (i.e. WM) and providing agents with feedback gates during evolution allows them to reach the goal more often. This is an important control because from a computational point of view there is no qualitative difference between WM and LTM as both methods allow for recall of the past.

Populations allowed to use feedback gates quickly evolve the ability to reach the goal in any of the 24 possible environments. The variance of their performance supports the same idea, that agents do not become better by just performing well in one environment,

but instead evolve the general ability to learn the mapping each environment presents (See Figure 2.2 panel B).

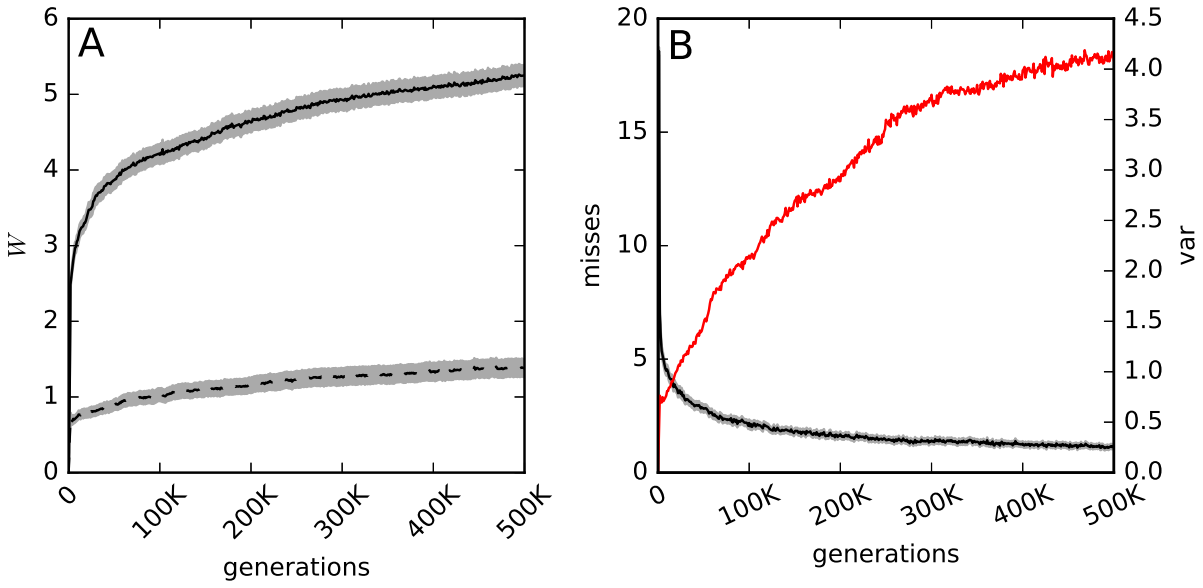

Figure 2.2: **Performance over evolutionary time.** Panel A solid line shows how often the goal was reached (W) by all 300 replicate experiments across all 24 possible environments for agents on the line of descent. The dotted line is the same average performance for the same agents when their feedback mechanism was disabled. The underlying gray-shade indicates the standard error. Panel B shows how often on average the 300 replicate agents on the line of descent could not reach the goal a single time in any of the 24 possible environments as a black line. In red is the variance in performance on the line of descent as an average over all 300 replicate experiments.

Now that we have shown that agents with feedback gates are capable of evolving a solution to navigate in this environment, we have to ask if they actually utilize the feedback gates. For that, all agents on the LOD were tested again but their feedback gates were kept from changing their probability tables. Comparing these results with the agents performance when using the feedback gates regularly reveals that the agents rely heavily on their feedback gates (see Figure 2.2 panel A). This implies that the evolved agents indeed store information about the environment for longer periods using their feedback mechanism. If they would have used hidden states as the only means to save information for later, blocking the feedback signal would not have caused a loss of function. Therefore feedback gates indeed allow for

the formation of LTM.

2.2.2 Feedback gates change over time

We find that the probability tables modified by feedback become specifically adapted to each of the 24 possible mappings the agents get tested in. See Figure 2.3 as an example of the best performing agent using only one feedback gate. Some rows in the probability tables converge to having a single high value that is specific to the environment the agent experienced (for more details see supplementary information Figures 2.9 and 2.10). This shows, that indeed feedback gates become specifically adapted to the environment the agent experiences. It also indicates, that agents change their computational machinery according to their environment and do not rely solely on WM to perform their task.

The change to the feedback gates' probability tables can be quantified by measuring the mutual information each table conveys at birth and after the agent completes the task. We find that the mutual information is generally lower at birth ($\sim$0.25) and higher at the end of the task ($\sim$0.8). This signifies that the agents have more information about the environment at death than they did at birth, as expected. We then compute the difference between both measurements, ($\bar{\Delta}$), which quantifies the change of mutual information over the lifetime of the agent. When discretizing these values for different levels of performance, we find a strong correlation ($r = 0.922$) between performance and increase in mutual information (see Figure: 2.4). This shows that agents who perform better increase information stored in their feedback gates over their lifetime.

2.2.3 Differences between agents using feedback gates and those who do not

We wanted to test if feedback gates improve the agents ability to learn. Those agents that evolved to use feedback gates generally perform very well in the environment, however, we also find other agents that only use deterministic gates and still have an acceptable

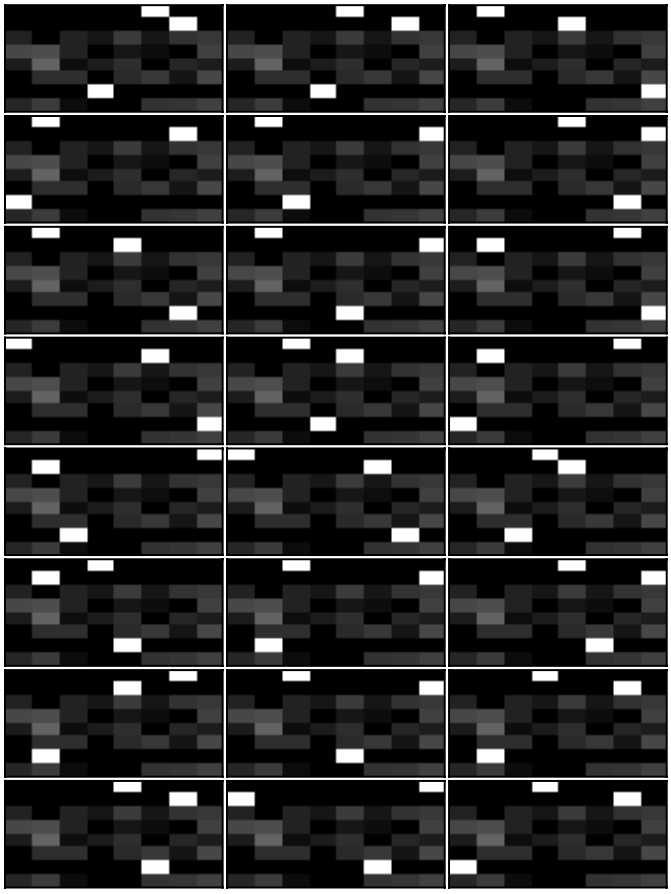

Figure 2.3: **Probability tables of a feedback gate after it learned each of the 24 possible mappings.** Each of the 24 gray scale images corresponds to a feedback table adapted during the lifetime of an agent to a different mapping. The darker the color, the lower the probability, observe that rows have to sum to 1.0. Some rows, apparently those of input combinations that were never experienced, remain unadapted. Rows one, two, and seven of each table converge to a single high value surrounded by low probabilities.

performance. This either suggests that feedback gates are wither not necessary or that they do not provide enough of an selective advantage to be used every time. It is also possible that there is more than one algorithmic solution to perform well in this navigation task. All of these points suggest that a more thorough analysis and comparison of the independently evolved agents is necessary.

We find that agents that do not use feedback gates require a much greater number of logic gates than those who do (see Figure 2.5). This seems intuitive, since feedback gates can store information in their probability tables whereas agents that do not use them need

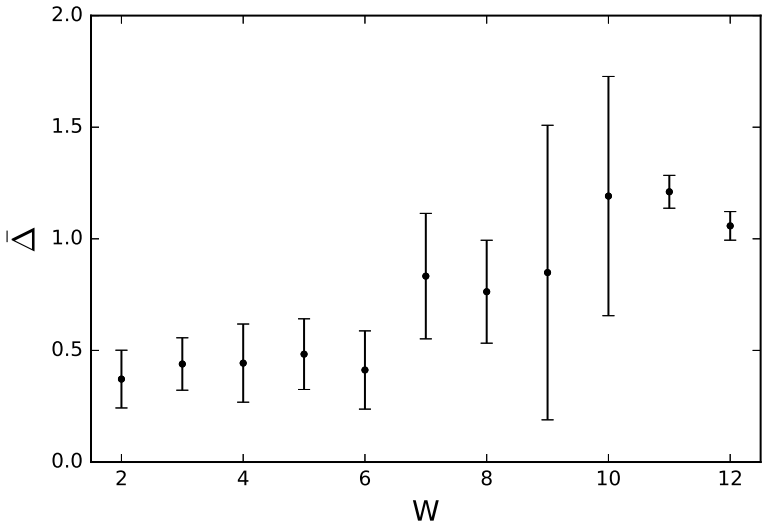

Figure 2.4: C**hange in mutual information of feedback gates for different levels of performance. The change in performance ($\bar{\Delta}$) is shown on the y-axis for different performances (W). The performance for all 300 final organisms using all types of gates was measured and put into ten bins, ranging from the lowest performance 2 to the highest performance of 12, with a bin size of 1. Error-bars indicate the variance.**

to store all information in their WM. This suggests that there might be a difference in the evolved strategy between those agents that use feedback gates and those who do not. When observing the behavior of the differently evolved agents we find that there are two types of strategies (see supplementary information Figures 2.11 and 2.12 for details). Agents that evolved brains without feedback gates use a simple heuristic that makes them repeat their last action when the arrow they stand on points into the direction they are facing. Otherwise, they start rotating until they move off a tile, which often results in them standing again on a tile that points into the direction they are facing, which makes them repeat the last action (see supplementary information Figure 2.13).

Agents that evolved to use feedback gates appear to actually behave as if they learn how to turn properly. They make several mistakes in the beginning, but after a learning period perform flawlessly. Of the 300 replicate evolutionary experiments where agents were allowed to use all types of logic gates, 56 did not evolve using feedback gates. Comparing

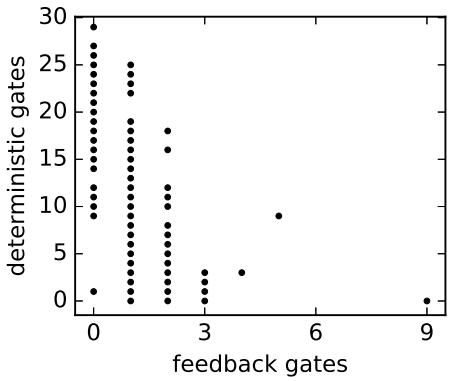

Figure 2.5: **Correlation of number of feedback gates to deterministic gates.** Each dot represents the number of feedback gates the last agent on the line of descent has versus the number of deterministic gates it has in 300 replicates. The more feedback gates an agent has the less deterministic gates it evolves; the feedback gates allow agents to decrease brain size while still solving the task.. The Pearson correlation coefficient between the number of feedback gates and deterministic gates is -0.52 with p-value of $8.64e^{-06}$

the actions those 56 agents take with the remaining 244 which do use feedback gates we first find that as expected the group using feedback gates reached the goal more often on average (5.33 times, versus 4.89 times for those agents not using feedback gates) which suggests a difference in behavior. The usage of actions is also drastically different during evolution (see Figure 2.6). Agents using feedback gates reduce the instances where they do nothing, minimize turns and maximize moving forward. Agents not using feedback gates are less efficient because they rely on forward movements while minimizing times where they do nothing and turns. In conjunction with the observations made before, we conclude that indeed agents not using feedback gates, use some form of heuristic with a minimal amount of memory while agents using feedback gates learn to navigate the environment properly and quickly (see supplementary information Figure 2.14 and Section 2.6.8 for a more details regarding the evolved solutions).

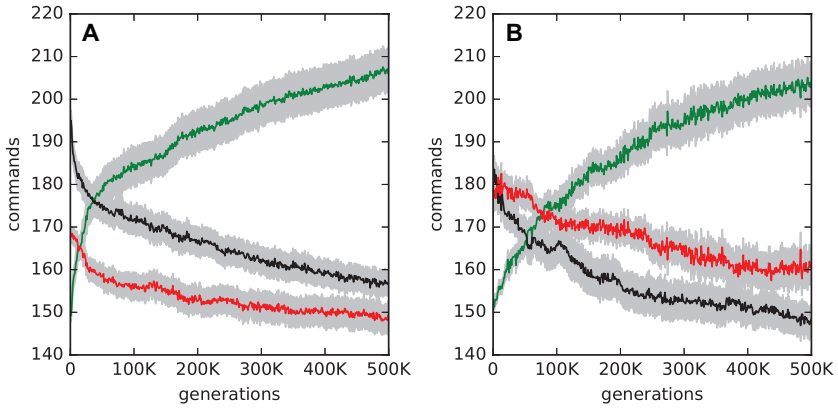

Figure 2.6: **Different actions performed while navigating over evolutionary time.** Both panels show the average use of forward (green), do nothing (red), and turn (black) commands over generations. Gray background indicates the standard error. Panel A shows those 244 agents that do use feedback gates, Panel B shows the remaining 56 independently evolved agents that do not use feedback gates. Agents on the LOD were analyzed.

2.3 Discussion

In prior experiments, Markov Brains could evolve to solve various complex tasks and even form memories that represent their environment [72]. However, these MBs use binary hidden states to store information about the environment similar to WM or STM, but lacked a feedback mechanism that allowed them to change their internal structure, very much like long term potentiation would lead to LTM. Other systems like artificial neural networks already allowed for a similar process, were weights could be adapted due to feedback. We augmented MBs with feedback gates that use the multiplicative weight update algorithm to change their probabilities given positive or negative feedback.

It is important to note that this feedback is not directly provided by the environment to the agents but must be internally generated by the MB. In addition, MB need to integrate these feedback gates into their cognitive machinery during evolution. We showed that we can successfully evolve MBs to use feedback gates. These gates generated internal feedback and they change their internal probabilities as expected. However, we also find competing strategies, which instead of evolving to learn deploy some form of heuristic. In the future it might be interesting to study under which evolutionary conditions either heuristics or

learning strategies evolve (similar to Kvam 2017). We used a biological inspired task and we see the future application of this technology in the domain of computational modeling to study how intelligence evolved in natural systems and to eventually use neuroevolution as the means to bring about general artificial intelligence. ANNs that are specifically optimized by machine learning will solve specific classification tasks much faster than the model introduced here. The idea is not to present a new paradigm for machine learning, but to implement a tool to study the evolution of learning. While using MBs augmented with feedback gates will probably not be competitive with other supervised learning techniques, it remains an interesting question: How would typical machine learning tools perform if challenged with the task presented here? (see supplementary information Figure 2.15 for a comparison with Q learning on a single environment).

One problem encountered by systems that receive external feedback is "catastrophic forgetting" [115, 116]. When the task changes, feedback will cause the system to adapt to their new objective but they forget or unlearn former capabilities. Requiring adaptive systems to interpret the environment in order to generate internal feedback might be a way to overcome this problem; assuming that the fitness function or evolutionary environment not only changes but also actively rewards organisms who do not suffer from "catastrophic forgetting".

We now have the ability to evolve machines inspired by natural organisms that can learn without an objective external signal. This gives us a tool to study the evolution of learning using a computational model system instead of having to use natural organisms. It will also allow us to study the interplay between evolution and learning (aka Baldwin effect) and explore under which circumstances evolution benefits from learning and when it does not; we propose to use this model to study these questions in the future. Another dimension we will investigate in the future is the ability of MBs to change their connections due to feedback, not just the probabilities within their gates.

