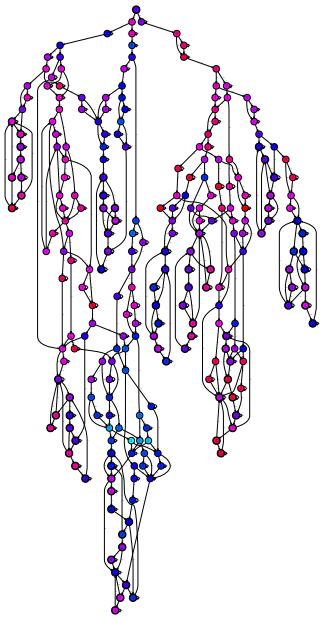
Master's thesis

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

Christopher Michael Vibe

# Practical Reservoir Computing & Echo State Property Metrics

Master's thesis in MTDT
Supervisor: Gunnar Tufte
June 2022

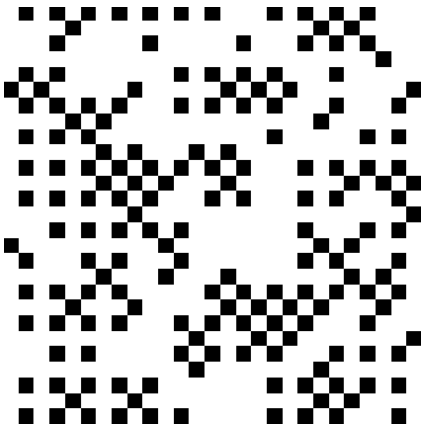


13x13 graph reservoir w. highlight on frustration. Christopher Vibe, Made w. Graphviz

NTNU
Norwegian University of
Science and Technology

Christopher Michael Vibe

# Practical Reservoir Computing & Echo State Property Metrics



Master's thesis in MTDT
Supervisor: Gunnar Tufte
June 2022

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

NTNU
Norwegian University of
Science and Technology

# Abstract

Reservoir Computing (RC) is part of a forced paradigm shift as traditional computing approaches arrive at the limits of physics with problems like the slowing of Moore's law and the power-wall. The unconventional computing framework, RC, exploits the dynamics present in natural systems, dubbed reservoirs, exemplified by the rippling surface of a bucket of water. Since many natural systems can be considered reservoirs, RC also provides a systematic way to study emergence, an intangible phenomena in nature. Furthermore, RC offers many advantages for computing such as low energy consumption in a system that scales well.

However, there is a need for metrics to gauge the potential of reservoirs, so that they can be selected or designed for a computational purpose. This thesis will attempt to measure the Echo State Property (ESP), which is analogous to short-term working memory; the amount of information temporarily present in the reservoir system over time. The end-goal is to quantify this property in units of bits by a metric developed in foundational work to the thesis, called the Echo State Buffer (ESB) [1].

Recalling that RC can be a number of natural systems, the reservoir of choice is an array of nano-magnets. The magnets are arranged on a flat grid, so that their north-south poles are aligned with this plane. An example arrangement is a chessboard, with each magnet glued to the center of each square in various directions. The magnets are unusual in that their poles can switch without physical movement, so the magnets poles reconfigure to avoid north-north, south-south clashes to the best of their ability. Note the parallel with a water surface that is settling. Having reduced clashes, the system enters one of many possible semi-stable steady states, a natural physical phenomenon. By perturbing the system further, say by flipping a single magnet, the system recommences its search for a stable configuration. With other reservoirs, like a bucket of water, it is hard to pause and analyse the reservoir response between perturbations as the system races towards a calm surface. The magnet poles however settle in various configurations and stay that way, without external stimulus. The reservoir is therefore ideal for applications like intermittent computing, where interruptions are expected during computation.

At present, interacting and measuring the physical magnet system is time-consuming, so the Spin Ice (SI) material will be simulated as an Artificial Spin Ice (ASI) with software that captures the physics of the interactions between the

magnets in the array. Simulations are relatively much faster to work with, but somewhat limiting. Therefore, any given ASI is converted to a graph as a surrogate model. The graph is a surrogate model, as it is a more succinct representation of the steady states and leads to a more scalable experimental setup. Furthermore the graph representation naturally allows for a discussion which borrows tools and ideas from graph theory.

The first part of the thesis will focus on setting up a practical RC system with the magnet reservoir with a non-linear function as a benchmark. With a more theoretical focus, the second part will propose metrics for measuring the ESP. The metrics are also evaluated by predicting the reservoir's success on a practical control system problem; balancing n-inverted pendulum on a cart, referred to as the balancing benchmark/problem.

On the non-linear benchmark, several RC architectures where developed, with the most accurate model achieving up to 67% accuracy on a challenging non-linear dataset. The ESB metric could not reliably predict success or failure on the balancing benchmark, but this was most likely due to a lack of experiment control. High accuracy's were also achieved in the balancing benchmark, with up to two balancing pendulums. Two pendulums was an insufficient amount of datapoints for nuanced comparison, but demonstrates potential for further development. Aside from the benchmark performances, a number of tools and approaches were developed that are relevant for further research. I.e., a promising method for analyzing reservoirs as graphs was developed. Furthermore, insight was made as to how a RC could be understood as a classifier, and how the RC system produces its results.

# Abstract (Norwegian)

RC er ein del av eit tvunge paradigmeskifte når tradisjonelle datatilnærmingar når fysikkens grenser med problem som nedbremsinga av Moores lov og kraftmuren. Det ukonvensjonelle datahandsamingsrammeverktøyet, RC, utnyttar dynamikken som finst i naturlege system, kalla reservoar, eksemplifisert ved krusningar på overflata i ei bøtte med vatn. Sidan ein kan sjå på mange naturlege system som reservoar, kan ein også nytte RC for å systematisk studere emergens, eit uhandgripeleg fenomen i naturen. Vidare tilbyr RC mange fordelar for databehandling, som til dømes lågt energibruk i eit system som lett kan skalerast.

Det er likevel eit behov for metrikkar som måler potensialet til reservoar slik at dei kan velgast eller utformast for eit berekningsføremål. Denne oppgåve gjer eit forsøk på å måle ESP, tilsvarande arbeidsminne. Det kan også skildrast som mengda informasjon midlertidig tilstades i reservoarsystemet over ei kortare periode. Sluttmålet er å kvatifisere denne eigenskapen i biteiningar ved hjelp av ein metrikk utvikla i forarbeidet til oppgåva, ESB [1].

Som nemnd kan RC gjere nytte av ei rekke naturlege system. I dette tilfellet er reservoaret ei samling av nanomagnetar. Magnetane er ordna i eit flatt rutenett, slik at nord-sør-polane er på linje med gitt plan. Eit døme på mogleg samansetting av plasseing er eit sjakkbreitt, der kvar magnet er plassert i midten av kvar rute med polane peikande i forskjellige retningar. Magnetane er atypiske, på den måten at polane kan byte plass utan fysiske rørsler i magnetane. Dette fører til at systemet kan unngå nord-nord og sør-sør samanstøyt etter beste evne. Her kan ein sjå likheitar med ei uroleg overflate av vatn som er i ferd med å roe seg. Etter å ha redusert samanstøyt, går systemet inn i ein av mange moglege halv-stabile *steady states*, eit naturleg fysisk fenomen. Ved å forstyrre systemet enno meir, til dømes ved å snu ein enkelt magnet, byrjar systemet å søke etter ein ny *steady state* konfigurasjon på nytt. Med andre reservoar, som ei bøtte med vatn, er det vanskeleg å sette systemet på pause for å analysere reservoarresponsen mellom forstyrringane. Magnetpolane finn dermed ein *steady state*, og blir verande i den tilstanden utan ekstern stimulans. Reservoaret er difor ideelt for bruk som til dømes *intermittent computing*, der forstyrringar er forventa under berekning.

No er samhandling og måling av det fysiske magnetsystemet tidkrevjande, så SI-materialet vil verte simulert som eit ASI med programvare som fangar opp fysikken til interaksjonane mellom magnetane. Simuleringar er mykje raskare å jobbe med, relativt sett, men noko avgrensande. Difor vert ein kvar gitt ASI kon-

vertert til ein graf som ein surrogatmodell. Grafen er ein surrogatmodell, sidan har er ein meir kortfatta representasjon av stabile tilstandar og fører til eit meir skalerbart eksperimentelt oppsett. Dessutan opnar grafrepresentasjonen naturlegvis for ein diskusjon som låner verktøy og idéar frå grafteori.

Fyrste del av oppgåva til fokusere på å setje opp eit praktisk RC system med magnetreservoaret, med ein ikkje-lineær funksjon som referanse. Med eit meir teoretisk fokus, vil den andre delen foreslå metrikkar for å måle ESP. Berekningane vert også evaluert ved å føreseie suksessen til reservoaret på eit praktisk kontrollsystemproblem; balanserande n-invertert pendel på ei vogn, referert til som balanseringsreferansa/problemet.

På den ikkje lineære referansa vart det utvikla fleire RC-arkitekturar, der den mest nøyaktige modellen nådde opp til 67% nøyaktigheit på eit utfordrande ikkje-lineært datasett. ESB-berekninga kunne ikkje føreseie suksess eller fiasko for balanseringsreferansa på ein pålitelig måte, men dette var mest truleg grunna mangel på eksperimentkontroll. Høg nøyaktigheit vart også oppnådd i balansereferansa, med opp til to balanseringspendlar. To pendlar gav ei utilstrekkeleg mengd datapunkt for nyansert samanlikning, men viser potensial for vidare utvikling. Sett vekk ifrå referanseresultata, vart det utvikla ei rekke verktøy og tilnærmingar som er relevante for vidare forsking. Det vil seie at det vart utvikla ei lovande metode for å analysere reservoar som garfar. Vidare vart det oppnådd innsikt i korleis ein RC kan verte forstått som ein klassifikator, og korleis RC-systemet produserer sine resultat.

- oversatt av Elise Skeide

# Acknowledgments

I would like to thank my advisor Gunnar Tufte for taking the time to fill his chalk board with figures as we would try to untangle the abstract subject matter. This past year has been confusing, but I feel you have inspired me.

Johannes Jensen deserves a special mention for his contribution in making the Flatspin simulation tools, as well as his other foundational work from which I could reference. He helping me grapple with the basics of the Flatspin simulation tools when I was getting nonsensical results during master project, which was the front runner to this thesis.

Finally, I would like to thank my friends, family, and those in-between, for supporting me and putting up with my constant talk of this thesis. Your contributions from proof-reading, making me food, and loving support, were invaluable, and I feel lucky to have you all in my life.

# Contents

# Figures

# Tables

# Code Listings

# Acronyms

**ASI** Artificial Spin Ice. iii–v, 3, 8, 13–15, 20, 34, 36, 43, 44, 79–81, *See:* ASI

**EML** Extreme Machine Learning. 2, 3

**ESB** Echo State Buffer. iii–vi, 4, 5, 45, 46, 51, 55–57, 59–61, 63, 65, 87–93, 95, *See:* ESB

**ESN** Echo State Network. 2, 4

**ESP** Echo State Property. iii–vi, 4, 5, 31, 41, 43–47, 55, 63, 65, *See:* ESP

**FPGA** Field Programmable Gate Array. 3

**MCA** Metric Component Analysis. 35

**NTNU** Norwegian University of Science and Technology. 10, 11

**PCA** Principle Component Analysis. 35

**RC** Reservoir Computing. iii–vi, 1–5, 7, 13–15, 18, 20, 21, 23, 31, 34, 38, 41, 43, 46, 47, 59, 65, *See:* RC

**RNN** Recurrent Neural Network. 2

**SCG** Strongly Connected Graph. 15, 43–45, 54–57, 60, 61, 65, 80, 81, 87, *See:* SCG

**SI** Spin Ice. iii, v, 3, 5, 7–11, *See:* SI

# Glossary

**ASI** An Artificial Spin Ice is a non-physical simulated SI material.. iii, 3, *See:* SI

**branch** A branch is a set of paths in a directed graph, all of which are capable of reaching the same SCG. A reference to a graph branch may refer to its vertices or edges. 43, 44, *See:* SCG

**ESB** Echo State Buffer, a proposed metric meant to quanitatively measure the ESP. iii, 4, *See:* ESP

**ESP** The Echo State Property implies a system has fading memory; for a given input, previous inputs have an influence on the next system state, but this influence fades over time. iii, 4

**k-means** A technique for automatic categorization, where the K specifies the number of categories the algorithm should return. The algorithm attempts to minimize the differences within each category while maximizing the difference between categories. E.g. With K=2, considering a group of average adults with basketball players, the algorithm may group the people in groups of two *tall*/*average* or maybe *male*/*female*. see. 35

**multiplexing** Multiplexing combines few inputs to generate many unique outputs over a shared medium. The easiest way to explain it is by understanding the binary number system, where a small number of bits can produce a surprisingly large set of unique natural numbers. E.g. 3 bits can represent numbers from 0 to 7: $0101 \rightarrow < 0,1,0,1 > \cdot < 8,4,2,1 >= 0x8 + 1x4 + 0x2 + 1x1 = 5$.. 17, 38

**random forest** A supervised machine learning technique for automatic categorization. The algorithm can be represented as a directed graph, such that each node is a filter and each edge is a path taken depending on the filter. For example, starting at node0, if a variable is less than 10, go to node1, else go to node2, etc. The filter threshold is usually set by machine learning to categorise inputs according to a target class.. 43–45, 54, 61

**RC** Reservoir Computing is a theoretical framework that leverages the dynamics of physical systems to extract useful computation. Classically, it is divided up into three layers; input, reservoir and readout. iii, 1

**SCG** A Strongly Connected Graph has at least one path between each pair of vertices, i.e., all vertices are reachable upon graph traversal. 15

**SI** A Spin Ice is a material with internal frustration leading to multiple equilibrium states. Internally, this can be exemplified by two neighbouring particles with their north poles pointing towards each other, causing a repelling force. Unchecked, these particles would flip to get a south-north alignment, but in this frustrated equilibrium state, these particles cannot reconfigure due to other physical circumstances which are even less favourable. iii, 3

# Chapter 1

# Introduction

## 1.1 Disclaimer

This work is a continuation of a master's project by the same author which can be found in appendix C. Thus, there may be instances where portions of the thesis recycle ideas, figures, or text from previous work. These instances will be marked by a sentence like this one referring to the previous work and a citation [1]. Most of the figures were originally made by the author, but some figures were manipulated images from the Flatspin simulation library [2]. The textual content was compared on www.prepostseo.com via their free online plagiarism tool to mitigate identical text instances.

## 1.2 Reservoir Computing

Having been independently discovered in various fields, this unconventional computing framework has taken on various pseudonyms, but has since the early 2000s been dubbed Reservoir Computing (RC) [3]. Since a reservoir in RC is merely a dynamic system, the theoretical framework presents an opportunity to study the phenomena of emergence in natural systems. The idea is simple: the sum of a system's parts being greater than their individual worth. The theoretical framework is quite flexible and shares many properties with systems in biology and physics. E.g. a slime mould manages to solve mazes as a collective entity without possessing anything remotely resembling a brain [4]. The intelligent behaviour seems to emerge from the unicellular organism as a whole. This is in stark contrast to the modular recipe for computation in classical computers, as described by the Von Neuman architecture [5].

The reservoir in RC is exemplified in the material computing subfield, which leverages the natural properties of materials for computation. As indicated by the name, a literal bucket of water can be used as a material reservoir. When this reservoir is perturbed, water waves arise on its surface, and can be interpreted as a form of computation [6]. Identifying the inputs and outputs in the RC system:

The perturbations represent a series of inputs exciting the water surface. Hence, a mapping has occurred translating a low dimensional input to a high dimensional water wave representation. The reservoir is now a representation of the inputs, distorted and mixed in various ways. The next step is a readout layer applied to the reservoir, which involves machine learning to recover information or extract added meaning from the new representation.

RC is a subfield to Recurrent Neural Networks (RNNs), and has been useful in solving various problems from learning melodies in Echo State Network (ESN) [7] to solving differential equations [8]. However, applying machine learning techniques such as back-propagation to RNNs is notoriously difficult due to their characterizing feedback property, where inputs can mix with previous outputs like feedback in a microphone near its speaker. The way RC gets around this limitation is to outsource the computational strain to its reservoir, a natural dynamical process with inherent computational power. RC can offer solutions with desirable characteristics found in biological evolution, such as low energy, parallelism, or fault tolerance [9]. The classical RC model can be described as a three step procedure as illustrated in figure 1.1.

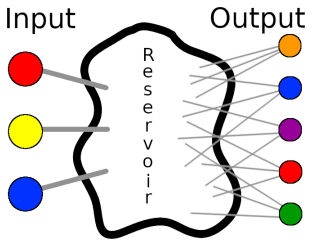



**Figure 1.1:** The classical RC model: input layer (left), reservoir (middle), and output layer (right). Note that the input has been mapped to a higher dimensional representation, artistically shown by the mixing of prime colours, resulting in a rich palette in the output layer. The reservoir is flexible; in this case, the inputs are spatially multiplexed with three input nodes, but it can be also be temporally multiplexed to a sequence of vectors, for example by feedback effectively connecting the output to the input. The output nodes are part of the readout layer, and typically consist of a single linear machine learning layer. This figure was directly taken from previous work [1].

Extreme Machine Learning (EML) [10] can be used as a way to understand RC from a machine learning perspective. In machine learning, the idea is to copy the connection strength of a biological neuron, as found in brains. Akin to real brains, the neuron strength is emulated by a weight parameter which scale signals in the systems as to favour desirable outputs. As opposed to learning an optimal value for each weight, EML sets most of the weights to a random value. This is because it takes great resources to learn satisfactory weight values through the learning mechanism known as back-propagation. Instead, the machine learning techniques are dedicated to the final layers in the neural net, reducing the amount of trainable weights. RC differs from EML in that the former in arranged so that

feedback connections are present. I.e., a reservoir's response is a product of its current and last state, just like in the aformentioned microphone example. EML acts as if the input enters a pipe and exits on the other side without interacting with the entry-point of the pipe; a feed-forward pipeline.

The randomly set weights in EML are surprisingly useful. Likewise, RC exploits subtle properties in physical systems. Research involving an Field Programmable Gate Array (FPGA), a massive programmable circuit, illustrates this well by Thomson [11]. The FPGAs were programmed with evolutionary algorithms to evolve circuits that solve a given task. To the researcher' surprise, he found that the FPGAs had been configured in non-sensical circuits, but that they performed well. Adding to the confusion, applying a successful solution to other FPGAs, by means of identical configuration, gave very poor results. The RC system had taken advantage of the underlying analog behaviours that are typically ignored during circuit analysis, which was tailored to the individual hardware of each FPGA. Thus, the takeaway in many physical dynamical systems it is likely that there exists inherent useful computation. With some manipulation of the system the useful parts can be synthesized.

