Acknowledgements

Dedicated to the trailblazers whose steps I'm retracing, to all those who helped me out asking nothing in return, and to all those who come after. More specifically, I'm grateful towards my advisor Gunnar Tufte for his thoughtful advice and for the freedom he has given me to choose my own approach, only correcting my course when absolutely necessary.

Abstract

The human brain is a vastly parallel, energy efficient and robust computing machine which arises from the self-organizing capabilities of billions of neurons. In contrast, digital computers are brittle devices whose exponential growth in processing power is now waning as moore's law is coming to an end. The main issues faced by processors is the energy consumption, and design complexity of attempting to parallelize the inherently sequential underlying von-neumann architecture.
This work is part of an interdisciplinary effort to investigate and exploit the properties of biological substrates, i.e neurons, for computation. A *cyborg*, short for cybernetic organism is being created as part of the NTNU cyborg project, serving as a proof of principle for a hybrid neuro-digital organism. The cyborg is a real world robot, controlled by a hybrid neuro-digital system, consisting on neural tissue cultivated in a laboratory, an interface between the analog domain of neurons and digital computer logic through electrical signals, and a flexible framework for utilizing reservoir computing to provide a bridge between the digital logic of the robot control system and the dynamics of the neural tissue.
The focus of this thesis is the design and implementation of a closed loop system where a digital computer interfaces with the neural tissue, forming a two-way bridge allowing the cyborg to learn how to navigate a simulated robot through a maze with no human intervention. The theoretical framework of reservoir computing is presented, providing a technique for communicating with neural cultures which is then used to implement a working proof of concept system, allowing neural tissue located at the neuroscience lab at the medical faculty at NTNU to control a small simulated robot remotely over an internet connection. All the nessesery software for interfacing with neurons, from the low level analog-digital level to the high-level interpretation of signals has been implemented as part of a platform for experimentation which handles networking, recording and playback of experiments, and the configuration of parameters for the embedded reservoir computer which is now available as the end product. A first experimental set-up is included as a working example.

Contents

List of Figures

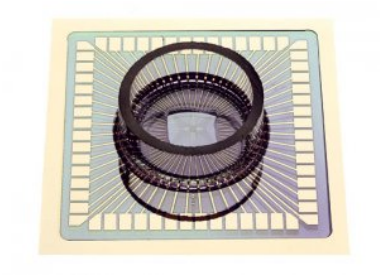
Chapter 1

Introduction

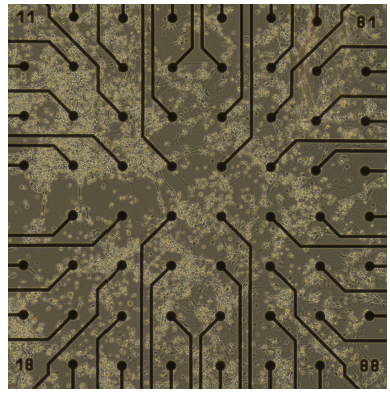
> They're trying to understand what space is. That's tough for them. They break distances down into concentrations of chemicals. For them, space is a range of taste intensities.
>
> Greg Bear, Blood Music

In order to meet the ever rising demand for computing speeds modern processor architectures have become increasingly parallel and complex to circumvent the inherent physical limitations of single core processors. However, designing complex circuits is a difficult task, especially considering the harsh demands on correctness, if even one in a billion instructions are executed wrong the program will crash, or worse, give an incorrect result. Nature on the other hand does not shy away from complexity, something the human brain exemplifies: Unlike the processor, it is not the result of some top down design philosophy, yet through a process of self organization neurons are capable of forming highly complex networks capable of solving complex tasks, with far greater energy efficiency, robustness and parallellism than any designed processor. In order to investigate the processes that create the neural networks that makes up the brain, a hybrid biological-digital robot has been created in a joint multidisciplinary effort. Neurons are grown *in vitro* on *Micro Electrode Arrays*, shown in figure 1.1 are interfaced with a digital computer throug lab equipment (figure 1.2) which can perform highly accurate voltage measurements forming a hybrid neuro-digital system. This system, known as a *Cyborg*[1] is used to control a small robot, allowing it to sense and maneuver a simple maze. The ultimate goal of the cyborg is to further our understanding of the underlying principles that governing how nature computes.

[1] A portmanteau word stemming from cybernetic organism

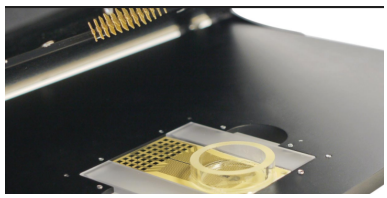


(a) an MEA

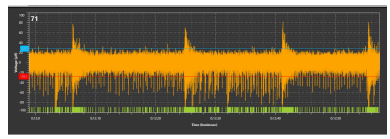


(b) neural tissue on an MEA seen through a microscope

Figure 1.1: A micro electrode array (MEA), lined with electrodes that extend to the middle chamber where neural tissue can be grown.



(a) An MEA2100 headstage



(b) Electrical activity measured by a single electrode

Figure 1.2: MEAs containing neurons are inserted into equipment capable of measuring and inducing electrical activity at high precision.

Complexity

In the 50's and 60's there was much optimism in the burgeoning field of artificial intelligence. In 1965 H. A. Simon claimed "machines will be capable, within twenty years, of doing any work a man can do."[8] , while Marvin Minsky boldly claimed in 1967 that "Within a generation [...] the problem of creating 'artificial intelligence' will substantially be solved." [8]. Had they chosen to predict any other field, such as logistics, information sharing or communications their statements would have been prophetic and visionary, yet even today *general artificial intelligence* seems no closer to our grasp than it did 50 years ago, so why did artificial intelligence turn out so differently? In their quest to make machines that could truly *think*, being capable of learning and reasoning about the world and solve problems they had never seen before, the researchers sought to make machines that could apply logic similar to that of high level human thinking. It followed that the machine had to be programmed with rules governing logic in order to reach sound conclusions, allowing it to store and manipulate its knowledge of the world. To represent the

prior and deduced knowledge, the researchers designed programming languages such as lisp[16] that could accurately describe these operations. In order to actually execute these lisp programs, underlying hardware had to be created, supporting the primitive operations such as addition, subtraction, and loading from memory. Regardless of the underlying platform, a lisp program did not change meaning, separating execution and intent in an elegant manner allowing the programmer to ignore implementation details. In short, each piece of the puzzle was self contained, allowing the researchers to build large systems where each "layer" interacted in a manner that could be reasoned about independendently, keeping the *complexity* of the system in check. Nature, on the other hand, applies a completely different method. Complex structures appear with no blueprint, arising from a process of *self-organization* driven by a set of growth rules. This self-organizing process is capable of producing incredibly complex, robust and diverse structures whose functionality arises not from specialized components working in isolation, but from the interaction of many components. The early AI researchers believed that to create an intelligent system it was sufficient to provide a platform that could reason logically, but as history shows this did not work out. The human brain, however, is not hiding some underlying logic beneath incidental complexity, instead it seems, the complexity itself might be the fundamental driving force behind intelligence. In short, intelligence does not arise from logic, instead our ability to reason logically arises from intelligence, and focusing on the complex interaction of neurons from which intelligence emerges could yield greater understanding of just what intelligence actually is.

Computation

The invention of the digital computer will be remembered as one of, if not the most significant technological advances of mankind. With this neatly designed clockwork machine the concept of computation seemed rather straight forward, where each executed machine instruction corresponded to some unit of computation. This idea of computation is very convenient, and it fails utterly when applied to biological systems. In a recent example of this from 2005 is Ray Kurzweils book The singularity is near [11]. Ray Kurzweil claimed that computers would be as powerful as a mouse brain by 2020, a prospect that looks rather unlikely in 2018. Other than in the head of computer scientists, neurons have no notion of floating point operations or branching logic, and measuring the computational capabilities of a brain in FLOPS (a measure of raw throughput for primitive math operations like addition and subtractions per second) is grossly underestimating what computing *can* be. In spite of the digital computers shortcoming compared to the human brain, the conventional digital approach to computing has been hugely successful in solving problems that humans are bad at, and is so ubiquitus that other approaches have been dubbed *Unconventional Computing*. Unconvential computing, as implied by the name, comes in many forms such as buckets of water [9], or blobs of carbon nanotubes [14] In these unconventional approaches it becomes harder and harder to pin down exactly what computation is and what distinguishes it from any ordinary physical process.

Cyborg

The main body of work done for this thesis is the design and development of a as part of the NTNU cyborg project[7], which comprises researchers from the department of neuroscience, nano science, computer science, cybernetics and more. The work in this thesis is a continuation of the work performed by the cyborg project [4], with focus on design and implementation of software, from low level interfacing between machine and neuron, to algorithms that give meaning to the dynamics of the neural tissue. Both by necessity and choice, most biological details are left out, the focal point of the thesis is the system in which neural tissue is a part, not the neurons themselves. Thus only details that are necessary for the computational model of neural tissue will be considered, leaving the intricate complexities of neurons to the neuroscientists, at least for the time being.

Chapter 2

Background

> The genetic code does not, and cannot, specify the nature and position of every capillary in the body or every neuron in the brain. What it can do is describe the underlying fractal pattern which creates them.
>
> Academician Prokhor Zakharov
> *Sid Meier's Alpha Centauri*

Creating a cyborg is a massive cross disciplinary effort, and if a scope is not clearly defined the background runs the risk of becoming equally massive. The goal of the thesis is to create a hybrid neuro-digital system capable of controlling a simple robot, and by extension to create a "bridge" between neural and digital. Each section in the background shares this bridge as a red thread, and consequently topics that are not directly related to the thesis' goal are discarded. The background is laid out as follows: First *Complex Systems* are introduced as a framework to discuss the computational capabilities in a wide range of systems that exhibit system dynamics similar to that of neurons. Modeling neurons as a complex systems is a first step towards establishing a common "language" between neural activity and digital logic. Next, *Material Computing* and *Evolution In Materio*, EiM for short, introduces computation done in unstructured matter through the process of evolution. The goals of EiM are closely aligned with the goal of this thesis, as both study massively parallel computation happening in physical matter shaped by the process of evolution. Next section, *Computing Neural Networks* introduces neurons with a strong focus on their computational capabilities. Building on the EiM section, this section helps reducing the necessary scope of modeling, proposing a simplified model of the computing neuron and a medium of communication between neuron and digital. Finally, *Reservoir Computing* is introduced, tying together the previous sections by introducing the framework of reservoir computing in order to establish a common "language" between neural cultures and digital.

2.1 Complex Systems

Before discussing complex systems it is necessary to provide a definition of complexity and system. In common parlance complexity one might describe a book as complex due to its intertwined narrative, where small details later turn into large plot points. A song can be described as complex because of how the different instruments play together, providing a context to the lyrics and vice versa. Giving an exact of what complexity is is awkward, but the common theme is highly interconnected interaction. In the first example this refers to how the narratives intertwine, where no narrative makes sense outside of the context of the other narratives, while in the second example it refers to how the differenent instruments can interplay, giving meaning to each other in such a way that one instrument alone, or the lyrics simply written down loses meaning and impact.

A system is defined as a set of elements and the relations between them. When a system changes over time it is a *dynamical* system, a property that is implicitly assumed for the rest of this thesis. When there are no interaction the system can only be described by its constituent elements, and these systems are not very interesting. When systems have interaction this interaction can be either linear or *nonlinear*. Linear systems are characterized by the independence of interaction. In these systems the behavior of the system as a whole is driven by the interaction of components, but each individual component behaves independantly of the rest of the system. When designing systems this is a desirable property because it enables a *top down* design process where each part of a machine can be designed independently before being put together, as each part does not alter its behavior when part of a larger system, allowing us to assemble large systems that are *complicated* while keeping complexity low. When the interactions become nonlinear this property is lost, and these systems become very hard to reason about and design. As Strogatz puts it, “Whenever pants of a system interfere, cooperate or compete there are nonlinear interactions going on. ...if you listen to your two favorite songs at the same time you won’t get double the pleasure!” [22] In *complex systems* nonlinear interaction is what shapes the systems behavior which becomes greater than the sum of parts.

Feedback and Self Organization

The nonlinearity of interactions in complex systems is often expressed as *feedback* which when positive can amplify small perturbations into cascading effects that change the system entirely, or when negative act as dampeners, givin the system some measure of stability. An example of this is how bees harvest nectar which must be carefully dried, a process that can be done on a large scale if all the nectar has the same water content. Because of this a hive will harvest from only one source at a time, which in beekeeping terms is known as “flow” (with raspberry-flow being the most important one). With no centralized decisionmaker the bees are still able to quickly agree on a new flow once the current flow dries up through local interactions. When a worker bee finds a suitable source of nectar it will fly back to the hive and perform a dance, informing other bees of the food source. Other bees that see this dance will then seek out the food source, and if successful they too will return to the hive to recruit new workers, however even after a successful foraging they can still be recruited to a different source. Often several food sources are found, and these sources “compete” against each other, but typically the closest source wins because the

faster trips cause recruitment to happen quicker. Through the amplification of local effects (one bee finding a food source) the bees settle on one type of nectar, a *global behavior* that has *emerged* in a *bottom up* fashion, i.e without the need for a centralized decisionmaker, through the process of *self organization*. Through collective behavior bees can adapt to changes in their environment, in fact a hive may even survive the death of their queen by refitting the cell of a worker larvae to a queen cell if it senses that the queen is absent or dying. To highlight how much of the bees sophistication arise from collective behavior, consider the fact that if a hive is moved even a meter, the forager bees will not be able to find their way back home, clearly as an individual the bee is not a great thinker!