2.4 Conclusion

As stated before, combining evolution with learning is not a new idea. We think that it is in principle very easy for other systems to internalize the feedback. For example, it should be easy to evolve an artificial neural network to first interpret the environment, and then use this information to apply feedback on itself. However, we need to ask under which circumstances this is necessary. By providing training classes for supervised learning situations we can already create (or deep learn) machines that can learn to classify these classes. In addition, we often find these classifiers exceed human performance [117, 46]. If we are incapable of providing examples of correctly classified data we use unsupervised learning methods and only need to provide a fitness function that quantifies performance. But we know that fitness functions can be deceptive and designing them is sometimes more of an art than a science. When interacting with future AI systems we should find a different way to specify what we need them to do. Ideally they should autonomously explore the environment and learn everything there is to know without human intervention - nobody tells us humans what to research and explore, evolution primed us to pursue this autonomously. The work presented here is one step into this direction and will allow us to study evolution of learning in a biological context as well as explore how we can evolve machines to autonomously learn.

2.5 Methods

Markov Brains are networks of probabilistic and deterministic logic gates encoded by a genome. The genome contains genes and each gene specifies one logic gate, the logic it performs and how it is connected to sensors, motors and to other gates [1, 72, 111]. A new type of gate, the feedback gate, has been added to the Markov Brain framework (https://github.com/LSheneman/autonomous-learning),and this framework has been used to run all the evolutionary experiments. The Markov Brain framework has since been updated to MABE [118]. See below for a detailed description of each component:

2.5.1 Environment

The environment the agents had to navigate was a 2D spatial grid of 64x64 tiles. Tiles were either empty or contained a solid block that could not be traversed. The environment was surrounded by those solid blocks to prevent the navigating agent from leaving that space. At the beginning of each agent evaluation a new environment was generated and $\frac{1}{7}$ of the tiles were randomly filled with a solid block. The randomness of the environments maintains a complex maze-like structure across environments, but no two agents saw the exact same environment.

A target was randomly placed in the environment, and Dijkstra's algorithm [92] was used to compute the distance from all empty tiles to the target tile. These distances were used to label each empty tile so that it had an arrow facing to the next closest block to the target. When there was ambiguity (two adjacent tiles had the same distance) a random tile of the set of closest tiles was chosen. At birth agents were randomly placed in a tile that had a Dijkstra's number of 32 and face a random direction (up, right, down, or left). Due to the random placement of blocks it was possible that the goal was blocked so that there was no tile that is 32 tiles away, in which case a new environment was created, which happened only very rarely.

Agents were then allowed to move around the environment for 512 updates. If they were able to reach the target, a new random start orientation and location with a Dijkstra's number of 32 was selected. Agents used two binary outputs from the MB to indicate their actions– 00, 01, 10, or 11. Each output was translated using a mapping function to one of four possible actions- move forward, do nothing, turn left, or turn right. This resulted in 24 different ways to map the four possible outputs of the MB to the four possible actions that moved the agent. The input sensors gave information about the label of the tile the agent was standing on. Observe that the agent itself had an orientation and the label was interpreted relative to the direction the agent faced. There were four possible arrows the agent could see– forward, right, backward, or left– and were encoded as four binary inputs,

one for each possible direction. Beyond the four input and two outputs nodes, agents could use 10 hidden nodes to connect their logic gates. Performance (or fitness) was calculated by exposing the agent to all 24 mappings and testing how often it reached the goal within the 512 updates it was allowed to explore the world. At every update agents were rewarded proportional to their distance to the goal (d), and received a bonus (b) every time they reached the goal, thus the fitness function becomes:

$$W = \prod_{n=0}^{n<24} ((\sum_{t=0}^{t<512} \frac{1}{1+d}) + b) \tag{2.1}$$

2.5.2 Selection

After all agents in a population were tested on all 24 option-to-action mappings at each generation, the next generation was selected using tournament selection where individuals are randomly selected and the one with the best fitness transmits offspring into the next generation [119]. The tournament size was set to five.

2.5.3 Mutation

Genomes for organisms in the first generation were generated randomly with a length of 5000 and 12 start codons were inserted that coded for deterministic, probabilistic, and feedback gates. Each organism propagated into the next generation inherited the genome of its ancestor. The genome had at least $1,000$ and at most $20,000$ sites. Each site had a 0.003 chance to be mutated. If the genome had not reached its maximum size stretches of a randomly selected length between 128 and 512 nucleotides got copied and inserted at random locations with a 0.02 likelihood. This allowed for gene duplications. If the genome was above 1000 nucleotides, there was a 0.02 chance for a stretch of a randomly selected length between 128 and 255 nucleotides to be deleted at a random location.

2.5.4 Feedback Gates

At every update of a probabilistic gate, an input i resulted in a specific output o. To encode the mapping between all possible inputs and outputs of a gate we used a probability matrix P. Each element of this matrix P_{io} defined the probability that given the input i the output o occurred. Observe that for each i the sum over all o must be 1.0 to define a probability:

$$1.0 = \sum_{o=0}^{O} P_{io} \tag{2.2}$$

where O defines the maximum number of possible outputs of each gate.

A feedback gate uses this mechanism to determine its output at every given update. However, at each update we consider the probability P_{io} that resulted in the gate's output to be causally responsible. If that input-output mapping for that update was appropriate then in future updates that probability should be higher. If the response of the gate at that update had negative consequences, then the probability should be lower. As explained above, the sum over all probabilities for a given input must sum to one. Therefore, a single probability cannot change independently. If a probability is changed, the other probabilities are normalized so that equation 2.2 remains true.

But where is the feedback coming from that defines whether or not the action of that gate was negative or positive? Feedback gates posses two more inputs, one for a positive signal and one for a negative signal. These inputs can come from any of the nodes the Markov Brain has at its disposal, and are genetically encoded. Therefore, the feedback can be a sensor or an output of another gate. Receiving a 0 at either of the two positive or negative feedback inputs has no effect, whereas reading a 1 triggers the feedback.

In the simplest case of feedback a random number in the range $[0, \delta]$ is applied to the probability P_{io} that was used in the last update of the gate. In case of positive feedback the value is increased; in the case of negative feedback the value is decreased. The probabilities cannot exceed 0.99 or drop below 0.01. The rest of the probabilities are then normalized.

The effects of the feedback gate are immediately accessible to the MB. However, because MBs are networks, the signal that a feedback gate generates might need time to be relayed to the outputs via other gates. It is also possible that there is a delay between an agent's actions and the time it takes to receive new sensorial inputs that give a clue about the situation being improved or not. Thus, allowing feedback to occur only on the last action is not sufficient. Therefore, feedback gates can evolve the depth of a buffer that stores prior P_{io} values up to a depth of 4. When feedback is applied all the probabilities identified by the elements in the queue are altered. The δ is determined by evolution and can be different for each element in the queue.

2.5.5 Line of descent

An agent was selected at random from the final generation to determine the line of descent (LOD) by tracing the ancestors to the first generation [120]. During this process, the most recent common ancestor (MRCA) is quickly found. Observe that all mutations that swept the population can be found on the LOD, and the LOD contains all evolutionary changes that mattered.

2.6 Supplementary Information

2.6.1 Feedback Gate

A feedback gate is very similar to a probabilistic logic gate, in that it has inputs, outputs and a probability table that defines the likelihood that an input gets mapped to an output at every given update (See Figure 2.7 1). However, it also possesses two additional connections, one to detect when positive feedback is given, and one for negative feedback (See Figure 2.7 red and green wires going into the gate). In our specific example here, we assume that at the first update the gate did, it received an input of 0 for A and 1 for B. It then used the probability table and returned an output of 1 (See Figure 2.7 2). In the probability table we marked the value that mapped the input 0,1 to the output 1 in blue. Technically, all probabilities in the row for that input are responsible, but here we call only the probability in blue the responsible one. The probability should be higher if that mapping was advantageous, and lower if the decision of the gate was incorrect or had negative consequences. In step 3, the gate now receives a positive feedback signal, and no negative signal otherwise (See Figure 2.7 3). Consequently, the formerly marked blue probability needs to be increased as indicated by the upward facing arrow in the probability table (0.7 becomes 0.8). Because each row in the table is summing to 1.0 in order to ensure that all entries remain probabilities, we need to re-normalize the row, and thus the other values need to be decreased (light gray probability with an arrow facing down). After feedback is computed, new inputs need to be processed. Here the gate receives a 1 for input A and a 0 for input B. This leads to a new output of 0, which identifies a new probability in the probability table, highlighted in blue (See Figure 2.7 3 blue highlight).

At the next update (See Figure 2.7 4), the gate now receives a new negative feedback, but no positive one. This means that the probability remembered (blue) was too high and needs to be decreased. The arrow facing down in the dark blue section of the table. Similarly, due to normalization, the other values in that row need to be increased, indicated by the

upwards facing arrow in the light gray box. Again, after feedback was applied, the gate still needs to map the newly received input (1,1) to an output, and remembers the responsible probability (blue).

This principle would only allow for immediate feedback. When implemented A memory queue is added to remember the sequence of all responsible probabilities. The depth of the queue is evolvable and when feedback is applied, all probabilities in the queue become modified (positively or negatively depending on the nature of the feedback). The magnitude of the change is also evolvable. The elements of the queue are processed sequentially for oldest to youngest, and after each step of the sequence the probability matrix becomes normalized again.

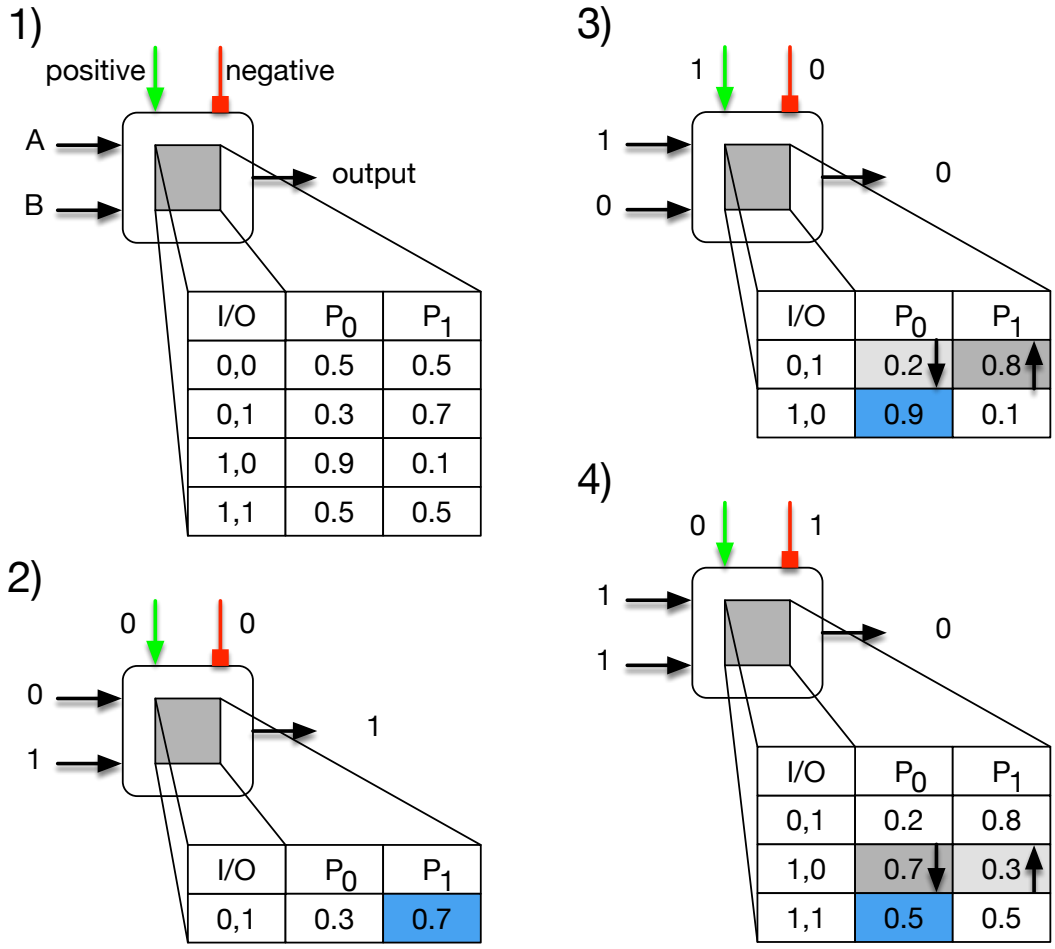

Figure 2.7: **Functionality of a feedback gate.** See the text for an explanation.

2.6.2 Minimum Performance Distribution

For each evolutionary experimental condition, we recorded the minimum number of times each of the 300 agents reached the goal across all 24 maps. We see that minimum number of times an agent reaches the goal has a left-skewed distribution in all three treatments, indicating that learning the task is non-trivial.

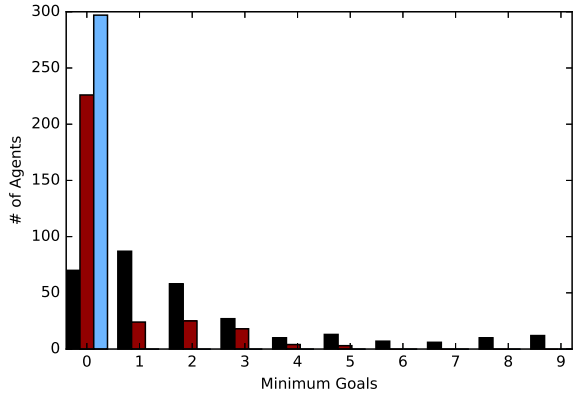

Figure 2.8: **Minimum Performance Distribution for three experimental conditions.** Black: all gates allowed, blue: probabilistic gates allowed, red: only deterministic gates.

2.6.3 Feedback Integration

Agents that were evolved using feedback gates have the potential to change their probability table within their lifetime. By computing the difference between the tables from each feedback gate at birth and after 512 updates, we find that indeed these tables change over their lifetime, but this difference neither increases or decreases over evolutionary time (see Figure 2.9). The difference was computed using the root mean square method. In addition, we find only a very weak correlation between the amount of change and performance (see Figure 2.10).

2.6.4 Computational Networks

When we compare the behavior of the best performing agent using feedback gates with that of the best performing agent that does not use feedback gates, we find one very striking

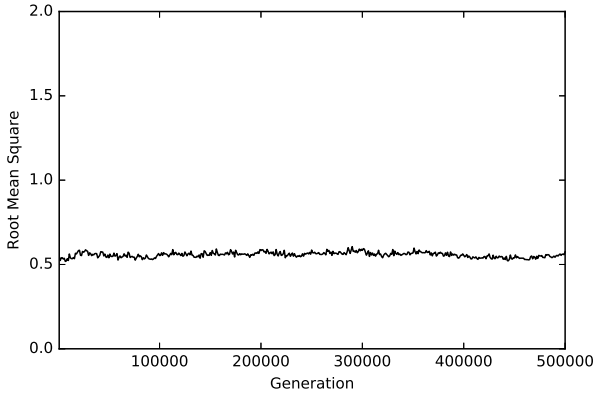

Figure 2.9: **Root mean square difference between probability tables at birth and after learning is over for each generation on the LOD.**

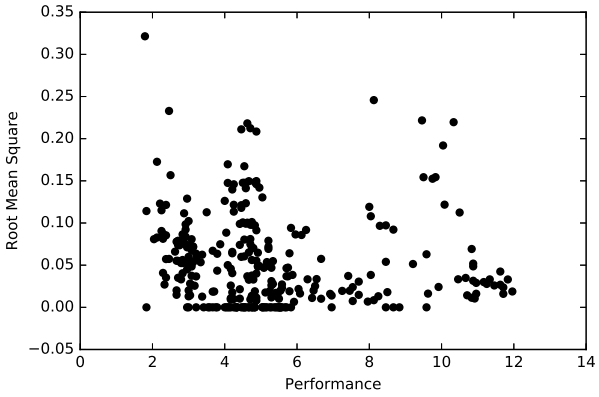

Figure 2.10: **Correlation of change in root mean square to performance at the end of evolution.**

difference. To exemplify this we show the best performing agent without feedback gates (see Figure 2.12) that evolved a complex network of deterministic gates (shown with square boxes) and seem to make use of the hidden nodes of the Markov Brain. The best performing agent that has feedback gates (see Figure 2.11) seems to write to the hidden states less and both output nodes receive a signal from the feedback gate (seen in the octagon).