## 1.3 Introduction Of The Chosen Reservoir

The chosen reservoir is an array of nano-magnets which belongs to a group of materials called Spin Ice (SI). The SI will be simulated with simulation software named Flatspin, and is therefore aptly named Artificial Spin Ice (ASI), a virtual material used as a reservoir. Both SI, and ASI are further detailed in section 2.

The ASI reservoir has been chosen for several reasons, the most important being that it is relatively easy to maintain experiment control, and saves time from not having to interact with a physical setup. Furthermore, the dynamic system state can be easily frozen in time, as opposed to a bucket of water, facilitating analysis. Most practically, the ASI simulation allows for a deep dive into the topic with minimal barriers to entry from a physical setup. Focusing on the properties of SI, the reservoir seems to be a promising candidate for RC showing signs of possessing the qualities discussed in section 1.4.

The reservoir simulations will be used to create useful graph abstractions of the system. The graphs are made by exhaustively exploring the reservoir states with a depth first search algorithm, and is thus a compressed version of all possible combinations of inputs and steady states. The new representation conveniently unlocks tools from graph theory in the analysis.

## 1.4 Identifying a Good Reservoir

Certain properties have been identified as desirable for a reservoir to be suitable for computation. The reservoir must have numerous components, with non-linear, local interactions, resulting in a highly dynamic response when perturbed. The

response may be chaotic, but should not be random. Additionally, the system must have a fading memory property, meaning the system must forget previous inputs over time. This last idea was introduced as the Echo State Property (ESP) in the ESN [7].

During the foundational work phase of this thesis [1], the idea was to explore ways to measure the ESP. The approach was a direct extension of efforts by Jensen et al. to uncover computational properties through two metrics: kernel quality and generality [12]. Kernel quality is a measure of a reservoir's ability to distinguish temporal input, quantified by the rank of *nxm* matrix. The matrix is organized so that the *n* rows represent the state of each of the reservoir's magnets spins, after perturbation from *m* unique input sequences. If kernel quality is high, then different inputs should map to unique states. On the other hand, generality is a measure of the reservoir's sensitivity to similar inputs. The method for generality is identical to kernel quality, except for the inputs, being sequences that are similar instead of unique. A good reservoir is then identified by high kernel quality and low generality. A high-scoring reservoir would then be capable of many classification categories simultaneously keeping similar inputs in the same category. The metrics are a good way to evaluate the potential of a reservoir, but the ESP remains difficult to quantify directly.

The goal of this thesis is to attempt the quantification of the ESP in the chosen reservoir. This is a direct continuation from a previous exploration on the topic where the concept of the Echo State Buffer (ESB) was introduced as a proposed metrics for measuring the ESP [1]. Recalling that the ESP is the retention of temporary information in a reservoir, the ESB attempts to quantify this property in bits.

The following is a metaphor aimed at providing more intuition on the ESP, as it is more fleeting than permanent storage memory. If you were to stand in an empty church and clap your hands once, a reverberating echo would come from the walls. If you were to introduce a pattern of clapping for a minute, then how long of a clapping history could the echo's temporarily store if it suddenly ceased? This is the ESP, roughly measured by the claps you can count, from the moment you stop the clapping pattern until the echos fade to silence.

As the sound waves reach your ears, they would have travelled various distances, despite coming from the same temporal origin at the moment of the clap. Furthermore, the sounds come back distorted, with some being filtered of high frequencies. The clap in this scenario can be thought of as an input to a RC system, and the computation is a combination of the inputs experiencing distortions, filters, and mixing. Since the ESP dictates how many claps are remembered at a given time, it also determines how much input mixing is possible. Therefore, the ESP acts like a form of working memory, and is fundamentally important in identifying a good reservoir for useful computation. From a more academic perspective, the ESP should be tightly related to the upper limit of instantaneous space complexity an algorithm can require of a given RC system.

## 1.5 Motivation for Metrics on the ESP

The thesis has two parts, a practical and theoretical chapter. Although the practical chapter is the largest, the experiments conducted have the end goal of developing metrics for the ESP. This is because the computational potential of a RC system greatly relies on the ESP. Furthermore, any practical implementation could arguably be a success or failure purely due to seemingly arbitrary design choices, making it difficult to compare reservoirs. However, by developing metrics for the ESP, it is possible to decouple the quality of a reservoir for a given task with any other design choices that might come into play. Note that an intuition for the ESP has been established, but there is no known baseline to compare to. Therefore, the approach is to validate the ESB by its ability to predict performance on a benchmark where the ESP is essential for success.

Performance will be compared to simple graph properties such as nodes, edges, cycles, as well as the ESB. Jump to figure 3.3 for an example of a graph consisting of nodes representing states, and edges representing transitions. For the graph related metrics, the method can be summarized by converting the reservoirs to graphs, and borrowing analysis tools from graph theory. And for the ESB metric, the method can be summarized as probing the reservoirs with repeated input sequences with special properties, and looking for patterns in the response. The ESB metric measures how many times a sequence of information can be repeated as input to a reservoir before the reservoir's state response starts repeating as well. It is argued that when the reservoir's transitions start predictably repeating, a point has been marked that indicates the system's memory has been reset. Upon a repeated input, if the system state starts from a point it has already started from sometime in the past, then it is as if the system has forgotten that this particular state has already been visited. The echo state property would influence how long the state transition history could be in such a scenario as an outcome of fading memory. The ESB and reservoir to graph conversions are further explained in sections 7.2 and 3.2, respectively.

The research is foundational, so priority is not primarily to produce a product. Looking ahead however, the chosen reservoir should be able to perform with minimal energy expenditure. The reservoir is also a good candidate for intermittent computing, due to an inherent property of SI, where the system state is frozen in time without a need for external maintenance. With a broader outlook, the research may aid in understanding many natural systems like the brain, as many of them can easily be considered reservoirs.

# Chapter 2

# Background

## 2.1 Unconventional Computing

Beginning with conventional computing, the personal computer is an example of top-down design with greatly engineered precision. Each component of the computer is highly modular, with well-defined scope and function. On the other hand, RC is an example of unconventional computing. The approach is bottom up, and tries to exploit the inherit intelligence in a system. As mentioned in section 1.4, a good reservoir has numerous components. Typically, each component is relatively simple in such systems, and zooming in on a couple of them, one would think the system would not have any potential of intelligent computation. The idea is that intelligence emerges from the dynamic response of these components; the intelligence of the system is greater than the contribution of each component in isolation. This is something typically observed in nature, but somehow the mechanisms for how these systems become intelligent often elude scientists. Hopefully, RC can be a more successful way to uncovering some of these mechanisms, with fewer preconceptions as to how intelligence should look like.

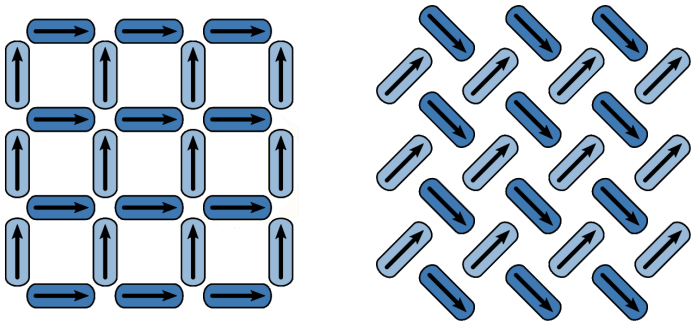
## 2.2 Spin Ice

Before further delving into RC, this section will allude to the reservoir used in this thesis. The reservoir is an array of magnets that is part of a group of materials named SI. Essentially, a SI material contains many small magnetic particles that are slightly loose. Specifically, the magnetic orientation of a particles north-south poles may change in search of the configuration with the lowest potential energy. This orientation is often referred to as the magnetic spin, and explains the first letter in the acronym SI. With many such loose spins, the material can have many possible unique configurations. Thus, by applying an external magnetic field to such a reservoir, a complex response is formed, which can be regarded as a calculation of sorts. This property is leveraged in RC and is elaborated on in section 2.3. The “I” in SI is just a reference to water in the ice phase, one of the first places

this molecular organization was observed [13].

The SI will be simulated with Flatspin, a GPU-accelerated simulation library in python, developed by Jensen et al. [2]. As put in the documentation: "flatspin can simulate realistic dynamics of millions of magnets within practical time frames." Hence, the reservoir is virtual, an *artificial* Spin Ice (ASI). Flatspin and ASI will be further discussed in section 2.3.

The SI reservoir is well represented both as a physically printed magnetic array, or simulated in Flatspin as an ASI. Features such as spacing, magnet geometry, and other properties can thus be engineered, and is henceforth referred to as the geometry. Figure 2.1 showcases some examples of a magnetic ASI with different geometries made with Flatspin. Note that the reservoirs discussed will be frequently referred to by their size. E.g. in the both of the figure below, the size is 3x3, meaning the pattern is repeated over 3 rows and 3 columns. As such, the description does not mean that there are 9 magnets in the array.



**(a)** Square **(b)** Diamond Pinwheel

**Figure 2.1:** Standard geometries generated with tools from Flatspin and modified from the cited: [2].

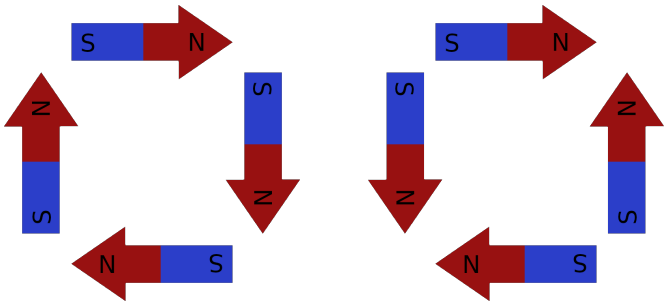
## 2.3 Spin Ice as a Reservoir

The scientific community has noticed SI for their special properties producing phenomena such as emergence, quantum tunnelling, magnetic mono-poles, and more [14]. Recently, in 2021, Giorgio Parisi was awarded the nobel prize in physics from his study of spin glass, a close cousin to SI.

It may come as a surprise that all known materials have particles with magnetic poles. On a macro scale, this may not be intuitive, as magnetic properties often cancel out. The cancellation occurs depending on if the internal orientation of a substance's magnetic particles or spins. In a fridge magnet, the particle spins are aligned on a micro scale, but they can also cancel out, like in a potato. In SI materials, the spin orientations settle in semi-stable configurations. This is because the magnetic particles are frustrated, meaning each magnetic particle with a north and south pole wish to avoid north-north south-south clashes. This is

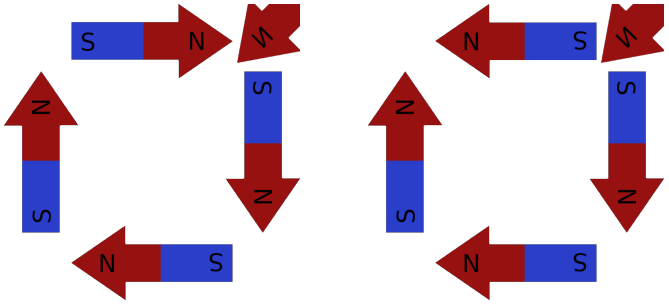
simply because magnet poles with the same polarity repel each other. The concept of frustration is also graphically explained in figure 2.2.

In natural dynamic systems, there is an ever present search for the lowest energy level. The idea can be compared to a water surface that is settling. This also holds true in SI, however, as the particles attempt to reduce frustration, there are many ways for the magnetic particles to arrange themselves. This is because physical circumstances in the substance partially limit the particles in their re-arrangement. The substance is considered semi-stable because the spins of a SI can be coaxed into various stable low energy states by external influences. Figure 2.3 illustrates two such configurations. Note how flipping the topmost middle magnet leads to the same amount of overall frustration in the system, allowing for many stable configurations. Flipping this top-magnet is unfavourable either way, as flipping polarity costs energy. I.e. the system is not stuck between states, but settles for one of them with indifference. Figure 2.1 shows examples of designed geometries that can be explored to see if they also possess this type of behaviour. In fact, some of these geometries have been manufactured, with figure 2.4 illustrating a real SI based on nanomagnets.




**(a)** Low frustration **(b)** High frustration

**Figure 2.2:** Two SI substrates exhibiting low and high frustration (left to right). In this simplified system, the physical circumstances allow the magnetic particles poles to flip, so north and south switch position, but restrict all other degrees of freedom. The number of north-north or south-south clashes in the substrates particles dictate both frustration and stability. Note that the high frustration case would need an external force to remain in this state or it would quickly transition to the low frustration state.




**(a)** Configuration 1 **(b)** Configuration 2

**Figure 2.3:** Two of many semi-stable SI configurations with the same minimal potential energy as well as frustration. In this simplified system, the physical circumstances allow the magnetic particles poles to flip, so north and south switch position, but restrict all other degrees of freedom. Note that flipping the topmost middle magnet in a) leads to b), but that the number of north-north south-south clashes remains the same.

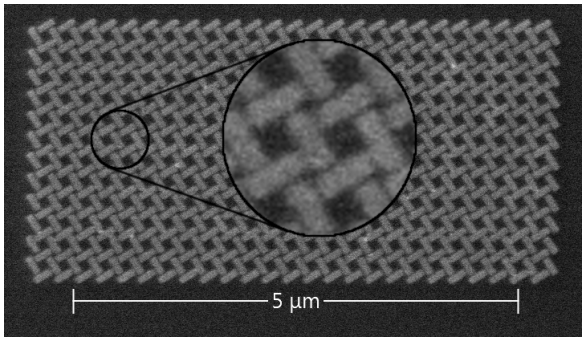



**Figure 2.4:** A physical reservoir: SI as a magnet array in a diamond pinwheel configuration. Courtesy of the SOCRATES project at Norwegian University of Science and Technology (NTNU). The black circle is a zoomed in area of the array, revealing individual magnets as white rectangles at the nanoscale. This figure was directly taken from previous work [1].

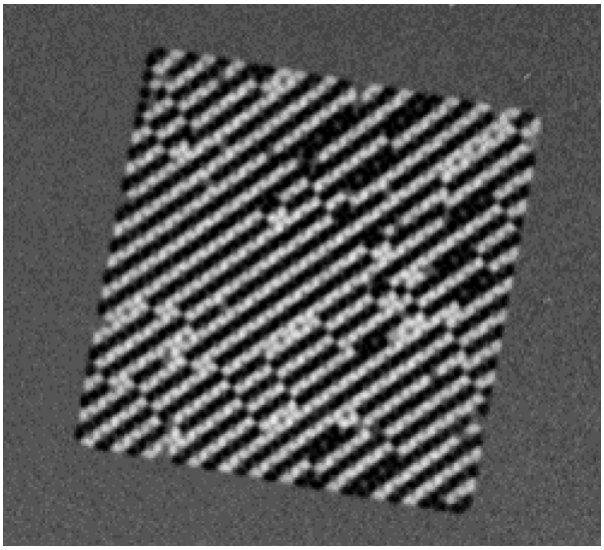


**Figure 2.5:** Physical reservoir readout of a physical SI can be achieved with PEEM, a photo-emission electron microscopy method. Courtesy of the SOCRATES project at NTNU.

# Chapter 3

# Method - Practical RC

This method chapter has hidden some numerical experiment parameters for clarity. See appendix A.1 for parameters pertaining to the ASI models. See appendix A.2 for parameters associated with section 3.3.

## 3.1 Simulating Spin Ice with Flatspin

This research will employ a virtual reservoir, a simulated 2D nanomagnet array, dubbed ASI. The simulations will be carried out with the Flatspin simulator [2]. Jensen et al. [12], has previously examined this virtual reservoir with Flatspin for properties needed for RC, demonstrating that they can be found after some tuning. Tuning in this context is a general term including parameters such as magnet spacing, magnet geometry, geometrical magnet arrangement, and more. The experiment parameters were tuned identically to this preliminary exploration of RC in ASI.

This research borrows Jensen's approach, using binary strings as input, which in turn translates to magnetic field perturbations on the reservoir. The reservoir model represents information based on concepts from mechanical computing [15] with 1s and 0s represented by magnet spins. The chosen reservoir models are designed so that they are frustrated. As discussed in section 2.3, this is desirable so that the systems behave dynamically as they attempt to reduce frustration. See figure 3.1 for a rendering of a virtual ASI reservoirs employed. Note that the figure features a diamond pinwheel geometry, as introduced in figure 2.1b. The idun cluster at NTNU was used to convert reservoirs to graph by a depth first search algorthim of the ASI reservoir [16].

## 3.2 A Reservoir as a Directed Graph

In RC, the impression of a given input or perturbation of a reservoir can be interpreted as traversing a directed graph. Each node would then represent a unique 2D matrix of spin states, and each edge a transition between these states for each

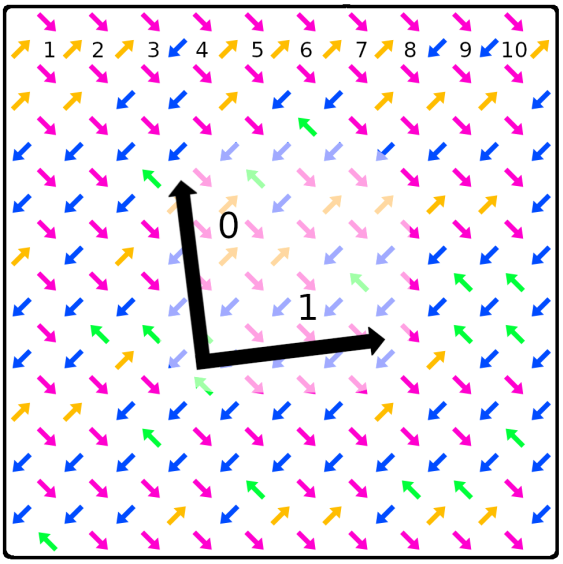



**Figure 3.1:** A virtual reservoir: ASI with a 10x10 diamond pinwheel configuration. Note that the small arrows define each magnet's directional reference point for their net magnetic spin. Their actual spin orientation is colourized based on their North-East-South-West orientation. The small coloured arrows reveal regions of frustration, as seen by the non-trivial colouring. The large black arrows represent the global external magnetic field used to perturb the reservoir. This figure was directly taken from previous work [1].

consecutive input. Such a graph is capable of a deterministic prediction of the reservoirs state for all possible inputs and is exemplified by figure 3.3. Each node in the graph is an array of spin states, as illustrated in figure 3.4. The graph representation is thus a more practical way to analyze the properties of the reservoir, and can serve as a surrogate model of the ASI. Figure 3.2 shows the conversion from the physical reservoir model to a directed graph. Larger directed graph examples can be found in section A.5.