Just like a bee does not operate in a vacuum, the beehive is part of a larger ecosystem. Local interactions on the *scale* of a single bee shapes the behavior at the scale of the entire hive, which in turn interacts with the surrounding flora at the scale of the eco-system. In short, every layer that is peeled away, from the scale of the eco-system to the scale of local flora down to the scale of a single bee reveals a new layer of complexity. Changing the scale of observation does not only reveal new complexities, it can just as well reveal simplicity and order. An example of this *emergent simplicity* is the earth itself which behaves as a simple celestial body in the solar system regardless of the complexities found in smaller scales of observation.

State Spaces and Attractors

The space of possible states, or configurations a system can be in, is known as a *state space*[1] As an example, the state space of a rubiks cube can be defined as all possible configurations a cube can be in. This space includes the solved cube as well as any permutation reachable from the solved cube when twisting it, but it does not include impossible rubiks cubes such as a cube with ten red faces. A single system may be defined by multiple state spaces depending on which parts of the system is observed and at which scale. The state space of the rubiks cube can be extended to include the current rotation the cube has in space, or it can be observed at a finer scale where two solved cubes occupy different states if the face of one cube has been slightly rotated. These states can be viewed as *macro states* and have been chosen specifically by the observer as opposed to the *micro state* which is the ground truth of the system[10]. For physical systems the micro state is not only intractable, but impossible to measure due to the uncertainity principle, and generally what happens at a subatomic level is not very relevant when describing a rubiks cube.

As a dynamic system evolves over time it follows a trajectory through its state space which depends on its initial conditions and outside perturbance. Analogous to gravity wells in cosmology, certain points in the state space act as *attractors*, “pulling in” the states in its *attractor basin*. The simplest such attractor is the point attractor in which a system in the point attractors basin will move towards a static state. Cyclic attractors are a sequence of states that repeat themselves sequentially. A system in a cyclic attractor can oscillate between two states, or it can follow a longer path, known as cycle length before repeating itself. For both cyclic and point attractors the amount of states a system goes through before reaching the attractor (that is, the amount of states visited only once) is

[1] For continuous systems the word phase space is more accurate, but it’s not a useful dichotomy for this thesis.

known as the *transient*. The length of the transient is a property of one trajectory leading to the attractor, but the average trajectory length is a property of the attractor itself. There is a third type of attractor which have have no cycles, known as *strange attractors*. Systems in strange attractors exhibits *chaotic* behavior, characterized by its sensitivity to initial conditions and unpredictable, seemingly random behavior. Figure 2.1 shows one solution for a nonlinear differential equation known as Lorenz attractor. Two near-identical initial conditions in this attractor will quickly diverge in behavior, however both will create the "butterfly" pattern. For a discrete system a strange attractor is impossible as any discrete system can only represent a finite amount of states, however a cyclic attractor with a cycle length near the total amount of expressible states may be informally considered as strange.

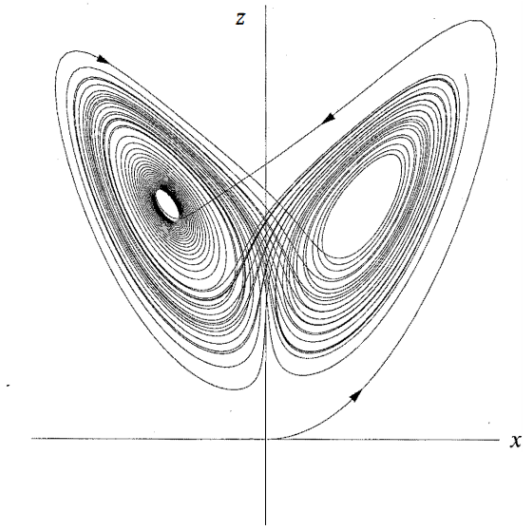


Figure 2.1: The trajectory of a particle orbiting the Lorenz attractor. Even miniscule differences in initial conditions will cause two trajectories to diverge from each other, however they will both form the butterfly pattern. Image taken from Strogatz - Nonlinear dynamics and chaos [22].

Universality Of Complex Systems

Not all complex systems are biological in nature. In Dynamics of Complex Systems Yaneer Bar-Yam illustrates this as shown in figure 2.4. In the classical school of thought different fields are seen as divergent, however Bar-Yam points out that each field deals with systems, and it turns out that complex systems have much in common regardless of how and where they originate. This *universality* is a powerful tool when studying systems such as neural networks because it allows us to explain and study much of the behavior with far simpler models, one such model being *cellular automatons*[6].

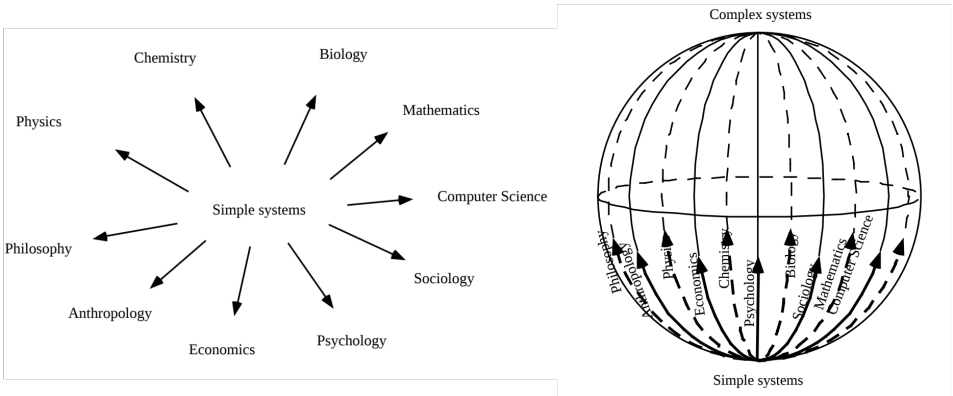


Figure 2.2: The figure depicts two viewpoints of how different fields diverge or converge. To the left the classical view is shown where, as model complexity grows, fields diverge further and further from each other. On the right side the opposing viewpoint where all fields converge towards complex systems is shown, hinting that seemingly different fields end up having much in common as model complexity increases. The figure taken from Yaneer Bar-Yams Dynamics of Complex Systems [5].

2.1.1 Cellular Automata

A *Cellular Automaton*[2], CA in short, is a simple discrete model of a single cell which changes between a discrete set of states based only on its immediate neighbors. Figure 2.7 shows an example of a *rule set*, or *transition table* for a cellular automaton with only two states, and an initial configuration, or "seed", which over several steps forms a pattern. CAs have many properties that make them ideal as a model for studying complex systems and how they can facilitate computation. In particular, each individual cell is very simple, each cell only communicates with its direct neighbor, and every cell updates in parallel, in short CAs are at the extremes of simplicity, parallelism and locality, visualized in 2.3 In other words, any computation spontaneously occuring in CAs must be self organized, bottom up and driven by local interaction which are the features that all complex systems share due to universality. An example of such a computation given by Sipper is contour extraction of an image, highlighting how a parallel local computation can yield a global result. While contour extraction is a computation, it is not very general, and the initial conditions are set up in a rather artificial way, so one might ask if CAs can perform more complex computation. As it turns out the answer is yes, as even simple 1D CAs are *Turing complete*, enabling them to express any computation. This is of little practical use though, as Sipper puts it: "This is perhaps the quintessential example of a slow bullet train: embedding a sequential universal Turing machine within the highly parallel cellular-automaton model", however it does show that more complex rules or models are not inherently more capable.

[2]Singular: Automaton, Plural: Automata

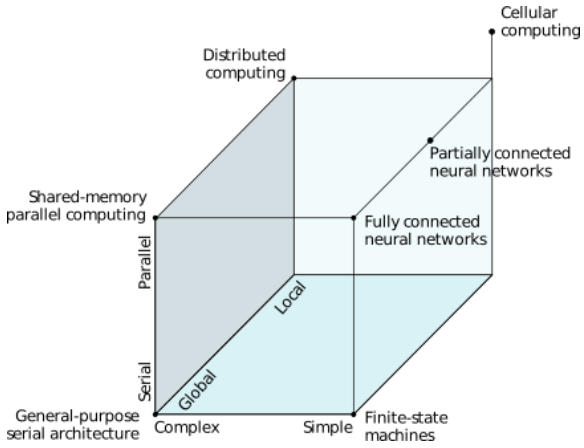


Figure 2.3: The figure shows how computation can be described along three axes: Complexity, Parallelism and Range of interaction (global or local interactions) In this domain cellular computing occupies the extreme point of simple, locally interacting parallel computations. Note that complexity here refers to the complexity of each individual computer, the resulting behavior of simple components can be highly complex. Figure taken from Sippers ermergence of cellular computation.[20]

Figure 2.4: The figure shows a specimen of the *Conus textile* cone snail against a background generated by the rule 30 cellular automata, suggesting that many patterns occuring in nature can be modeled by cellular automatas.

Order, Criticality and Chaos

Contour extraction and turing machines are artificially constructed computations, neither spontaneously occuring or particularily complex. A pioneer in the field, Stephen Wolfram, was interested not only in expressing computations with CAs, but in which conditions computation could spontaneously occur. He divided CAs into 4 classes based on their behavior[25], from least to most complex, as shown in figure 2.5. Class 1 and 2

quickly resolved to static or simple periodic behavior, which corresponds to an attractor landscape with many point and cyclic attractors with shallow basins. Class 3 CAs exhibited chaotic behavior where small differences in initial conditions caused the systems to diverge quickly into cycles so long they could effectively be considered to be in strange attractors. Class was characterized by having very long transients, forming complex patterns of localized structures corresponding to an attractor landscape with fewer attractors with very large basins. Wolfram correctly suspected that the fourth class was capable of computation, even of the universal[3] variety. In Langton's pioneering paper *Computation on the Edge of Chaos* [12] Langton explores the space of possible transition tables for one dimensional cellular automata, linking the space of CA behavior to phase changes in material. Langton ascribed a parameter, λ, to each ruleset which described how evenly matched rules leading to the dead state was vs the live state. When all configurations lead to one type of cell the rulesets λ parameter was set to 0, while the rulesets where the amount of states leading to dead and live were equal obtained a λ of 1.0. Langton found that the λ parameter modeled how rulesets and wolframs CA classes interact in a way analogous to how matter changes phase at different temperatures, shown in figuer 2.8. Figure 2.6 is a phase diagram for water where the colored lines shows first order phase changes, such as freezing below 0 degrees at ocean-level pressure, analogous to how CAs go from fixed to periodic in figure 2.8. More interesting, the figure also shows a *critical point* where matter undergoes a *second order phase change* in which it exhibits properties of several phases at once. In the critical phase matter forms structures that are self-similar, a property known as *scale invariance*, which causes classical assumption about "smoothness" of behavior at lower scales that allows simplifying the lower scale behavior break down. This is a recurring theme for systems, at the *edge of chaos* there is a critical point where the forces of feedback and dampening are closely matched, allowing a system to be responsive and dynamic without disintegrating into chaos.

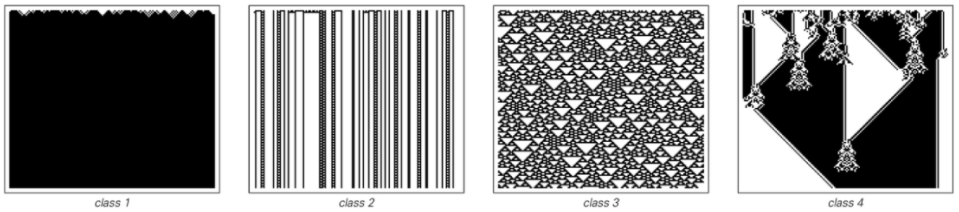


Figure 2.5: The figure depicts the four different "regimes" a cellular automata can exhibit, corresponding to Stephen Wolframs four categories. Class 1 CAs quickly converges to an all white or all black state. Class 2 CAs form static or simple oscillating patterns. Class 3 CAs exhibit chaotic behavior creating dense patterns. Class 4 CAs exhibit *complex* behavior. Figure source: Stephen Wolfram - A new kind of science.

[3]Not to be confused with universality of complex systems

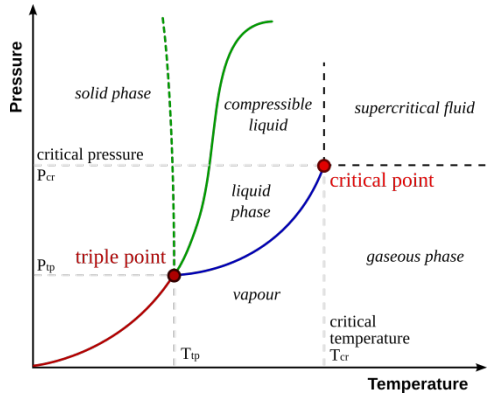


Figure 2.6: The figure shows the phase space of H_2O. Under normal pressure and temperature H_2O undergoes ordered transitions from ice to water, water to vapor and even ice to vapor, known as 1st order transitions. As temperature and pressure reaches the *critical* point however a *critical point* occurs. At this point H_2O is in neither state, but exhibits properties from both in a manner similar to that of CAs in the critical phase. Figure source: Wikimedia commons.

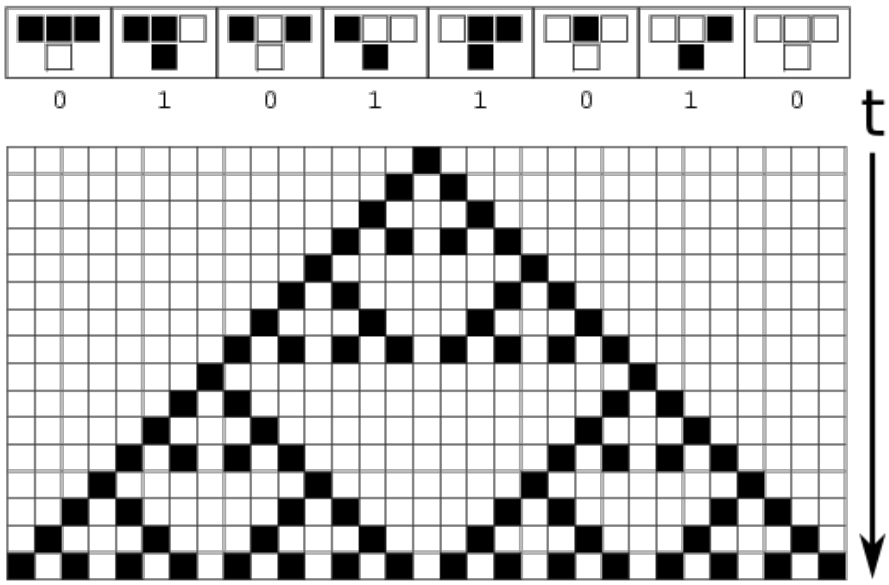


Figure 2.7: The figure shows the pattern generated by the rule 22 cellular automata when starting from a single starting cell. The system is one-dimensional, but when rendering each timestep as the system evolves over time a two-dimensional image is created.