2.6.5 Performance Plots

Each image (see Figure 2.13) depicts an environment where the cells are labeled using a gray arrow, pointing towards the optimal path. Dark gray boxes are walls or obstacles in

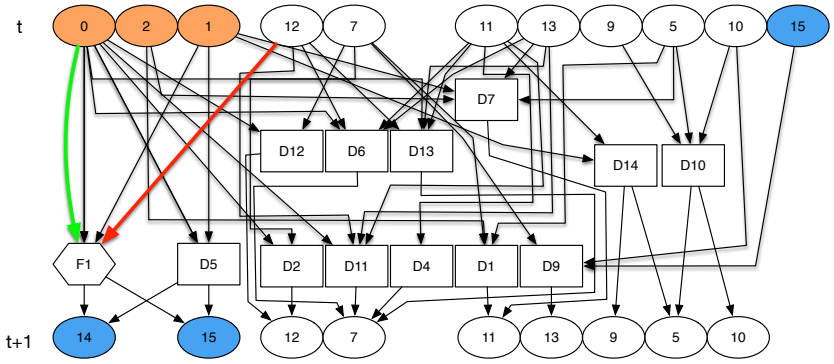

Figure 2.11: **Computational network of best performing agent using feedback.** Input nodes are orange, output nodes are blue, hidden nodes are white. Positive feedback connections are green arrows, and negative feedback are red arrows. Logic gates are rectangular boxes, while the feedback gate is an octagon. The top row of nodes represents the states at time point t, while the bottom row of nodes represents the internal states of the Markov Brain at time point $t+1$.

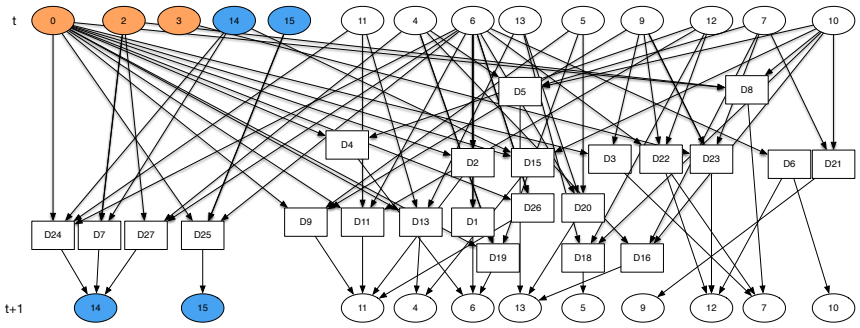

Figure 2.12: **Computational network of best performing agent not using feedback.** Input nodes are orange, output nodes are blue, hidden nodes are white, logic gates are rectangular boxes. The top row of nodes represents the states at time point t while the bottom row of nodes represents the internal states of the Markov Brain at time point $t+1$.

the environment. The red arrows indicate the orientation and location an agent had during its life. Once agents reached the goal, they were put back to a random location 23 steps

away from the goal. The left two images, labeled A and C, depict the path in environment 16, and the right two images, labeled B and D, show environment 21. We see that either evolved agent moves on a straight line without attempting to take turns when they move on a straight line of forward facing arrows. However, the agent using feedback gates makes fewer mistakes when it comes to turns, whereas the agent not using feedback gates struggles to find the correct direction at every turn and reaches the goal less often.

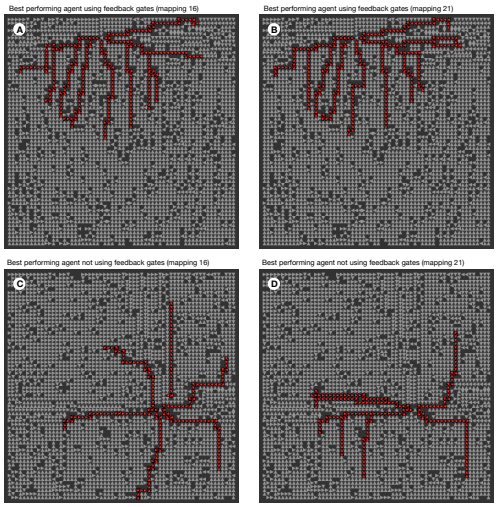

Figure 2.13: **Path comparison of the best performing agent with feedback versus best performer without feedback.**

2.6.6 Average Updates to Goals

For the top 50 performing agents using all three types of gates–deterministic, probabilistic, and feedback gates– we recorded the number of updates it takes to get to the first five goals (see Figure 2.14). This allows us to estimate the progression of learning. We find that the time to reach the first goal takes about 118 steps, to the next one only 54, and finally this trend stabilizes at about 50 steps to the goal.

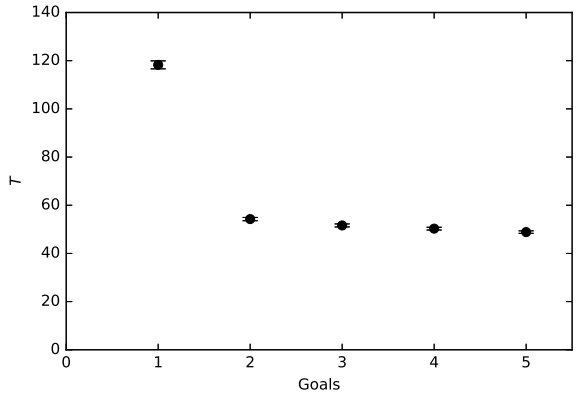

Figure 2.14: **Average updates required to reach first five goals.**

2.6.7 Learning a Single Map

To create a fair comparison between Q-learning and MBs we evolved 300 replicates of MBs to solve one map for 5,000 generations. Here we plot the mean performance over evolutionary time for agents on the LOD (see Figure 2.15). Agents were able to match the performance of the Q-learners fewer than 400 generations.

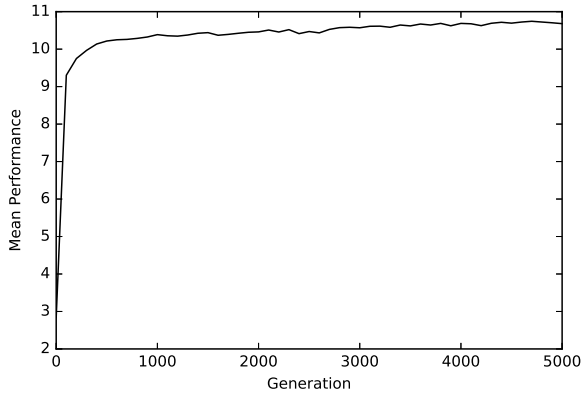

Figure 2.15: **Performance over evolutionary time when evolved on a single map.**

2.6.8 General function of feedback learning agents

Reconstructing the set of logic instructions for each agent is simple, however the algorithm implemented by an agent remains epistemically opaque (See Marstaller 2010). All computations happen in parallel and representations about concepts are spread over nodes

instead of being neatly compartmentalized. This makes it impossible to distill a narrative. However, general statements like the number of nodes or the number of gates are possible. Also, different evolutionary experiments might result in different types of solutions, which further complicates the attempt to create a narrative for the function of an agent. When trying to understand the evolved agents, we found at least one remarkable design feature. Very much like the agents controlled by a heuristic, feedback gates preferred to receive positive feedback from the sensor indicating that the agent now faces forward. It seems reasonable to do that since actions that turn the agent to be facing forward are always good as they can advance to the next tile.

Chapter 3

The Effects of Feedback learning on Generalization and Reversal Learning

3.1 Introduction

One goal of Artificial intelligence (AI) research is to create algorithms that are able to adapt to their surroundings. But to adapt in a truly flexible way may requires applying concepts learned to accomplish one task to a new task not previously attempted. This kind of flexibility is known as generalization. For example, if a hypothetical robot learned to walk from point A to point B in a flat room and was asked to go up a flat of stairs to point C, it would have to use the basic skills of balance and muscle control skills to reach the next floor. There has been extensive research done to develop systems which adapt through their physical presence [121, 122, 123] and recent research has shown how we can adapt the cognitive abilities of the systems [81, 82, 83, 84, 85, 90, 86, 87, 88, 89, 91, 124] (and others).

The cognitive abilities of these systems are often controlled through learning algorithms, which use past experiences to correctly predict future outcomes. Typically these algorithms use information gained from a training set, or a subset of the possible data, to construct a model capable of making accurate predictions. The model is then tested on a different subset of the data, or test set. Generalization occurs when a model is able to correctly predict the test set of data [125, 126]. Similarly, great advances have been made using feedback learning

where an objective external signal informs the learner regardless if the actions were good or bad [127, 128, 46].

However, these machine learning algorithms suffer common pitfalls of *overfitting* and *underfitting* that limit the robots from generalizing to a variety of task [126]. Overfitting occurs when the model produces a low bias with a high variance [129] and is a result of either having a training set that is not representative of the data or using poor feature selection. In nature overfitting would be akin to an organism being able to perform one task extremely well and failing at all others. Underfitting demonstrates a high bias with a small variance [129] and results from not having enough information in the training set to capture a trend. Underfitting would be similar to an organism that performs many tasks but is not highly skilled in any of them. Research often focuses on eliminating the overfitting of an algorithm to a particular task [130, 131, 132]. The same holds true for classifiers or regression models.

Recently we presented a different way to address the problem of creating adaptive machines. Typically, objective external feedback is given to the system, and then the application of this feedback directly affects the system. In nature, however, we find that everything starts with a sensorial percept, that needs to be interpreted so negative or positive feedback can be generated internally by the organism receiving the percept. Loosely speaking, you had to first learn that the phrase "good job" is a positive feedback before you could use it as such. Therefore, we evolved not only an agent to navigate an environment, but also required it to evolve the machinery that interprets the consequences of the actions it takes and then apply feedback internally. This resembles much closer what occurs in nature, breaks with the former paradigm of feedback learning, and raises new questions.

If feedback is not applied directly, but the machine has to evolve the ability to interpret the environment before it can apply feedback, what does this mean to over- or underfitting? After all, feedback is now synthesized by the machine, and thus we have to ask if the machine either evolved to learn only those environments it has seen, or if it evolved the ability to provide general feedback, so that it can learn an arbitrary environment. In many machine

learning algorithms positive or negative reinforcement is explicitly provided to the machine by a external source. Here feedback is not provided by an external source but rather is internally generated from observations of how past actions influenced the environment. This means that the typical concept of positive or negative reinforcement might not apply to machine learning systems anymore.

We can think of the internal feedback that our system uses in terms of the binding or symbol grounding problem seen in nature. In general terms, a binding problem is when the brain must integrate multiple stimuli to form a representation of its environment. Assuming that internal feedback and the binding problem are linked allows us to think about our system in a different way. In a nutshell, our sensors are general purpose machines that receive stimuli which first need to be interpreted before we can perceive a *qualia* and thus a binding between sensorial inputs and external reality must first be computed (For a more detailed introduction see [133, 134]). When providing external objective feedback, as in machine learning, we bypass the binding problem. One could argue that in order to understand cognition (and learning) we cannot simply detour such a fundamental problem but we instead need to integrate it in our models. To this end, MBs evolved to perceive the world, and then also needed to interpret (bind) the environment in order to know whether or not actions were good or bad. However, it still needs to be tested if such change allows for generalization to happen, or if the machinery that interprets the environment and generates the feedback is not prepared to facilitate such change.

Additionally, if a system can still generalize then the question becomes: can they adjust to different environments using reversal learning? Reversal learning is the ability to become trained on a stimulus and after some period the stimulus changes and something else or the opposite needs to be learned [135, 136]. After all to generalize on the system is be able to adapt to an environment; reversal learning is simply using the same principles of adaptation to adjust to the changes. Here we will investigate the extent to which Markov Brains (MBs) [2] are able to generalize from a small subset of test cases and how they will be able to

reversal learn. The MBs were chosen for this investigation since they already have learning capabilities [124].

Evolvable Markov Brains, are networks of computational logic units (gates) that are genetically coded and subject to evolution [1, 72] and have a built in feedback mechanism [124]. Each MB is represented by a sequence of numbers that serves as the genome and contains stretches of numbers that encode computational units similar to how genes encode proteins. These genes determine each gate's type and starting states along with the networks inter-connectivity and connections to their sensors and actuators.

Typically, MBs can contain deterministic, probabilistic, or feedback gates and each gate maps an input i to an output o. In deterministic gates each input correlates to a specific output, while probabilistic gates are coded to map inputs to outputs using a probability table. Since inputs and outputs to gates are binary (0 or 1) each probabilistic gate contains a table of size $2^i x 2^o$ to account for every possible mapping of inputs to outputs. This table is defined by the genome, and can be adapted by evolution. During the lifetime of the agents these values never change. However, the process of learning creates long lasting memories which typically involve the neural substrate to change. Therefore, feedback gates also use a probability table but the values are allowed to change due to feedback during the lifetime of the agent [124].

Feedback gates have been shown to improve an agent's performance within their lifetime using a navigation task that contains a large amount of ambiguity. In this experiment agents are given the task to reach a randomly selected goal. The path to the goal is indicated to the agent as arrows on the floor that instruct the agent to move forward, to turn left or to turn right. However, the agent cannot just turn or move; it is given four actions to choose from. These four actions (A,B,C,or D) map to one of four behaviors (turning 90 degrees to the left or right, doing nothing or by moving forward) and the way they are mapped is environment specific. It is this mapping that the agent has to learn, but instead of receiving feedback of a beneficial or detrimental action, the agent needs to observe the environment, draw the right

conclusions, and apply feedback appropriately. For example, in map one action A will move the agent forward, B turns it left, C turns it right and D does nothing. In map two A results in a forward movement, B turns the agent left, C turns it right and D does nothing (for more details see 3.1). There are 24 permutations of possible action-to-behavior mappings, some of which may be withheld during evolution so generalization abilities can be tested. After the agent has seen all that is allowed during evolution tournament selection was performed to select the agent with the highest fitness to produce offspring for the next generation for 500,000 generations. Through this evolutionary process agents learn how to associate the what action results in which behavior to navigate their environment.

This environment has been used to show lifetime learning [124] and to examine how learning impacts the neuro-correlate Φ [137], which is based on Information Integration Theory [94, 3, 95, 96, 4, 97]. Here we use the navigation task from Sheneman and Hintze 2017 but evolve agents in a subset of the possible mappings, then determine a) if agents are able to navigate novel mappings and, if so, b) the minimum number of maps required during evolution to facilitate generalization. Additionally we will test the agents' ability to reversal learn, by first exposing them to one environment which they must adapt to, then just to move them into a new environment with a different mapping. The question is whether agents can adapt to a new environment once they have learned to navigate in another. We will show that the ability to reversal learn evolves for free and that it evolves without being explicitly selected for.