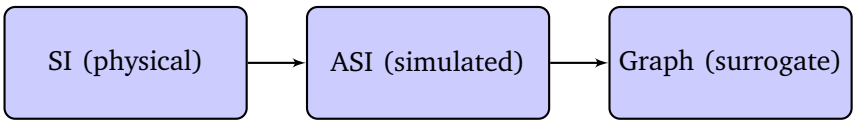



**Figure 3.2:** Reservoir conversion pipeline showing the relationship between figures 2.4, 3.1, and 3.3.

## 3.3 Experimental Setup

Recall from figure 1.1 that the classical RC model is quite flexible. A series of architectures were devised to reveal how the RC system should be orchestrated.

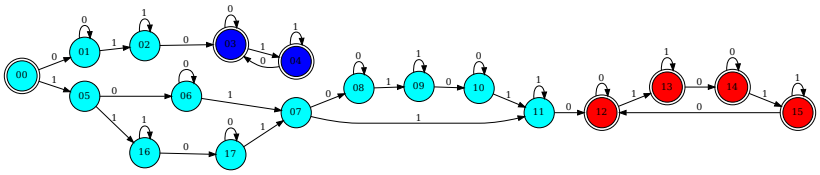
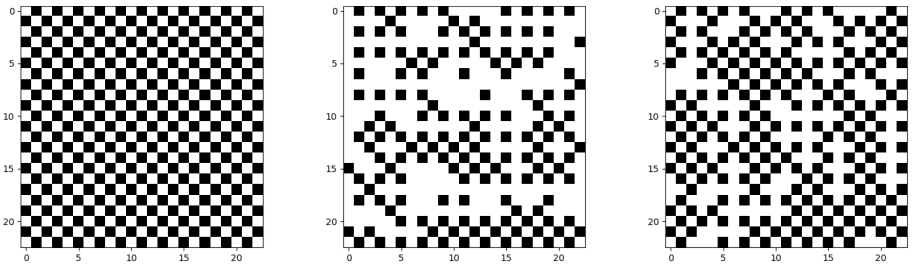



**Figure 3.3:** An ASI reservoir with a 11x11 diamond pinwheel configuration as a directed graph. All possible unique spin states are nodes, with inputs as transitional edges. See figure 3.4 for closeups of some node spin states. All Strongly Connected Graph (SCG) sub-graphs are coloured by size and marked with a double circle. The start node is also marked with a double circle.



**(a)** Node "0" **(b)** Node "1" **(c)** Node "16"

**Figure 3.4:** Spin State representation of select nodes from figure 3.3. The magnet spins are represented by black and white. Note that the inter-magnet spacing in the pinwheel configuration is also white, thus the initial saturated state, node 0, is just a white square. Compared to figure 3.1, there is less information, because information about frustration between individual magnets is lost. On the other hand, it is easier to see emergent patterns in the state response.

To compare the models, a non-linear 2D target image was selected as a benchmark challenge, illustrated in figure 3.6. The idea was to perform a qualitative and quantitative analysis of the model's ability to predict the target image, revealing insight into a good RC design. The quantitative analysis was made with a simple accuracy metric defined in section 3.3.1. The qualitative analysis was made by comparing the target image with a model's predicted image.

The machine learning optimizer AdamW [17] was chosen in an effort to keep the setup the same for all experiments, as it is known to perform well without too much fine tuning of hyper-parameters. Furthermore, high values were chosen for batch size, epochs, to reduce the chance of unfair comparisons.

### 3.3.1 Defining Accuracy

Accuracy can be defined in many ways for a predictive model, even for a single value regression task. This metric was kept as simple as possible by defining a

successful prediction as accepting a 5% error from its target value. See listing 3.1

**Code listing 3.1:** Definition of accuracy

```
def calc_accuracy(model, x, y, tolerance=.05):
    '''
    Returns the accuracy over m data samples with n features.

            Parameters:
                    model (torch.nn.module): Predictive model
                    x (torch.tensor)       : Data samples to be predicted [mxn]
                    y (torch.tensor)       : Target values [m]
                    tolerance (float)      : Error tolerance

            Returns:
                    accuracy (float): The fractional accuracy.
    '''
    predictions = model(x)
    deltas = torch.abs(predictions - y)
    correct = torch.sum(deltas < tolerance).item()
    m = len(y)
    accuracy = correct / m
    return accuracy
```

### 3.3.2 Defining a Non-linear Target

A non-linear 2D target image was chosen such that few values on the image could be determined with only one of its parameters. Furthermore, as seen in figure 3.6, the image has many gradients which makes it easier to qualitatively assess fine-grained feature potential. The three large columns makes it possible to assess more coarse grained features.

$$f(x,y) = -\cos((x-0.1)y)^2 - x\sin(3x+y)10$$

**Figure 3.5:** The non-linear target function. Visualized in figure 3.6.

## 3.4 Architectural Layer Definitions

Note that the input and output layers are linear for all tested architectures. This is to ensure that all non-linear computation can be attributed to the reservoir layer, and not solved in either of the input or output layers. This is strict, and not needed for practical application, but provides this research with more experiment control. Section 3.5.1 also aims to gain further experiment control in this regard.

Below is a quick definition of terms used to describe the method. A pipeline, is a series of sequential steps applied to something like a factory line. Input goes in the pipe on one end, and the pipe produces and output on the other end. A more general term expressing the same concept of a pipeline in networks, is feed-forward network, where information flows one way. Entanglement refers to which

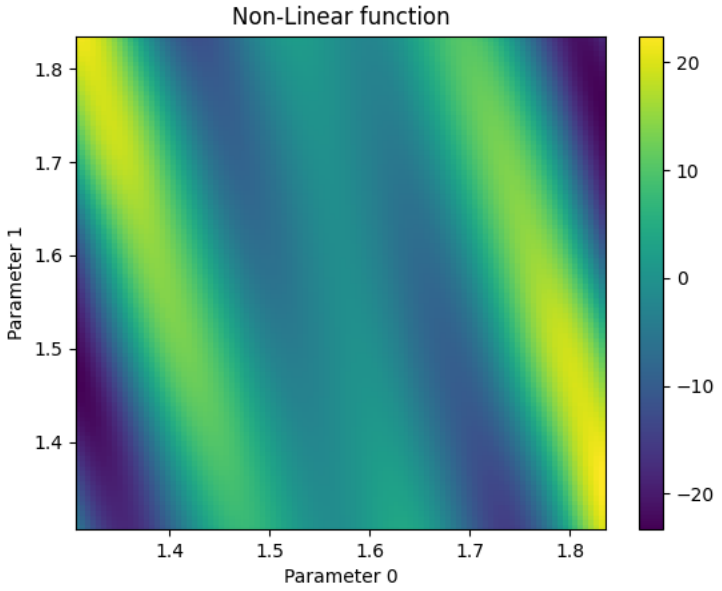



**Figure 3.6:** A non-linear target image. Defined by equation 3.5

input variables have informational presence in a given reservoir. I.e. if a reservoir is not entangled with a variable, then the value of the variable cannot be inferred by the reservoir's state alone. The term can also be used in the case of encoding, if there is no entanglement, it means the input variables were encoded independently, which makes it easier to allocate variables to separate pipelines.

### 3.4.1 Input Layer and Encoding

The input layer accepts a set of variables represented by normalized floating point numbers in the range $[0, 1]$. It subsequently converts each number to a string of 1's and 0's. E.g with two variables and an encoding scheme consisting of multiplying by 10, rounding, then translating to binary: $\{0.1, 0.5\} \rightarrow \{0001, 0101\}$. Note that the example indirectly introduces multiplexing as a concept, and is useful for understanding the input layer. See the glossary for a definition.

The input layer introduces a upper-bound on the prediction image resolution. If the input variables can maximally express $s$ states, then they can maximally produce an image with $s^2$ pixels. Therefore, it was beneficial to have a binary encoding in the input layer, as this is the encoding with the most combinations. However, practical efforts so far with a binary encoding did not work well. There are many self-references in the graph-input bits which can easily be "lost" in the reservoir, and may partly explain why the binary encoding was difficult. When translating a binary number to decimal, it is crucial to correctly read the leftmost bits, while the rightmost bits are not as important. On the opposite side of the spectrum, a tally encoding was introduced based on these experiences, defined

below.

Introducing a tally encoding remedies this issue because each bit has equal value, i.e. all equal to one upon conversion to decimal, e.g., $000111 = 3$. A weakness was introduced however as the state may get repeated after an input, due to self-referencing edges. This can be quickly verified by traversing a graph, such as 3.3, with any imaginary binary string. In other words, one should avoid too many consecutive 0's or 1's. To remedy this a fixed shuffling technique was introduced, so that the input encoding had a better distribution of 0's and 1's, e.g. $000111 \rightarrow 100101$. Note that some attempts were made to compromise these two encoding methods with a cross between tally and binary encoding called distributed encoding. To lower the scope of this thesis it can be found in appendix A.1, but will not be further discussed. Recall that the input layer introduces a upper-bound on the prediction image resolution. If there are b bits per input variable, then note that the tally encoding can only express b states, while the binary encoding could express $b^2$ states. This means the prediction image resolutions must be $bxb$, assuming there is no multiplexing, as considered in section 5.2.4.

Many encoding schemes were tested, but none were as stable as the tally encoding. This property was critical to get sensible results from the various architectures. To ensure there was influence from all variables in the encoding the variables where first encoded separately and concatenated. Subsequently the resulting bit string was shuffled, in a fixed manner, and interleaved before moving on to the reservoir layer, discussed in section 3.4.2.

### 3.4.2 Reservoir Layer

The reservoir layer accepts a single binary input string and outputs a high-dimensional binary state that potentially represents the input. The layer represents the reservoir by a graph, as discussed in section 3.2. Hence, the reservoir response can be emulated by processing one input bit at a time in a graph traversal. After the traversal, the reservoir layer returns an end-state describing all the magnets spins, just like in figure 3.4.

### 3.4.3 Output Layer

The output layer is the only part of the RC system that uses machine learning and is a dense single-layer readout of a reservoir state. The layers' purpose is to extract meaning from the reservoir state automatically.

The linear readout transformation can be summarized by equation 3.7. Each magnet state spin was encoded as $\{-1, 1\}$, because if they were encoded as the more intuitive $\{0, 1\}$, then there would be no effect from multiplication with the learned weights when the spin state is 0.

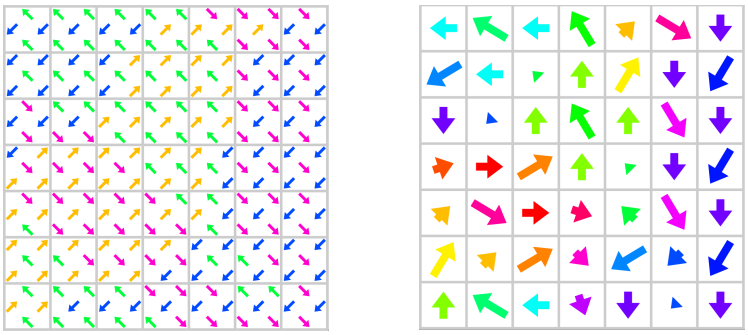
$$O(R,W) = \sum R_i \cdot W_i + B_i, \qquad R_i \epsilon \{-1,1\}$$

**Figure 3.7:** A mathematical definition of the output layer: the scalar multiplication of each reservoir magnet spin with a weight followed by the addition of a scalar bias term. Both the weight and the bias term are floating point scalars set by machine learning. Note that the operation can intuitively be thought of as a dot product, overlaying the spin state image with a weight image. This is somewhat inaccurate however, as the bias term should be added before summation.

### 3.4.4 Pooling Layer

The pooling layer is used to reduce the high-dimension of the reservoir states. Such a reduction is needed to allow architectural connections between several reservoir layers, and can be seen as an encoding of a reservoir state, as opposed to encoding a normal decimal input variable. Imagine a grid superimposed on the 2D reservoir state. A pooling layer would take the average of the reservoir state values for each rectangle in the grid and return a smaller matrix of values. Ideally the pooled result preserves what is encoded in the reservoir state representation.

Note that all non-linear transformations should be attributed to the reservoir layer, therefore some care was taken when choosing a linear reduction technique. Other types of pooling, such as the max pool, is not a linear transformation. The average pool reduction however, is linear transformation, and is therefore a good candidate.



**(a)** Before pooling **(b)** After pooling

**Figure 3.8:** A pooling layer example taken from a publication by Jensen et al. [12], originally referred to as squinting. The figure was generated with tools from Flatspin. Note that the numerical representation going from figure 3.8a to 3.8b is a 2D matrix with reduced dimensions. There are many types of pooling in machine learning, for this layer it is a simple average pool, with a simple grid just like in the figures.

### 3.4.5 Walk-through of a Basic RC Design

The following is a walk-through description of what happens in the three RC layers in figure 1.1 for the simplest feed-forward design. In the input layer analog inputs are scaled from 0 to 1, translated to binary strings, and finally, converted to a sequence of global magnetic field perturbations.

In the the reservoir layer, the first magnetic field perturbation will force the reservoir to find a new equilibrium with regard to the magnetic field. This may cause a cascade of reservoir magnets to flip to adjust their state configuration accordingly. The cascade effect is due to a tug of war of competing local magnets simultaneously trying to reach the lowest energy state possible. Once a steady state has been reached, depending on the encoding, the magnetic field will be changed again, and the cascading flipping mechanism repeats.

Finally, with a single readout layer, the magnetic spins of the reservoir are read as a binary -1 or 1 depending on their local reference direction. Each binary value is then multiplied by its own weight parameter set by machine learning. Typically the final layer is the sum of these multiplications, but this may be adapted to the problem domain.

## 3.5 RC architectures

### 3.5.1 Vanilla Architecture

The term vanilla is an established term for the most simple version of something, referencing an order at an ice cream store. The vanilla architecture is almost identical to the description made in section 3.4.5. This was the starting point and most basic configuration with regard to figure 1.1. The basic pipeline was employed once for each of the variables, resulting in a parallel setup with no entanglement of the variables in any given reservoir.

The architecture is illustrated in figure 3.9. Starting from the left, the architecture accepts two variables as input, each in the range of $[0, 1]$. The variables are passed to the input layer, where the variables are encoded as a string of bits. Next, these same bits are passed to the reservoir layer, where the input bits emulate a series of global magnetic fields perturbing a magnet array. Remember that the dynamics behind the ASI has been replaced by a surrogate model in the form of a graph. This means the reservoir layer is merely a graph traversal, returning a set of spin states. Finally, a prediction is made in the output layer by performing a linear transformation of the spin states with trained weights set by machine learning.

The purpose of this design was to investigate if a non-linear problem could be solved without entangling information, i.e., the upper pipeline reservoir has information about the first variable, but no information from the second variable, and vice versa. This variable is marked in figure 3.9 by the superscript as either 0 or 1. If there is an entanglement, then this means non-linear properties can be in-

troduced when the various output layers are summed to the final prediction value. This would be problematic for experiment control, because it becomes difficult to know if the non-linear properties of the system are attributed to the reservoir.

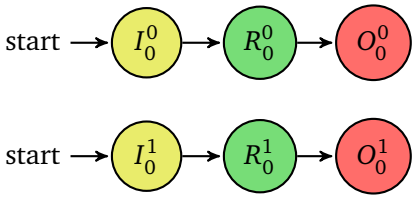



**Figure 3.9:** The Vanilla architecture. The superscript denotes the variable, while the subscript denotes the encoding. Settings are identical in the input, reservoir, and output layers, but the two variables are handled in parallel pipelines.

### 3.5.2 Unit Architecture

The unit architecture is almost identical to the vanilla architecture in section 3.5.1. The main difference is that there is now only a single pipeline with information from both input variables in one reservoir. Note that the vanilla architecture in figure 3.9 has superscripts that denote information from both variable 0 and 1 are dedicated to each their pipeline. In the unit architecture, however, the superscript indicates that information from both variables is present in all the layers.

The purpose of this design was to see if a non-linear problem could be learned when mixing the information from the variables. The architecture represents a starting point to see how a reservoir may handle multiple variables.

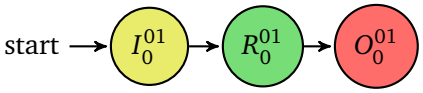



**Figure 3.10:** The Unit architecture. The superscript denotes the variable, while the subscript denotes the encoding. The input layer encodes both variables and merges them by interleaving before moving on to the reservoir layer.

### 3.5.3 Ensemble Architecture

Preliminary research for this thesis included investigating how others conduct RC. A technique used by Appeltant et al. [18] inspired the following architecture ensemble architecture. The researchers applied a random mask to their input, a technique also known as applying jitter. This means an input is used as input with some redundancy and variation. E.g. if a single variable input is $0.5 \rightarrow [0.505, 0.498, 0.509]$. Theoretically, these values with jitter will all traverse the reservoir graph with an almost identical traversal in the reservoir layer. Hence, many of them should be neighbours. This should give the readout layer more options when considering how to extract meaning from the reservoir state. As an extreme counter example,

if all the inputs map to the same neighbourhood in the graph, it is difficult to extract meaning.

A similar effect was achieved by making several parallel pipelines, but with different encoding. In figure 3.10, the subscript indicates that the upper and lower and pipelines have a different encoding. Note that the information from both input variables is in both pipelines, as indicated by the identical superscripts.

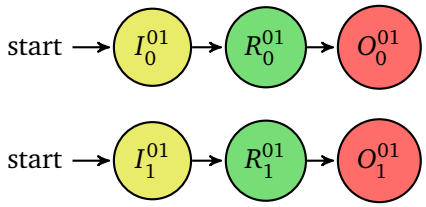



**Figure 3.11:** The Ensemble architecture. The superscript denotes the variable, while the subscript denotes the encoding. The input layer encodes both variables and merges them by interleaving before moving on to the reservoir layer. Note that the raw input is identical in both pipelines, but the encoding layers are different so that we get an ensemble of interpretations.

### 3.5.4 Pool Architecture

In the pool architecture, the first variable is confined to the top pipeline in figure 3.12, and spills over to the second pipeline through a pooling layer. As explained in section 3.4.4, the reservoir state is represented by fewer numbers, and converted to a string of 0's and 1's. This string is then fed into the reservoir layer in the bottom pipeline together with the string representing the second variable. Thus, the second pipeline has information from both variables starting from the reservoir layer.

The pooling layer may seem awkwardly put together, seeing as the input in the top pipeline is converted to a reservoir state, only to be pooled and fed as input again in the bottom pipeline. Actually, the pool architecture is similar to the vanilla diagram in figure 3.9. The variables start in separate pipelines, but are connected by a pooling layer. The idea is to see if information from the first variable can be transmitted to the lower pipeline, even after the reservoir and pooling layers. If such an information transfer is possible, then this proves the pooling mechanism can preserve information.