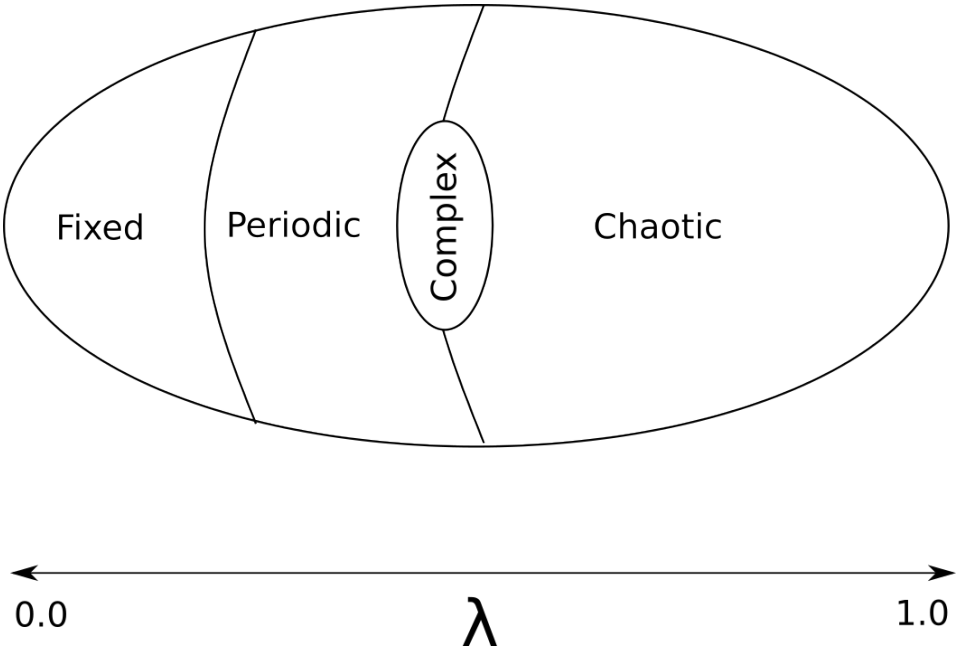


Figure 2.8: The phase space of cellular automata as a functioc of Langtons parameter. As the value of goes from 0 to 1 the behavior of the cellular automatas go from still to periodic to chaotic, however an additional phase exists between order and chaos, analogous to the 2nd order phase in phase transition, shown in Figure 2.6. Figure source: [12]

2.2 Material Computing

Modeling neurons as cellular automata gives an idea of how neurons organize themselves into computationally capable networks and how this computation may look like, but the question of how to make sense of these dynamics and how to physically interact with them remains. Before tackling neurons it is therefore necessary to extend our model to the physical realm and develop a model that can be extended to perform computation. A very natural extension of the CA model is *material computing* in which the atoms that make up matter locally interact in similar ways to CAs. Two pioneers in this field were the british duo Pask and Beer which studied how unstructured matter could be used to perform computational tasks. In one experiment [18] the duo used silver in an acidic solution which would form short-lived silver filaments when subjected to electric currents. By tuning the various knobs controlling parameters such as voltage this solution could be made to perform the task of tone discrimination, proving that unstructured matter could in fact perform computation. In Langtons CAs the goal was not to compute, but to explore what conditions were necessary for computation to occur, while Gordon and Pask sought to actually compute by observing how the matter reacted to sound and tuning the parameters in response. Tommaso Toffoli argues that “Nothing makes sense in computing except in the light of evolution” in his eponymous paper [23], which separates Langtons compute

capable CAs and the tone discrimination which had been evolved by repeatedly observing its performance and altering the parameters accordingly. A corollary of the turing completeness of cellular automatas is that CAs can be "tuned" in a rather heavy handed way to compute, but a more interesting conjecture is that a CA in the complex regime with a random seed will eventually exhibit local self-replicating behavior, spontaneously giving rise to what Toffoli would regard as computation, or possibly even as life.

Whereas the pioneers in material computing sought to perform specific tasks with their experiments more recent work has been made for general computation. One approach is *Evolution In Materio* (EiM) [17] where various materials have their parameters tuned algorithmically to perform a transformation between input and output. Figure 2.9 shows how the properties of a material can be tuned in response to its performance on some task. A recent effort, realizing the design in figure 2.9 is the NASCENCE project which has developed a physical device, "mecobo" that can host a variety of materials which can then be interacted with, using an array of electrodes. By using the same basic loop as the early material computing reasearchers the Mecobo board has been used to create XOR logic gates using unstructured carbon nano-tubes [14].

$\mathbf{(DS)^2}$

In the introduction Fernando's water bucket computer [9] was briefly mentioned. In their paper they refer to the bucket as a "liquid brain"[4], performing calculations by observing the resulting wave pattern made by small motors. Unlike its biological counterpart the "liquid brain" does not evolve over time, in fact it has no structure to evolve at all, at least not at the chosen scale of observation. $(DS)^2$ is shorthand for dynamic systems with *dynamical structure*[21], and materials that exhibit this behavior can be used to study and model biological systems. In these material systems the *state space*, that is the set of reachable states, and the transition function between them evolve over time in tandem with the systems dynamics, both shaping them and being shaped by them. In [24] Lykkeb shows that a mixture of table salt and water exhibits (DS)2 behavior using the mecobo board by showing that the systems response to perturbation evolves over time. Similarly, experiments using simulated artificial spin ice [10] (ASI) shows that in a lattice of nano-magnets several underlying microstates map to the same macro-state as shown in figure 2.10. In the figure the ASI has been simulated on a computer, which necessitates simulating the micro-statese in order to calculate the correct macrostates, however on a physical ASI lattice only the macrostates would have been observable. In the context of this thesis, the unobservable microstate of neural cultures is the inner life of each cell, the chemical gradients of neurotransmitters both inside and between neurons, and the activation of various genes, while the macrostate is the observable properties of the network which is its topology and electrical activity. If the focus of the study were on single neurons a different definition of macro and micro state would be useful, highlighting the fact that the choice of micro and macro state is not necessarily an intrinsic property of the subject of study, but a choice made by the experimenter and method of observation.

[4] Also in quotation marks in the original paper

2.3 Computing Neural Networks

Material computing provides both a theoretical framework and practical approach for modeling and interacting with the computational capabilities of living neural networks. The $(DS)^2$ viewpoint is a very natural fit when modeling how neurons organize into networks. The structure in a neural network continously evolves as each neuron individually seeks out other neurons to forge new connections while letting old connections wither and die off. The dynamics of a neural network is the electro-chemical communication between neurons in the network. These signals are obviously shaped by the topology of the neural network, but they are also the driving force between the continued evolution of the networks topology in ways that we only have a rudimentary understanding of. Although both table salt and neural networks exhibit $(DS)^2$ behavior only the neurons do this on purpose. When viewing Pask's tone discrimination experient from a $(DS)^2$ viewpoint we can catch a glimpse of the same process that shaped the rules governing neural self organization. Although this was not the view of the experimenter, the tuning of parameters performed by Pask altered how silver filaments formed and decayed, which can be viewed as a rudimentary approximation of how neurons form connections. In Pask's experiment the observer tuning the knobs served the same role as evolution did in shaping neurons, and neither force were particularily concerned about exactly what the underlying structure of the material did as long as the behavior of the material performed as desired on a macro-scale.

The Computing Neuron

The neuron, or nerve cell, is the basic building block of both the human brain and the nerve system. The culmination of billions of years of evolution, the neuron is a vastly complex entity, featuring complex chemical pathways and gene regulatory networks. However, as discussed in the section on complexity, this behavior is nescessary in order for evolution to function. As a corollary, most of the complexity of the neuron is not nescessary for it to function, it is simply a byproduct of evolution, an implementation detail. It is the view of the author that the behavior of a neural network can be modeled with a much simpler neuron such as in artificial spiking neural networks.

Although the computation performed by neurons can be modeled with simpler models, the fact of the matter is that the cyborg project utilizes real neurons rather than a digital approximation, so a cursory introduction of the neuron is in order. While there exists a multitude of different neurons in the human body it is sufficient to consider a simplified model neuron, shown in figure 2.11 The three main parts are the body, or *Soma*, the *Dendritic network* and the *Axon*. The dendritic network acts as a receptor sensing electrical activity around the neuron, while the axon transmits electric pulses to neighboring cells. The connection between two neurons is called a *Synapse*. In addition to chemical signals neural networks communicate and regulate their behavior through chemical signals. These chemical signals, known as neurotransmitters, correlate strongly with electrical activity however, thus measuring the chemical gradients in neural networks is not a priority from a compuational perspective.

Neural Dynamics

The "medium" of communication between neurons chosen as the observable macro state, that is the dynamics we are interested in measuring for the neural network, are electrical bursts of activity, known as *spikes*. Figure 2.12 shows recorded activity from an electrode. Each electrode measures the activity of the surrounding neurons rather than the state of a single neuron. Given that spikes propogate through the network in a cascading manner this level of observation is sufficient and more fine grained measurements is not a priority. Over time these dynamics evolve in tact with the structure of the network, as expected in the section on $(DS)^2$. Not only does the dynamics of the neural network change like the table salt experiment, it moves in the complexity space as shown in figure 2.13 towards a critical phase as discussed in the section on cellular automata.

Artificial Neural Networks

In recent years ANNs[5] (shorthand for artificial neural network) have revolutionized AI due to their ability to solve hard classification problems such as image recognition. The simplest, and most successful variation is the *feed forward* topology shown in figure 2.15 in which each layer only interacts with the layer directly in front and behind it. Shown in figure 2.14, the model for a single neuron consists of a summing function that calculates a weighted sum of incoming connections and a function that calculates the outbound activation of that neuron. More advanced models of neurons often retain state from their previous activation, one such variant being the spiking neuron which emulates the biological neuron more faithfully. Networks with neurons that retain state can have more interesting topologies where the connections between neurons can go in all directions. Cleary recurrent spiking neural networks should be more capable than feed forward networks, if nothing else for their ability to encode time-series as part of their state, yet the simple, non-spiking variant has proven much more successful. This gap stems from the relative simplicity of the fitness landscape of simpler network topologies which allows training algorithms to manually adjust the weights between layers iteratively until a good result is achieved. In back-coupled networks however the relation between cause and effect is far more obscure, thus algorithms which essentially assign blame and alter weights based on this are unable to cope with these topologies.

2.4 Reservoir Computing

Material computing has provided a tractable model of how the neuron computes and how to physically interface with them. However, still absent is a model that allows us to actually access the computational capabilities. In fact, from the field of material computing Susan Stepney extends the following warning "the biological substrate is extremely complex and complicated, having evolved over billions of years to exploit specific properties. In some sense, biological substrate is as far (or further!) removed from a primitive substrate as are our own designed abstract digital computational media." As the heading indicates, the ace up our sleeve comes in form of *Reservoir Computing* (RC), a technique that fittingly

[5] Or rather, the advent of hardware capable of efficiently running and training them.

enough originated from the study of ANNs of the back-coupled variety[13][15] (back-coupled here means that the flow of information is not uni-directional as is the case with feed forward networks.) In [13] randomly connected neural networks are used to solve classification problems, forgoing training the network altogether. Instead the randomly generated network was used as a *reservoir* of dynamics that reacted to the input while a simple linear output function had to be trained. The principle behind reservoir computing is to employ a complex system as a reservoir which, as Schrauwen puts it [19] "... acts as a complex nonlinear dynamic filter that transforms the input signals using a high-dimensional temporal map, not unlike the operation of an explicit, temporal kernel function." Figure 2.16 shows this setup, in which input perturbs a reservoir and the resulting dynamics is classified using a linear filter. In order to explain why this works, Schrauwen makes a comparison to the the machine learning technique of support vector machines (SVMs) work, as shown in figure 2.17. The general idea behind an SVM is to expand the input into a higher dimensional "feature space", which in the figure is represented by the transition from 2D to 3D. Similarily, when perturbed by some initial condition the resulting dynamics from the reservoir can be interpreted as a feature space and a trained *readout layer* can interpret the resulting dynamics. Schrauwen points out two major differences between SVMs and RCs. First, SVMs only implicitly expands the input to high dimensional space in order to make the problem tractable, while reservoirs do not. Secondly, kernels are not capable of handling temporal signals. The second difference is very important, it is what allows reservoirs to implicitly encode temporal signals in their dynamics, making reservoirs a natural fit for tasks such as speech recognition where each part of the input is context sensitive.

2.4.1 Linear and nonlinear output layers

In classical reservoir computing emphasis is put on the readout layer being linear. This is important because when solving a classification problem as the one in figure 2.17 a linear classifier ensures that the actual problem-solving happens in the reservoir rather than the output layer. When using an artificial neural network as readout layer with more than a single layer (known as a perceptron) the layer can, given enough nodes approximate any nonlinear function. This is a reasonable constraint when proving that nonlinear computation is done by the reservoir, but the case for removing this constraint can be made. A simple case can be made showing a reservoir with a nonlinear output layer can achieve a better performance on a classification problem. Another case is made for problems with a temporal aspect where the readout layer has only a limited memory. In this case simply showing that the problem can be solved is not sufficient, the reservoir could be a simple memory bank doing no nonlinear classification itself, thus it is still necessary to show a performance increase over traditional methods. It is not clear whether there is any advantage to using a nonlinear filter for neural cultures, but for a proof of concept both approaches can be employed, keeping in mind the caveats of using a nonlinear filter.