3.2 Materials and methods

3.2.1 Agent

The Brain of the agents had 4 input sensors, 2 output motors, and 10 hidden states. Each of the four detectable instructions were wired to an input sensor: sensor 0 indicates that the agent should stay in the same place, 1 tells it to go right, 2 go forward and 3 turn left. Each square of the environment contains an instruction that the agent can detect. These

e

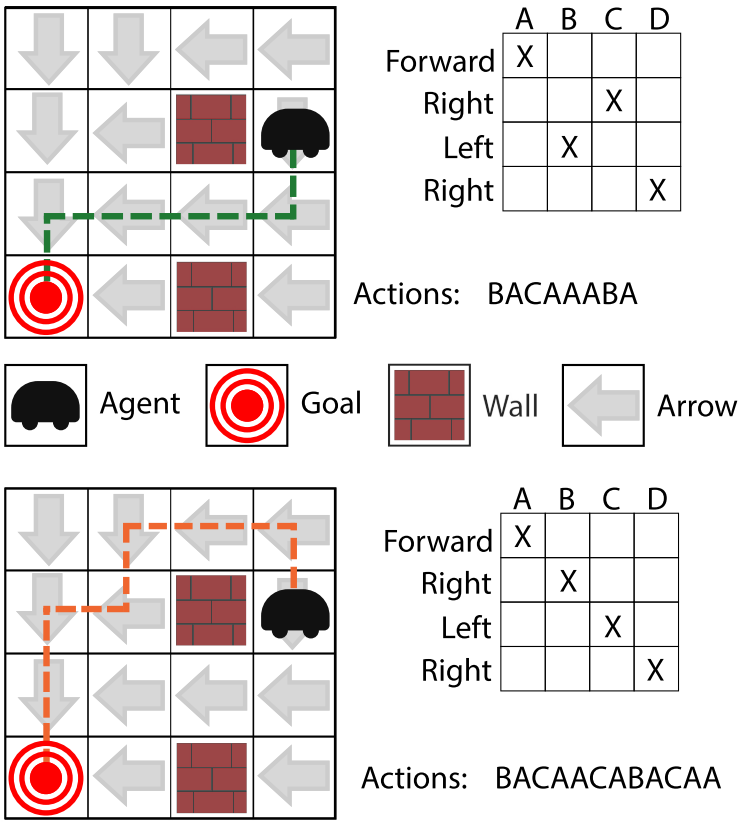

Figure 3.1: **Navigation task overview.** In both panels, the agent (black vehicle) is tasked with navigating to a goal (red bulls-eye). An arrow (gray arrow) is placed in each cell that points to the next cell in the shortest path to the goal. Walls and obstacles (red brick) are present to make the task non-trivial. The action-to-behavior mapping changes is each time in agent is placed in a new environment, and the agent is only able to choose actions which are then mapped to movements. In map one the correct sequence of actions is BACAAABA and in map 2 it is BACAACABACAA. (This figure is derived from Sheneman and Hintze 2017 *The Evolution of Neuroplasticity and the effect on Integrated Information*

instructions are encoded as 0 and 1 which are used as inputs to the sensors of the agent. For example, when a square indicates the agent should go forward next, the agent's input states are 0,0,0,1. The agents control how they move around their environment through two binary motors that output 00, 01, 10, or 11 to indicate what action to take. The binary signals are translated to four possible actions– move forward, do nothing, turn left, or turn right. The way that the binary signals are determined depends on the environment the agent is currently in. For example, in one map go forward might be 01, and in another forward is

encoded as 00.

3.2.2 Feedback Gates

Similar to all other MB gates, feedback gates have inputs, outputs and a probability table that controls the percentage of time an input results in a particular output at any given update. Additionally, feedback gates also have two other connections to detect positive and negative input. When these connections are fired, the probability table for the gate is updated so that the correct output is fired more often then the wrong one. While the internal function of the feedback gates changes within the agent's lifetime, the connections to other nodes within the MB do not change. For a complete description of how these gates work see Sheneman and Hintze 2017.

3.2.3 Environment

Agents were placed in a $64x64$, 2D spatial grid where the grid cells were either were empty or contained an impassable solid block. To keep agents within the environment the grid was surrounded by the solid blocks. Each time an agent was evaluated a new environment was produced where $\frac{1}{7}$ of the cells contained solid blocks creating complex maze-like environments.

A random cell was chosen as the agent's target, and the distances from all remaining empty cells to the target was calculated using Dijkstra's algorithm [92]. An arrow was then placed in each empty cell that pointed to the next closest distance to the target. If two adjacent blocks contained the same distance the arrow direction was chosen to point to one of these randomly. Agents were randomly placed in a cell with a Dijkstra's number of 32 and face a random direction (up, right, down, or left).

The agents were then allowed 512 updates to traverse the environment. If they reached the target they were relocated to a new random start cell with a Dijkstra's number of 32 and faced a random direction. There were four binary output states an agent could produce–

00, 01, 10, or 11 – which were translated via a mapping function to four possible actions- move forward, do nothing, turn left, or turn right. Observe that there are 24 combinations of action-to-behavior mappings that an agent could be placed in. Agents were exposed to a subset of the 24 possible mappings during evolution.

Performance was obtained by testing how often the agent reached the goal in 512 updates within each mapping of the subset. The agent was rewarded proportional to their distance to the target (d) each update and received a bonus for reaching the target (b), making the function for fitness (W):

$$W = \prod_{n=0}^{n<24} ((\sum_{t=0}^{t<512} \frac{1}{1+d}) + b) \tag{3.1}$$

3.2.4 Selection

After each generation in an evolutionary run 100 agents were tested on a set of action-to-behavior mappings, tournament selection was used to select the population of the next generation. In tournament selection individuals were selected at random and the agent with the best fitness was subjected to mutation then placed in the next generation [119]. Here the tournament size was five.

3.2.5 Mutation

The genomes of the first generation of agents were randomly generated with $5,000$ genes and contained 12 start codons for deterministic, probabilistic, and feedback gates. The next generation consisted of agents that propagated from the genome inherited from its ancestor. All genomes maintained a length between $1,000$ and $20,000$ sites where each site had a 0.003 chance to be mutated. Genomes that were not at the maximum size, randomly a selected region of between 128 and 512 nucleotides was cloned and inserted into a random location with a 0.02 likelihood, allowing for gene duplication. All genomes above $1,000$ nucleotides had a 0.02 chance of a stretch of a randomly selected of length between 128 and

255 nucleotides to be deleted from a random location.

3.2.6 Line of Descent

A random agent from the final generation is selected and its ancestory is traced to the first generation to determine the line of descent [120]. The most recent common ancestor (MRCA) is found during this process allowing for mutations swept the population to be recorded.

3.2.7 Mutual Information

Mutual information [138] quantifies how much information is shared between two random variables. It can assess how much one variable is capable of predicting the other. Here we use mutual information to gauge how much predictive power the input and output to a logic gate have about each other. Here all input states are characterized by the random variable S_i and all output states are defined by the random variable S_o. A logic gate that randomly returns outputs that are independent of inputs should have no mutual information between the inputs and outputs, whereas a deterministic logic that maps an unique output to an unique input has perfect mutual information between S_i and S_o. Mutual information was calculated using:

$$I(S_i; S_o) = \sum_{o \in S_o} \sum_{i \in S_i} p_{io} \log(\frac{p_{io}}{p_i p_o}) \tag{3.2}$$

k-means

To analyze the performance of agents a derivation of k-means clustering [139] was used. The k-means classification algorithm divides the data into k partitions. The algorithm uses k centroids, computes the distance of all data points to all centroids, and assigns each data point to the cluster defined by the closest centroid. Afterwards, the positions of the centroids are recomputed as the mean positions defined by all data points assigned to each cluster.

This is repeated until the centroids stop moving. For this experiment the centroids were predefined so that they were defining specific classes of behavior. These behavior classes were based on how agents performed on two sets of mappings– familiar and novel– when they were allowed to use feedback and when they were not. Agents were categorized into the clusters accordingly. Each distance from an agent behavior vector to predefined centroid was computed and the centroid with the shortest distance defined the class. Since centroids were not moved in this procedure the initial centroids define each cluster.

3.3 Results

Many natural organisms use feedback about past actions to improve their performance on a novel task. Here agents evolved to decode environmental cues to correctly control their motor outputs in order to reach a goal, as seen in Sheneman and Hintze 2017 [124]. To test the agents ability to generalize we evolved 100 replicates for 500,000 generations where at each generation agents were exposed to a subset of the 24 possible mappings. During analysis we exposed organisms from the line of descent (LOD; see materials and methods) of each of the replicates to all 24 environments. Performance was averaged over five replicate transfers. This allows the performance of the agents to be accurately evaluated given that environments are randomly generated at run-time. Prior research has shown that agents evolved in this navigation task demonstrate a wide distribution of behavior. Some agents fail to perform, some evolved the ability to take full advantage of feedback gate and learn to navigate environments quickly. A third observed behavior was a heuristic which allows agents to somewhat navigate environments equally poorly. Since we are interested in agents who are able to exploit patterns encountered during learning we rank the agents by their average performance and use the top 20 agents for most of our analysis. These 20 agents have a high average performance and evolved to use feedback gates; agents with lower performance are removed since they might have some bias or systematic error keeping them from performing optimally.

As seen in Fig 3.2, agents were able to solve the task in different ways based on the number of maps they saw during evolution. Agents shown 1 to 3 mappings during evolution reach the goal ∼12 times on average when placed in familiar mappings (i.e. those seen in evolution) and ∼1 on novel mappings. It stands to reason that agents are able to memorize the small subset of mappings seen during evolution. These agents are able to perform using only hidden states to memorize the familiar mappings and thus would not see a performance decrease when feedback is suppressed. In fact, we find that agents exposed to only 1 mapping rarely evolve feedback gates (4 out of 100 replicates). This trend increases when agents are exposed to at least 2 maps (in 36 out of 100 experiments agents incorporate feedback gates into their brains). After agents see three mappings, we find the majority of experiments to result in agents incorporating feedback gates into their cognitive architecture (see Table 3.1).

Maps in Evolution	**All Replicates**	**Top 20 Replicates**
1	4	3
2	36	20
3	84	20
4	86	20
...		
24	94	20

Table 3.1: **Number of agents who evolve feedback gates for each subset of maps seen in evolution.**

When agents are shown 4 to 16 maps during evolution their ability to reach the goal in the mappings seen during evolved lessens while they are able to reach the goal more on the new mappings. Agents that see more than 4 maps begin to show behavior consistent with an ability to generalize. Specifically, these agents are able to perform reasonable well on unfamiliar mappings. The explanation for this is that the agents are using a heuristic to distinguish between mappings they saw during evolution. Though agents can still reach the goal in novel mappings using a heuristic they follow a step-wise procedure and do not learn from their experiences. This procedure results in the agents taking longer to reach the goal, thus demonstrating reduced performance on novel mappings. It is possible that these

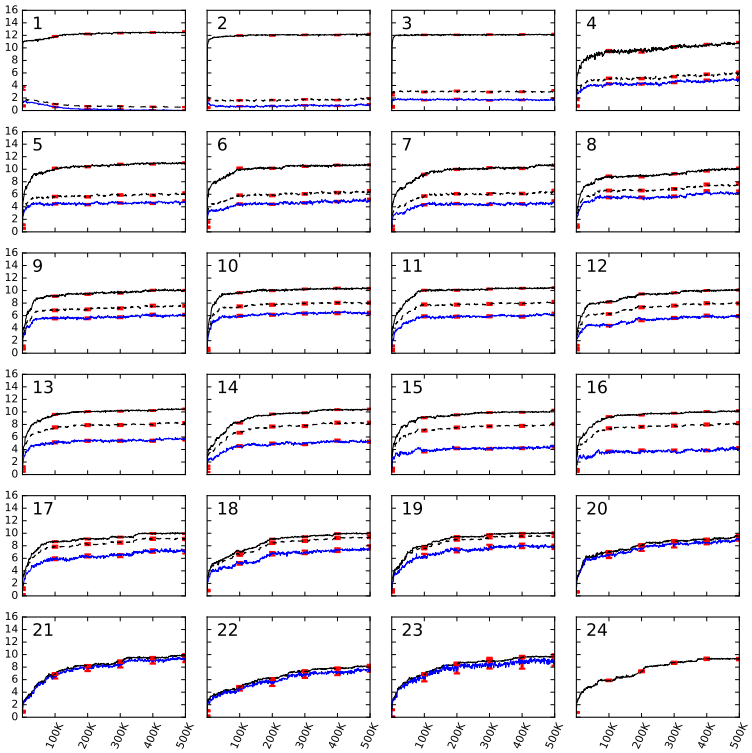

Figure 3.2: **Agents ability to generalize given the number of maps seen in evolution.** Each subplot represents the number of maps agents saw with during evolution. The solid black lines represent the average number of goals reached (W) on maps previously seen by the top 20 of 100 replicate experiments. The blue line shows the W achieved on maps the agents were not shown during evolution. The dotted line shows the average W in all 24 environments. Agents who see 24 mappings during evolution are not presented any novel mappings.

heuristics are difficult to evolve using only hidden states so agents who incorporate feedback gates emerge during evolution– a theory supported by Table 3.1.

Agents shown 17 to 24 mappings obtain an average performance of ∼8 on familiar maps and this performance never deteriorates by an average of more than ∼2 goals when presented with never before seen mappings. These agents perform well on all types of mappings which indicates that they are no longer using a heuristic to solve the task rather they are learning within their lifetime.

The performance increase between the agents who saw 16 mapping and those who saw 17 indicates a change in the method used to determine the agent's current environment. One possible explanation for this that the agents who see less than 17 mappings during evolution

have not evolved the connections necessary to use the feedback gates. This is supported by the finding that within the top 20 performers who were shown 2-24 mappings all contain feedback gates (see Table 3.1 top 20 replicates).

Not all gates contained in a MB contribute to the changes in the agent's behavior, thus simply evolving a feedback gate does not mean that feedback is actually used to help the agent solve the task. To test if the feedback gates are being used we can measure the change in the amount of information, i.e. mutual information, the gates inputs convey about their output from birth to task completion. A change to the probability that helps an agent solve the task would result in a positive change in mutual information, whereas a detrimental change would result in a loss of mutual information. Figure 3.3 shows agents who see a subset of 4 mappings during evolution began to gain mutual information during their lifetime, which is the same subset where feedback gates start to evolve (as seen in Table 3.1). Agents who see more mappings during evolution display a larger change in mutual information, and when paired with Figure 3.2, suggests an increased ability to generalize. We find a strong correlation ($r = 0.684$) between the gain in mutual information and an agent's ability to perform on novel maps (See Figure 3.4).

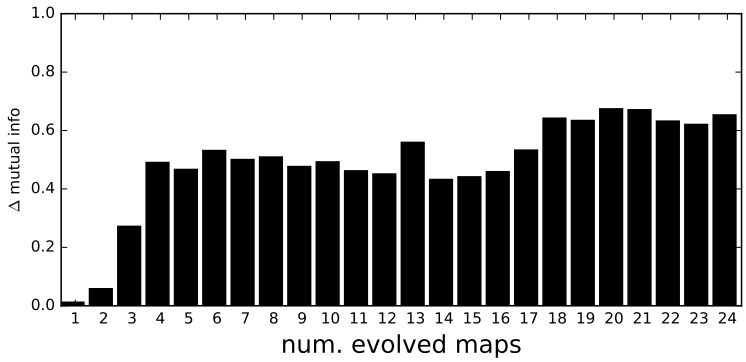

Figure 3.3: **Mean delta mutual information by subset of mappings seen in evolution.** Each bar represents the mean change in mutual information conveyed from birth to after task completion for 100 replicates by subset of mappings seen in evolution.