Practically, a limited amount of bits are available to encode the input. When an input goes from being a string of bits to a reservoir state, the information is represented with significantly more numbers. This makes it difficult to make architectural connections such as the one attempted in this architecture, with pooling as a way to bridge the gap. If this architecture is successful, architectures as seen in figure 3.13, should be possible.

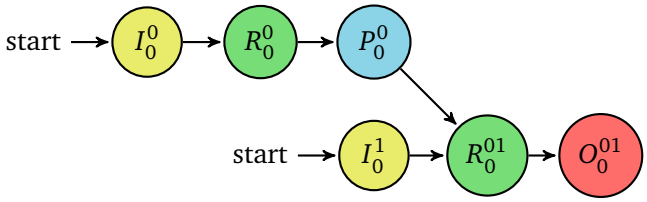



**Figure 3.12:** The pool architecture. The superscript denotes the variable, while the subscript denotes the encoding. The first variable is handled in the upper pipeline, leading to a pooling layer. The lower pipeline then accepts both the pooling layer and the second variable input layer. These two input streams are merged by concatenation, so that the pooled layer perturbs the preceding reservoir before the input layer.

### 3.5.5 Feedback Architecture

The feedback architecture focuses on introducing feedback into the system, which implies connecting the output layer back into the input layer. The technique is a common in RC, and was also used in the research aforementioned in section 3.5.4 [18].

Introducing feedback is not trivial, as the input layer dimension is much smaller than the output layer dimension. The input layer dimension cannot be increased, as this implies more bits being fed to the reservoir layer, and the reservoir layer has a finite number of states. The output layer dimension is set by the number of magnets in each reservoir state, and should be a large dimension in a good reservoir, as discussed in section 1.4. One solution to this issue is by passing the reservoir state to a pooling layer as detailed in section 3.4.4.

Note in figure 3.13 that the pooling and input layers show bits entering their respective reservoirs with some ambiguity. This is because the bits from these sources were combined in two ways to reduce the effective input length. The first way was by concatenation, with input bits followed by pool bits in a ratio of 1/4, respectively. This seemed to be a good setup for early experiments. The second method was performing a logical *AND* operation between the pool layer bits and input bits before being applying to the reservoir layer.

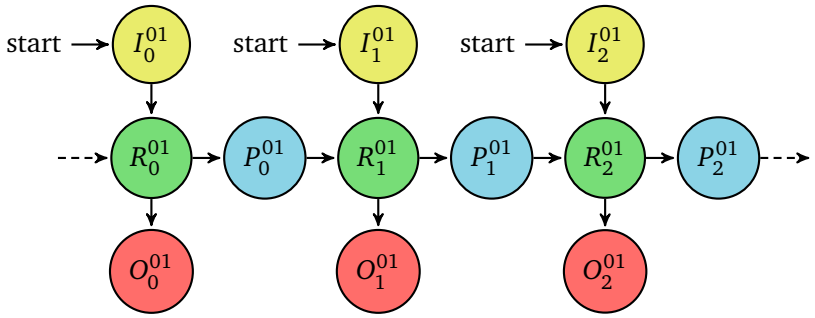



**Figure 3.13:** The Feedback architecture. The superscript denotes the variable, while the subscript denotes the encoding. The input layers are identical in that they encode the same input, with both variables, and merges them by interleaving. They differ in that they only select a portion of a B bit input stream, distributing B/n bits over n input nodes. Each pool, input, reservoir, output layers can be considered a module, identified by their subscript. Information propagates across the n modules via the pooling layers from left to right in a daisy-chained fashion. Each module is then fed by concatenated input and pool layers, in that order. Note that reservoir initialization is done on the rightmost module by ignoring the pool layer from the left. This allows use to evaluate the rightmost pooling layer which feeds the leftmost reservoir layer, and so on.

# Chapter 4

# Results - Practical RC

## 4.1 Architecture Performance on Non-linear Function

In this section, the various architectures from section 3.3 will be quantitatively examined, followed by a qualitative analysis. Each model had the task of predicting an image generated from equation 3.5. The idea is to gauge how the various architectures performed in terms of accuracy, resolution, and qualitative patterns in each model's predicted image. The reservoir size was 20x20 for this experiment, the largest size considered. This choice was made with the assumption that the largest reservoir will perform the best, so that we could also qualitatively draw conclusions. The insight gained from this would then be transferred to the tuning and architecture of smaller reservoir. See appendix A.2 for training parameters associated with this section. The 20x20 reservoir can also be found in appendix A.5.

Note that the predictive images below all have the same high resolution as the target image, and yet there is a clear pixilation. This is an artifact from the input layer, which only allows each variable n unique states. The term resolution will be referred to below, and can be thought of as the effective image resolution of a predicted image, limited either by the model or the encoding.

### 4.1.1 Quantitative Comparison

It is difficult to compare qualitative impression. It is therefore better to use the quantitative score as summarized in table 4.1 to recognize this architecture as the most optimal. The quantitative results are summarized in table 4.1 and indicate that the most successful architectures are Feedback with 67% accuracy followed by Ensemble with 64% accuracy. This should be a somewhat fair comparison, considering that all parameters are equal except their architecture. Note that the resolution of the feedback architecture is also higher than the ensemble architecture.

**Table 4.1:** Architecture comparison summary. Encoding legend: T=Tally, S=Shuffle, I=Interleave.

| Architecture | Accuracy | Readouts | Resolution | Encoded Bits | Encoding |
|---|---|---|---|---|---|
| Vanilla | .33 | 2 | 40x40 | 80b | TS |
| Unit | .34 | 1 | 20x20 | 40b | TSI |
| Ensemble | .64 | 8 | 20x20 | 320b | TSI |
| Pool | .59 | 8 | 40x40 | 640b | TS |
| Feedback-concat | .67 | 8 | 40x40 | 80b | TSI |
| Feedback-and | .54 | 8 | 120x120 | 320b | TSI |

### 4.1.2 Qualitative Comparison

The various architectures form a story of incremental improvement in designing a model architecture. The Vanilla architecture shows how keeping variables in separate reservoirs for a non-linear problem is problematic, as it forces the model to converge on a solution that focuses on only variable, see figure 4.1.

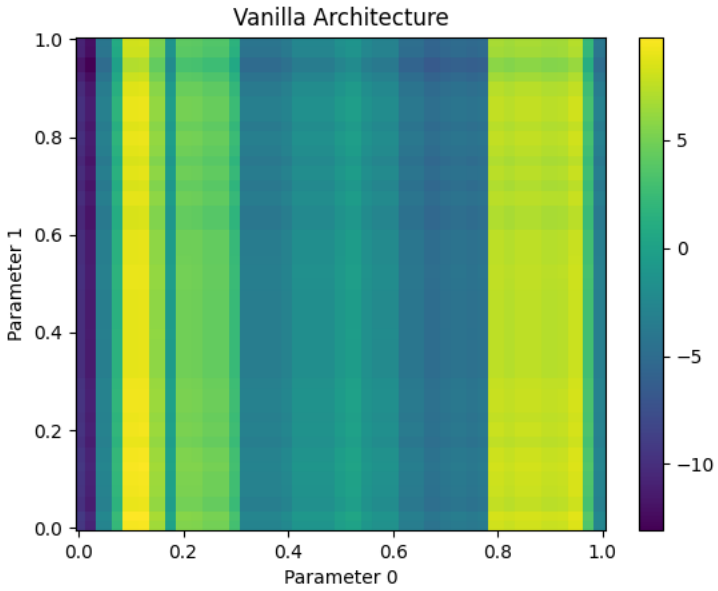



**Figure 4.1:** Vanilla architecture prediction image relative to figure 3.6. The resolution is 40x40 due to the input layer. Note that the model has attempted to focus on the three vertical bars in the target image, but parameter 1 has no effect on the prediction. A similar result was obtained when rotating the target image by 90 degrees; horizontal bars with no effect from parameter 0.

The Unit architecture amends this issue by entangling the variables before entering a single reservoir, see figure 4.2.

The Ensemble architecture improves on the Unit architecture by adding more

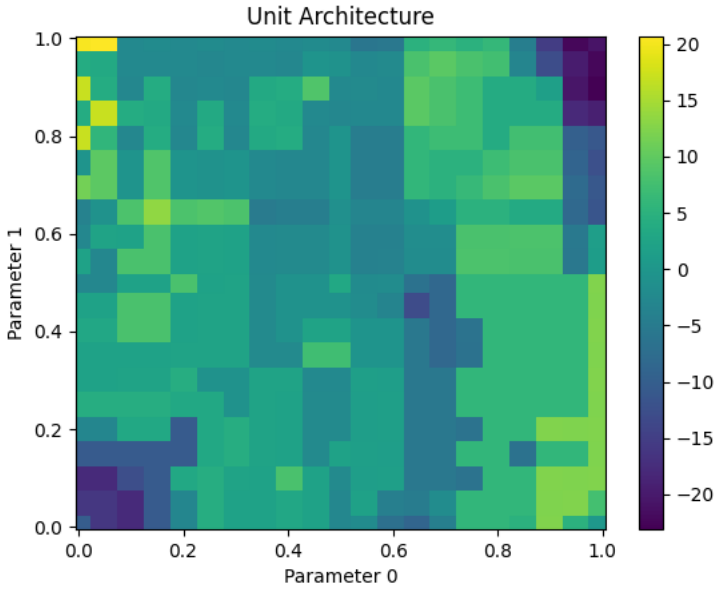



**Figure 4.2:** Unit architecture prediction image relative to figure 3.6. The resolution is 20x20 due to the input layer. Note that the model is starting to learn the two most prominent columns, as well as the dark regions on the bottom left and top right. It seems a single output layer is not sufficient to learn the target image accurately.

parallel readout layers, where each output layer interprets a state stemming from a different encoding, see figure 4.3.

The Pool architecture merely tests if pooling can be applied, while still preserving reservoir state information, see figure 4.4.

Finally, the Feedback architecture leverages feedback using the pooling mechanism from the Pool architecture to solve dimension issues that arise when connecting the input (small) and output layers (big), see figure 4.5. The idea with feedback is to act as an extra memory of previous states.

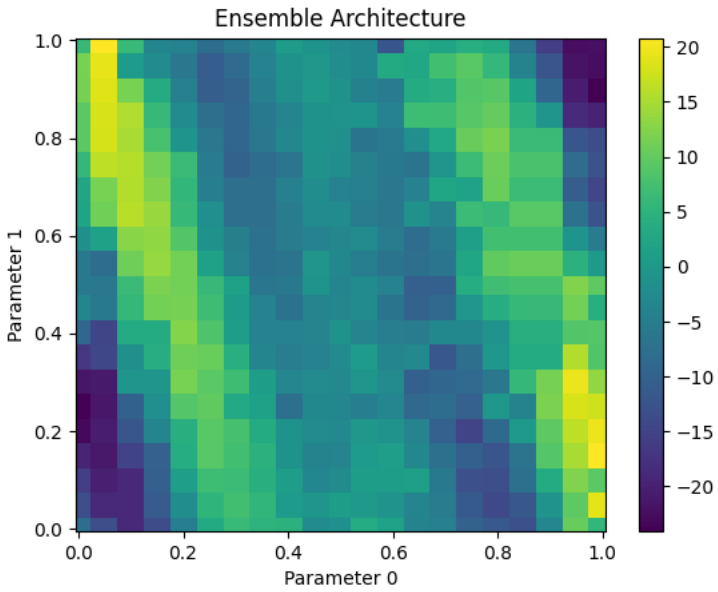



**Figure 4.3:** Ensemble architecture prediction image relative to figure 3.6. The resolution is 20x20 due to the input layer. The model is now essentially identical to the target image, but is suffering from a low resolution. The image proves that an ensemble approach, introducing various encoding schemes, is effective for improving performance.

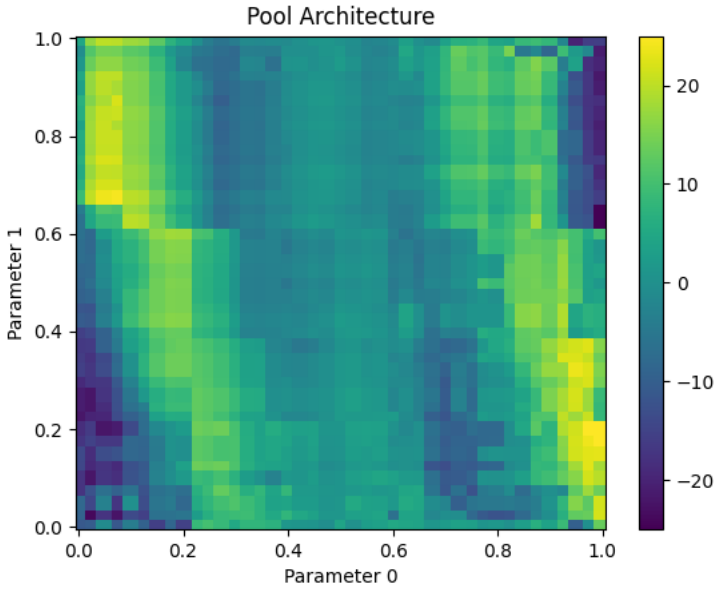
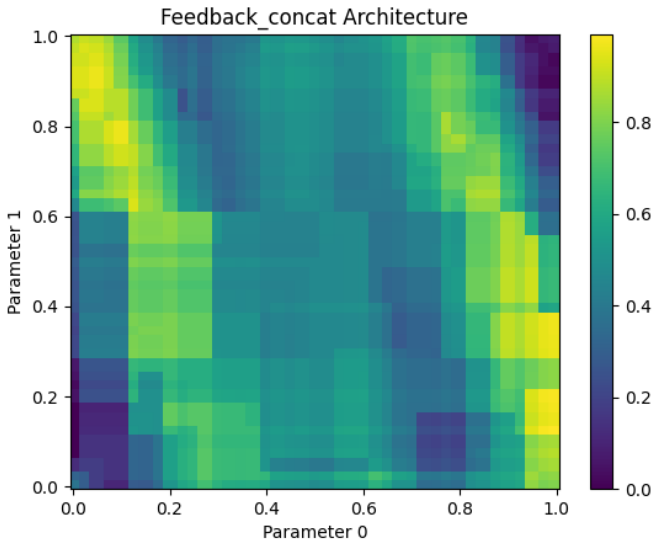
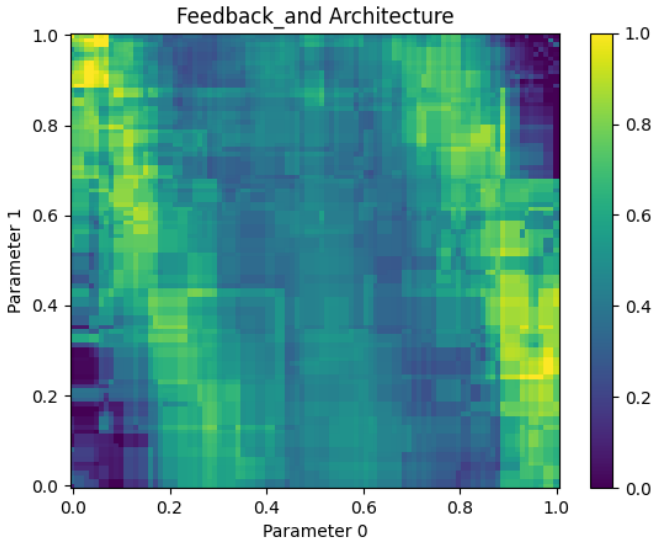



**Figure 4.4:** Pool architecture prediction image relative to figure 3.6. The resolution is 40x40 due to the input layer. The purpose of this experiment was merely to prove that pooling a reservoir will preserve information transfer, coincidentally it performed quite well. See the architecture in figure 3.12 to see how pooling must preserve information considering both parameter 0 and parameter 1 effect the image. If pooling didn't preserve information, then the predicted image would be similar to figure 4.1. Since the pooled reservoir representation of parameter 0 is entangled with the input layer of parameter 1 by a logical AND operation, the architecture allows for more bits compared to concatenation.




**(a)** Feedback-concat architecture prediction image relative to figure 3.6. The resolution is 40x40 due to the input layer, a restriction set by the concatenation of bits from the input and pool layers, which must sum to 40b.




**(b)** The figure is the same as above, but with 120x120 resolution. The figure demonstrates that the resolution can be increased at the cost of accuracy. This was done by replacing the concatenation step mentioned above with a bit-wise AND operation, which in turn allows for many more bits in the input layer.

**Figure 4.5:** Feedback architecture predictions images.

# Chapter 5

# Discussion - Practical RC

## 5.1 Deciding on a Model

The purpose of having a practical non-linear function in equation 3.5 as a benchmark was to find a model suitable for RC. The idea is that once a good model is found, it can be used on another benchmark used for validating ESP metrics. The result from the non-linear function benchmark was that the feedback-concat and ensemble architectures had the highest accuracy. Despite this, the feedback-and model was chosen, as it is able to handle much more bits in the input layer, which translates to more bits per input variable. This will be important for the next part of the thesis as the benchmark changes to the n-inverted pendulum balancing problem, where the variables increase from 2 up to 8. Henceforth, the feedback-and architecture will be loosely referred to as the feedback architecture.