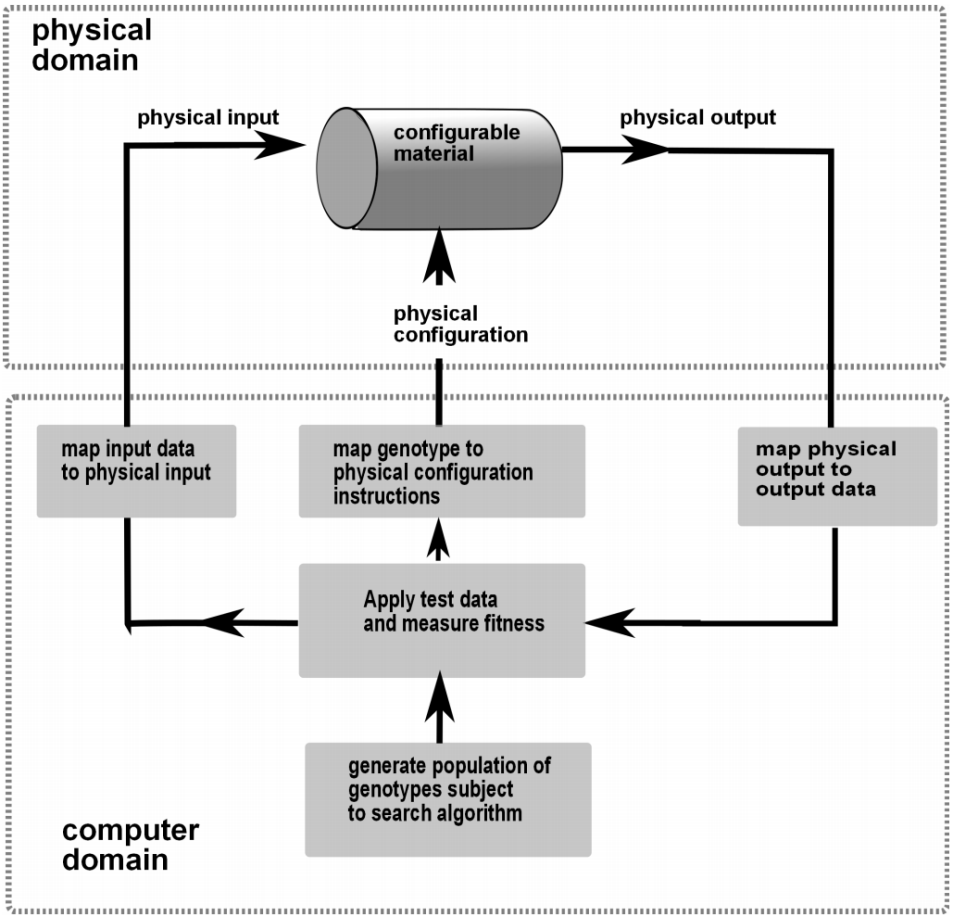
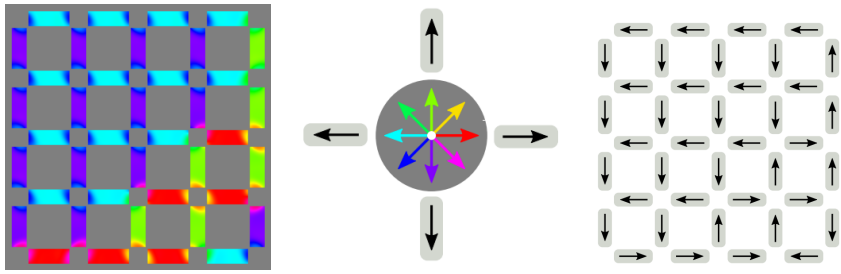


Figure 2.9: A conceptual model of an Evolution in Materio computer. In the software domain a computer searches for a configuration, that when applied to the matter in the matter domain, approximates some function between input (perturbation) and output (resulting dynamics). Each configuration is assigned a fitness based on how well the matter to provide a new generation of configurations. Figure source: [17]



(a) Microstate of a nanomagnetic lattice

(b) The correspondence microstate color and magnet direction

(c) Macrostate of a nanomagnetic lattice

Figure 2.10: A simulated lattice of nanomagnets. The color of each magnet in (a) corresponds to the direction of its magnetic field shown in (b). The corresponding macrostate for (a) is shown in (c).

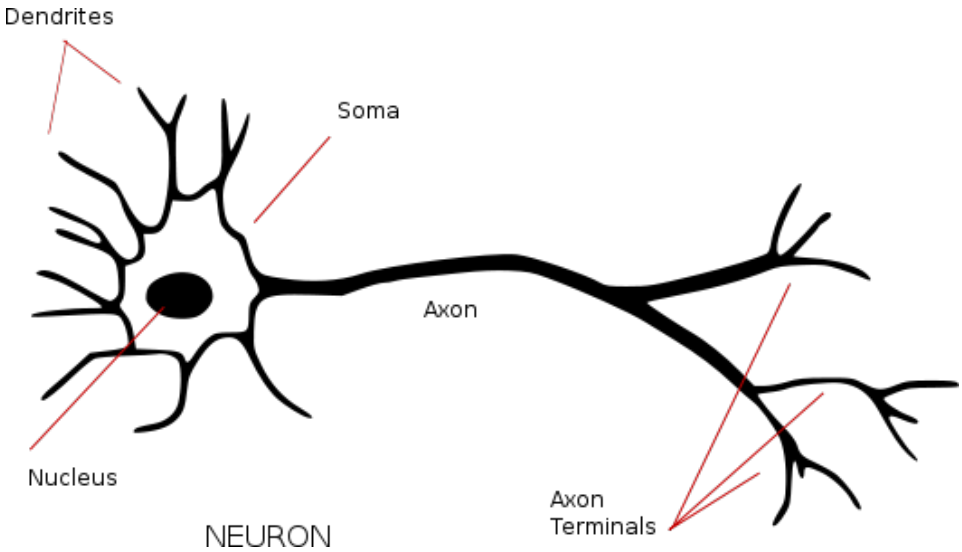


Figure 2.11: A simplistic model of a neuron. Dendrites sense electrical activity, which after reaching a certain treshold causes the neuron to fire off an electric pulse of its own, travelling down its axon which in turn can be sensed by the dendrites of other neurons close to the axon. After a pulse a cell must undergo a short refactory period before it can fire again. Figure source: Wikimedia commons.

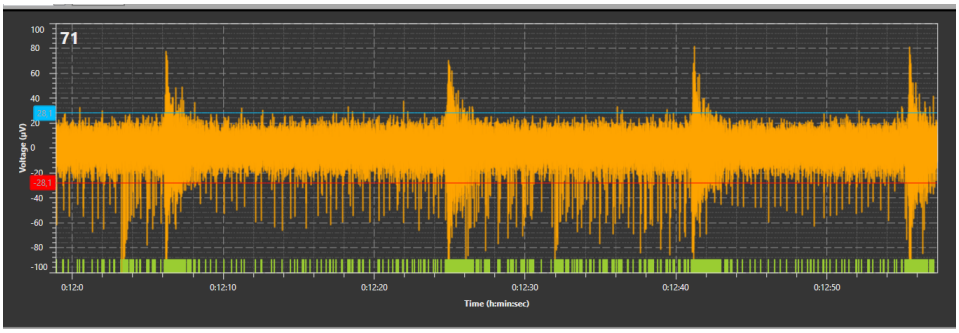


Figure 2.12: Electrical activity from the neural tissue is recorded at a micro-volt scale. In the depicted recording there is constant spiking interspersed with large bursts of activity.

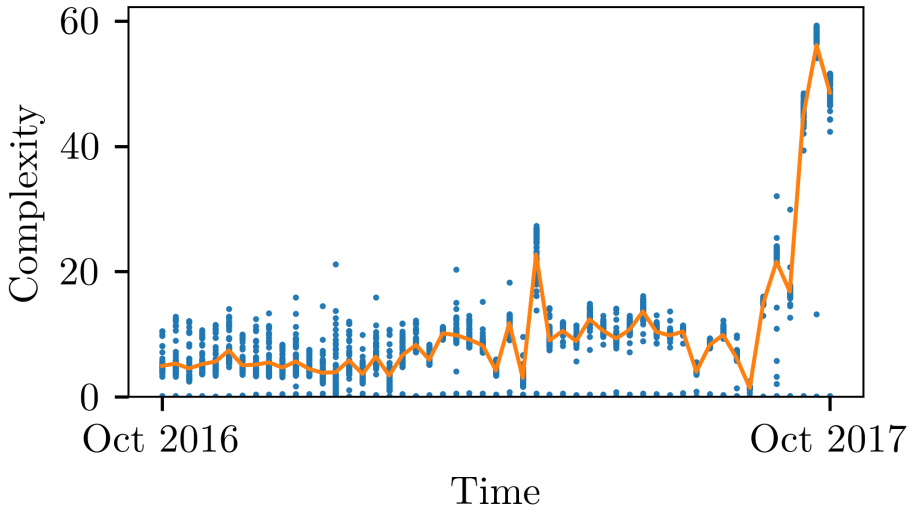


Figure 2.13: Approximated Kolmogorov complexity of the firing patterns from recordings taken from a single culture over a one year period.

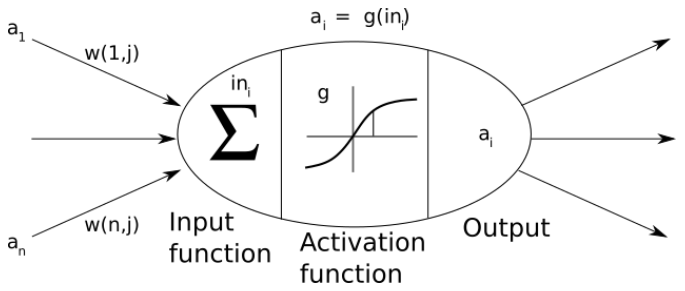


Figure 2.14: A simple model of an artificial neuron. The activation a of neuron i is determined by the input weights which are summed and evaluated by an activation function g.

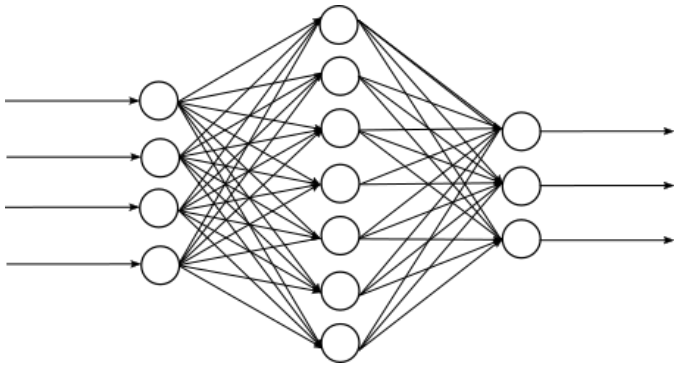


Figure 2.15: The figure depicts a network of artificial neurons as shown in Figure 2.14. When the neurons can be ordered into layers such that information only flows in one directions, which is the case in the figure, the network is called a feed forward network.

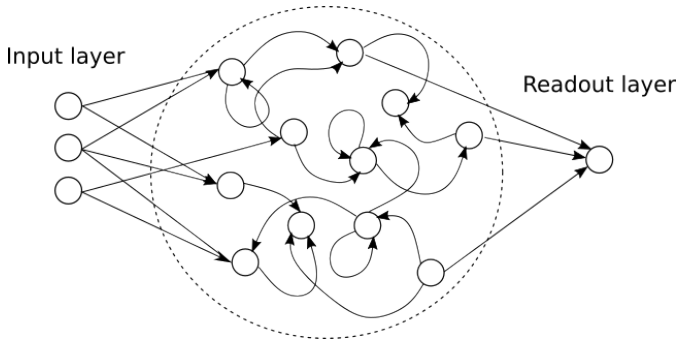


Figure 2.16: The figure shows a model reservoir computing setup. The reservoir can be a physical or virtual system as long as its dynamics responds in a nonlinear fashion when it is perturbed. In order to approximates some function $f(x)y$, input (x) is translated to a perturbation suitable for the chosen reservoir, while the resulting dynamics is interpreted to the resulting value (y) by the readout layer. In contrast to the EiM approach shown in Figure 2.9 in which the characteristics of the matter is altered to reach the target function, the reservoir computer approach only alters the readout layer which is not part of the reservoir itself.

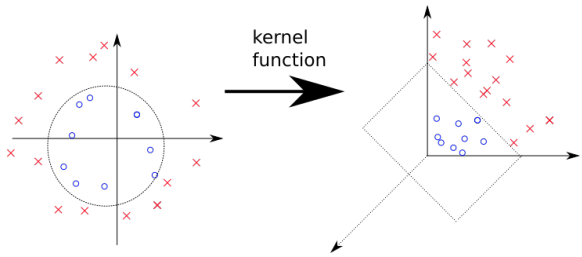


Figure 2.17: The idea behind support vector machines visualized. It is impossible to separate the crosses and circles using a single line in the left figure, but by projecting the 2D space to a 3D feature space using a kernel function the resulting space can be separated by a plane with only crosses on one side and only circles on the other. Typically the dimensionalities of input and feature space is far greater than 2D and 3D, but the concept remains the same.