Sheneman and Hintze 2017 [124] observed that agents can develop multiple strategies to solve the navigation task when all 24 mappings where seen during evolution. Under the

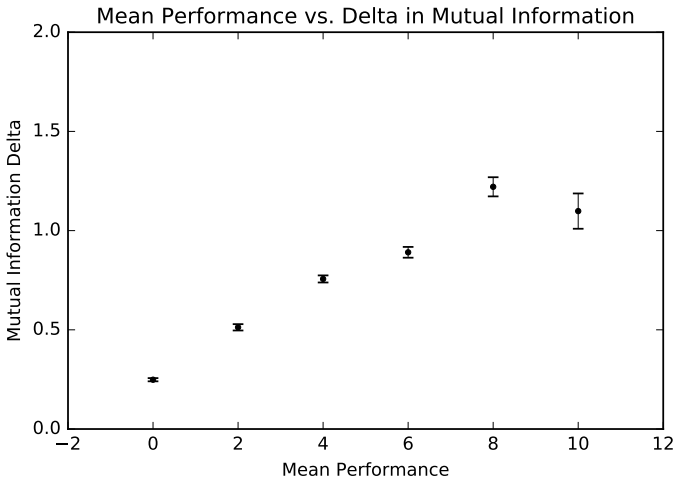

Figure 3.4: **Mean delta mutual information by levels of performance on novel maps.** The gain in mutual information ($\bar{\Delta}$) is shown on the y-axis for categories of performance on novel mappings. All replicates for the 23 conditions where a novel mappings were available for tests were placed in six bins ranging from a performance of 0 to the maximum performance of 12 with a bin size of 2. Variance is represented by error bars.

same principle, we expect evolving agents on a subset of the mappings would result in the emergence of three different strategies– memorizing, advanced heuristics, and learning– due to the changes in task complexity. To test this we recorded the agents performance on mappings seen in evolution and novel mappings each under two conditions: when allowed to use feedback and with feedback disabled. Agents who are simply memorizing the mappings they are presented would perform optimally on familiar maps regardless if they are allowed to use feedback or not. However, when they are placed in a novel map the agents would fail to adapt to any feedback thus perform poorly both with feedback on and with feedback off. When agents are using an advanced heuristics they find the goal through a constant series of trials and errors designed to identify which of the familiar maps they are in. The path these agents would take to reach the goal would be sub-optimal each time, which results in fewer goals reached than an agent who has memorized the mappings. Further, disabling the feedback gates in these agents would also lead to greater reduction in performance. Agents who are learning have a high performance on all mappings when using feedback but experience a significant decline in performance when feedback is disabled.

We expect that the agents who use memorization to solve the task emerge from evolution when a low number of mappings were seen in evolution since they are presented with a low complexity task. However, agents who learn emerge when they see many mappings because they must develop a way to quickly distinguish which mapping they are in to perform well. The agents who rely on heuristics would evolve in the intermediate number of mappings are seen. Using k-means clustering we find that the 100 agents who see only one mapping memorize the mappings (see Figure 3.5). When looking at all 100 replicates that saw four mappings during evolution 38 agents memorize the mappings, 33 solve the task with heuristics and 29 agents learn the mapping they are in. By the subset of 19 familiar mappings, agents use heuristics in 67 replicates and use feedback gates to learn in the other 37 replicates.

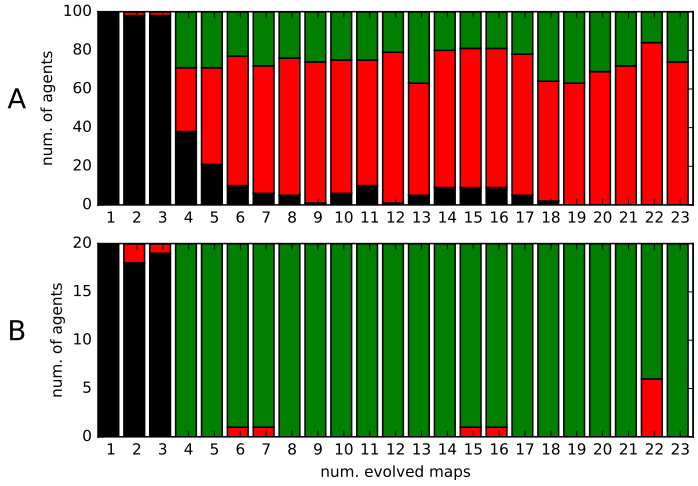

Figure 3.5: **Number of agents in each performance cluster.** Black bars represent the number of agents who memorize the maps. While red bars represent agents who use heuristics to solve the task and the green bars represent the number of agents who use learning to solve the task. Panel A represents all 100 replicates for each subset of mappings seen in evolution, while Panel B shows the 20 top preforming agents.

As previously mentioned (see Figure 3.2) agents who see a subset of 17 or more mappings maintain a high performance level regardless if they have seen the mapping before or not. Artificial intelligence researchers are seeking a solution that allows robots to adapt to a

changing environment. Since the agents evolved for this task can learn how to interpret novel mappings, is it possible they can learn one environment and then reversal learn to a different environment.

We explore the agents ability to reversal learn by measuring performance in one mapping for 512-steps and then switching to a new mapping and measuring performance in an additional 512-steps. When analyzing at all 100 replicates in each subset of mappings seen by evolution, Figure 3.6 Panels A and C we find that agents who are not simply memorizing the environments seen in evolution are able to adapt to a new environment (those who saw a subset of 1 to 3 mappings during evolution), achieving between 1 to 4 goals in the second mapping. We also found that agents who are better generalists, i.e. the top 20 performers, are worse at reversal learning (see Figure 3.6 Panels B and D), i.e. reached the goal a maximum of 2 times in the second mapping.

3.4 Discussion

In the past, Markov Brains have been used to study how populations evolve so that agents have the ability to solve a specific task or set of tasks. In Sheneman and Hintze [124] agents evolved the use feedback gates which allowed them to use the results of past actions to influence future decisions. These agents were capable of solving the same complex navigation task we used here, but were exposed to all 24 possible mappings during evolution. Through lifetime changes of their probability tables they were able to determine which environment they were placed in and navigate to the goal. Here we used the same navigation task, but restricted the number of mappings agents saw during evolution. Through a series of experiments that controls how many variations of the environment the agents are exposed to, we evolved and tested agents who are able to both generalize to learn a novel mapping within their lifetime and reversal learn.

In order to generalize to perform novel tasks agents must be exposed to a minimum number of possible mappings during evolution. We find that after 17 (out of 24) environ-

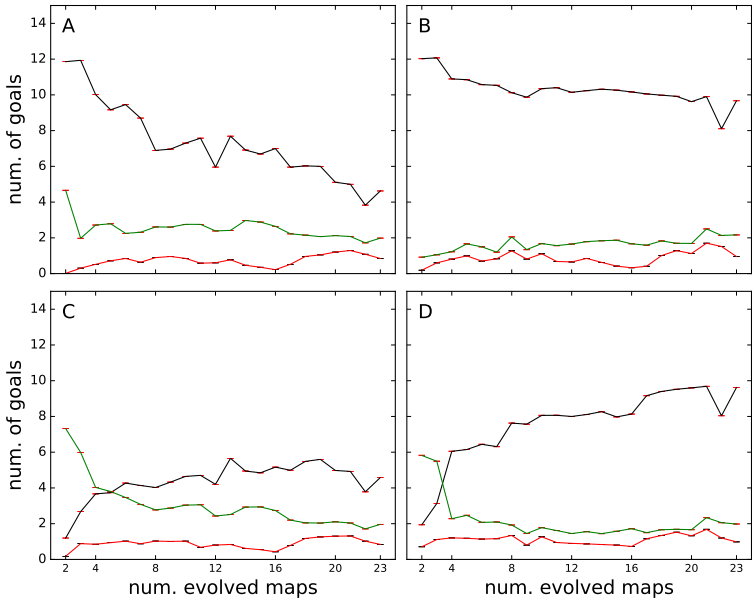

Figure 3.6: **Agents ability to reversal learn.** Panel A shows the mean performance as the number of goals reached as the average over of all 100 replicates when first shown a mapping they saw during evolution. The black line indicates the number of goals reached in 512-steps in the first environment they see. The green line shows the number of goals reached when the second environment seen was also seen during evolution and the red line is the number of goals reached when the second environment is novel. Panel B is a subset of the top 20 performers given this treatment. Panel C shows the mean the number of goals reached of all 100 replicates when shown a novel mapping in the first environment they see. The green and red lines represent the same treatments as panel A. Panel D is a subset of the top 20 performers in panel C.

ments agents generalize to the remaining 24. It seems obvious that there is some critical threshold of experiences that organisms must be exposed to before they generalize. We found that agents who see as few as 4 of the 24 different environments during evolution deploy a complex heuristic to reach the goal in novel environments, and do not manage to learn a new mapping. Most of these agents have evolved and are using feedback gates but they still require continual trial and error to reach the goal without learning from their mistakes in the long run. Agents who saw at least 17 environments during evolution show only a marginal loss of performance in the novel environments, while those who experienced fewer than 17 environments drastically lost performance. We also observe this in nature when training dogs to detect explosives [140]. When the dogs were exposed to a single component

of the explosive they were not able to find the explosives. However, when exposed to multiple components separately they were able to use olfactory generalization to find explosives containing mixtures.

Throughout all conditions we find agents to evolve Brains that incorporate feedback gates, which allow agents to update their probability tables based on experience. However, only when agents had the ability to generalize or to reversal learn, did blocking the feedback signal lead to a loss of function. Therefore, we argue that feedback gates were used opportunistically in agents who were not able to generalize or reversal learn. Though these agents possessed feedback gates, the feedback mechanism did not play an important role in their performance. Once the agents needed to evolve the ability to generalize feedback gates were used including their feedback mechanism. In this model the threshold to evolve generalization was high (17 out of 24 environments). Other systems will have such thresholds as well, but their exact limit will probably differ.

We also find that agents evolve the ability to reversal learn as a byproduct of the generalization ability. However, agents who generalized extremely well were not as effective at reversal learning. We believe that in order to produce perfect reversal learners the environment must select for the ability to reversal learn. A future extension of this work should test if presenting the agent with mappings that change during the agents lifetime would indeed improve the ability to reversal learn.

3.5 Conclusion

The experiments presented here show that agents controlled by Markov Brains which can use feedback gates evolve the ability to generalize and to reversal learn if the variability experienced during evolution is high enough. In the context of machine learning, the system is no longer learning the environments that it sees, but is rather evolving the ability to collect and apply general feedback. The implication is that they systems can now learn an arbitrary environment as opposed to being trained to perform a specific task.

In the task set presented here the system needs to see about $\frac{2}{3}$ of all training cases before generalization is evolved. These findings suggest that natural organisms' ability to generalize new concepts require a minimal amount of experiences to extract common concepts. However, it is not clear to what extent natural organisms need a similar ratio to evolve the ability to generalize their learning ability to other objectives. In addition, we established that some degree reversal learning is a by-product of the evolution. Future improvements of the feedback learning mechanism in MBs might reduce that ratio further, which will shed more light on the generalization in nature.

All this work was performed within the same task domain. It is entirely possible that generalization would work also across task domains, which needs to be tested in the future, but this requires us to show within task domain generalization before this question can be addressed. For now, we showed that Markov Brains equipped with feedback gates can generalize from a small subset of examples to the whole set, and by circumstance evolved the ability to reversal learn as a bonus.

Chapter 4

The Evolution of Neuroplasticity and the effect on Integrated Information

4.1 Introduction

Information integration aims to measure a system's ability to make a decision by combining information obtained from multiple environmental sensors with knowledge of past outcomes. Imagine a child playing soccer for the first time trying to score a goal. To kick the ball she must visually locate the ball and determine how to make contact with her foot to move the ball in the desired direction. The cognitive structure she has as a child is drastically different from that of the professional soccer player she might become. We assume that the amount of information integration is different for the child and the professional because the neural structures are different [96], but we are unsure if the amount of information integration increases or decreases with neural development. The child's cognitive system might need to process a lot more information while the professional's cognitive system is already streamlined by training to the point that kicking goals becomes a near unconscious reflex. Alternatively the professional, being much more accurate and having more experiences, might integrate more information than the child. Here we investigate which assumption is correct using computationally evolved artificial animats that can change their architecture through lifetime learning.

The neuro-correlate Φ is used to quantify the amount of information integration a cognitive system performs. Since the introduction of information integration theory [96, 94, 95, 97], it has been shown that the amount of integrated information a system has increases over the course of Darwinian evolution [1, 74]. Further we have established that task complexity defines a lower bound of Φ [74] for the task to be solved and that more complex environments require an ever increasing amount of Φ [77]. These observations were made using computationally evolved animats (others use the term agent which is synonymous) controlled by a Markov Brain (MB) [1, 74, 77, 79].

MBs are networks of computational logic units (which from now on we will refer to as gates) that are genetically coded and subject to evolution[1, 72, 77, 73, 76, 74, 78, 79, 75] (For a detailed description of Markov Brains see `https://arxiv.org/abs/1709.05601` [2]).

The MBs contain deterministic and probabilistic gates each of which can be thought of as a $2^i \times 2^o$ state transition table, P, where i is the number of inputs and o is the number of outputs . Probabilistic gates are coded so each input is associated with a vector of probabilities that determine the likelihood for each possible output to occur. Deterministic logic gates work the same, except that their vector contains a single probability of 1.0 and the rest are 0.0. Each MB is represented by a sequence of numbers that serves as the genome and contains stretches that encode computational units similar to how genes encode proteins. These genes determine the number, type and if needed as in probabilistic and feedback gates the probability tables. In addition, each gene contains information about how gates are connected to each other and how they interface with sensors or actuators.

To this framework, we added feedback gates, which are similar to probabilistic gates but change their probabilities within their lifetime using the success or failure of previous executed actions. For a detailed description see Sheneman and Hintze 2017 [124]. In essence, feedback gates have two additional inputs for positive or negative reinforcement. At each update, a feedback gate records which output was mapped to the current input. In the case of positive feedback, the probability that lead to this mapping becomes increased, in the case

of negative feedback this probability becomes decreased. After that, the probability table is normalized so that the sum or each row of the probability table sums to 1.0. The strength of the feedback as well as the number of time points the feedback gates record is evolvable as well, and encoded in the genes describing each feedback gate. Observe, that the feedback itself is not supplied externally from a supervisor, but that that MB itself has to evolve a mechanism to detect which actions have a positive or negative outcome. Based on this, MB then have to apply feedback to their feedback gates internally.

A key component of evolving MBs is that they interact with their environment which requires that they be embodied [80] as an animat. These learning animats evolve to use an internal feedback mechanism to change the function of feedback logic gates. As a consequence, we can now compare animats that are naïve about their task with those that have experienced the task before. These experiences change the neural substrate and thus Φ can change as well.

Animats can store representations about the environment in states of their MBs that are not observable to the outside world [72], i.e. hidden states. So far, when evolving animats, Φ was always measured over the animat's lifetime using the information about the environment that was stored as brain states, or by evaluating their brain directly, which do not change in topology or function over their lifetime.

However, natural brains change in many different ways. The transition from wakefulness to sleep or anaesthesia has the most profound effects on information integration [141, 142]. Neurotransmitters are chemical messengers that can modify the function of the brain, halt functions, and change the function of the brain, disrupt functions, and signal the need for unconsciousness rest, which prevents the brain from integrating information. Similarly, patients in a coma do not integrate information [143]. A more subtle type of change can be the neural development, thus the ability to integrate, that occurs during the transition from an juvenile to an adolescent or senescent brain, or even simpler changes such as learning. Modeling sleep, unconsciousness, or development in a digital animat seems challenging, but

a recent development in MBs allow animats to learn [124]. Here we use learning animats to study how neuroplacsticity affects Φ.

In prior work, we evolved animats who learn to solve a navigation task (see Figure:4.1) and we can now use these animats to investigate the effect of learning on Φ. To evolve these animats we placed them in a 2D lattice (64 x 64 cells wide) environment whose outer edges and a small fraction of inter cells contained walls. From the remaining cells we selected a random goal and computed Dijkstra's path [92] to the goal from all other cells. Each of the cells on the path then are assigned an arrow that points to the next cell on the path to guide the animat's navigation. The animats were then randomly placed a set number of steps from the goal (here 32 steps) and oriented in a random direction. The animat has 512 updates to reach the goal as many times as it can.

To navigate to the goal the animat chooses one of four actions (A,B,C,or D) which maps to one of four behaviors (turning 90 degrees to the left or right, doing nothing or by moving forward). There are 24 possible ways the set of actions can map to the behaviors and every time an animat is placed in a new environment the animat is shown a different combination. For example, in one map the action A may result in a left turn, B in a right turn, C does nothing and D goes forward. In the next map the selection of A moves the animat forward, B does nothing, C leads to a left turn and D leads to a right turn. Once each animat in the population had experienced all 24 combinations tournament selection was preformed to determine which animats reproduce. Over 500,000 generations animats evolve the ability to properly navigate their environment. In order to do so, they needed to learn over a trial and error phase, how actions mapped to turning and forward locomotion, so that they could reach their goal in all possible environemnts.