## 5.2 The Cooperation of Output Layers

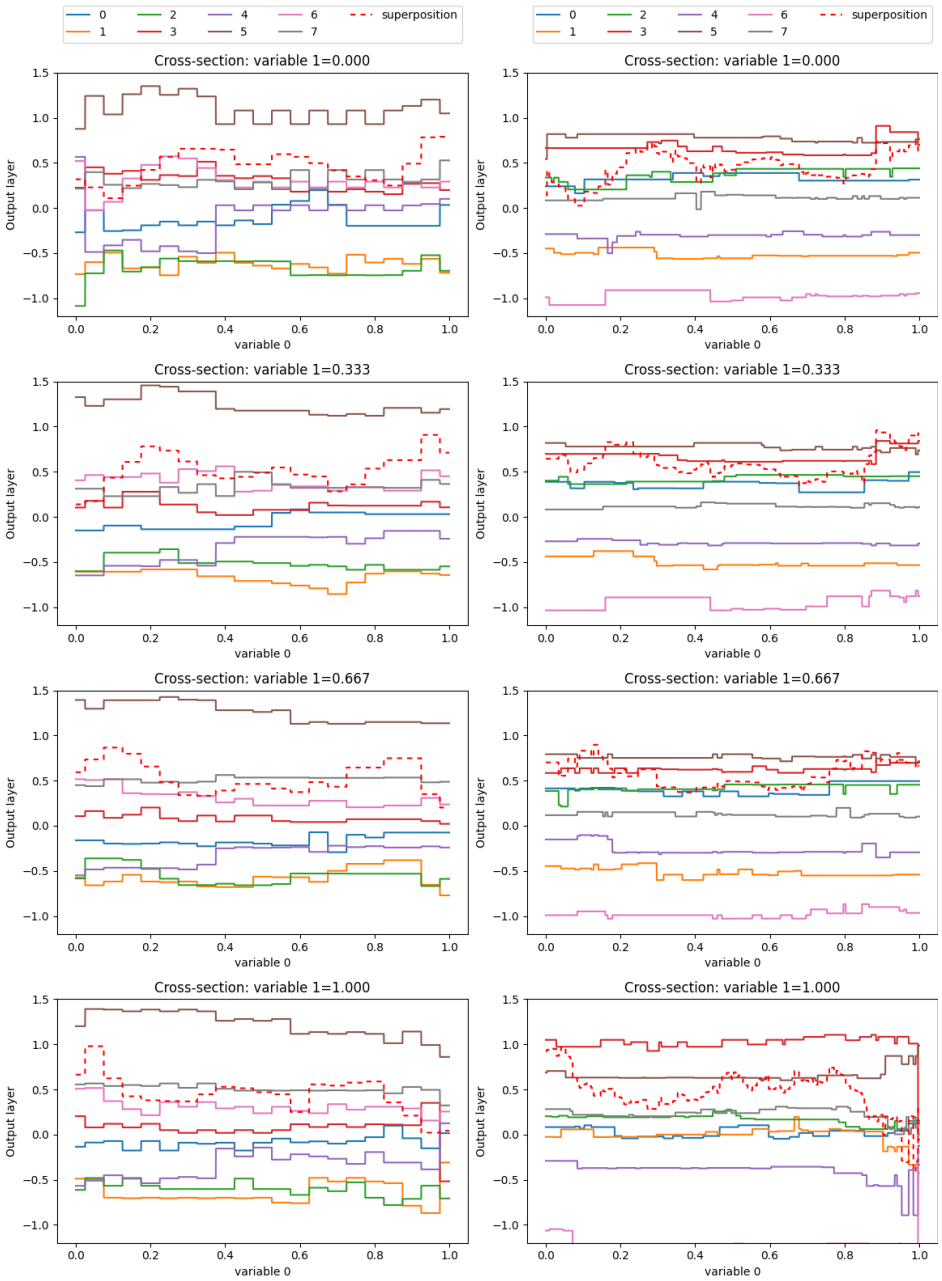
### 5.2.1 Output Layer Signal Analysis

For all explored architectures, the output layers are summed to calculate a model's single prediction value for each pair of inputs. In figure 5.1, the output layer values are tracked individually as they produce a prediction, shedding light on their cooperation mechanism. This was done by cross-sections of the non-linear target image in figure 3.5. Thus, the system is placed in a signal theory context, with the output layers as signals. The sum of the output layer signals is equal to the model's prediction value, a phenomenon known as superposition in a signal context.

The following is a walk-through for interpreting the cross-sections in figure 5.1. The cross-sections are horizontal slices of the prediction image, keeping variable 1 constant, and varying variable 0. The figure's left column is the ensemble architecture, and the right column is the feedback architecture. Choose one of them and proceed. The cross-sections have been arranged analogously to looking left-right and down-up, as if one were looking at a 2D prediction image. Thus, the x-axis of each sub-figure corresponds to looking left-right. Similarly, looking

down-up the y-axis of the prediction image is like navigating sub-figures vertically, noting that variable 1 changes.

Next are the contents of each subgraph, in the same figure 5.1. Per graph, the filled lines are signals tracking the values of the output layers before summation. The output layers are labelled by id, after their position in their respective architectures in figures 3.11 and 3.13, respectively. The dashed line is the architecture prediction value, equals the sum of the signals, or superposition.




**(a)** Ensemble architecture **(b)** Feedback architecture

**Figure 5.1:** Cross-sections of the ensemble and feedback prediction images relative to figures 4.3, and 4.5, respectively. See the walk-through in section 5.2.1 to learn how to interpret cross-sections in this layout. The solid lines are coloured by their layer position in the architecture, and represent the values of each layer prior to summation. The dashed red line is the sum of the solid lines, which is the output value of the models.

In figure 5.1 there are three things to note: The square quality of the superposition signals are due to encoding restrictions discussed in section 3.4.1, previously referred to as resolution. The signals are relatively straight, with a tendency to stay roughly parallel. And lastly, between a models cross-sections, the signals seem to maintain somewhat fixed vertical ordering, with small vertical shifts up and down.

From figure 5.1, it is clear that the superposition of the ensemble architecture is more coarse than the feedback architecture. The square quality of the superposition signals is due to encoding restrictions, as discussed in section 3.4.1, previously referred to as resolution. Note that both the ensemble and feedback architecture, in the left and right columns respectively, are trying to predict the same target image, hence the dashed red line is similar.

Interestingly, the signals are relatively straight, with a tendency to stay roughly parallel, and are shifted up and down between the cross-sections. This indicates that the cooperation mechanism strategy is to dedicate each signal to a limited range. One potential benefit of a relatively straight signal is that is easier to reduce overlap with other signals. This may reduce contention among signals, effectively assigning each output layer with a generalized role. Note that it is easier to reuse the signals for different combinations when they are not specialized.

In the brain, there are many inhibitory mechanisms in place to prevent the excess firing of neurons. Without this, the brain would quickly malfunction with phenomena such as epilepsy. There seems to be a similar interplay between balancing a positive and negative gain among the output signals. The signals seem to be evenly distributed between $[-1, 1]$, although their target is always in the range of $[0, 1]$. The negative signals and positive signals balance each other out, and in some cross-sections the signals, even seem to mirror each other over the neutral output layer axis (y=0).

### 5.2.2 The Reservoir as a Classifier

This section will focus on understanding the reservoir as a classifier. The same architectures from section 5.2.1 will be compared. The core idea of this analysis is to get an idea of how a model's input and reservoir layer classify inputs without influence from the output layer. Thus, one can assess how much potential there is for the RC system as a classifier even before the final categorization from the readout layer.

The states in figure like 3.3 are labelled based on their discovered order during the depth first search mapping process converting the ASI to a directed graph. A single reservoir can represent $2^m$ different categories with $m$ magnets, but in this analysis, we limit the number of categories to 128. This is due to the limitations in resolution from the input layer in section 3.4.1, making 128 categories a reasonable choice, as the models cannot express more categories than their resolution listed in table 4.1. To get a more objective state labeling, the various state spin configurations were categorized via four unsupervised categorization techniques.

A single technique was not chosen, because boolean data is not ideal for typ-

ical dimension reduction techniques like Principle Component Analysis (PCA), or k-means. Labeling is a common image processing technique for giving pixels an id from 0 to n, where each id is a group of pixels in a picture. The method was modified by renaming the groups by the size of each group, and then normalizing by the largest group found in the range $[0, 1]$. Corex reduces the boolean data by optimizing an information goal function [19]. Metric Component Analysis (MCA) is ideal for boolean data, and attempts to make a 2D plot where similar data form clusters, which can be categorized with k-means. The details of how the categorization techniques work are not important. The takeaway is that four quite different automatic categorization techniques were used to see if an objective classification of spin states could be achieved. The approaches are listen in table 5.1.

**Table 5.1:** Categorization techniques

| id | approach | color control |
|---|---|---|
| i) | MCA $\rightarrow$ K-means | normalized cluster id as a hsv color |
| ii) | Labeling $\rightarrow$ PCA | 4 principal components independently control RGBA |
| iii) | Labeling $\rightarrow$ K-means | normalized cluster id as a hsv color |
| iv) | Corex | normalized binary number as a hsv color |

There might be unwanted artifacts from the various objective categorization techniques categories, but there is a clear agreeance among them, except Corex, which had a very limited number of categories. To restrict the scope of this discussion, the focus is on the label-k-means approach.

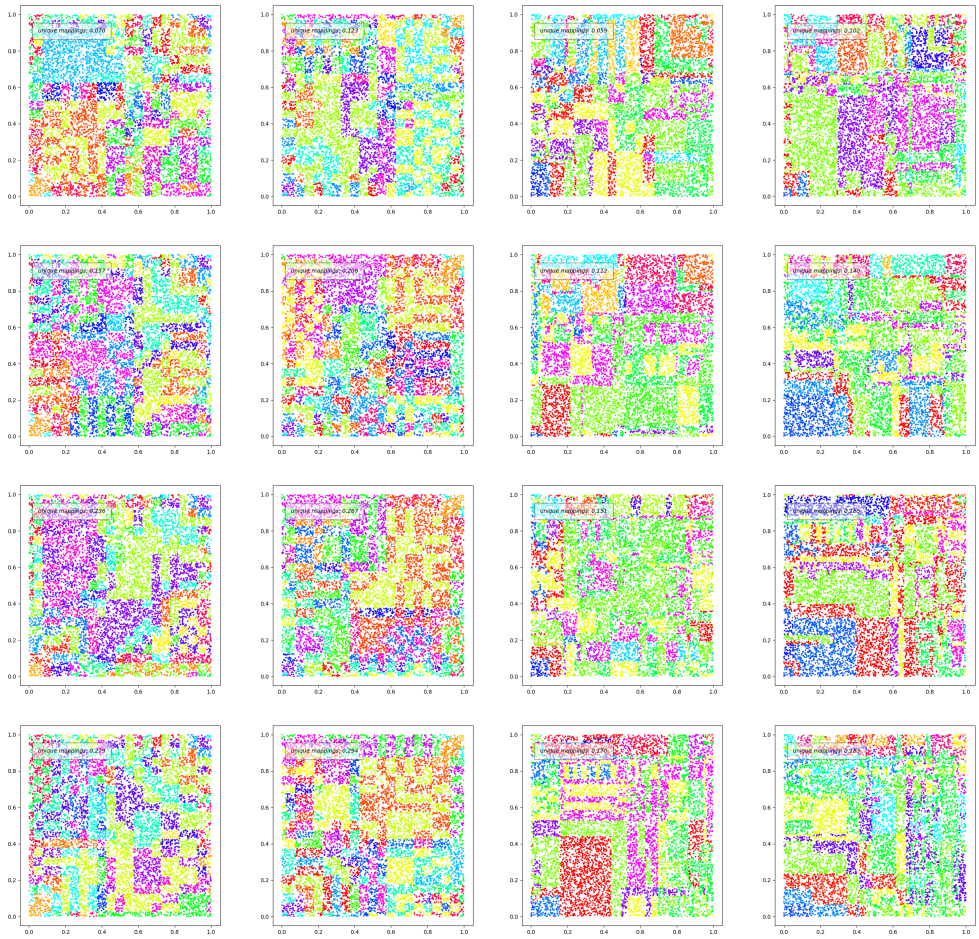
Figures 5.1 and A.2 are closely related. Each square in figure 5.2b represents one of the output signals in the cross-sections. Note how making a horizontal slice in a classification sub-figure is equivalent to tracking an output signal in the cross-section's sub-figure. By making a scatter plot of random inputs and observing which objective states they get mapped to, one can reveal what the readout layer has to interpret with machine learning, expressed as the output signals in figure 5.1.

Figure 5.2b maps inputs to objective categories. Choosing a subfigure is equivalent to one of an architecture's output layers, while choosing a point on the 2D plane is equivalent to selecting a combination of two of its input variables. Thus, for each output layer, a set of state mappings are produced for a given input. The mappings reveal that there are well-defined, rectangular concentrations of states. The concentrations are likely due to the self-references in the graphs, causing similar input to categorize similarly. The tendency for rectangular shapes can be explained by resolution limitations in the input layer, as explained in section 3.4.1.

In the same figure, a set of state mappings is then a set of colours for each of the architecture output layers. E.g. if one sets $x = y = 0.4$, one would get a classification from each reservoir layer: $pink, yellow, orange, blue, red, purple, orange, green$. Hence, the readout can be viewed as a repeated low-pass filter, accumulating con-

tributions from each reservoir layer. A set of mappings could then uniquely define a given output by combining the best contributions selected via the output layers, effectively exploiting a kind of decentralized classification. Note that order is not relevant during readout going from a set of mappings to a single output. A need for a collaboration mechanism is needed to avoid contention between the mappings, f.e., if two collaborating reservoirs have significant informational overlap, it would quickly lead to a double counting. This example makes it seem intuitive to orchestrate signals by an averaging of outputs, especially for the ensemble architecture. However, as discussed in section 5.2.1, both ensemble and feedback architectures seem to use a similar collaboration mechanisms.

The mapping was also projected on the ASI graph as seen in appendix A.4, and also shows that neighbouring nodes/states tend to be classified the same. Within the ensemble architecture each row is a different encoding, while each column has an inverted definition of 1 and 0. This inversion seems to transpose the mapped images in the corex mappings in the aforementioned appendix, and more generally seems to be an effective way to force different state responses based on encoding.



**(a)** Ensemble architecture **(b)** Feedback architecture

**Figure 5.2:** Input to category maps for the 8 reservoir layers in the Ensemble and Feedback architectures. Each point is a random combination of the two input variables, which together represent the normalized XY plane. The colouring indicates the reservoir state as categorized by labeling -k-means approach. The little opaque boxes in the top left corners are unrelated to the categories, and are the ratio of all mapped states to all possible states, cumulatively increased as each reservoir layer expresses unseen states. This is a measure of how exhaustively the reservoirs explore their potential state space. Note how making a horizontal slice in a sub-figure is equivalent to tracking the signal from one of the eight lines in the cross-section sub-figures in figure 5.1.

### 5.2.3 Performance in Relation to a Lookup Table

A quick definition: A simple example of a lookup table is a phone-book, where a name can be looked up, and you get a number. The idea can be generalized to any problem with this input-output format. In this discussion, the speed of lookup is not relevant, but rather the size of such a table. How big would a look-up table be to memorize all the RC systems input to output mappings?

The highest performing model, in terms of accuracy, uses the feedback-concat architecture in 4.1 with 40x40 resolution. The model uses a 20x20 reservoir with 840 magnets that can configure into 1540 unique states. This means that the RC model readout layer has around 840 linear nodes, each made up of a weight and a bias. This translates to 840x2 learned parameters, rounded to 1600 for simplicity. Theoretically, if each of these nodes were set to represent a combination of input using two variables, then you could store a 40x40 image in the readout layers nodes. If one were to repurpose the RC models readout layer as a lookup table, it would probably perform better than the model itself, due to the sheer amount of trainable wights in the output layer. Thus, assuming perfect accuracy, a model should achieve at least a 40x40 resolution with a single readout layer, if the RC model should be competitive against such a theoretical lookup table. Comparing this result with the results from section 4, they are not performing well enough to beat a repurposed version of their readout layer as a lookup-table, leaving the reservoir and input layers redundant.

On a more optimistic note, a lookup table will grow exponentially as it contains more variables. The lookup table in the example is only for two variables, but this already scales to many entries, ie., a resolution of 100 requires a $100^2 = 10000$ size lookup table to capture all combinations. The reservoir may have the ability to effectively compress such a lookup table, thus getting a spacial complexity advantage over a lookup table. Another way to reduce the size of the output layer is by pooling the reservoir layer before readout, an idea referred to as "squinting" in earlier work [12].

### 5.2.4 Resolution as a Heuristic for Multiplexing

The pigeon hole principle is a simple but powerful logical argument. The term pidgeon hole is just a bird box. Imagine a group of holes and pigeons, and each pigeon wants its own hole. If there are P pigeons and B holes, and $P > B$, then at least one hole will have more than one pigeon. This same argument will be used below. See the glossary for a definition of multiplexing.

Recall that several output layers may have to cooperate to produce mappings, by the sum of their values. What would be interesting to know is if the inputs are interpreted in the same way as they were encoded. I.e. "011" is $1+1=2$ in tally encoding, but $0x4+1x2+1x1=3$ in binary. Depending on how the reservoir and output layers collectively interpret the input, then it could allow for a much larger map space. Note that a tally can maximally express 3 states, while the binary equivalent can express $2^3 = 8$. By the pigeon hole principle, an achieved resol-

ution can serve as evidence for this phenomenon called multiplexing. Consider that n multiplexing output layers have their values set by an input string with b bits. With a tally encoding, the input string could maximally express $T$ states by itself. As a binary encoding, however, it could maximally express $B^2$ states. The binary encoding is a perfectly multiplexed, as discussed in section 3.4.1. Thus, for encoding schemes like the distributed encoding, the input could express an $D^2$ resolution, which is somewhere in-between these bounds.

Using these defined limits, if the resolution is higher than the $T^2$ threshold resolution, it has been proved that the output layers must be partially multiplexing. In the aforementioned pigeon hole principle, each bit in the boolean input string can be used to access a grid of holes, while the resolution achieved represents the pigeons. E.g., if input has two variables, both with 3 bits, they can maximally form a 3x3 grid. On the other hand, if they multiplex completely, they form a $2^3x2^3$ grid. If there are more pigeons than holes, then this means that the resolution will be higher than the $T^2$ lower bound. In this case it would be proven that the inputs are partially multiplexing. The $T^2$ lower bound also marks an upper limit for all tested models in section 4.1, since values are set by tally encoding. Qualitatively, the resolution seems to be equal to this limit, which indicates that there is no multiplexing mechanism. For example, the input layer for the feedback model in figure 4.5 has maximally 40 states per variable, and the prediction image has 40x40 squares. Breaching this limit should not be possible with tally encoding, but may be exemplified in the experimental distributed encoding in appendix A.3.

# Chapter 6

# Conclusion - Practical RC

Since RC is a relatively open framework for computation, an architectural investigation was necessary before delving into testing the ESP metrics. By carefully designed architectures, a robust RC was identified. The floating point inputs will thus be converted to bits using the tally-shuffle-interleave encoding. Going forwards, it is perhaps most important to note that the practical RC exploration lead to proceeding with the feedback-and architecture for the next part of the thesis. This was not the highest performing model quantitatively, but qualitatively, it allows for a much higher resolution. The high resolution means that more bits can be encoded, which will be highly required in the next part of the thesis, when the number of input variables increases. The non-linear benchmark was learned reasonably well, with 54% of the model's predictions within 5% of their target values.

In addition to choosing an architecture and encoding, a thorough analysis of the model and the reservoir was undertaken. The models output layers were analyzed as signals, examining their cross-sections as they predict the target image in figure 4.5. Here, it was revealed that the output signals tended to be relatively flat and parallel, manipulating their accumulative prediction signal through small vertical translations. This behaviour was apparent for both architectures, so the output layers probably function in a similar way. The relatively flat output signals either meant that there were very few bit flips in the reservoir, or the readout layer selectively prioritized stable reading configurations. Thus an examination of the reservoirs themselves ensued, confirming that once a bit had flipped, it tended to get stuck in its new state. Furthermore, automated classification techniques were applied to the various reservoir states to see how various inputs might get classified by a RC system without a readout layer. This revealed qualitatively that the reservoirs seemed to be classifying similar inputs expressed as patches of similar classifications in figure 5.2. The connection was also made in how the reservoir first classifies the input like a repeated low-pass filter, proceed by the superposition of these values to produce a prediction.