Chapter 3

Making Of A Cyborg

> What is real? How do you define 'real'? If you're talking about what you can feel, what you can smell, what you can taste and see, then 'real' is simply electrical signals interpreted by your brain.
>
> Morpheus to Neo - The Matrix

The mission statement of the NTNU cyborg is "... to enable communication between living nerve tissue and a robot. The social and interactive Cyborg serves as a platform for studying neural signaling properties, robotics and hybrid bio-robotic machines." [7] The work in this thesis is strictly focused on enabling communication, leaving the robotic and social aspects for later work. This section presents the design and implementation of a system capable of communicating with neural tissue and controlling a robot in three parts, wetware, hardware and most importantly software.

Concept

In figure 3.1 the conceptual cyborg is shown. This conceptual cyborg is comprised of three main components: An *MEA*, short for *Micro Electrode Array* in which a biological neural network is grown. A *neural interface*, allowing two-way communication between the neural network and the outside world. A robotic body, responding to movement commands and equipped with a sensor allowing it to perceive its environment. In this concept neural readouts are transformed into a Left and Right signal by a feed forward neural network, controlling the direction of the robot. Simultaneously, data retrieved from the sensors of the robot are processed in a feedback processor and fed back to the neural network. The conceptual cyborg is a closed loop system: The only input to the system is what the sensors perceive, which the cyborg in turn acts upon.

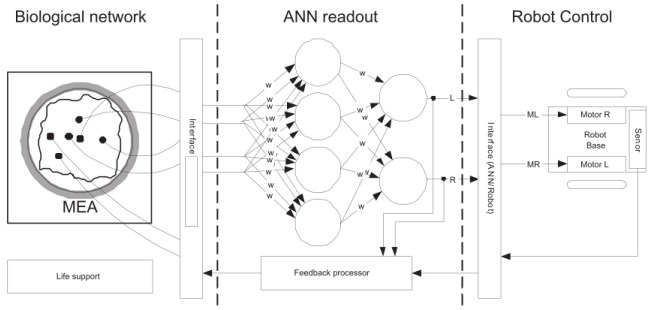


Figure 3.1: A conceptual model for a cyborg. In the conceptual model, a robot equipped with a sensor relays the input captured by the sensor to the neurons in form of stimuli generated by the feedback processor. The resulting dynamics of the neural tissue is then recorded and transformed into input values for a feed forward neural network which tells the robot which way to move.

Architecture

As shown in figure 3.2, the actual cyborg adds a lot of parts compared to the initial concept. This design expands on the initial concept of figure 3.1, fleshing out the "interface" box from the conceptual cyborg, juxtaposed in figure 3.3. As the conceptual cyborg indicates, what goes on in the box labeled interface is mostly implementation detail, thus the robot control and reservoir readout is known as the *core reservoir computer*, differentiating it from the interface components. The more detailed overview in figure 3.2 focuses on the main data loop, depicted as red and green lines, with red denoting the data pipeline from the reservoir to the core reservoir loop, and green from core reservoir loop to reservoir. The expanded architecture reveals a very important detail in the final design: The closed data loop between neurons and robot extends over network. This design choice allows the neural cultures to be safely stored in a laboratory, which has ramications both in a practical and philosophical sense. By decoupling the reservoir and reservior computer, the neurons are not restricted to a single robot. In fact, the robot does not have to be a physical robot, it can be fully virtual without the neurons noticing any difference. Additionally, the reservoir computer can interface with any reservoir as long as the constraints set by the interfaces are satisfied. This allows other reservoirs to be used in place of the neurons, both for testing purposes, and to compare the capabilities of different reservoirs.

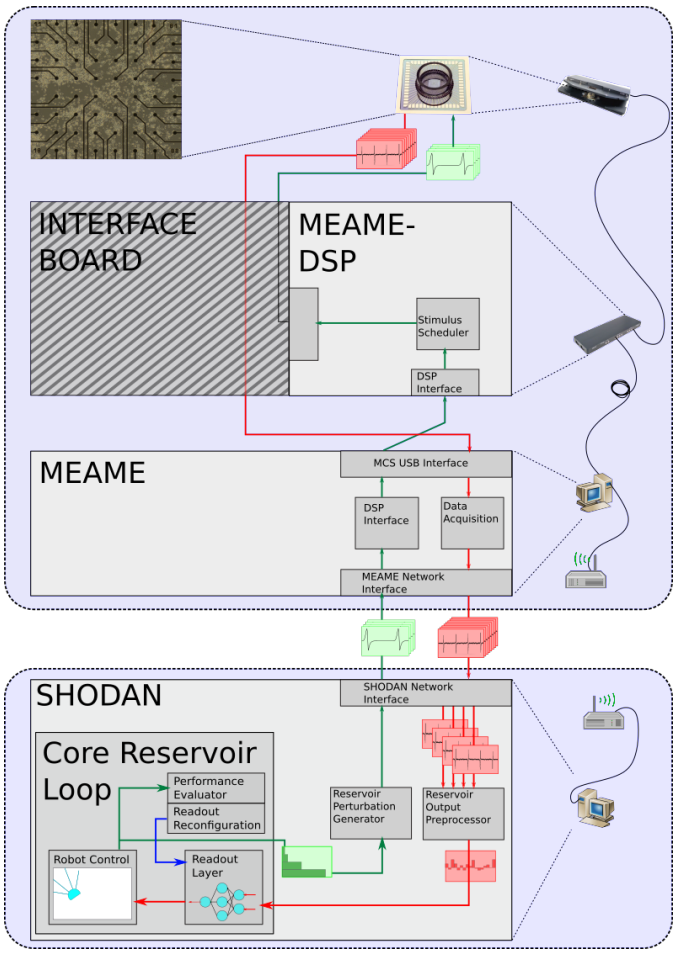


Figure 3.2: The full overview of the cyborg, divided into the three systems: MEAME, MEAME-DSP and SHODAN. The arrows denote dataflows, where the red arrows show data going from the neural tissue to the core reservoir computers, and the green arrows going from robot sensors back to the reservoir. On the left side the various devices where each system resides on is shown, including how they communicate with each other. Physical proximity is indicated by the blue tinted box, and as hinted by the routers communication crossing this physical gap happens over network. (If the colors are hard to see, the red line goes from the MEA to the readout layer as indicated by the arrowtips.)

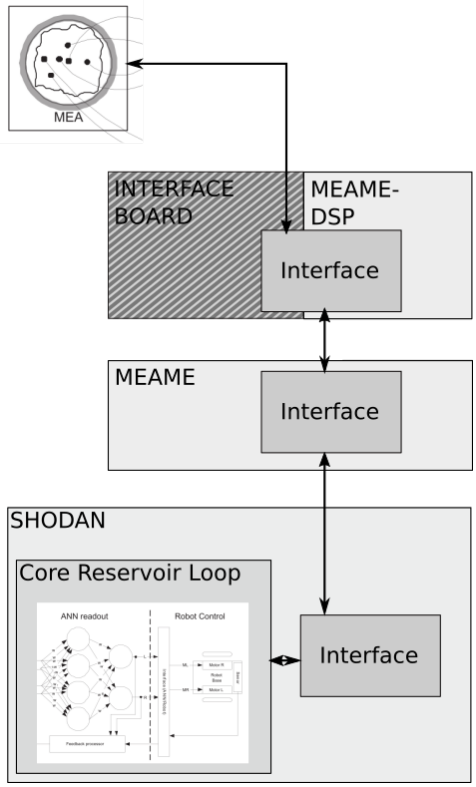


Figure 3.3: When juxtaposing the pieces of the conceptual cyborg in figure 3.1 onto the fleshed out design it becomes apparent that the interface box makes up a majority of the cyborg.

3.1 Wetware

As opposed to hardware and software, the term wetware describes system components of biological origin, i.e "wet" components. The wetware of the cyborg is thus the neural networks which are being grown in MEAs at the department of neuroscience located at the St.Olavs research hospital. The MEAs are *seeded* by introducing a solution containing neurons that have formed from stem cells of human origin[4]. Just after seeding there is no network at all, only a "soup" of dissociated neurons which over the course of several weeks start forming networks. The activity from these so-called pacemaker neurons can be seen in figure 3.4. In the pacemaker figure, each cell in the grid corresponds to one of the electrodes seen as black lines in figure 3.7 At this stage the monotonic spiking activity tends to be transient, starting and stopping randomly.

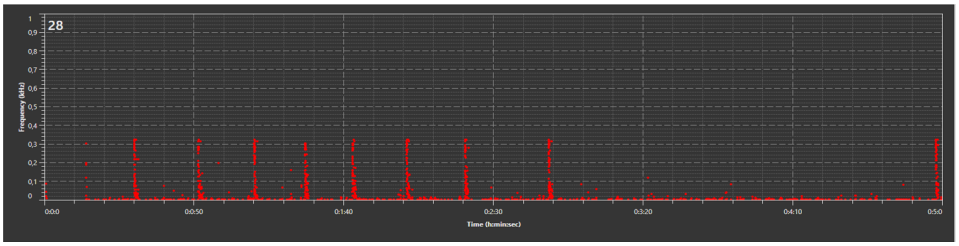


Figure 3.4: A readout showing spikes from a neural network on an MEA measured on a single electrode. The spikes follows a "pacemaker" pattern for a while, before petering off, This pattern typically forms after a few weeks. Figure source: [4]

3.1.1 Neural Interface

The *Neural Interface* is the bridge between the biological and digital, responsible for providing a convenient API (application programming interface) for reading out digital waveform data and inducing stimuli. The MEA2100 is built to conduct in-vitro experiments on electrically active cell cultures such as neurons contained in micro electrode arrays, by measuring and and inducing voltages with its high precision electrodes. Although not originally intended for use in close loop systems, the *MEA2100* system features the necessary hardware to implement a neural interface with the necessary functionality for a closed loop system. In addition to voltage and current, the MEA2100 system comes with a life support module, allowing neural cultures to survive for prolonged experiments for up to 30 minutes. Figure 3.5 shows a physical overview of the components that make up the neural interface, which is also marked in figure 3.2.

Micro Electrode Array

Shown in figure 3.6, an MEA containing a neural culture is slotted into the headstage for measurements. The MEA can easily be removed, the one shown in the picture is only one of many MEA's which are kept in an incubator when not experimented on. These MEA's feature 60 electrodes, which when accounting for the reference electrode provides 59 individual measurement and stimuli points, evenly spaced out in a grid. Each MEA is seeded with a neural culture, i.e once seeded an MEA will be the host of a single culture, each capable of living for over a year, like the culture shown in figure 3.7.

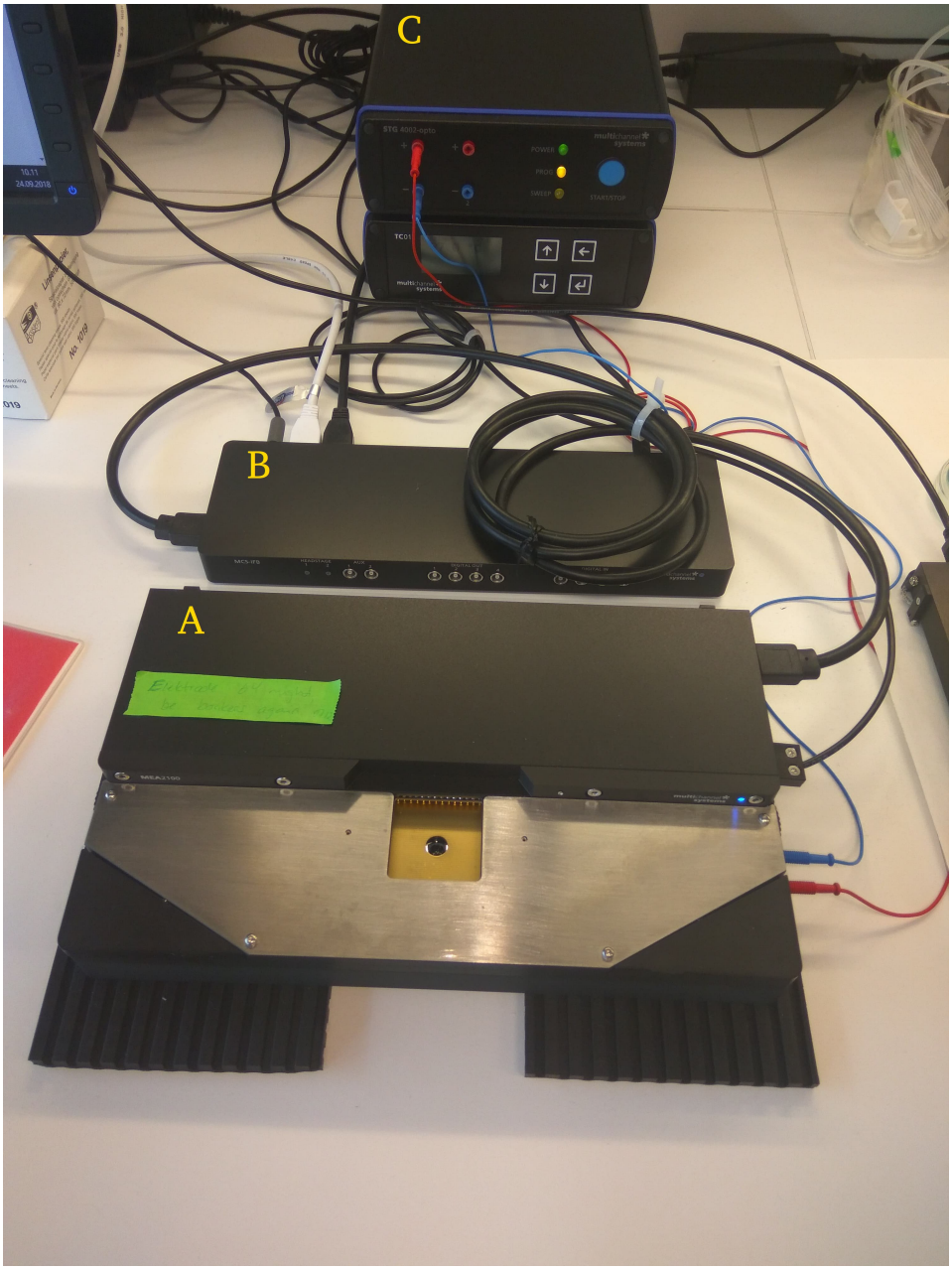


Figure 3.5: An overview of the lab equipment used for interfacing with neurons. The components shown are the headstage(A), the interface board(B) and the life support(C).