Using animats evolved in this environment we can now measure the amount of Φ animats have over their lifetime. The method used to measure Φ has changed over the history of information integration theory. The latest version of Φ^{Max} [77] seems to be the most appropriate method to measure the amount of information integration because this version

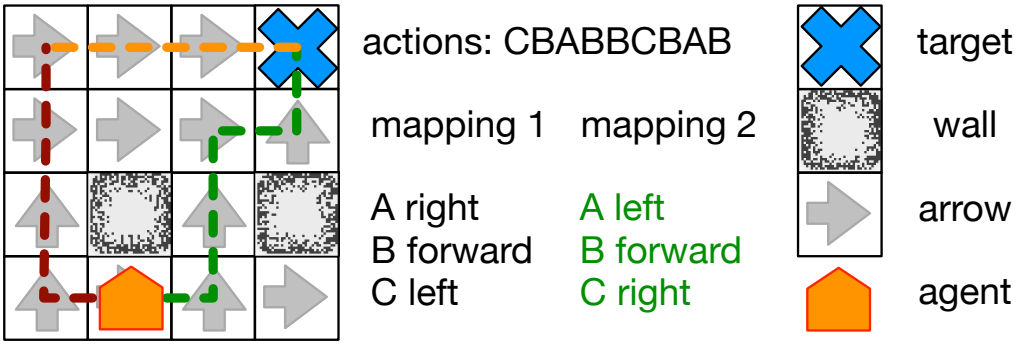

Figure 4.1: **Overview of the navigation task.** An agent (orange) has to navigate towards a goal (blue cross). Each cell contains an arrow (gray arrow) indicating the shortest path towards the goal, circumventing obstacles and walls (black/gray blocks). However, the agent has no definite control over its body and can only choose actions which are mapped to movements. An example sequence of action, as seen in the figure (CBABBCBAB) might lead towards the goal (mapping 2 green path) while given a different mapping (mapping 1 dark red path) it might lead the agent astray. Assuming the agent can integrate the information it collected while navigating erroneously (dark red path) it could learn the correct mapping and proceed to the goal (orange path).

can measure Φ at a particular moment in time. However, Φ^{Max} requires all components to be solely determined by their inputs. The probabilistic and feedback gates we use here violate this requirement due to *instant causation* which requires that each gate have exactly one output (for a much more detailed explanation see Schossau et al. [144]) and therefore prevent us from using the latest version of Φ [144]. In order to avoid overestimating Φ, which is also called the effective information EI. EI is defined in Equation 4.1 where X is a random variable at time points, P is a partition of the system whose states are compared across time. Note that for each time point t the state of all k parts P must be known in order to compute the entropies used here. Earlier versions searched through all possible partitions of the system to find that combination of elements that integrates the least amount of information [1].

$$EI(X_0 \to X_t|P) = \sum_{i=1}^{k} H(P_0^{(i)}|P_t^{(i)}) - H(X_0|X_t) \tag{4.1}$$

The complexity of the Φ calculation depends on the number of nodes in the system and

can be approximated as $O(B_n N^2)$, where B_n is the Bell number of n, which scales super exponentially. To put this in perspective, with current technology, assuming one million partitions per second, it takes about one month to compute Φ for a system of 16 nodes. Here we would need to repeat this for 512 times per animat (once per update), and we would need to do this for 50 animats, which prevents us from pursuing this approach. Fortunately Φ_{atomic} (see equation 4.2 and 4.3) assumes that each component of the system belongs to its own partition, which reduced the problem to a complexity of $O(n^2)$ (for a more detailed description of Φ_{atomic} see [1]).

$$SI_{atomic} = I(X_t : X_{t+1}) - \sum_{i=1}^{n} I(X_t^{(i)} : X_{t+1}^{(i)}) \tag{4.2}$$

$$I_{total} = I(X_t : X_{t+1}) = H(X_t) - H(X_t|X_{t+1}) \tag{4.3}$$

At the same time it is a very good correlate to more complex versions of Φ. In prior experiments [1, 74], Φ_{atomic} has been calculated using all the animat's brain states over its entire lifetime, and not for specific time points. Here, we compartmentalize the lifetime of an animat into different phases using milestones, record the brain states in each phase separately, and are thus capable of measuring changes to Φ_{atomic} within its lifetime.

4.2 Results

Animats were evolved to perform a navigation task where they had to use environmental clues to determine the correct motor outputs as seen in Sheneman and Hintze 2017 [124]. Given the difficulty of the task not all replicates produced animats that performed well at the end of evolution. To isolate animats capable of learning we tested the same 300 animats that were evolved for the previous original navigation task again. This time, to assess their performance more accurately, they were exposed to each of the possible 24 mappings five times. The animats start 32 steps from the goal. Only 45 of the 300 replicates reached the

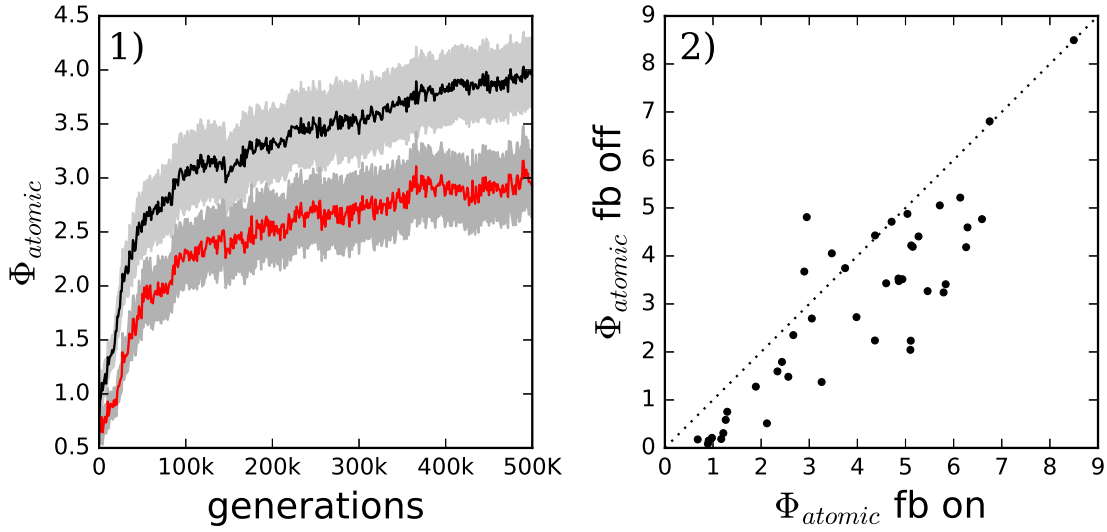

Figure 4.2: Φ_{atomic} **over evolutionary time.** Panel 1) in black Φ_{atomic} on the line of decent, measured for animats using feedback. In red Φ_{atomic} for the same animats, but no feedback applied. The gray background indicates the standard error. Panel 2) shows the last generation of the same data measured with (fb on) and without feedback (fb off) plotted against each other. The dashed line indicates where no change in Φ_{atomic} would occur.

goal an average of five times or more. From these 45 elite performers, all agents on the line of decent (LOD, see Methods for details) were tested in the same environment to analyze Φ.

4.2.1 Effects of Evolution on Phi.

Earlier experiments with Φ_{atomic} showed that Φ increases over evolutionary time [77, 1, 74, 79], which we observe here as well (See Figure 4.2 panel 1). While Φ_{atomic} increases rapidly over the first 100,000 generations, it eventually arrives at $\sim$4.2 after 500,000 generations, when evolution was stopped. When analyzing the last agent on each replicates LOD, we find that Φ_{atomic} drops in almost all cases when feedback is disabled as expected (See Figure 4.2 direct comparison with regular performance panel 2, and over evolutionary time red line panel 1).

In addition to the increase of Φ over evolutionary time, it was observed that performance defines a lower bound on Φ [74]. There is a point where animats cannot evolve better performance unless Φ also increases. We make the same observations here (See Figure 4.3). The

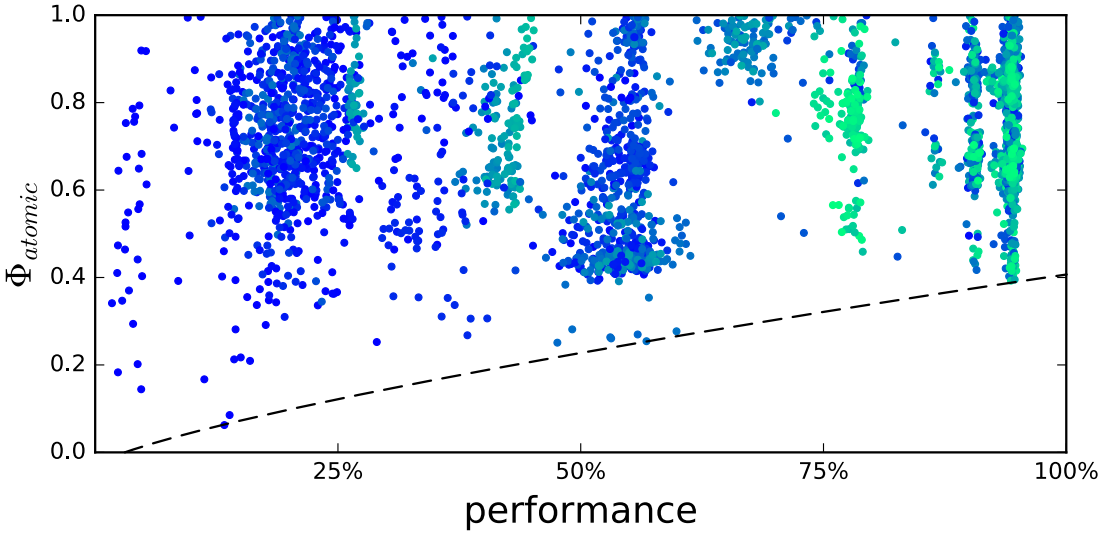

Figure 4.3: **Lower bound relation of Φ_{atomic} vs performance.** For all best animats performance and Φ_{atomic} was recomputed over 5000 time steps for all 24 possible mappings to improve the accuracy of the measurement. The performance was normalized to the maximum performer (animats reach the goal at most 400 times over 5000 time steps). All data points are from the line of decent and are color coded so that early generations are shown in blue and sequentially turn green the further evolution progresses. The dashed line was added to illustrate the lower bound relation between Φ_{atomic} and performance. Values for Φ_{atomic} higher than 1.0 were not shown, since only the lower bound is of interest here, and the data points above 1.0 do not effect our conclusions.

correlation between the performance and Φ can also be exploited to accelerate evolutionary adaptation [79]. These tests confirm that the learning-navigation task performed here also increases over evolutionary time and with performance.

4.2.2 Effects of Learning on Phi.

The addition of feedback gates to MBs allows these brains to not only change from generation to generation, but to also change during their lifetime. This allows us to test how these lifetime changes (learning) effect Φ. To analyze the effects of learning on Φ we need to measure Φ at various time points within an animat's lifetime. A large amount of time points are needed to establish an accurate measure of Φ_{atomic} [96], here we require a minimum of 10,000 data points per measurement of Φ_{atomic} to ensure accuracy.

In the navigation task animats were placed in an environment for 512 updates, not resulting in the required 10,000 measurements. Further subdividing these 512 time steps would result in a highly inaccurate calculation. To compensate for lack of precision, we sample the animats' brain states multiple times in the same environment resulting in a sufficient number of data points. We further define intervals within in the agent's lifetime. Interval boundaries are defined by the time points where agents reach goals. Animats are placed at new random locations after successfully finding the goal, which makes this moment a natural point to subdivide the lifetime. For each interval Φ_{atomic} was calculated separately.

As a result, we can now quantify Φ_{atomic} for each phase of an animat's life in such a way that we can accurately measure the information integration at each milestone. We use the Φ_{atomic} computed when feedback is turned off as the amount of Φ contained at the beginning of the experiment. Figure 4.4 shows that when no feedback is used animats have a Φ_{atomic} of ~ 2.5. The first and second time the goal is reached Φ_{atomic} peaks at $\sim$3.52 then very slowly decreases each sequential time the goal is reached. In this environment animats learn the most when navigating to the goal the first time [124]. The sharp increase in Φ_{atomic} at goal one combined with the knowledge that the animats learn the most on their path to the first goal indicates that animats integrate more information as they learn. A drop in Φ_{atomic} due to learning would have indicated that the cognitive process controlling the actions becomes more streamlined and there is less integrating. Here however, we find evidence for the opposite, suggesting that learning causes the system to integrate more information, and that this increase in Φ_{atomic} allows for better performance. A positive correlation of 0.500 between changes in Φ_{atomic} and performance due to learning supports that notion.

Over the course of evolution animats evolve the ability to navigate their environment. They do this by first incorporating feedback gates in their cognitive architecture and then learning to use them. Animats who maximize the utilization of their feedback gates perform better, and thus can increase the difference between performance with and without feedback. This performance differential can be defined as $\Delta_{performance}$ as seen in Equation 4.4 where

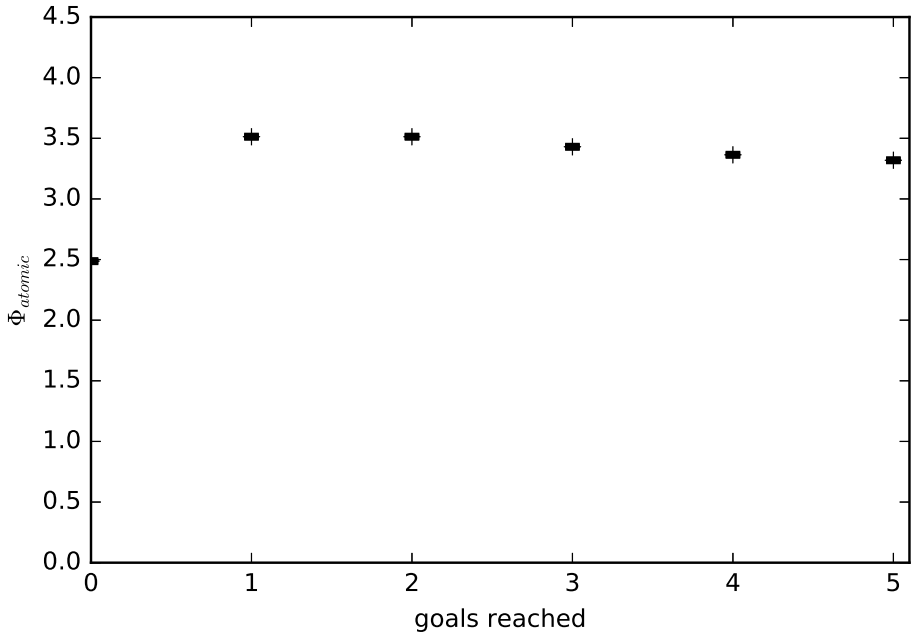

Figure 4.4: **Mean Φ_{atomic} each time the goal is reached.** Each point represents the mean Φ_{atomic} measurement of the 45 elite performers when the goal is reached. The error bars indicate the standard error of the measurement. To increase the resolution for this measurements animats were placed only 10 steps away from the goal.