In section 5.2.3, the reservoirs output layers were compared to lookup tables. This comparison was possible, because the trainable weights in the output layer

could theoretically be re-purposed as a lookup table. The comparison was made to get an idea of how much of the computation could be realized without the reservoir. The conclusion was that even the highest performing model would not come close to beating a lookup table made out of the resources dedicated to its output layer. On a more optimistic note, the reservoir should be able to compete as the resolution and number of variables increases, as this would cause the lookup-table to explode in terms of resources.

Lastly, in section 5.2.4, it was concluded that there was no multiplexing in the predicted images with tally encoding by an argument based on the pigeon hole principle and qualitative resolution. The key takeaway from this discussion point is that the output layers could theoretically express a much higher resolution by a multiplexed encoding. This would be yet another way to surpass the theoretical lookup-table at producing useful input to output mappings.

# Chapter 7

# Method - ESP metrics

In section 7.1, it may seem like the topic strays from its purpose; describing a method for measuring ESP metrics. The section is necessary, however, because it describes how the graph metrics where chosen, as well as giving a background to relate the metrics to.

## 7.1 Underlying Structure of the Reservoir

By converting the various sized RC systems into ASI graphs, it was discovered that the graphs can be interpreted as having two distinct components when traversing from the start node to its leaf nodes. Putting the graph on its side like in figure 7.1, and traversing from left to right, the start is reminiscent of a random forest, but near the graphs leaves, the structure is characterized by SCG regions. In other words, the two regions represent a shift in graph topology from being monotonically directed to randomly directed.

Based on this observation, it is intuitive that the top component may function as a random forest classifier, while the bottom SCG regions could function as a secondary, more fine-grained, classification. Recall that each node in the graph is a unique state, and note that it is impossible to traverse from one graph branch to another. A branch in this discussion is defined as a set of paths leading to the same SCG. Most likely, this implies that certain state bits, or physical magnet spins, become permanently set for a branch, effectively blocking inter-branch transitions. Note that many of these branches are also similar in the graph topology, further indicating that the difference between them is most likely a handful of state bits. An intuitive analogy to this behaviour is the hardening of a melted solid in a dish that is slowly cooling. Small solid islands form on its surface first, allowing only the liquid parts to move. Slowly but surely, the whole material becomes solid, shrinking the liquid regions as each solid island grows. In this analogy, a liquid region is a magnet that can flip, while the solid region is a cluster of magnets that collectively resist external stimuli, and are hence effectively fixed.

These observations were quantitatively verified by a logical AND operation of all state bits within a SCG region. This operation produces a state matrix where

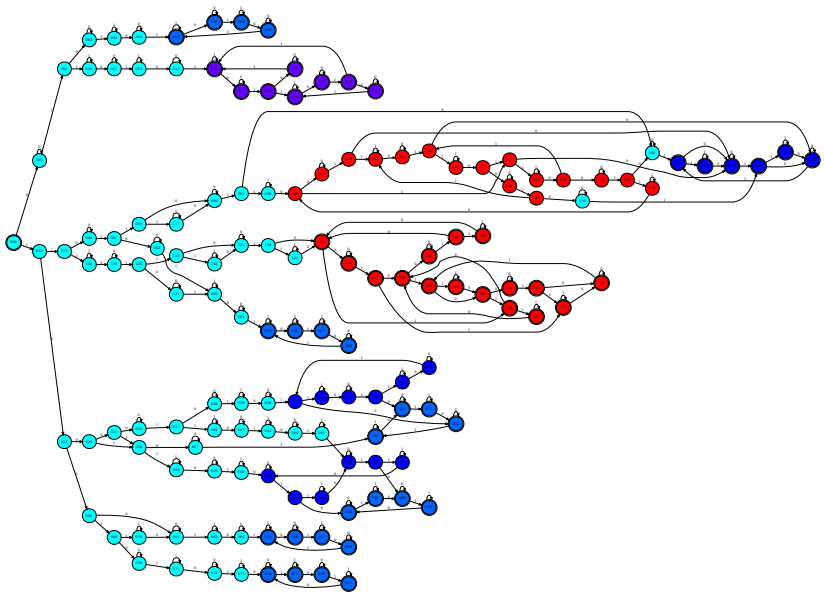
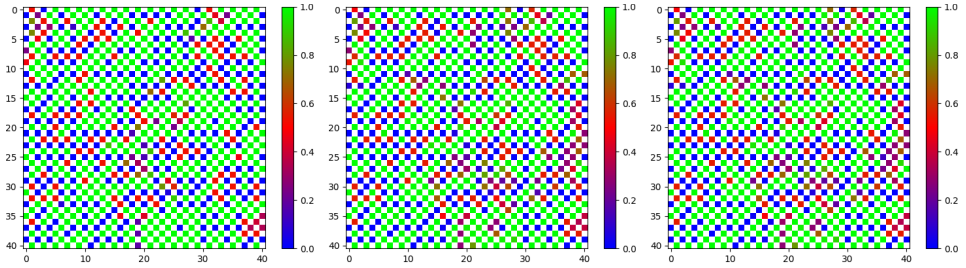


**Figure 7.1:** An ASI reservoir with a 14x14 diamond pinwheel configuration as a directed graph. See figure 7.1 for more description on how to interpret the graph. All SCG sub-graphs are coloured by size and marked with a double circle. The start node is also marked with a double circle. The turquoise nodes are reminiscent of a random forest near the origin node (left), while the other colored regions are the SCGs, often located near leaf nodes (right).

a zero means the bit has flipped at least once, and a one means the bit has never been flipped. In fact, in the 20x20 reservoir, roughly half of the state bits are fixed within a given SCG region. The random forest regions were also inspected using the same logical AND operation, and it was confirmed that around half of bits become fixed, after a few steps from the graph root. The amount of fixed bits barely decreases upon traversing a branch before reaching its SCG region. Finally, all status bits in the entire graph were compared, revealing that 37% of all state bits are always fixed. In other words, a relatively small amount of state bits are utilized to distinguish states both within branches and SCG regions. There is a large redundancy in the state representation, since many state bits tend to be fixed. The phenomenon is intuitively illustrated in figure 7.2 as bit state probability maps.

Continuing with the aforementioned two component analysis, with the random forest and SCG regions, it is argued that the random forest components are an expression of the systems memory, while the SCGs components an expression of the systems ESP. Consider a string of nodes as an example of a purely directed graph. It would be easy to backtrack to the origin from the last node, and it could be argued that the last node “remembers” the first. A way to see this is by inter-



**(a)** Intra SCG **(b)** Inter SCG **(c)** All states

**Figure 7.2:** Bit state probability maps (left to right): within the largest SCG (node count), between the top 5 SCGs, and for all states. The figure was made for the 20x20 reservoir by superimposing all state images (summing all values), and normalizing. Note that the strong blue and green colors indicate regions that are fixed, while the red end of the scale represents bits that are free to combine into many states. Note that the "Inter SCG" and "All states" figures are almost identical. The "Intra SCG" is different for a select few bits.

preting each node spin state as a kind of binary counter, as no states are repeated. The counter may also just be based on a one-step increment from the last state, but then the state is bound to repeat if no more unique states are possible. In this case, the SCG arises, as the traversal runs out of unique states. Within a SCG it is much more difficult to backtrack in the same manner. This is then an indicator that the states “forgot” previous states, and allow for their repetition. The fading memory property of the ESP is therefore expressed in a reservoir graph’s SCG regions. Furthermore, the ESP can then be discussed in terms of the number of SCGs, their node connectivity, and their size measured in nodes.

## 7.2 The ESB

In previous work [1], the concept of an ESB was introduced to measure the ESP. In short, the ESB metric perturbs a reservoir’s SCG with a repeated input until there is a predictable cyclical state response from the reservoir. The length of the transition, is the ESB, and should indirectly measure the ESP. As in the previous work, when a repeated input leads to a predictable sequence of states, then it will be referred to as saturation. The ESB is artistically exemplified in figure 7.3.

In figure 7.3, the ESB is defined as the transition length between two cyclic responses. This implies that the random forest region cannot be chosen as the start of the ESB, because cycles are not possible. On the other hand, SCGs are cyclic graphs implying that the ESB must have its starting point in one of them.

The proposed ESB metric predicts that wave theory phenomena may be observed in the reservoirs response to a varying input string lenghts. Wave phenomena includes the existence of a fundamental input length. The idea is that certain length input strings will resonate periodically with the system, just like pushing a swing is best done at the natural frequency of the swing. This translates to op-

timal performance at multiples of several fundamental input lengths, ie., with one fundamental with B bits, it may resonate at, 2B, 3B, etc. In the in intuitive swing example, the swing will respond strongly if you push at half its natural frequency. This prediction will be looked for in figures such as 8.5.

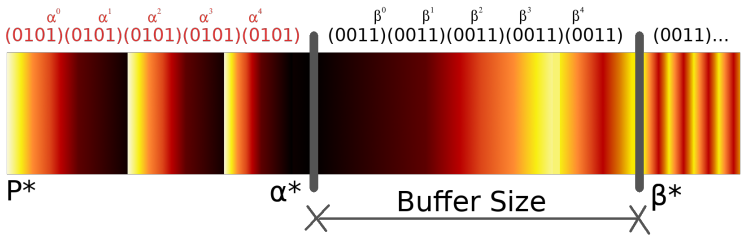



**Figure 7.3:** Artistic illustration of ESB, a metric to quantify the ESP. Each vertical color strip represents a reservoir state, i.e. if a color is repeated, a state is repeated. A input sequence $\alpha$ is repeated until there is saturation; a predictable sequence of state transitions. Upon saturation, a different input sequence $\beta$ probes the reservoir until saturation a second time. The ESB is the the difference, measured in bits. After $\alpha^*$, memory of the saturation fades incrementally, replaced by memory of $\beta^*$. $P*$ represents a optional warm-up input pattern, something which may be needed to get to saturation type behaviour. This figure was directly taken from previous work [1].

## 7.3 Engineered Inputs as Probes

In previous work [1] the concept of engineered inputs was developed as a way generate high information density bit strings. E.g., a bit string with no repeating substring is harder to compress, and thus contains more information. The idea is to probe the reservoirs with these engineered inputs, to help uncover properties like the ESB. I.e., it was found that the balanced density bit strings, having an equal number of 1s and 0s, traverse to all tested reservoirs leaf nodes the fastest. The engineered inputs will be limited to random and 50% density inputs as justified in section 8.1.2.

## 7.4 Comparing Model Success with ESP metrics

In the previous method, in section 3, the performance metric was to emulate a non-linear function. In this section, the performance will be gauged based on balancing a n-inverted pendulum on a cart, as illustrated by figure 7.4. Henceforth the benchmark will be referred to as the balancing benchmark or dataset. This task was chosen because it is a well studied benchmark for control systems with feedback. A certain amount of short term memory is needed to efficiently balance the inverted pendulums. I.e. If RC system is perfectly balanced with the exception

of the top pendulum, but this parameter was fed to the RC system last, the information might be forgotten. Arguably, the benchmark can therefore be used to validate findings related to ESP metrics. Note that the encoding was switched to tally encoding for this section, as it seemed to work much better than tally-shuffle-interleave, which was used during architecture development in section 5.1. This encoding was very bad as discussed in 3.4.1, but the feedback-and entangles the input with the pooling layer to achieve a good distribution of 0s and 1s. See appendix B.1 for more details on the systems parameters.

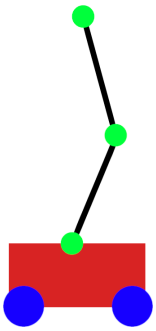


**Figure 7.4:** A benchmark for performance: N-inverted pendulum on a cart with N=2. The idea is to balance the inverted pendulums by moving the cart left and right, implying the system is 2D. The degrees of freedom of the system are illustrated by green circles as hinges, and blue circles as wheels. Mass is emulated as point masses at each node, with the mass at the bottom node representing the cart.

The balancing dataset was created based on an article featuring the SymPy python package for dynamics [20]. The idea was to balance n inverted pendulums, and log how a LQR controller pushed the cart left and right in response to the system states. An LQR is a linear approximation of a non-linear system emulating a spring model. The further a parameter goes from a equilibrium value, the more it contributes to a sum, like pulling a spring. The systems state parameters were comprised of the position and velocity of all the nodes, like in figure 7.4. In supervised learning the idea is to copy a known solution to a problem, in this case the LQR controller. Thus, by simulating the pendulum system balances by a LQR controller, data could be generated for supervised machine learning with the cart forces as the labelled data.

The initial conditions for pendulum balancing was set so that the task was difficult during training. All pendulum node angles were set to be within a reasonable range determined by manual experimentation and intuition. For each simulation, the angle and cart position was set by a uniform-random distribution. If the LQR controller failed to balance the system, the simulation was reset, discarding the data. The process was then repeated with a new initial condition until 2000 balancing acts were logged. The reason for making the initial conditions challenging was to improve the robustness of the balancing, as it was anticipated that the RC system might need to correct from difficult positions to compensate for potential

mistakes.

To further increase the data quality and reduce computation time, the simulations automatically stopped if one of two events occurred: balanced or fallen. The balanced event was defined as when the absolute sum of angles was vertical within a small error threshold. The fallen event was defined as when the base pendulum was below the horizontal. This should be a fair definition as the problem changes character past this point and becomes a pendulum swing up problem instead.

While creating the dataset, it was noticed that the pendulum simulations would spend little time in the unbalanced position, and spend more time with micro-movements striving towards a perfectly stacked system. To mitigate potential problems with an unbalanced dataset, the most common training samples were filtered out in such a way that the kurtosis and skew where closer to a bell curve shape to get a more balanced dataset. Kurtosis and skew are shape descriptors of a bell curve.

In the N inverted pendulum balancing simulations the N was limited to 2, as the number of variables are $2x(N+1)$, where each node has 2 variables for position and position rate. The first two representing the cart node, and the rest representing the pendulum nodes. This translates to $2x(2+1) = 6$ variables for the most difficult balancing problem.

## 7.5 Configuration Sweep and Evaluation

Ideally the pass/fail distribution over the reservoirs should be evenly distributed, so that it is easier to draw conclusions. Thus, the reservoirs were methodically categorized as having passed or failed a balancing with a easier set of criteria compared to training. The initial conditions were manually tweeked so that only a handful of reservoir would fail, giving more information on how to compare them based on their computing potential. The initial conditions are summarized in appendix B.2.

A parameter sweep was made to reduce biases of configurations in the design, as well as to uncover how the number of bits per reservoir effected performance. The sweep parameters are summarized in appendix B.3. Furthermore, each balancing benchmark test was attempted with the top 5 scoring model configurations from the sweep. Figure 7.5 is a sample training run from the sweep made to configure the model parameters used for the balancing dataset. Note how there is a clear gap in accuracy between the train and test data-sets as the curves begin their asymptotic phase, indicating that there may be some over-fitting.

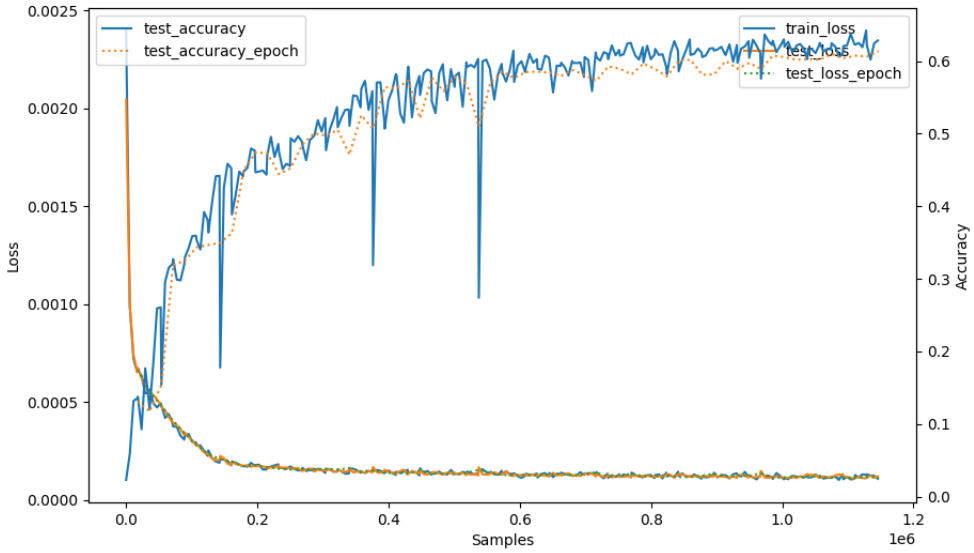



**Figure 7.5:** Example training curves from one of many configuration in the sweep. This run was for the one pendulum dataset with eight readouts, 20 bits per reservoir, a 10x10 reservoir size, and was able to balance. Note how accuracy and loss are inversely related, and stabilize asymptotically as expected after 64 epochs of training. The dashed epoch lines are the same as their solid counter-part, but are sampled only at every epoch; each time a full pass is done on the train-dataset.

# Chapter 8

# Results - ESP Metrics

## 8.1 Predicting Performance on the Balancing Benchmark

In this section, the ESB metric will be compared with quantitative results from the n-inverted pendulum on a cart balancing problem, or balancing problem for short. The idea is to see if the ESB can predict success or failure, which would validate the proposed metric. Other simple graph metrics were also calculated to expand the discussion and facilitate the discovery of correlations. The balancing benchmark proved to be more difficult than anticipated, so the highest number of pendulums balanced was two. Since this is a limited way to make comparisons, the number of pendulums balanced serves as a rough classification, and loss is used for further comparison. The loss algorithm is the L2 loss, which is merely the normalized difference between the target value and the predicted value. Therefore, a comparison of loss values on the machine learning test-dataset is reasonable. The results with both n=1, and n=2 for the balancing benchmark are in tables 8.1, 8.2, respectively. The listed entries are the first configuration to balance from the sweep, as mentioned in section 7.5.

Note from appendix B.2 that the initial conditions were much harder for n=1 than for n=2. All the reservoirs could balance one pendulum, even with initial conditions that were much harder than in the training environment.

In table 8.2 some reservoirs find the benchmark challenging and fail to balance two pendulums despite trying five of the most promising configurations from the sweep.