Figure 3.6: The headstage with an MEA inserted.

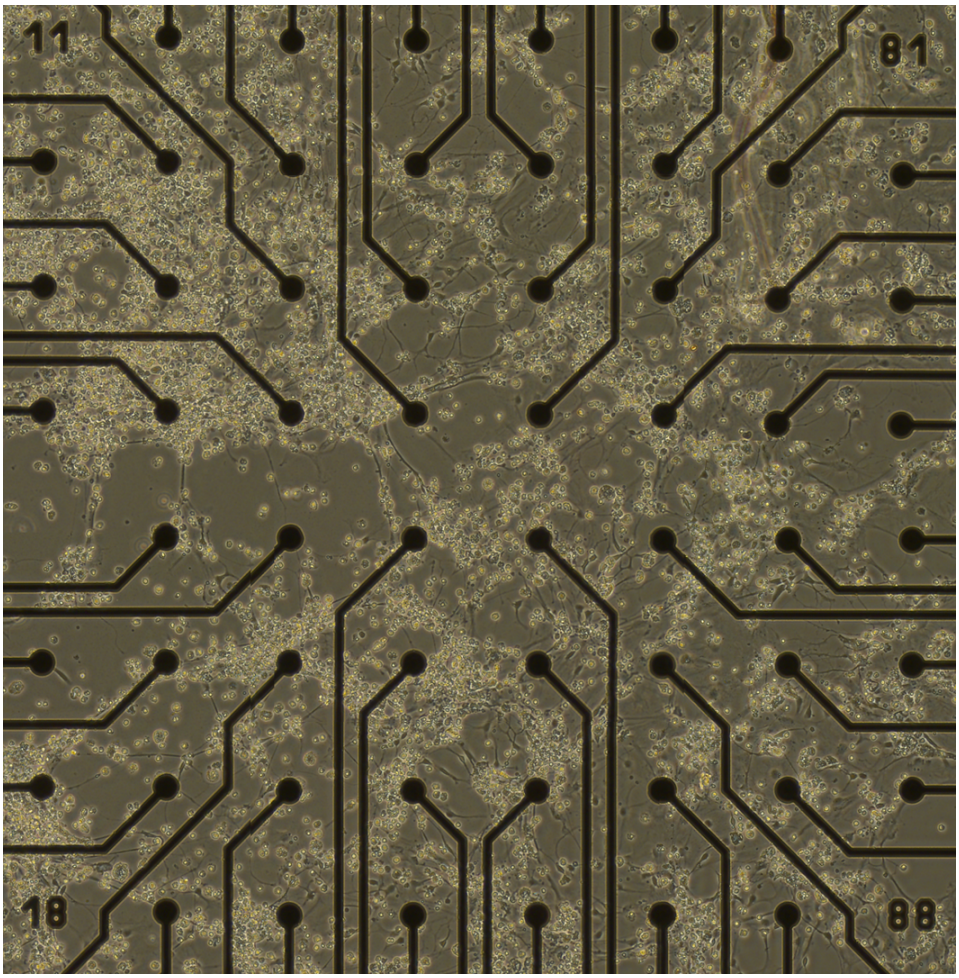


Figure 3.7: A neural culture on an MEA seen through a microscope. The figure shows a picture taken through a microscope of a culture grown as part of the cyborg project. The straight black lines are electrodes that can be used to record and apply stimuli.

Headstage

Equipped with 60 high precision electrodes, shown in figure 3.8, the headstage can connect to an MEA when inserted into the measurement bay of the headstage. Thanks to the headstage each MEA is relatively expendable as they need only provide electrodes, leaving the measuerement to the headstage.

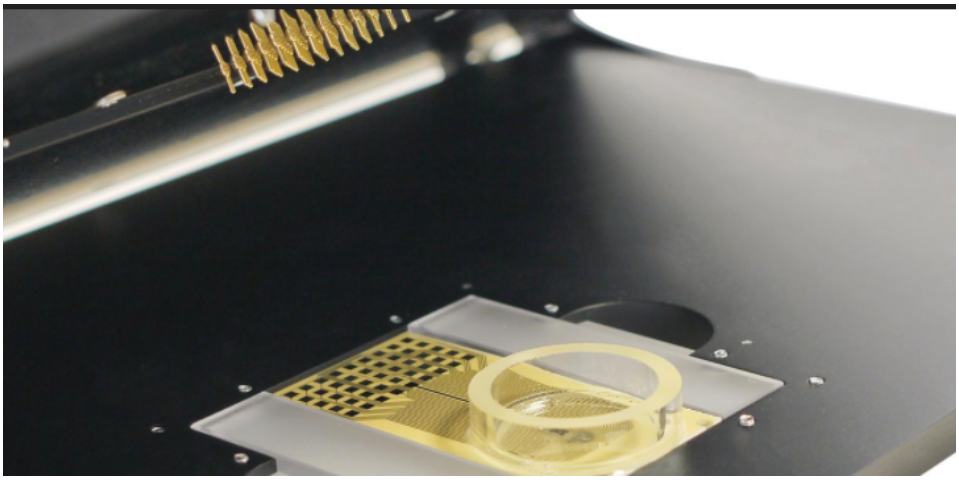


Figure 3.8: The MEA2100 headstage. In the very top of the image the rows of electrodes that measure electrical activity can be seen.

Interface board

The interface board connects to up to two head-stages and is responsible for interfacing with the data acquisition computer, as well as auxiliary equipment such as temperature controls. For this project, the most interesting feature of the IFB is a user programmable digital signal processor (DSP), which can access the raw stream of data from the headstage and issue stimuli. This DSP has access to the interfacing hardware in the IFB which is otherwise off limits and not reprogrammable for the user.

3.2 Software

Creating a cyborg is a massive undertaking, thus a quite extensive software suite has been created to make research feasible. As the system architecture overview in figure 3.2 hints, the software is the main body of work performed for this thesis. Another important detail is the strong emphasis on the dataflow, which is what connects the individual components. What goes on inside a component is less relevant compared to the transformation of data, which is visible from the outside. Finally, the figure shows a division between two systems, MEAME and SHODAN. MEAME runs close to the neural cultures, residing on the DSP and lab computer, and is specialized to interface with the MEA2100 system specifically.

SHODAN on the other hand resides on the other side of the “network chasm”, and is closer to a framework, capable of handling any reservoir as long as the necessary transformations are implemented.

3.2.1 MEAME

An overview of MEAME is shown in figure 3.9, revealing that MEAME internally consists of two distinct sub-projects exposed by a unified REST interface which is a common protocol for communicating over network. While typically REST interfaces imply that a service is public, the MEAME REST interface is only intended to be used by SHODAN. Behind the REST interface there are two modules: data acquisition and DSP interface. The data acquisition module is responsible for configuring recording parameters such as sample rate, and starting or stopping recordings, while the DSP interface provides a very thin layer of abstraction for writing to the DSP memory. Both these modules are built on top of a very thin API provided by the equipment vendor, making it clear that the closed loop cyborg system is pushing the equipment further than the vendor intended, for better or worse. In addition to the REST interface, MEAME exposes raw TCP sockets, outputting the raw waveform data once the lab equipment has been configured. The format of the raw output is shown in figure 3.10, which must be demultiplexed by the receiver in order to separate data into individual electrode readouts, referred to as *channels*.

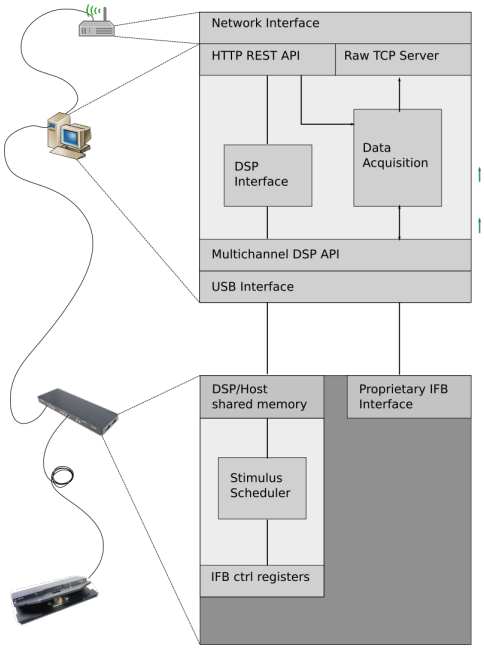


Figure 3.9: A block diagram of MEAME. The purpose of MEAME is to perform readouts on the neural tissue and broadcast this on the network, as well as providing an interface for applying stimuli to the neural tissue. In the conceptual cyborg shown in figure 3.1 the functionality provided by MEAME is contained entirely in the box labeled “interface”.

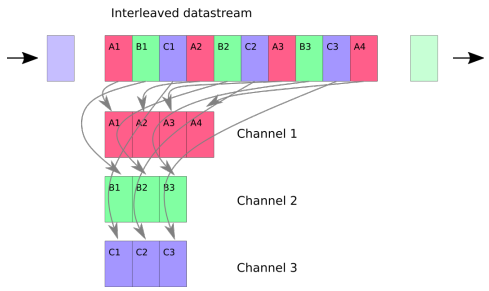


Figure 3.10: The format used when sending recorded neural activity over network. The interleaved datastream can be broken down into its individual channels. Here a datastream with only 3 streams are shown, whereas the stream is normally comprised of data from 60 channels.

MEAME-DSP

Housed on the IFB, the DSP is not in use during normal operation and thus fully user programmable. Communication between MEAME's DSP interface and the DSP itself is done with a primitive remote procedure call system, implemented using shared memory on the DSP. When SHODAN wishes to execute a DSP procedure this is done by submitting a list of memory writes via the REST interface, and then reading a special register that is incremented when the procedure call has been executed by the DSP. This indicates that a new call can be executed, and that return data is valid and can be safely read. This rather thin API is admissable because of the fact that MEAME is only accessed by SHODAN which internally translates procedure calls to memory writes and reads. The stim control module maintains a list of *stim groups*, which contain a list of electrode numbers and a desired frequency. Consequentially, stimulating neurons is done by specifying which electrodes should be stimulated, and at which frequency, rather than explicitly sending a signal whenever stimuli should be applied. The stimulus group configuration also specifies a pattern for each group, sine and square respectively. These patterns can be uploaded by directly writing to memory from SHODAN. However, for this project simple square waves are adequate.

3.2.2 SHODAN

As shown in figure 3.2, SHODAN is responsible for storage, filtering, generating perturbations and maintaining the core RC loop. During an experiment SHODAN can be divided into two parts, reservoir interfacing shown in figure 3.11, and and the core RC loop which can be seen in the overview figure 3.2. The former can in turn be subdivided into input and output processing with regards to the main RC loop.

Input Processing

The MEAME comms interfacing module mirrors the corresponding interface exposed by MEAME, consuming raw waveform data over a TCP connection, and configuring experiment parameters and issuing stimuli requests using a HTTP client. The transformation

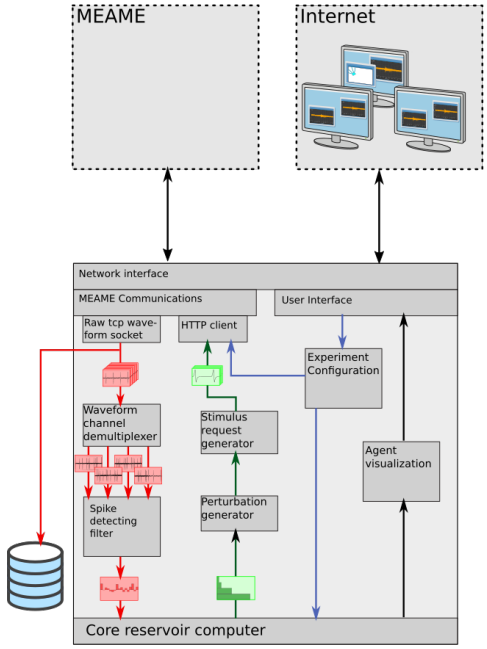


Figure 3.11: A block diagram of the frontend of SHODAN. The frontend of SHODAN, responsible for demuxing the datastream (shown in figure 3.10) and preprocessing the input data (shown in figure 3.12) before handing it over to the core reservoir computer. The frontend is also responsible for translating sensory data from the core reservoir computer to neuron-compatible perturbations (shown in the right part of figure 3.13). The frontend is also responsible for hosting a web-interface allowing configuration and visualization of data through a browser. In the conceptual cyborg shown in figure 3.1 the functionality provided by SHODANs frontend is contained in the boxes labeled "interface" and "feedback processor".

of data is shown in figure 3.12: First the channel demultiplexer splits the raw datastream from the TCP sockets into individual channels before entering the filtering module. Next, the filtering module is responsible for translating the raw waveforms to a format that can be utilized by the main RC loop, which in the current implementation is done by a basic "spike detection" transform where a binary spike event is recorded whenever a certain voltage threshold is reached, followed by a short cooldown where the detector is disabled for a short period before it can detect the next spike. Finally, the spikes are input into a moving average filter, smoothing out the staccato spike/no spike filter output to the amount of spikes that have occured between t and $t - \Delta t$ where Δt is in the order of 100ms to 1000ms. The final input to the core RC loop is a vector of values, one for each channel that can be periodically sampled. As the data crosses the boundary to the core RC loop, its shape reveals little facts about its origin. For any other reservoir it would be just as natural to translate the output dynamics to a vector of scalars, thus for any reservoir the filter module is what translates from specific to generic.