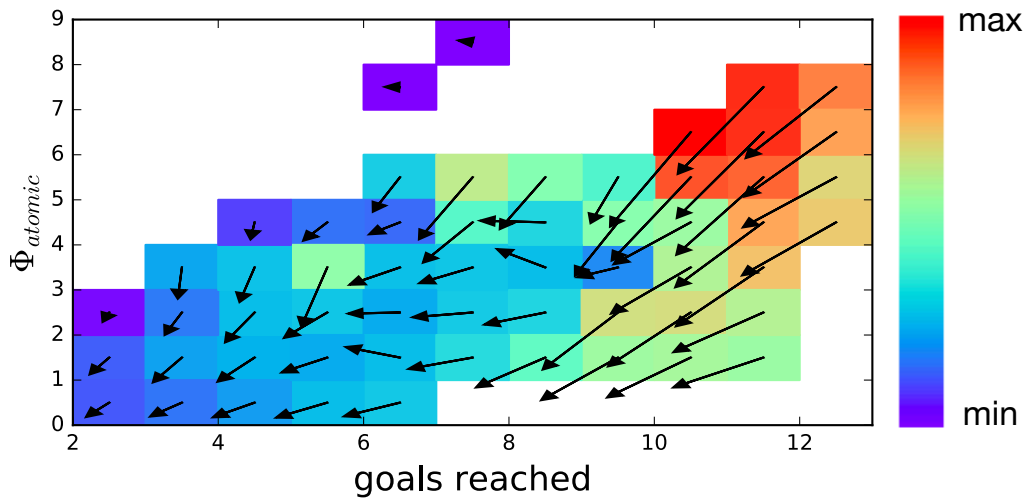

Figure 4.5: **The effect of feedback loss on performance and Φ_{atomic}.** All animats performance and Φ_{atomic} was measured on the line of decent with and without feedback. As a consequence, we can display the change in Φ_{atomic} and performance as a vector, originating at the point with feedback, and pointing towards loss of feedback. All vectors were then binned on both axes and their average vector of change is displayed for bins containing 50 or more data points as black arrows. The effect on performance is scaled by 0.1 to reduce the overlap between arrows. The colored patches indicate bins, and are colored by the length of the vector so that strong effects are more red, while weak effects are more blue.

W is performance.

$$\Delta_W = W_{with} - W_{without} \tag{4.4}$$

In addition Φ_{atomic} not only increases over generations but also as a result of learning, i.e. $\Delta\Phi_{atomic}$ (see Equation 4.5 and Figure 4.5). Since performance and Φ_{atomic} depend on each other, we can correlate the change in performance with the change in Φ_{atomic} (correlation coefficient 0.5 with $p < .0001$).

$$\Delta\Phi_{atomic} = \Phi^t_{atomic} - \Phi^{t+1}_{atomic} \tag{4.5}$$

4.3 Discussion

This work is purely computational, and the learning mechanisms used here might not reflect natural learning mechanisms. Here we argue that animats which increase their performance either during evolution or during lifetime learning. Though, this work is purely computational, and the learning mechanisms used here might not reflect natural learning mechanisms. The system we MB framework is in principle similar enough to natural systems for us to make meaningful predictions. Obviously, one should test if indeed Φ changes over a learning period in natural systems to confirm our predictions.

The way Φ is measured has iterated over time; here we used the more computational tractable Φ_{atomic} which computes information integration over some period instead for a single time point, as provided in Φ^{Max}. As explained before, the types of probabilistic logic gates used here, prevents us from applying Φ^{Max}. This can be remedied by either using gates that have only a single output and not up to four as we used them here, or gates would need to be *decomposable* [144]. Decomposable means, that a more complex gate can be described as a combination of simpler ones. While this has been achieved for probabilistic gates, feedback gates with more than one output are not decomposable yet. If this can be achieved, it would be interesting to compare these results to ones obtained using other versions of Φ.

4.4 Conclusion

From earlier experiments we already knew how Φ changes over evolutionary time, that there is a lower bound so that performance can only increase together with an increase in Φ, and that increasingly complex environments require an increase in Φ as well. While we knew that learning increased the animats performance, we did not know how lifetime learning affects Φ. Further, it was unclear if the increase in performance is due to streamlining the computation and thus reducing the amount of integration, or if better performance requires

more integration of information? To determine how the integration of information was affected by learning, we tested how Φ_{atomic} behaves over evolutionary times as well as it is affected by lifetime learning. Through these test we confirmed all prior observations about Φ. In addition, we find that animats not only evolve to integrate more information over time, but that lifetime learning increases their capability to integrate information.

Animats improve their ability to integrate information during initial learning, but once the task is learned the change in Φ_{atomic} becomes minuscule. When we relate this to our example of the child learning to play soccer we see that she has been evolved to improve her ability to integrate more information as her career progresses. This supports the idea that Φ increases not only over evolutionary time, but also during lifetime learning. In addition, we find that the system's ability to increase performance is positively correlated with a system's ability to increase Φ. Therefore, we suggest that a system that can dynamically adapt Φ will learn better.

4.5 Methods

To study how information integration changes over an organism's lifetime, we create an animat based simulation where animats are required to correlate an action with a symbol. The brain of each animate is a probabilistic finite state machine that consist of sensor nodes (input), memory nodes (hidden) and action nodes (output), or a Markov Brain (MB) [1]. The animates were evolved using a genetic algorithm (GA) that selects for individuals that successfully decode a variety of symbols.

4.5.1 Environment.

The animats were originally evolved to perform a navigation task for Sheneman and Hintze [124] and were placed 32 cells from the goal. The final generation of the initial animats were then tested in all 24 mappings between action choices and behavioral outcomes in five unique environments. Agents that did not reach the goal at least 5 times in each mapping

were discarded. The process results in 45 elite performers out of 300 replicates.

4.5.2 Animat.

Animats brains consist of 16 states: 4 input sensors, 2 output motors, and 10 hidden states. Each input sensor is assigned a direction to detect: sensor 0 fires when the animat is told to stay put, 1 indicates turn right, 2 go forward, and 3 turn left. The sensors each read the block that the animat inhabits. When the sensor detects its given direction the state is set to 1, otherwise the state is 0. For example, if the map indicates the animat's next closest step is to the left, its first four states are: 0,0,0,1.

4.5.3 Line of Descent.

The line of descent (LOD) for an experiment is determined by choosing a random animat in the final generation and tracing its lineage to the first generation [120]. Through this method the most recent common ancestor (MRCA) and all mutations that reached fixation in the population are recorded.

4.5.4 Feedback Gates.

The key feature of feedback gates is their probability that an input i results in output o has the potential to change with each update. Each time these numbers change so does the amount of predictive information shared between nodes resulting in a change to Φ_{atomic} [1]. It is important to note that the typology, or the connections, between nodes does not change, only the information contained in the connection changes.

4.5.5 Phi.

All computations of Φ_{atomic} are done using the same method as in Marstaller et al. 2013.

Chapter 5

Conclusion

Markov Brains (MBs) have been evolved to solve many complex tasks [77] and even demonstrated a way to mimic memory [72]. This research has focused on harnessing the evolutionary processes of MBs to solve tasks. However, as stated in Chapter 1, there is a difference between learning– where past actions influence future decisions– and memory– which allows knowledge gained during learning to be stored. While learning takes place within one's lifetime, the neural mechanisms used by memory can be developed through evolutionary principles. Here I introduced feedback gates which allow MBs to evolve the ability to learn, and thus provide a more in depth model for the evolution of neural plasticity, i.e. learning, in natural organisms.

5.1 Summary

In Chapter 1, I explained the mechanics of feedback gates, and how when incorporated in MBs these gates do not rely an objective external feedback signals to learn. Instead feedback gates use the multiplicative update algorithm to update their probability tables based on positive or negative feedback. This learning process is similar to how natural organisms interpret the environmental effects of their actions in order to generate internal feedback, which results in learning.

In Chapter 2, feedback gates are proven to work within the MB paradigm by presenting

agents to solve a complex navigation task. In order to optimally reach the goal agents must first decipher how their actions translate to behavior and then incorporate the new knowledge in future action selections. Agents are allowed to turn to the left or right 90 degrees or move forward, but they must do so by selecting from four intermediate options (A,B,C,or D). Given that there are four actions and each one can result in any of the four behaviors, there are 24 action-to-behavior mappings. In this experiment agents experience all mappings during each generation of evolution (See Section 2.5.1 for a detailed description). I show that feedback gates not only are able to modify their probability tables as I assumed, but that these changes align with increased performance: those who learn more perform better. In fact, the probability tables of these gates partially resemble that of a deterministic gate (see Figure 2.3), i.e. some rows converge to a single high value for an input/output combination. I further show that the majority of learning takes place during the agents first trek to the goal, which is analyzed in Figure 2.14. There are some agents who forego the use of feedback gates and instead evolve an advance heuristic to solve the task. However, these agents do not perform as well as their peers who use feedback gates. These findings establish that MBs with feedback gates are a viable system for further research on the evolution of neural plasticity.

Chapter 3 extends the same navigation task to test agents' ability to generalize and reversal learn. First, generalization is the ability to apply concepts organisms learn from past experiences and apply them to new situations. Here agents were evolved in subsets of the possible mappings, and then their generalization abilities are tested using all 24 possible action-to-behavior mappings. I find that agents needed to experience 2/3 of the mappings before they are able to generalize. Second, I examined agents' ability to reversal learn, which is where an agent learns how to process the stimuli from one environment before the meanings of the stimuli are switched. The switch in stimuli forces the organisms to adapt their neural mechanisms to survive. I test the ability of the MBs to learn by exposing agents to one mapping and switching them to a different mapping. Results show that reversal learning is

a natural by-product of generalization.

While modeling generalization and reversal learning provides insight to how natural organisms learn, it contributes a new dimension of *autonomous* learning to machine learning. Autonomous learning allows the agent to learn new concepts without human intervention. Previous work in machine learning has focused on developing advanced methods of supervised and unsupervised learning. I found MBs with feedback gates are able to generalize and are no longer focusing on learning the environments they see during evolution. They instead evolved to interpret the environment to generate internal feedback.

In Chapter 4, I use MBs to study how learning effects the neuro-correlate Φ which measures the amount of information integration a cognitive system performs. I use the same agents evolved for Chapter 4, but performed additional analysis where I specifically looked at measurements of Φ_{atomic}. When populations evolve the ability to learn Φ_{atomic} increases over evolutionary time. I also subdivide the lifetime of the agent into natural milestones to measure how Φ_{atomic} changes. I find that individual agents increase their Φ_{atomic} as the probability tables of the feedback gates stabilize.

5.2 Future Work

The advent of feedback gates in MBs has already proven to advance both the artificial intelligence field and the study of natural organisms. However, the research presented here is merely the beginning of how these gates can contribute to these endeavors. There are open questions on:

- Future updates to feedback gates.
- The extent of generalization.
- Biological insights.
- The impacts on information integration theory.

Though feedback gates are already powerful tools for studying the evolution of learning, there are additional features that will add to that power. For instance, we know that in nature, neural mechanisms are reconfigured during consolidation. In feedback this is analogous to changing the connections to the state buffers, a process that is only accomplished though evolution in the current implementation. Future work may focus on adding process to an agent's abilities within their lifetime, as with this ability we could then have insight into consolidation. Nonetheless, there are still concepts that can be explored within both machine learning and biology with the current implantation of feedback gates.

A current hot topic in machine learning is creating autonomous robots. To achieve this goal machine learning needs to have a wide range of generalization capabilities. Though I have shown that MBs with feedback gates are capable of creating an autonomous system, I use a limited task domain. A reasonable next step to having autonomous robots controlled by MBs would be to explore cross domain generalization– that is agents are able to generalize to solve multiple types of tasks. First we must answer if the agents who are capable of performing cross domain generalization will emerge from evolution. Instinctively we know that populations must be exposed to multiple domains in order to extrapolate a wide range of concepts to apply to new challenges. However, it is unclear if there is a lower bound for the number of domains that agents need exposure to, to develop global generalization capabilities. It is quite possible that global generalization is not feasible; after all a single human cannot be an expert in all things. Previous computational models have suffered from "catastrophic forgetting" which means that systems that attempt to learn new information but in doing so simply forget or unlearn former capabilities. Feedback gates may in fact overcome these challenges given the right fitness function and/or evolutionary environment, but more research is required to determine this.

As I mentioned in Chapter 1, there are many aspects of natural learning we don't currently understand. For instance, I found that reversal learning evolves with generalization, but would an agent be able to have greater success at adapting to the second stimuli if we

selected for that in evolution? Further, would generalization then be an ability that these agents automatically possess?

The measure for integrated information theory, Φ, has been iterated throughout its history. In order to capture the amount of information integration over multiple periods of time, I choose to use Φ_{atomic} in Chapter 4. However, a newer version of the measurement exists, Φ_{Max} [77], which requires that all outputs of the computational components of the MB brain have independent outputs (decomposable); which probabilistic and feedback gates do not have. However, if we are able to replicate the logic of these more complex gates with a network of single output gates– in essence *decomposing* them– we can use the newer measurement. Schossau et al. presented decomposable probabilistic gates that allows Φ_{Max} to be calculated. However, since the feedback gate's probability table can change from update to update there is currently no method to decompose them. Once a method is developed, a comparison of how Φ_{Max} and Φ_{atomic} respond to learning could prove interesting.

As mentioned above, the way integrated information is measured has changed several times. In Chapter 4, I transformed $\Phi atomic$ into a time local measurement by measuring the amount of integrated information between time points t and $t+1$, which I termed a *milestone*. However, there is a difference between the amount of integration a system performs and the idea that new information becomes integrated into the system. The work I presented did not include a measure that focused on the new information integrated between milestones. In addition, we humans perceive learning as a conscious endeavor, while the application of something learned can be done unconsciously. This change in attention might affect our perception of our own consciousness, however Φ as a measure of information integration is not designed to take attention into account. It is still unclear how integrated information theory would address any of this. Future versions of integrated information theory might incorporate a way to account for attention. My model would be a tool to test future Φ theories in a convenient way.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Jeffrey A Edlund, Nicolas Chaumont, Arend Hintze, Christof Koch, Giulio Tononi, and Christoph Adami. Integrated Information Increases with Fitness in the Evolution of Animats. *PLoS Comput Biol*, 7(10):e1002236, October 2011.

[2] Arend Hintze, Jeffrey A. Edlund, Randal S. Olson, David B. Knoester, Jory Schossau, Larissa Albantakis, Ali Tehrani-Saleh, Peter Kvam, Leigh Sheneman, Heather Goldsby, Clifford Bohm, and Christoph Adami. Markov brains: A technical introduction. *arXiv preprint arXiv:1709.05601*, 2017.

[3] David Balduzzi and Giulio Tononi. Integrated information in discrete dynamical systems: motivation and theoretical framework. *PLoS Comput Biol*, 4(6):e1000091, 2008.

[4] Masafumi Oizumi, Larissa Albantakis, and Giulio Tononi. From the Phenomenology to the Mechanisms of Consciousness: Integrated Information Theory 3.0. *PLoS Computational Biology*, 10(5):e1003588, 2014.

[5] Sievert Rohwer and Gregory S Butcher. Winter versus summer explanations of delayed plumage maturation in temperate passerine birds. *The American Naturalist*, 131(4):556–572, 1988.

[6] Dolph Schluter. Adaptive radiation along genetic lines of least resistance. *Evolution*, pages 1766–1774, 1996.

[7] J Curtis Creighton. Population density, body size, and phenotypic plasticity of brood size in a burying beetle. *Behavioral Ecology*, 16(6):1031–1036, 2005.

[8] Eric R Kandel, Yadin Dudai, and Mark R Mayford. The Molecular and Systems Biology of Memory. *Cell*, 157(1):163–186, March 2014.

[9] Sam McKenzie and Howard Eichenbaum. Consolidation and Reconsolidation: Two Lives of Memories? *Neuron*, 71(2):224–233, July 2011.

[10] Larry R Squire and John T Wixted. The Cognitive Neuroscience of Human Memory Since H.M. *Annual Review of Neuroscience*, 34(1):259–288, July 2011.

[11] Leon N Cooper and Mark F Bear. The BCM theory of synapse modification at 30: interaction of theory with experiment. *Nature Reviews Neuroscience*, 13(11):1–13, November 2012.

[12] Charles Darwin. *On the Origin of Species*. 1859.

[13] Carl Zimmer and Douglas John Emlen. *Evolution: Making sense of life*. W. H. Freeman, second edition, 2015.

[14] Emilie C Snell-Rood. An overview of the evolutionary causes and consequences of behavioural plasticity. *Animal Behaviour*, 85(5):1004–1011, May 2013.

[15] P Beldade, ARA Mateus, and R A Keller. Evolution and molecular mechanisms of adaptive developmental plasticity. *Molecular Ecology*, 2011.