**Table 8.1:** N=1 balancing benchmark

| size | balanced | readouts | bits-per-reservoir |
|---|---|---|---|
| 10 | True | 8 | 3 |
| 11 | True | 8 | 3 |
| 12 | True | 8 | 4 |
| 13 | True | 8 | 3 |
| 14 | True | 8 | 4 |
| 15 | True | 8 | 16 |
| 16 | True | 8 | 31 |
| 17 | True | 8 | 3 |
| 18 | True | 8 | 32 |
| 19 | True | 8 | 3 |
| 20 | True | 8 | 5 |

**Table 8.2:** N=2 balancing benchmark

| size | balanced | readouts | bits-per-reservoir |
|---|---|---|---|
| 10 | True | 8 | 7 |
| 11 | False | n/a | n/a |
| 12 | False | n/a | n/a |
| 13 | True | 8 | 27 |
| 14 | False | n/a | n/a |
| 15 | False | n/a | n/a |
| 16 | True | 8 | 33 |
| 17 | True | 8 | 25 |
| 18 | True | 8 | 22 |
| 19 | True | 8 | 5 |
| 20 | True | 8 | 30 |


**(a)** 10x10 reservoir balancing 1 pendulum **(b)** 13x13 reservoir balancing 2 pendulums

**Figure 8.1:** Balancing animations. Note that the figures can be viewed as animations with the following pdf readers: AcrobatReader, PDF-XChange, acroread, and Foxit Reader. The simulations and animations were based on a blog featuring SymPy, a symbolic maths library for python [20]

### 8.1.1 Graph properties

In figure 8.2, there are six graph properties describing all the tested reservoirs. The properties considered were the number of nodes in the graph, both for the entire graphs, and in SCGs. For the node count in SCGs, there was placed a condition that there had to be more than one node, excluding a large amount of self-referencing nodes. The SCG node count is thereby a more accurate representation viewing the graphs a combination of random forests followed by SCG regions in section 7.1. The number of edges is directly related to the number of nodes, as each node has the option of traversing the graph further by an input of either 1 or 0. Thus, the edge count is double the number of total nodes. The self-referencing edges were therefore excluded from the edges metric, to make a metric with new information relative to the nodes metric. In section 7.1, it was pointed out that the magnets could be compared to a solidification of a liquid over time, as only a handful of magnets were flipping. This inspired a metric for the number of flippable magnets in the whole graph and cumulatively in the SCG regions.

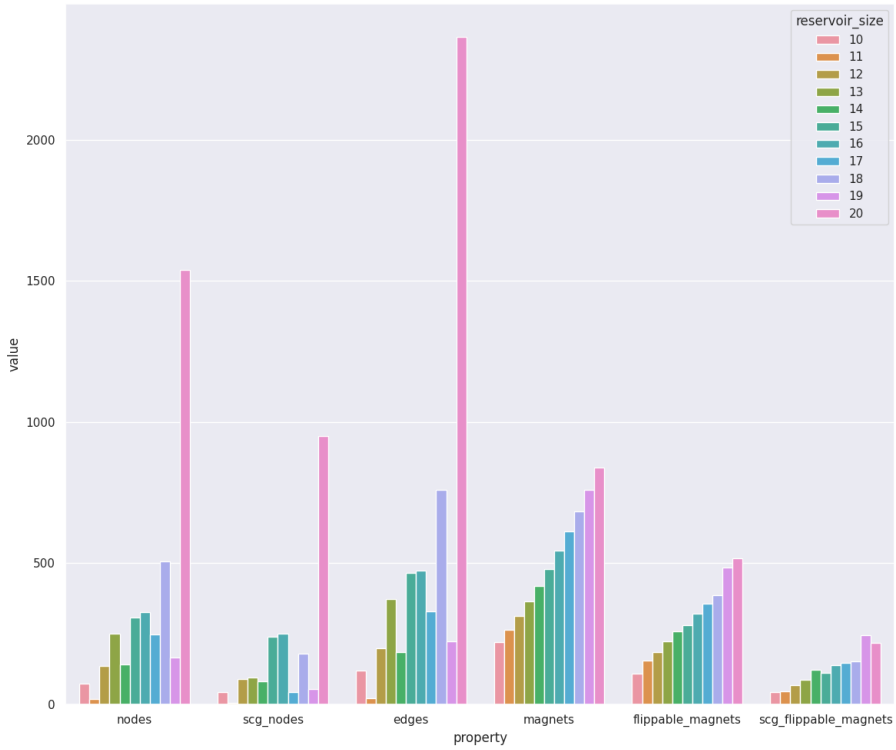



**Figure 8.2:** Graph Properties

Since the metrics in figure 8.2 are cumulative, a more detailed visualization was made for the actual distributions of SCGs in the reservoirs in figure 8.3. To avoid the overlapping of points in the scatter-plot, the x-axis was made continuous, despite the reservoir size being a discrete value. I.e., for the reservoir with

size 10, all relevant points are in the continuous range [10, 11). The individual size of each SCG can be read off the y-axis. As expected, there is a trend of growing SCG sizes with higher reservoir sizes.

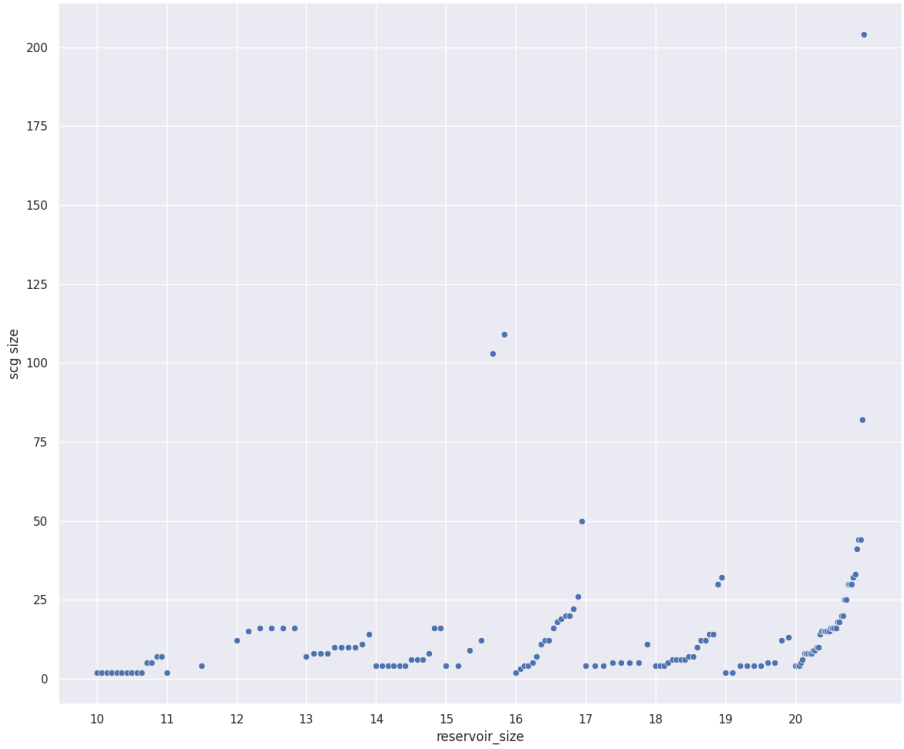



**Figure 8.3:** SCG distribution over the various reservoir sizes. Note that the reservoir size is still discrete, but has been spread over a continuous range to avoid the overlap of points.

### 8.1.2 ESB

As an attempt to measure the ESP, the ESB is the most important metric in regard to the goals of this thesis. In previous work [1], where the metric was introduced, the idea was to probe the reservoir engineered inputs, which will affect the shape of the ESB curve. Only two of many proposed engineered inputs were chosen, due to their simplicity and ease of producing numerous samples. The problem with the other engineered inputs, is that they are meant to be unique, but with few bits, the set of inputs is very small. Furthermore, the analysis would be more complicated if all the engineered inputs were to be considered. Hence, only the random and fixed density inputs are featured in the figures below. On the second axis to the right of the plot, an overlay of the machine learning test-loss was plotted to see if there was a correlation. A figure was made for each reservoir, and can be found in appendix B.4.

It is beneficial for the learning process in the output layer when the test-loss is low. The top five lowest points in the plot are therefore the points where the balancing benchmark was tested. It is difficult to comment on general tendencies for all of the graphs, as many of them are quite different. There seems to be little correlation between the plotted ESB and the loss curves. Some of the plots show a positive relationship with increasing bits, while others show a negative relationship. Surprisingly, in figure 8.4, the fixed density curve is almost flat. Nonetheless, there are some trends. Note how the test-loss has many points of optimal performance along the n_bits axis, keeping in mind that a low loss is desirable. The loss curves seem to make big fluctuations over small changes of input bits.

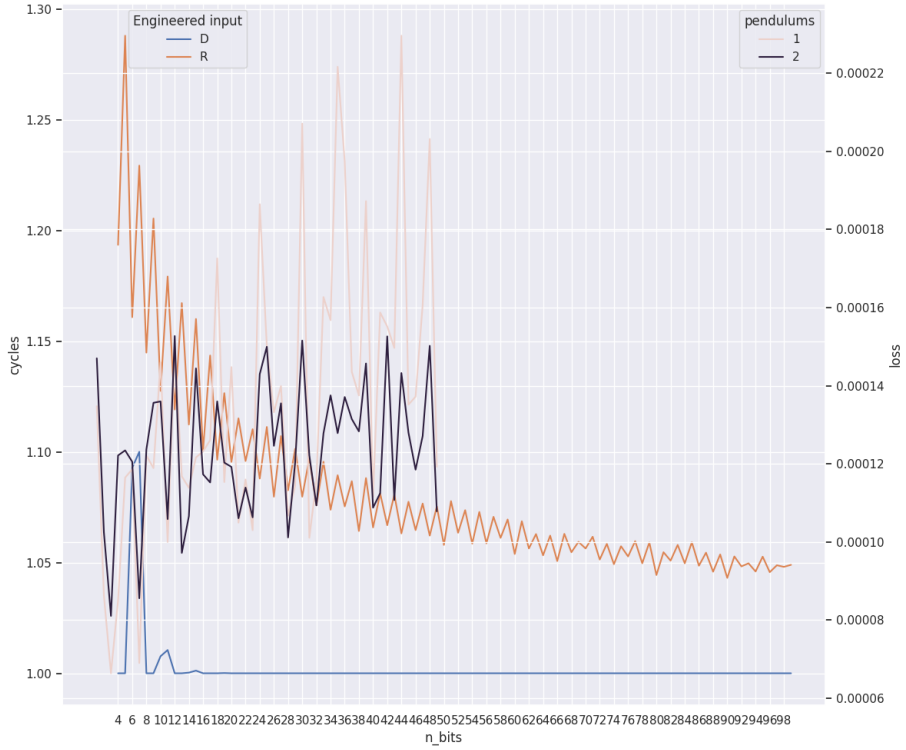



**Figure 8.4:** ESB metric with random and 50% fixed density inputs for the 10x10 reservoir. The graph is an average over 10k samples. The start nodes are random from within a SCG. On the second axis the loss from the test-dataset was plotted for easier detection of correlations. Note that the n_bits axis represents the length of the engineered input for the ESB metric, and the number of bits per reservoir in the loss metric.

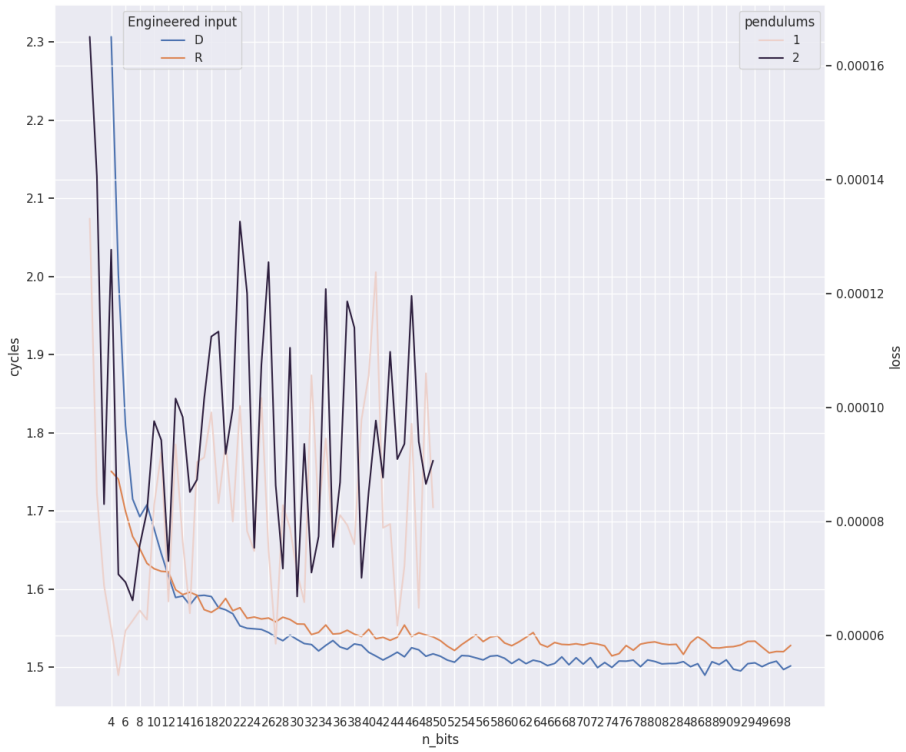



**Figure 8.5:** ESB metric with random and 50% fixed density inputs for the 20x20 reservoir. The graph is an average over 10k samples. The start nodes are random from within a SCG. On the second axis the loss from the test-dataset was plotted for easier detection of correlations. Note that the n_bits axis represents the length of the engineered input for the ESB metric, and the number of bits per reservoir in the loss metric.

# Chapter 9

# Discussion - ESP Metrics

It was discovered that when going from the non-linear benchmark to the balancing benchmark that the architecture performed better with tally encoding as opposed to tally-shuffle-interleave. This may be because the RC system is trying to emulate a LQR controller, which functions like a dot product of scalar weights with the systems state vector. The dot product result is the force needed by the cart to balance the system. The state vector holds the position and velocity information of each node as shown in figure 7.4. A dot product may be easier to copy if the variables in the RC system are kept separate, with minimal interaction between variables. Thus, it could be advantagous to keep the variables less entangled across reservoirs, which is achieved by not interleaving input during encoding. The damage cause by shuffling, however, is most likely because the feedback-and model entangles its inputs with the pooling layer, which means that it already has a good distribution of 1s and 0s. Too many consecutive 1s and 0s were flagged as a problem in section 3.4.1.

There is a general lack of experiment control in regard to the balancing problem. For one, the metric was biased based on how well the machine learning could be tuned for a particular reservoir. To counter this issue, a sweep was made to search for various configuration candidates. However, the same issue can be said of the encoding scheme, as the tally-shuffle-interleave mechanism may have worked well as a general encoder, but some reservoirs worked better with other encoding schemes. As mentioned above, this was the case, but the distributed encoding exemplified in figure A.1 may have worked even better. Having the model and output layer as part of the assessment made it difficult to evaluate the quality of a given reservoir due to factors such as these. With this uncertainty, it was difficult to correlate performance with the ESB.

Even assuming that the comparisons were fair, there is another issue. In simulations with the LQR controller it was possible to simulate up to five pendulums with the parameters set in B.1. The analysis was capped at two pendulums, since none of the reservoirs could balance more. The failure to balance more pendulums was most likely due to the increasing amount of variables required, as noted in section 5. The problem also becomes much more difficult for each added pendu-

lum, so even the LQR controller could not balance five pendulums in attempted simulations.

Next, consider the graph metrics. In section 7.1, it was noted that most magnet spins seem to be relatively fixed within their SCG. Revisiting figure 7.1 note how there are nine leaf SCGs. Traversing between the SCGs is impossible by their definition of being strongly connected, so this means that each of them could be considered its own category. Thus, traversing this graph with a large amount of input bits will always lead to one of these SCGs. Now, observe in figure 9.1, the same graph, but arranged in a way that interesting substructures are exposed. If the magnets where arranged in a less fixed manner, it is predicted that more substructures will exist. Theoretically, by increasing the number of possible neighbourhoods a traversal can end in, the graph should act as a better classifier. Therefore, the number of dots in figure 8.3 for a given reservoir should be indicative of the number of categories a reservoir can classify with the aforementioned mechanism. However, this did not function well as a predictor of success or correlate with ESB.

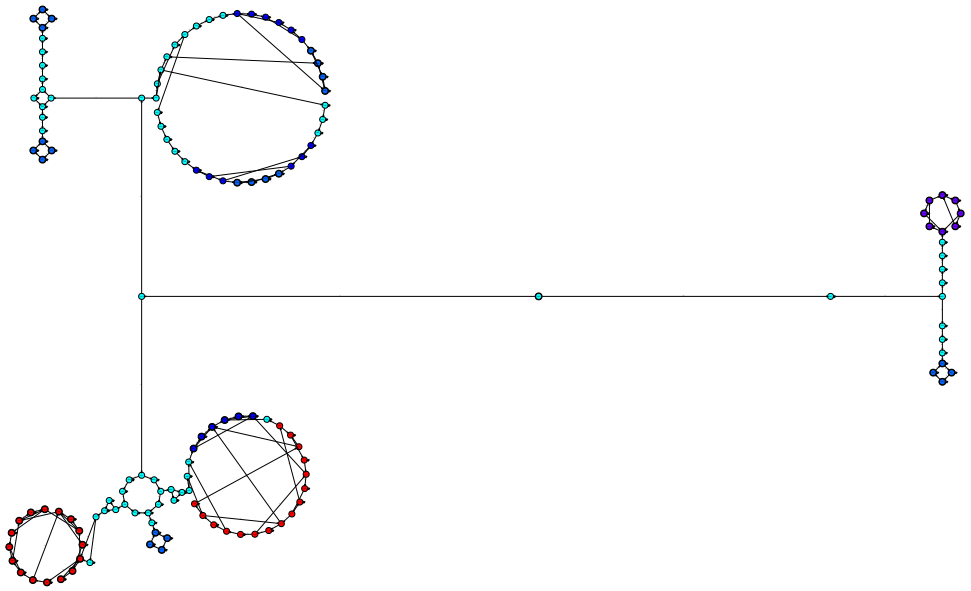


**Figure 9.1:** 14x14 graph, identical to 7.1, but arranged so that its substructures are exposed.

The ESB metric was introduced as measurement in bits, but on the left axis of the ESB figures it is represented as cycles, as this is a more normalized expression. The bit representation is calculated by multiplying the cycles by the number of bits. Several local optima seem to cluster around the transition of the ESB from vertical to horizontal. It is hard to support this claim beyond a generous qualitative assessment. A more tangible behaviour is that a small change in input bit length leads to a large change in loss. In figure 8.5, the ESB and training test-loss

are overlaid to facilitate the detection of cyclic performance, as predicted by wave theory comparisons mentioned in 7.2. This pattern may have been detected in the dramatic fluctuations of the loss curves shown for both loss curves with pendulums set to n=1, and n=2. When the loss is low, the model performs better. Note that in many plots the the curves for both pendulums respond fluctuate similarly, indicating an expected indifference to the dataset. Further analysis is needed to ensure that all these behaviour can be confirmed in a more systematic way. The rest of the reservoir ESB plots are in appendix B.4.