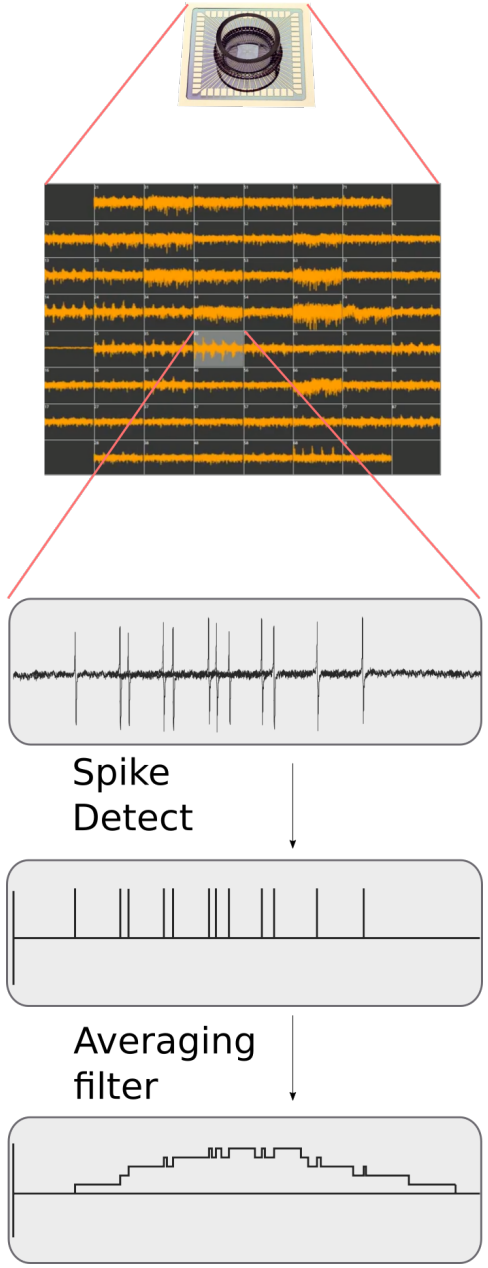


Figure 3.12: The preprocessing pipeline in more detail. The figure shows how neural activity is sampled and preprocessed to be compatible with the input layer of the reservoir computer. For each channel a spike detector singles out spikes which is then input into an averaging filter. In the conceptual cyborg shown in figure 3.1 the preprocessing is contained in the box labeled “interface”.

Output Processing

Being the dual of the Input processor, the output processor does the same thing in reverse, transforming data from generic to specific. In the figure 3.11 the output of the main RC loop is shown as a list of distances, but this is merely our interpretation of the data. The underlying shape is just the same as the output of the filter from the input processor: a vector of scalars which does not hint that these particular scalars encode distances. Since the goal of the project is to create a cyborg, the focus is naturally on robotics, virtual or not, but the main RC loop can just as well handle other tasks such as classifying images, and the resulting output would still be expressed as a vector of scalars. Figure 3.13 shows this duality between the input and output processor: Just as the input processor masks the identity of the reservoir from the core RC loop, the output processor masks the identity of the task being performed!

The datapath of the output processor is rather similar to the input processor with the perturbation transform module serving the same purpose as the filter in reverse, turning scalars into frequency ranges. In the figure an additional module is shown, transforming a stim request into a set of memory writes for the DSP as described in the MEAME section, but this is just a practical matter added for completeness.

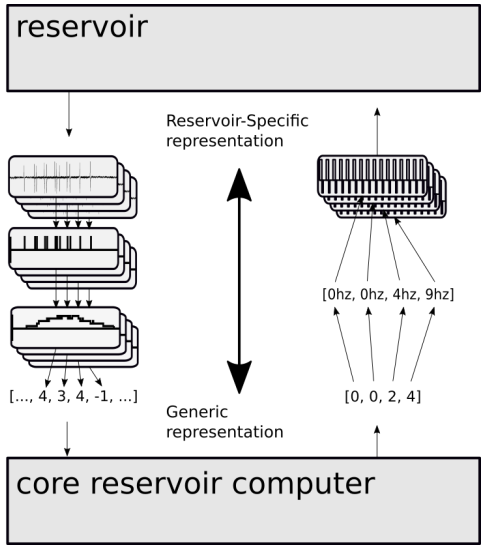


Figure 3.13: Figure showing the "anonymization" that happens between reservoir and core reservoir computer. When using neural tissue as a reservoir, the preprocessing stages transforms the recordings from neural tissue to a vector of scalars which would be indistinguishable from any other reservoir. Likewise, the perturbations generated from the perturbation generator does not reveal anything about what sort of task the neurons are actually performing since the sensory input is simply reduced to a square wave!

Core RC Loop

In the conceptual cyborg (figure 3.1), all of the software described previously can be fit into the interface between the ANN and MEA. The core of the reservoir computing system is the ANN readout and the robot control module, but the concept leaves out a very important piece of the puzzle: How can the correct ANN readout layer be chosen? The Core RC loop consists of three components, the robot control, the RC filter, and the reservoir performance evaluator. Just as the robot control does not have to be an actual robot, the RC filter can employ any filter, it does not necessarily need to be an ANN. The last module is what separates SHODAN from a classical reservoir computer. In classical reservoir computing the reservoirs themselves are fairly static, exhibiting little changes in behaviour over time. A reservoir computer using a random boolean network can store an RBN in memory and access this data to reconstruct the experiment at any time. This simplifies the task of training the RC output layer, since the reservoir will not change between runs. Unlike digital RBNs, neural cultures have no reset button, their behaviour changes both long term and short term. Long term, a cluster of neurons might "emigrate" away from an electrode, new connections form and old connections wither. This process happens over days and weeks, but change also happens on a short basis, both due to natural fluctuation, and because of the fact that no two experiments are truly independendent, there are no ways to make neurons forget and revert to previous behaviour. The solution to this problem is to back link the robot control to the RC filter via a performance evaluation module, which creates a feedback loop capable of coping with the moving target presented by a neural culture.

3.2.3 Storage And User Interface

This section will feature the UI and a description of the database system and the sort of offline analysis that can be done. It's not that interesting, but it does provide a significant boost to the usability of the system.

3.3 Current State

MEAME and SHODAN are both still in development and are far from complete. All functionality described in this chapter has been implemented and is in a running state, but the stimuli control module that is responsible for applying voltage to the electrodes does not work correctly. This means that the closed loop system does not work at all until the issue can be fixed. This frustrating issue reflects how the closed loop system goes far beyond the intended use for the lab equipment, which is marred by poor software and nonsensical documentation, leading to development often being frustrating process of reverse-engineering rather than engineering. While this flaw means the system does not fulfill its purpose, it's enough to verify that the remaining modules work. Recordings can be made and played back, which means that MEAME/SHODAN is already able to provide a better storage solution than the vendor provided solution. MEAME and SHODAN are available on github [2][3][1]

Chapter 4

Experimental System and Setup

> I guess cyborgs like myself have a tendency to be paranoid about our origins.
>
> Motoko Kusanagi - Ghost In The Shell

Because of the unresolved issues with stimuli generation the experiment setup has not been employed on living neural tissue. As a consequence, the current experimental setup is designed to be easy to modify whenever testing becomes possible and to verify functionality of the rest of the system. Consequentially, the focus on this section is on the capabilities for experimentation provided by the system, and the parameter space that can be explored.

When leaving out all implementation details, the setup for conducting experiments provided by SHODAN and MEAME shown in figure 4.1 becomes much simpler. The setup can be divided into three separate components. The first two components are the *primary* and *secondary* dataloop, which both interact with the third component, the *robot control* module. The focus of this chapter is on the role each of the three components of an experiment serve, and a survey of the parameter space which can be explored.

Deviation From Classical Reservoir Computing

Before going further, it is necessary to clarify where the experimental setup deviates from a classical reservoir computer setup, as well as definig the terminology. Henceforth, a *run* is defined as continously running the reservoir computer with no outside intervention for some duration of time. An *experiment* can consist of several runs in which the dataloop between reservoir and reservoir computer is restarted between runs. Finally, the goal of an experiment is to optimize the performance at a given *task* according to some criteria. In the classic RC setup the input layer is not altered during an individual run, instead it is modified between runs, and the resulting experiment iterates towards an input filter capable of solving some task specified by the experiment. From this angle the classic model is very

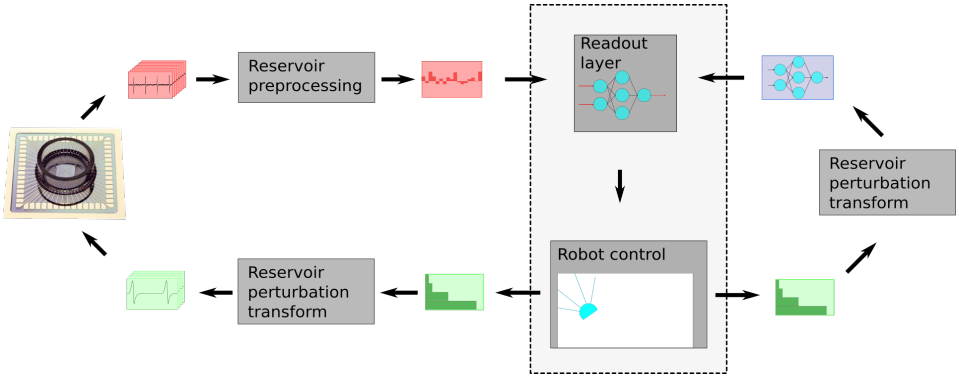


Figure 4.1: The figure shows direction of the dataflows that make up an experiment, stripping away all implementation details. By following the arrows two distinct loops are revealed, interersecting at the flow between the readout layer and robot control module.

close to the one shown in figure 4.1, however there is one major difference, namely that the two loops run in a *sequential* lockstep fashion rather than in *parallel*, or *on-line* as is the case with the cyborg. Running each loop sequentially makes more sense than parallel when the reservoir can be reset to an initial state, but this is not the case with living tissue whose topology and function evolves both over time, and as a response to previous experiments. This effect is exacerbated by the imperfect measurement of electrodes, where a cluster of neurons can seemingly vanish by physically moving away from an electrode.

A Sample Experiment

Before detailing the component systems, it is useful to describe an experiment to make the intended operation of the system clearer when described in detail in the following sections.

Before starting parameters which are considered static are pulled from a configuration file, which is also recorded into a new database record, ensuring that all parameters for a recording are stored as metadata. These parameters describe constants such as max electrode stimuli frequency, agent turn rate and similar. Once good values have been found they should not be altered (which is why they are stored in a file, rather than being user configured on a per-experiment basis). When the experiment is started the secondary dataloop will "cold start" the system by generating a set of random reservoir readout layers. The main dataloop will then start pulling data from the reservoir and feed it into the generated input layer, which in turn controls the robot. Output from the robot is then simultaneously sent through both dataloops. The primary loop uses it to generate reservoir perturbations while the secondary loop uses it to evaluate the current input layer. After all the randomly generated input layers have been evaluated, the secondary dataloop creates a new set of input layers based on the most successful layers from the previous generation. This is repeated until the experiment is terminated, hopefully after converging on a set of input layers capable of adequately solving the presented task. Since the system stores all data in a database it is possible to play back previous recordings without the system being aware of this (of course the perturbation will not do anything in this case), but it helps debug-

ging and ensures that all relevant information is stored in the database. For convenience all configuration can be done through a web interface which supports loading previously stored settings, visualizing live data and provide diagnostics in case of equipment failure.

4.1 Primary Data Loop

A more detailed view of the data loop is shown in figure 4.2. Since the primary dataloop interacts with the robot control, it must be part of the diagram (but is considered a separate component for configuration purposes). The two configurable parts of the primary dataloop are shown in the figure as the reservoir output filter and the perturbation transform. In the classical RC model the reservoir readout layer is typically illustrated as a single transform between reservoir and task. However when dealing with a physical reservoir, it is useful to break up this step into two separate filters, namely the preprocessor and the readout layer. The difference between these transforms is that only the readout layer may be modified as a function of its task performance. That does not mean that the preprocessor has to be static, the only constraint is that the update function cannot be aware of the task performance. Similarily, the reservoir perturbation generator may not be modified using information on task performance, but it can otherwise be modified. Exploring the parameters space of the primary dataloop is not pursued, as long as they are "good enough", and changing them will only render previous experimental results incomparable.

In the current implementation the chosen preprocessor is a simple averaging filter, but it can be changed to use a different model, such as linear of exponential decay if the experimenter so desires. The reservoir perturbation transform, which transforms a set of distances perceived by the robot into stimulus frequencies has three parameters: A list of electrodes which are receive the stimulus signal, the period between applying stimuli, and the stimuli waveform which is to be applied. Much like the choice of using a simple averaging filter, the default settings for the perturbation transform favors simplicity. For each sensor on the robot, a single electrode is chosen, and this choice is generally considered outside the interesting parameter space. Similarily a simple waveform is chosen, either a sine wave or a square wave. These samples are fixed, increasing the period results in a longer period between each sample, but the samples themseleves remain fixed. Again, for simplicity, a simple square wave is the default choice, thus the only parameter that is not considered fixed is the transform between distance and period. The chosen default for this transform is an exponential decay, when the sensor is facing a wall directly the square stimuli frequency is set to $30hz$ with an exponential dropoff to $1/3hz$ when the sensor perceives a wall at the maximum distance.

4.2 Core Reservoir Computer

The core reservoir computer is at the heart of both dataloops and is comprised of a readout layer and a robot control module. In the classical reservoir computer model there is little more than the core reservoir computer, a fact that can be illustrated by comparing it to the conceptual cyborg (figure 3.1) in which the reservoir itself and the core reservoir computer makes up the entire figure, save for a small interfacing box.