[16] John Garcia, Donald J Kimeldorf, and Robert A Koelling. Conditioned aversion to saccharin resulting from exposure to gamma radiation. *Science*, 1955.

[17] J O'keefe and DH Conway. Hippocampal place units in the freely moving rat: why they fire where they fire. *Experimental brain research*, 31(4):573–590, 1978.

[18] May-Britt Moser, Edvard I Moser, Elma Forrest, Per Andersen, and RG Morris. Spatial learning with a minislab in the dorsal hippocampus. *Proceedings of the National Academy of Sciences*, 92(21):9697–9701, 1995.

[19] Mark F Bear, Barry W Connors, and Michael A Paradiso. *Neuroscience*, volume 4. Wolters Kluwer, 2016.

[20] C David Allis and Thomas Jenuwein. The molecular hallmarks of epigenetic control. *Nature Reviews Genetics*, 2016.

[21] J Mark Baldwin. A new factor in evolution. *The american naturalist*, 30(354):441–451, 1896.

[22] B Sznajder, MW Sabelis, and M Egas. How adaptive learning affects evolution: reviewing theory on the baldwin effect. *Evolutionary biology*, 39(3):301–310, 2012.

[23] Helen Hodges. Maze procedures: the radial-arm and water maze compared. *Proceedings of the Fifth International Meeting of the European Behavioural Pharmacology Society (EBPS)*, 3(3):167–181, 1996.

[24] Lynn Nadel and Oliver Hardt. Update on Memory Systems and Processes. *Neuropsychopharmacology*, 36(1):251–273, September 2010.

[25] Wei Ji Ma, Masud Husain, and Paul M Bays. Changing concepts of working memory. *Nature Neuroscience*, 17(3):347–356, February 2014.

[26] DO Hebb. The organization of behavior; a neuropsychological theory. 1949.

[27] Wickliffe C Abraham and Anthony Robins. Memory retention – the synaptic stability versus plasticity dilemma. *Trends in Neurosciences*, 28(2):73–78, February 2005.

[28] Larry R Squire and Pablo Alvarez. Retrograde amnesia and memory consolidation: a neurobiological perspective. *Current opinion in neurobiology*, 5(2):169–177, 1995.

[29] Helmut W Kessels and Roberto Malinow. Synaptic AMPA Receptor Plasticity and Behavior. *Neuron*, 61(3):340–350, February 2009.

[30] Theodosius Dobzhansky. Nothing in biology makes sense except in the light of evolution. *The american biology teacher*, 75(2):87–91, 2013.

[31] Aimee S Dunlap and David W Stephens. Experimental evolution of prepared learning. *Proceedings of the National Academy of Sciences*, 111(32):11750–11755, 2014.

[32] Daniel C Dennett and U Mittwoch. Darwin's dangerous idea: Evolution and the meanings of life. *Annals of Human Genetics*, 60(3):267–267, 1996.

[33] Robert T Pennock. Models, simulations, instantiations, and evidence: the case of digital evolution. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(1):29–42, March 2007.

[34] Claus O Wilke and Christoph Adami. The biology of digital organisms. *Trends in Ecology & Evolution*, 17(11):528–532, 2002.

[35] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[36] V Mnih, K Kavukcuoglu, D Silver, A A Rusu, and J Veness. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[37] Yiwen Wang, Fang Wang, Kai Xu, Qiaosheng Zhang, Shaomin Zhang, and Xiaoxiang Zheng. Neural Control of a Tracking Task via Attention-gated Reinforcement Learning for Brain-Machine Interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pages 1–1, April 2015.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[39] Huisheng Zhang, Wei Wu, and Mingchen Yao. Boundedness and convergence of batch back-propagation algorithm with penalty for feedforward neural networks. *Neurocomputing*, 89(C):141–146, July 2012.

[40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[41] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.

[42] Ghassan Kawas Kaleh and Robert Vallet. Joint parameter estimation and symbol detection for linear or nonlinear unknown channels. *IEEE Trans. Communications ()*, 42(7):2406–2413, 1994.

[43] P M Baggenstoss. A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces. *IEEE Transactions on Speech and Audio Processing*, 9(4):411–416, 2001.

[44] Y Freund and R E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[45] S Arora, E Hazan, and S Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 2012.

[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[47] Stuart Russell and Peter Norvig. Ai a modern approach. *Learning*, 2(3):4, 2005.

[48] K O Ellefsen, J B Mouret, and J Clune. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput Biol*, 2015.

[49] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

[50] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K Jain. Face detection in color images. *IEEE transactions on pattern analysis and machine intelligence*, 24(5):696–706, 2002.

[51] Takashi Kimoto, Kazuo Asakawa, Morio Yoda, and Masakazu Takeoka. Stock market prediction system with modular neural networks. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 1–6. IEEE, 1990.

[52] Javed Khan, Jun S Wei, Markus Ringner, Lao H Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R Antonescu, Carsten Peterson, et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673, 2001.

[53] Zhi-Hua Zhou, Yuan Jiang, Yu-Bin Yang, and Shi-Fu Chen. Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, 24(1):25–36, 2002.

[54] Murat Karabatak and M Cevdet Ince. An expert system for detection of breast cancer based on association rules and neural network. *Expert systems with Applications*, 36(2):3465–3469, 2009.

[55] Ingo Rechenberg. Evolution strategy: Natures way of optimization. In *Optimization: Methods and applications, possibilities and limitations*, pages 106–126. Springer, 1989.

[56] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[57] A Globus, G Hornby, D Linden, and J Lohn. Automated antenna design with evolutionary algorithms. *Space 2006*, 2006.

[58] M D Johnston. Multi-objective scheduling for NASA's future deep space network array. *AAAI2006*, 2006.

[59] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science Stanford, CA, 1990.

[60] Dennis Garcia, Anthony Erb Lugo, Erik Hemberg, and Una-May O'Reilly. Investigating coevolutionary archive based genetic algorithms on cyber defense networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1455–1462. ACM, 2017.

[61] Lee Spector and Alan Robinson. Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40, 2002.

[62] Omid E David, H Jaap van den Herik, Moshe Koppel, and Nathan S Netanyahu. Genetic algorithms for evolving computer chess programs. *IEEE Transactions on Evolutionary Computation*, 18(5):779–789, 2014.

[63] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[64] Shimon Whiteson, Peter Stone, Kenneth O Stanley, Risto Miikkulainen, and Nate Kohl. Automatic feature selection in neuroevolution. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1225–1232. ACM, 2005.

[65] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE transactions on evolutionary computation*, 9(6):653–668, 2005.

[66] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2009.

[67] Jacob Schrum and Risto Miikkulainen. Evolving multimodal behavior with modular neural networks in ms. pac-man. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 325–332. ACM, 2014.

[68] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. 2008.

[69] T Baumert, T Brixner, V Seyfried, M Strehle, and G Gerber. Femtosecond pulse shaping by an evolutionary algorithm with feedback. *Applied Physics B: Lasers and Optics*, 65(6):779–782, 1997.

[70] Kalyanmoy Deb. Optimal design of a welded beam via genetic algorithms. *AIAA journal*, 29(11):2013–2015.

[71] Erin Jonathan Hastings, Ratan K Guha, and Kenneth O Stanley. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.

[72] Lars Marstaller, Arend Hintze, and Christoph Adami. The Evolution of Representation in Simple Cognitive Networks. *Neural Computation*, 25(8):2079–2107, August 2013.

[73] Samuel Chapman, David B Knoester, Arend Hintze, and Christoph Adami. Evolution of an artificial visual cortex for image recognition. *ECAL*, pages 1067–1074, 2013.

[74] N J Joshi, G Tononi, and C Koch. The minimal complexity of adapting agents increases with fitness. *PLoS Comput Biol*, 2013.

[75] R S Olson, A Hintze, F C Dyer, D B Knoester, and C Adami. Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface*, 10(85):20130305–20130305, May 2013.

[76] Arend Hintze, BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, and Masoud Mirmomeni. Evolution of Autonomous Hierarchy Formation and Maintenance. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 366–367. The MIT Press, 2014.

[77] Larissa Albantakis, Arend Hintze, Christof Koch, Christoph Adami, and Giulio Tononi. Evolution of Integrated Causal Structures in Animats Exposed to Environments of Increasing Complexity. *PLoS Comput Biol*, 10(12):e1003966–19, December 2014.

[78] Peter Kvam, Joseph Cesario, Jory Schossau, Heather Eisthen, and Arend Hintze. Computational evolution of decision-making strategies. *arXiv preprint arXiv:1509.05646*, 2015.

[79] Jory Schossau, Christoph Adami, and Arend Hintze. Information-Theoretic Neuro-Correlates Boost Evolution of Cognitive Systems. *Entropy*, 18(1):6–22, January 2016.

[80] Andy Clark. *Being there: Putting brain, body, and world together again*. MIT press, 1998.

[81] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[82] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Evolving adaptive neural networks with and without adaptive synapses. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 4, pages 2557–2564. IEEE, 2003.

[83] Faustino Gomez and Jürgen Schmidhuber. Evolving modular fast-weight networks for control. *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pages 750–750, 2005.

[84] Joseba Urzelai and Dario Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolution*, 9(4), 2006.

[85] Andrea Soltoggio, John A Bullinaria, Claudio Mattiussi, Peter Dürr, and Dario Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th International Conference on Artificial Life (Alife XI)*, number LIS-CONF-2008-012, pages 569–576. MIT Press, 2008.

[86] Paul Tonelli and Jean-Baptiste Mouret. On the relationships between synaptic plasticity and generative systems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1531–1538. ACM, 2011.

[87] Sebastian Risi and Kenneth O Stanley. A unified approach to evolving plasticity and neural geometry. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[88] Oliver J Coleman and Alan D Blair. Evolving plastic neural networks for online learning: review and future directions. In *Australasian Joint Conference on Artificial Intelligence*, pages 326–337. Springer, 2012.

[89] Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi. Evolving neural turing machines. In *Neural Information Processing Systems: Reasoning, Attention, Memory Workshop*, 2015.

[90] Benno Lüders, Mikkel Schläger, and Sebastian Risi. Continual learning through evolvable neural turing machines. In *NIPS 2016 Workshop on Continual Learning and Deep Networks (CLDL 2016)*, 2016.

[91] Rasmus Boll Greve, Emil Juul Jacobsen, and Sebastian Risi. Evolving neural turing machines for reward-based learning. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 117–124. ACM, 2016.

[92] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[93] Jake Lever, Martin Krzywinski, and Naomi Altman. Points of significance: Model selection and overfitting. *Nature Methods*, 13(9):703–704, 2016.

[94] Nihat Ay and Thomas Wennekers. Temporal infomax leads to almost deterministic dynamical systems. *Neurocomputing*, 52:461–466, 2003.

[95] Nihat Ay, Nils Bertschinger, Ralf Der, Frank Güttler, and Eckehard Olbrich. Predictive information and explorative behavior of autonomous robots. *The European Physical Journal B*, 63(3):329–339, 2008.

[96] Giulio Tononi. Integrated information theory of consciousness: an updated account. *Arch Ital Biol*, 150(2-3):56–90, 2012.

[97] Nihat Ay. Information geometry on complexity and stochastic interaction. *Entropy*, 17(4):2432–2458, 2015.

[98] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[99] Geoffrey E Hinton and Steven J Nowlan. How learning can guide evolution. *Complex systems*, 1(3):495–502, 1987.

[100] JF Fontanari and R Meir. The effect of learning on the evolution of asexual populations. *Complex Systems*, 4:401–414, 1990.

[101] Mauro Santos, Eörs Szathmáry, and José F Fontanari. Phenotypic plasticity, the baldwin effect, and the speeding up of evolution: The computational roots of an illusion. *Journal of theoretical biology*, 371:127–136, 2015.

[102] Aimee S Dunlap and David W Stephens. Reliability, uncertainty, and costs in the evolution of animal learning. *Current Opinion in Behavioral Sciences*, 12:73–79, 2016.

[103] Stuart Jonathan Russell and Peter Norvig. *Artificial intelligence: a modern approach (3rd edition)*. Prentice Hall, 2009.

[104] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[105] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[106] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[107] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[108] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[109] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

[110] Jason Gauci and Kenneth O Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural computation*, 22(7):1860–1898, 2010.

[111] Jory Schossau, Christoph Adami, and Arend Hintze. Information-Theoretic Neuro-Correlates Boost Evolution of Cognitive Systems. *Entropy*, 18(1):6–22, January 2016.

[112] Peter Kvam and Hintze Arend. Rewards, risks, and reaching the right strategy: Evolutionary paths from heuristics to optimal decisions. *Evolutionary Behavioral Sciences, invited submission for the Special Issue on Studying Evolved Cognitive Mechanisms*, (under review).

[113] Samuel D Chapman, Christoph Adami, Claus O Wilke, and Dukka B KC. The evolution of logic circuits for the purpose of protein contact map prediction. *PeerJ*, 5:e3139, 2017.

[114] Frank Wilcoxon, SK Katti, and Roberta A Wilcox. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics*, 1:171–259, 1970.

[115] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[116] Kai Olav Ellefsen, Jean-Baptiste Mouret, and Jeff Clune. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput Biol*, 11(4):e1004128, 2015.

[117] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[118] Arend Hintze and Clifford Bohm. Mabe. `https://github.com/ahnt/MABE`, 2016.

[119] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

[120] Richard E Lenski, Charles Ofria, Robert T Pennock, and Christoph Adami. The evolutionary origin of complex features. *Nature*, 423(6936):139–144, 2003.

[121] Satoshi Murata and Haruhisa Kurokawa. *Self-organizing robots*, volume 77. Springer, 2012.

[122] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *arXiv preprint arXiv:1407.3501*, 2014.

[123] Josh Bongard. Using robots to investigate the evolution of adaptive behavior. *Current Opinion in Behavioral Sciences*, 6:168–173, 2015.

[124] Leigh Sheneman and Arend Hintze. Evolving autonomous learning in cognitive networks. *Scientific Reports*, 7(1):16712, 2017.

[125] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[126] P N Tan, M Steinbach, and V Kumar. *Introduction to Data Mining*. Always learning. Pearson Addison Wesley, 2006.

[127] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[128] Ying Chen, JD Elenee Argentinis, and Griff Weber. Ibm watson: how cognitive computing can be applied to big data challenges in life sciences research. *Clinical therapeutics*, 38(4):688–701, 2016.

[129] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[130] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[131] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[132] Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. Starting small-learning with adaptive sample sizes. In *International conference on machine learning*, pages 1463–1471, 2016.

[133] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990.

[134] Petr Bob. *Brain, mind and consciousness: advances in neuroscience research*. Springer Science & Business Media, 2011.

[135] Ivan Petrovich Pavlov. Lectures on conditioned reflexes. vol. ii. conditioned reflexes and psychiatry. 1941.

[136] Theo Mota and Martin Giurfa. Multiple reversal olfactory learning in honeybees. *Frontiers in behavioral neuroscience*, 4, 2010.

[137] Leigh Sheneman and Arend Hintze. The evolution of neuroplasticity and the effect on integrated information. 2017.

[138] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[139] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[140] Lucia Lazarowski and David C Dorman. Explosives detection by military working dogs: olfactory generalization from components to mixtures. *Applied Animal Behaviour Science*, 151:84–93, 2014.

[141] Giulio Tononi. Consciousness as integrated information: a provisional manifesto. *The Biological Bulletin*, 215(3):216–242, 2008.

[142] Giulio Tononi. The integrated information theory of consciousness: an updated account. *Archives italiennes de biologie*, 150(2/3):56–90, 2011.

[143] Olivia Gosseries, Aurore Thibaut, Mélanie Boly, Mario Rosanova, Marcello Massimini, and Steven Laureys. Assessing consciousness in coma and related states using transcranial magnetic stimulation combined with electroencephalography. In *Annales francaises d'anesthesie et de reanimation*, volume 33, pages 65–71. Elsevier, 2014.

[144] Jory Schossau, Larissa Albantakis, and Arend Hintze. The role of conditional independence in the evolution of intelligent systems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 139–140. ACM, 2017.