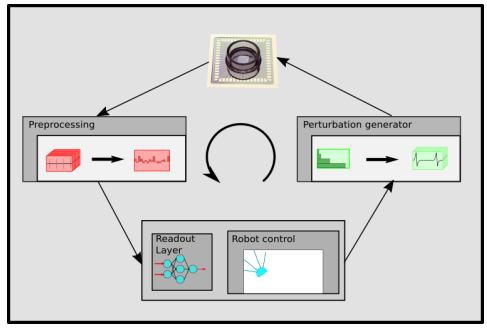


Figure 4.2: The figure shows the primary dataloop in more detail, with focus on the preprocessing transform and the perturbation generator. Since the readout layer and robot control is part of both the primary and secondary loop it is covered separately, thus from the point of view of the primary dataloop these two modules make up a single, opaque data-transform.

Readout Layer

The chosen readout layer is a simple feed forward neural network. As mentioned in the background, the choice of using a network with multiple layers as indicated in the figure implies that a non-linear problem may be solved entirely by the readout layer. For the proof of concept the topology of the network is not fixed to a topology. It is therefore up to the experimenter whether a network with a single layer or one with multiple layers should be used. In fact, there is no strict requirement that the readout layer must be an artificial neural network at all. While it might seem a wise choice to use a readout layer that has similarities with the reservoir itself, this line of thinking ignores the "anonymization" that happens between reservoir and reservoir computer, as shown in the figure 3.13 in the previous chapter. The final word on the choice of readout layer is that the artificial neural network approach has the following benefits: It can easily be extended from a linear to nonlinear classifier, and it is easy to modify during an experiment.

Robot Control

The robot control is a simulator that controls an agent with four "eyes" in a box world as shown in figure 4.3. As with all other parts of the system, variables such as the amount of eyes, sight range, the function between the output of the reservoir filter and robot movement, max turn rate and max speed can be changed on a per-experiment basis. As long as these parameters are "good enough", there is little scientific value in exploring them and they will remain fixed. The goal of the "game" is for the robot to not collide with walls. Since the best way to avoid this is to simply run in circles, the performance is evaluated by how close the agent came to crashing in a series of trials where the agent is faced towards the wall at different angles as shown in figure 4.4. Since the actions of the robot are decided by the output of the readout layer the action of the agent indirectly depends on the evaluation function since it is this evaluation that shapes the next readout layer.

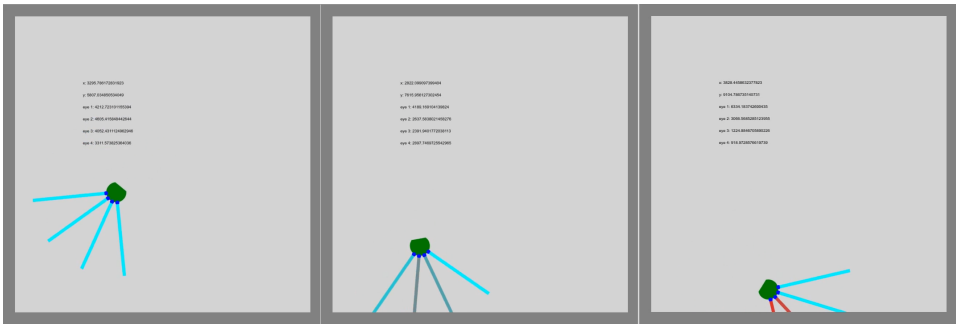


Figure 4.3: The cyborg maneuvering the simple walled world. In the first pane none of the cyborgs sensors are in range, thus no perturbation is generated. In the second pane the sensors detect a wall which is then relayed back to the neural cultures as stimuli. In the third pane the robot is very close to the wall, illustrated by the color change of the sensor "raycaster".

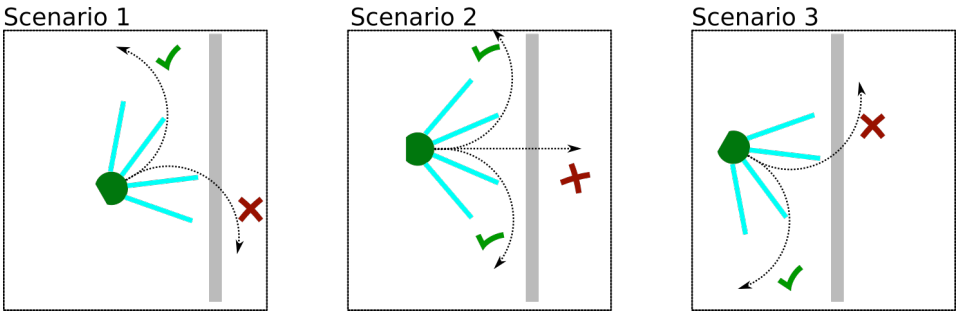


Figure 4.4: The evaluation strategy for a readout layer. In order to ensure that a given agent isn't avoiding walls simply by running in circles the agent is placed near a wall at different angles to see if it can successfully choose a left or right based on how it faces the wall.

4.3 Reconfiguration Loop

The reconfiguration loop, shown in figure 4.5 is the dataflow responsible for the on-line reconfiguration of the reservoir readout layer. On experiment start the reservoir filter reconfigurator supplies the readout layer module with a set of weights for the artificial neural network and records the resulting behavior. Each attempt is scored according to a scoring function as described in the previous section, and from this the reservoir filter reconfigurator generates new readout layers. The chosen method of readout layer generation is a *genetic algorithm* because of its ease of implementation and relative lack of bias.

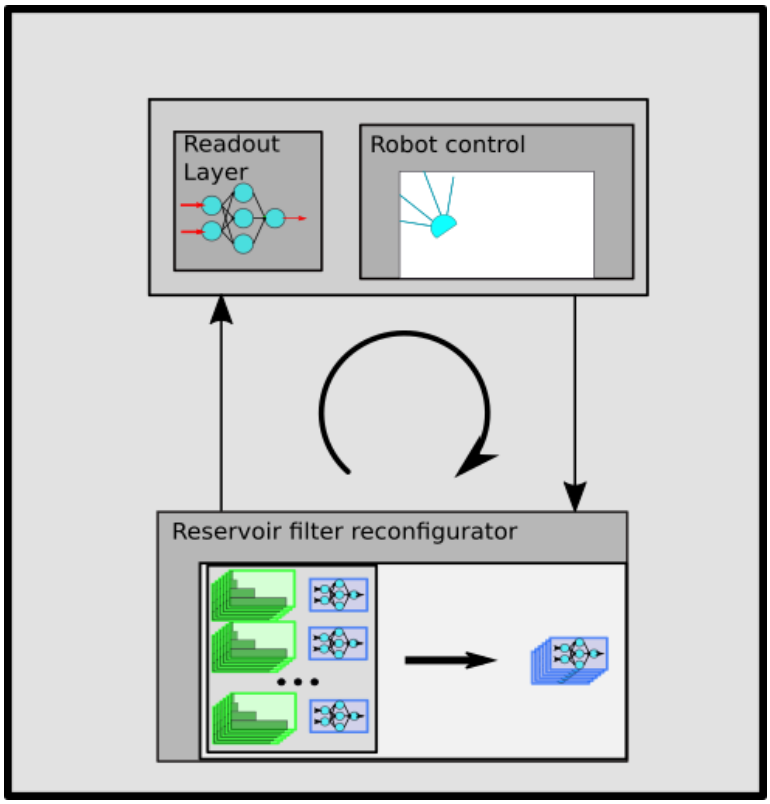


Figure 4.5: The reconfiguration loop evaluates the performance of the robot when using a given readout layer. The best performing readout layers are then used as a base in order to create a new generation of readout layers.

Chapter 5

Conclusion And Further Work

The future will be better tomorrow.

Vice president Dan Quayle (R)

The goal of this thesis has been to produce a working proof of concept for a neuro-digital hybrid. The system can record, store and broadcast neural recordings which can be accessed not only by the main experimenting platform, but by anyone with a working internet connection. Furthermore, the experimentation platform hosts an online self-reconfigurable reservoir computer capable of handling reservoirs that are "moving targets", i.e reservoirs whose behavior may change both during and between experiments. For ease of experimentation a web front-end is available, currently hosted locally only which provides a convenient and portable frontend for visualizing and setting up or scheduling experiments. Lastly, the reservoir computer can request perturbations to be applied to the reservoir, which is translated into stimuli requests which are broadcasted to the lab equipment. Every step in this process is verified to work, save for the last step in which setting the control registers of the stimulus generatior fails to trigger the reguested stimuli with the result that the closed loop system is not yet a loop. In spite of this, the system can be run as intended, verifying that the filtering of data, rendering of waveforms and simulated robot as well as perturbation requests are generated as specified.

5.1 Further Work

Finishing the stated goal of the thesis

While the lack of results make it hard to draw a scientifically valuable conclusion it at least makes the further work section easy to write. The obvious first task is to figure out what is causing stimuli to not work which is simply a matter of time. To this end a special DSP debugger chip has been ordered which should make stimuli working a matter of time.

Restoring Characteristics of damaged networks

In addition to running the experiment sketched described in this thesis, the flexibility and genericity of the pipeline allows other research with little effort. This flexibility will be leveraged in a planned experiment where the pipeline will record certain characteristics of a healthy neural culture and then attempt to restore characteristics of the network after it has been damaged.

Exploring capabilities of other reservoirs

The Socrates project is currently exploring the capabilities of artificial spin ice (ASI). Although the project is currently on the simulator stage, the computational capabilities of ASI have been shown to have many similarities to those of neurons. Once the project has produced a physical prototype it will be easy to plug in to SHODAN, which can provide a head-to-head comparison of neural and ASI capabilities.

Modeling the growth rules of neurons

By learning more about how neurons organize themselves models can be made that can more closely mimic the growth done by real neural tissue. This could help neural networks close some of the gap between their current specific intelligence bringing them at least one step closer to general intelligence.

Bibliography

[1] Aaser, P., 2018. Meame dsp github repository.
URL https://github.com/PeterAaser/MEAME-DSP

[2] Aaser, P., 2018. Meame github repository.
URL https://github.com/PeterAaser/MEAME

[3] Aaser, P., 2018. Shodan github repository.
URL https://github.com/PeterAaser/SHODAN

[4] Aaser, P., Knudsen, M., Ramstad, O. H., van de Wijdeven, R., Nichele, S., Sandvig, I., Tufte, G., Bauer, U. S., Halaas, O., Hendseth, S., Sandvig, A., Valderhaug, V., 2017. Towards making a cyborg: A closed-loop reservoir-neuro system. In: ECAL.

[5] Bar-Yam, Y., 1997. Dynamics of Complex Systems. Perseus Books, Cambridge, MA, USA.

[6] Burks, A. W., 1970. Essay one: von neumann's self-reproducing automata. In: Burks, A. W. (Ed.), Essays On Cellular Automata. University of Illinois Press, pp. 3–64.

[7] Cyborg, N., Jan. 2018. NTNU Cyborg.
URL https://www.ntnu.edu/cyborg

[8] Dreyfus, H. L., 1992. What Computers Still Can't Do: A Critique of Artificial Reason. MIT Press.

[9] Fernando, C., Sojakka, S., 2003. Pattern recognition in a bucket. In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J. (Eds.), Advances in Artificial Life. Vol. 2801 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 588–597.
URL http://dx.doi.org/10.1007/978-3-540-39432-7_63

[10] Jensen, J. H., Folven, E., Tufte, G., 2018. Computation in artificial spin ice. The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE), 15–22.

[11] Kurzweil, R., 2005. The singularity is near : when humans transcend biology. Viking, New York.

[12] Langton, C. G., 1991. Computation at the edge of chaos: phase transitions and emergant computation. In: Forrest, S. (Ed.), Emergent Computation. MIT Press, pp. 12–37.

[13] Lukosevicius, M., Jaeger, H., 2009. Reservoir computing approaches to recurrent neural network training. Computer Science Review 3 (3), 127 – 149.
URL http://www.sciencedirect.com/science/article/pii/S1574013709000173

[14] Lykkeb, O. R., Tufte, G., 2014. Comparison and evaluation of signal representations for a carbon nanotube computational device. In: Evolvable Systems (ICES), 2014 IEEE International Conference on. IEEE, pp. 54–60.

[15] Maass, W., Natschlager, T., Markram, T., 2009. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14 (11), 2531 – 2560.

[16] McCarthy, J., Aug. 1978. History of lisp. SIGPLAN Not. 13 (8), 217–223.
URL http://doi.acm.org/10.1145/960118.808387

[17] Miller, J. F., Harding, S., Tufte, G., 2014. Evolution-in-materio: evolving computation in materials. Evolutionary Intelligence 7 (1), 49–67.
URL http://dx.doi.org/10.1007/s12065-014-0106-6

[18] Pask, G., 1959. Physical analougues to the growth of a concept. In: Mechanisation of Thought Processes. No. 10 in National Physical Laboratory Symposium. Her Majesty's Stationery Office, London, UK, pp. 877–922.

[19] Schrauwen, B., Verstraeten, D., Campenhout, J. V., 2007. An overview of reservoir computing: theory, applications and implementations. In: Proceedings of the 15th European Symposium on Artificial Neural Networks. pp. 471–482.

[20] Sipper, M., 1999. The emergence of cellular computing. Computer 32 (7), 18–26.

[21] Spicher, A., Michel, O., Giavitto, J.-L., 2004. Cellular Automata: 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004, Amsterdam, The Netherlands, October 25-28, 2004. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, Ch. A Topological Framework for the Specification and the Simulation of Discrete Dynamical Systems, pp. 238–247.
URL http://dx.doi.org/10.1007/978-3-540-30479-1_25

[22] Strogatz, S., 2014. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. Avalon Publishing.

[23] Toffoli, T., 2004. Nothing makes sense in computing except in the light of evolution. Int. Journ. of Unconventional Computing 1 (1), 3–29.

[24] Tufte, G., Lykkeb, O. R., 2016. Evolution-in-materio of a dynamical system with dynamical structures. Proceedings of the European Conference on Artificial Life, 242–249.

[25] Wolfram, S., 1984. Universality and complexity in cellular automata. Physica D 10, 1–35.