The ESB has been simplified from 10k samples to an average, but it may be that an average is not a good way to represent the ESB. This is mentioned because there is no apparent correlation between the fluctuations and the ESB. However, it may be that different engineered inputs can reveal how a reservoir classifies strings, as pointed out by the flat fixed-density curve in figure 8.4. This was manually verified in the reservoir graph, as there was concern it was due to a mistake. Thus, the 10x10 reservoir can separate random inputs from fixed-density inputs, despite it having a very small SCG node population.

Assuming that the ESB is not correlated with test-loss, then this may indicate that the output layer may favour the random forest mechanism described in section 7.1, where the ESB is argued to interact only with the SCG regions in a graph. Furthermore, when searching for reasonable model and training parameters for the sweep, it was noticed that the bits per reservoir setting was relatively small compared to the depth of the graph. A sign that the training didn't seek out the SCGs at the leaf nodes of the graph, but rather focused on early traversal. The sweep parameters are further justified in section B.3.

In section 5.2.3, it was argued that the reservoir was performing poorly, as its output layer could out-compete it. From the bit state probability map in figure 7.2 and the graph metrics in 8 such as the flippaple magnets metric, it is clear that many of the bits were actually fixed, very roughly estimated at a ratio of 1/4. Furthermore, it would seem the spin states were highly determined by if their last edge was a 1 or a 0 throughout the graph topology. This claim is supported by automatically classified graphs in appendix A.4. This means most of the reservoirs' 2D area is not being used effectively. It is therefore impressive that the results achieved where possible from only a handful of changing bits. Hence, the flippable magnet metrics indicate the output layer signals may be multiplexing their collective outputs; they effectively rely on a very small number of magnets states to produce numerous outputs.

# Chapter 10

# Conclusion - ESP metrics

The ESP metrics could not predict success or failure on the balancing benchmark. As discussed in section 9, there may be many reasons for this, including a general lack of experiment control. Most notably, there were only two pendulums balanced, which makes it difficult to rate the reservoirs on such a small scale. Furthermore, there were too many ways for unwanted influence from the input and output layers. And from the reservoir layer, it was discovered that the magnet spins were relatively fixed. This may indicate the geometry wasn't optimal, assuming more dynamic magnet flipping is beneficial for performance on the chosen benchmark. The performance with tally encoding instead of tally-shuffle-interleave was quite drastic for this benchmark. The results in section 4 make it clear that many design features and settings could still have been adjusted to balance more pendulums.

The analysis could not show how the graph metrics were related to the ESB metric. Furthermore, neither the graph metrics nor the ESB could predict success in the balancing benchmark. However, the main goal of the thesis was not the practical task in itself, but rather testing the ESB as a metric for the ESP. From figure 8.5, there seems to be several optimal bit lengths for reducing the test-loss training in the output layer. The loss curves seem to behave as predicted in section 7.2 by discussions of a fundamental input lengths. Whenever the input bit length is a multiple of such a fundamental length it may lead to a large response. The fluctuation over small input length may be an expression of this mechanism. However, the ESB metric does not seem to correlate directly with the ESB curve. Due to time limitations, a fourier analysis could not be conducted to uncover the fundamental frequencies which might be expressed. It is suggested that this may be due to the averaging of ESB values, effectively removing fluctuations in its current representation.

The cyclical behaviour of the the loss curves may also be due to a lack of experiment control, as other factors may play a bigger role than the ESB. More investigation is needed to see if the engineered inputs from the ESB can be used as a way to uncover how a reservoir classifies inputs as plots like figure 8.4 indicated engineered input have this potential.

# Chapter 11

# Conclusion & Future Work

The first part of the thesis focused on practical RC with the non-linear dataset. It was relatively successful considering the design search space is huge. Many parameters had to be frozen to limit this design space, such as the reservoir size, geometry, encoding, definition of inputs, etc. In this list of constraints to the design was also the machine learning approach, which was set to supervised learning. Regrettably, an unsupervised machine learning approach could not be explored where the reservoir itself would have more opportunity to learn how to balance the pendulums themselves. The machine learning would then take the form of an indirect guidance with metrics like balance time, as opposed to imitating the LQR controller. The techniques for converting reservoirs to graphs were also useful, but there are many more interesting tools from this field which could not be explored due to time restraints. I.e. a von-nueman entropy graph metric was to be explored, which could potentially capture information about the quality of a graph's connectivity, but this will be left for future work.

The ESP can be intuitively understood, but there is no true baseline from which results can be verified. This may give the impression of a weak conclusion. Hence, future work hinges on the maturity of the understanding of the ESB as a baseline. It was ambitious to correlate the ESB metric with the balancing benchmark, and it may have been wiser to continue with well-defined functions instead, like the non-linear dataset. The number of variables could have been increased before switching to the balanced dataset, so that issues due to the increased number of variables wouldn't be a problem. However, the techniques and metrics developed for evaluating reservoirs are still interesting for future analysis. The need for metrics on the ESP has not been satisfied by the investigation from this thesis, but many approaches have been identified, including metrics from graph theory as well as the ESB. Most interestingly is the supposition that SCG regions are intricately tied to the ESP, an idea which should be investigated further.

# Bibliography


[1] C. M. Vibe, *Echo state metrics in reservoir computing*, 2021.

[2] J. H. Jensen, A. Strømberg, O. R. Lykkebø, A. Penty, M. Själander, E. Folven and G. Tufte, *Flatspin: A large-scale artificial spin ice simulator*, 2020. arXiv: 2002.11401 [physics.comp-ph].

[3] D. Verstraeten, B. Schrauwen, M. D'Haene and D. Stroobandt, 'An experimental unification of reservoir computing methods,' *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007, Echo State Networks and Liquid State Machines, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2007.04.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360800700038X.

[4] T. Nakagaki, 'Smart behavior of true slime mold in a labyrinth,' *Research in Microbiology*, vol. 152, no. 9, pp. 767–770, 2001, ISSN: 0923-2508. DOI: https://doi.org/10.1016/S0923-2508(01)01259-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923250801012591.

[5] A. W. Burks, H. H. Goldstine and J. von Neumann, 'Preliminary discussion of the logical design of an electronic computing instrument,' in *The Origins of Digital Computers: Selected Papers*, B. Randell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 399–413, ISBN: 978-3-642-61812-3. DOI: 10.1007/978-3-642-61812-3_32. [Online]. Available: https://doi.org/10.1007/978-3-642-61812-3_32.

[6] C. Fernando and S. Sojakka, 'Pattern recognition in a bucket,' in *Advances in Artificial Life*, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich and J. T. Kim, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 588–597, ISBN: 978-3-540-39432-7.

[7] H. Jaeger, 'The" echo state" approach to analysing and training recurrent neural networks-with an erratum note',' *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, Jan. 2001.

[8] V. Bush, 'The differential analyzer. a new machine for solving differential equations,' *Journal of The Franklin Institute-engineering and Applied Mathematics*, vol. 212, pp. 447–488, 1931.

[9] M. Dale, J. Miller and S. Stepney, 'Reservoir computing as a model for in-materio computing,' in Jan. 2017, vol. 22, pp. 533–571, ISBN: 978-3-319-33923-8. DOI: 10.1007/978-3-319-33924-5_22.

[10] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, 'Extreme learning machine: Theory and applications,' *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006, Neural Networks, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2005.12.126. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385.

[11] A. Thompson, *An evolved circuit, intrinsic in silicon, entwined with physics*, Undetermined. DOI: 10.1007/3-540-63173-9_61.

[12] J. H. Jensen and G. Tufte, 'Reservoir computing in artificial spin ice,' in *Reservoir Computing in Artificial Spin Ice*, 2020.

[13] L. Pauling, 'The structure and entropy of ice and of other crystals with some randomness of atomic arrangement,' *Journal of the American Chemical Society*, vol. 57, no. 12, pp. 2680–2684, 1935. DOI: 10.1021/ja01315a102. eprint: https://doi.org/10.1021/ja01315a102. [Online]. Available: https://doi.org/10.1021/ja01315a102.

[14] S. T. Bramwell and M. J. Harris, 'The history of spin ice,' *Journal of Physics: Condensed Matter*, vol. 32, no. 37, p. 374 010, Jun. 2020. DOI: 10.1088/1361-648x/ab8423. [Online]. Available: https://doi.org/10.1088/1361-648x/ab8423.

[15] H. Yasuda, P. Buskohl, A. Gillman, T. Murphey, S. Stepney, R. Vaia and J. Raney, 'Mechanical computing,' English (US), *Nature*, vol. 598, no. 7879, pp. 39–48, Oct. 2021, Funding Information: Acknowledgements H.Y. and J.R.R. acknowledge support from Army Research Office award number W911NF-1710147, Air Force Office of Scientific Research award number FA9550-19-1-0285 and DARPA Young Faculty Award W911NF2010278. P.R.B., A.G. and R.A.V. acknowledge support from the Materials and Manufacturing Directorate and the Air Force Office of Scientific Research of the Air Force Research Laboratory. T.D.M. acknowledges support from NSF 1837515 and ARO MURI award W911NF-19-1-0233. S.S. acknowledges support from the SpInspired project, EPSRC grant number EP/R032823/1. Publisher Copyright: © 2021, Springer Nature Limited., ISSN: 0028-0836. DOI: 10.1038/s41586-021-03623-y.

[16] M. Själander, M. Jahre, G. Tufte and N. Reissmann, *EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure*, 2019. arXiv: 1912.05848 [cs.DC].

[17] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2017. DOI: 10.48550/ARXIV.1711.05101. [Online]. Available: https://arxiv.org/abs/1711.05101.

[18] L. Appeltant, M. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. Mirasso and I. Fischer, 'Information processing using a single dynamical node as complex system,' *Nature communications*, vol. 2, p. 468, Sep. 2011. DOI: 10.1038/ncomms1476.

[19] G. V. Steeg and A. Galstyan, *Discovering structure in high-dimensional data through correlation explanation*, 2014. DOI: 10.48550/ARXIV.1406.1222. [Online]. Available: https://arxiv.org/abs/1406.1222.

[20] J. K. Moore. [Online]. Available: https://notebook.community/oliverlee/pydy/examples/npendulum/n-pendulum-control.

# Appendix A

# Practical RC

## A.1 Experiment parameters

This description is taken from previous work [1]. The magnet geometry parameters are tuned to produce a rectangular stadium-shaped magnet, as used in [12]. Their geometrical meaning is defined in [2]. Note that the grid size is the number of geometry instances on the grid, not the number of magnets; i.ex. number of pinwheel instances. Consult flatspin's documentation for more parameters default to the PinwheelSpinIceDiamond class, which was used for all experiments, and not listed below. To ensure the system was deterministic for experiment control, a random seed was set, as well as a deterministic "flip_mode" mechanism during simulation.

**Table A.1:** Experiment parameters

| Parameter | Value | Comment |
|---|---|---|
| $G_d$ | [10:20] | grid dimension sweep (nxn geometry instances) |
| $H_i$ | 78mT | input field strength |
| $H_c$ | 200mT | coercive field strength |
| $H_r$ | $7^o$ | anti-clockwise rotation of global field vectors |
| $M_l$ | 220nm | length of magnets |
| $M_w$ | 80nm | width of magnets |
| $M_h$ | 20nm | height of magnets |
| $M_\alpha$ | 1.02e-3 | spacing of magnets (dipolar coupling strength) |
| $b$ | 0.41 | magnet geometry parameter |
| $c$ | 1.0 | magnet geometry parameter |
| $\beta$ | 1.5 | magnet geometry parameter |
| $\gamma$ | 3.9 | magnet geometry parameter |
| $random\_seed$ | 0 | random seed for flatspin++ |
| $flip\_mode$ | "max" | flipping mode in flatspin |

## A.2 Architecture experiment parameters

The configuration settings below where the same for all architectures in section 3.3. Citation for AdamW: [17].

**Table A.2:** Architecture Experiment parameters

| Configuration | Value |
|---|---|
| reservoir size | 20x20 |
| learning rate | 1e-4 |
| optimizer | AdamW |
| cost function | L1 Loss |
| epochs | 512 |
| batch size | 512 |
| dataset size | 20k (random values) |
| train/test/dev split | .7/.1/.2 |

## A.3 Distributed Encoding

The following prediction image is a demonstration of how a cross between binary and tally encoding may look like. The ensemble architecture, like in figure 4.3 was used for this demonstration, copying its setup parameters. Each bit in this encoding is a power of two: 1, 2, 4, 8, repeated, ie. $(1011)(0100))dot(1248)(1248) = (1x1 + 0x2 + 1x4 + 1x8) + (0x1 + 1x2 + 0x4 + 0x8) = 13 + 2 = 15$. The full approach is not explained here, but note that it leads to several ways of encoding the same number. The approach is not discussed further, as tally encoding was much more stable for machine learning.

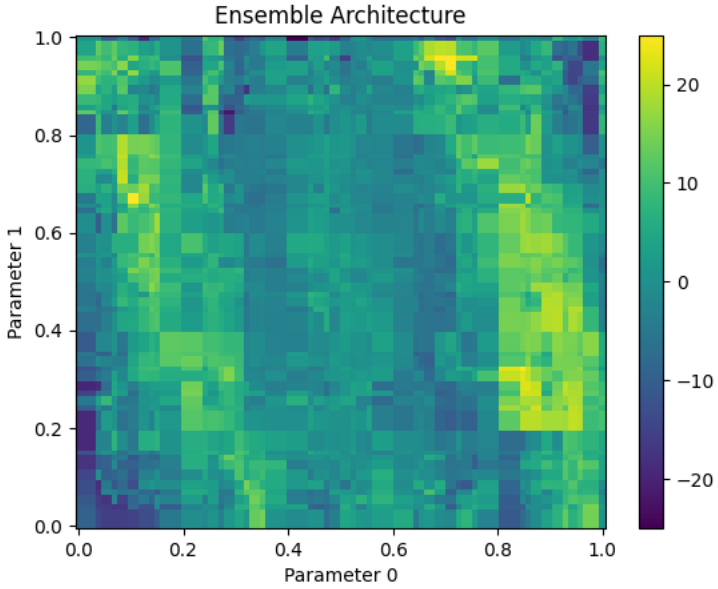



**Figure A.1:** Ensemble architecture prediction image relative to figure 3.6. The resolution is approximately 32x32, and the prediction accuracy is 43%.

## A.4 The Reservoir Graph as a Classifier

The following graphs demonstrate automated classification techniques on the 20x20 reservoir. Each graph uses 32 colors to classify each state. The graphs are three of four approaches as discussed in section 5.2.2. Observe the categorization with label-k-means, mca-k-means, and corex, in figures A.2, A.3, A.4, respectively. Note how many substructures are classified the same, this is because the difference in magnets spins is actually very small. There also seems to be separate subcategories depending if the last input was a 1 or a 0. The Labeling-PCA approach was simply not graphed due to time restraints, as it controls RGBA independently, while the other approaches use discrete categories.

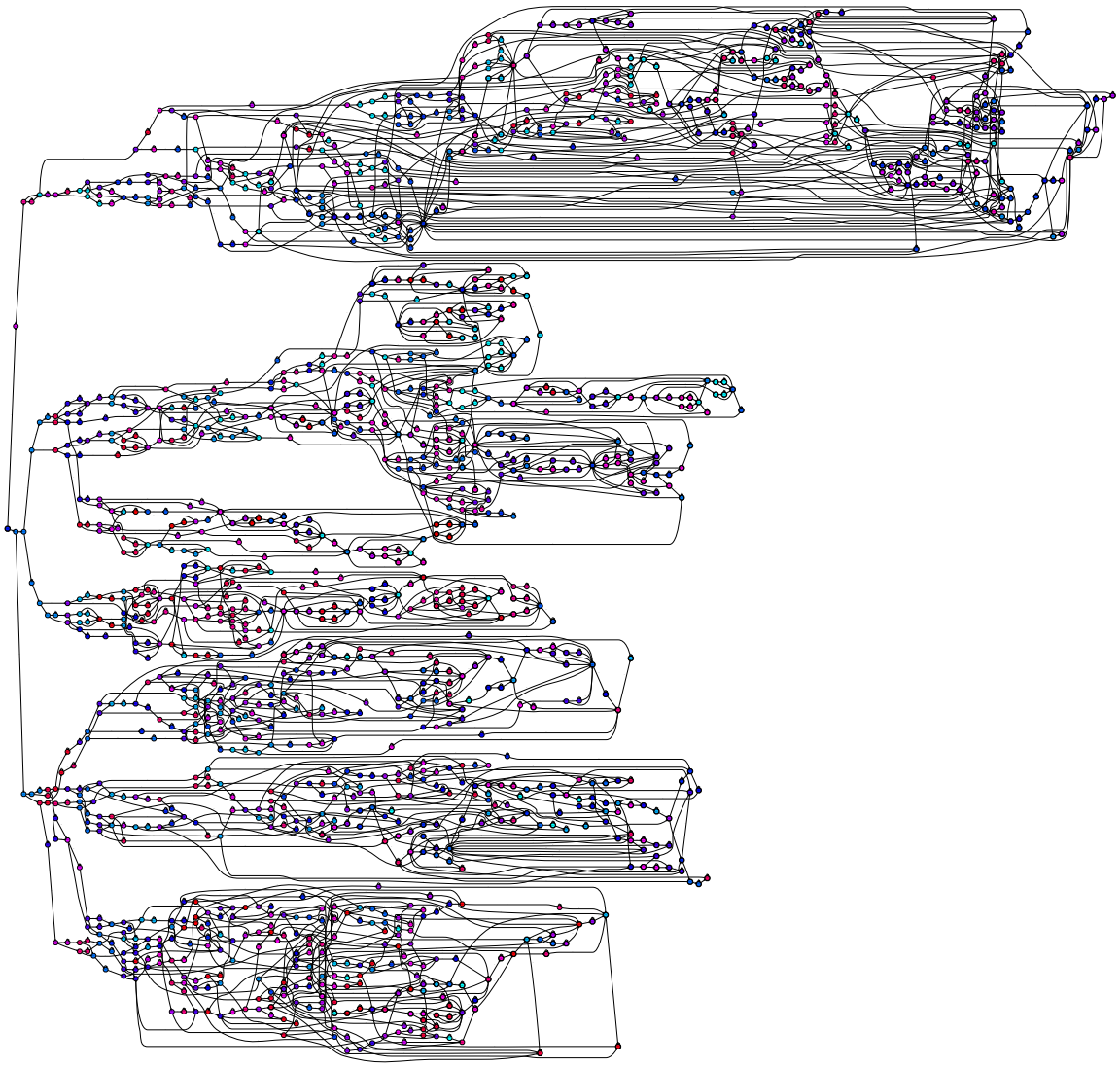


**Figure A.2:** This 20x20 reservoir graph has been coloured based on the classification of the Label k-means strategy discussed in section 5.2.2.

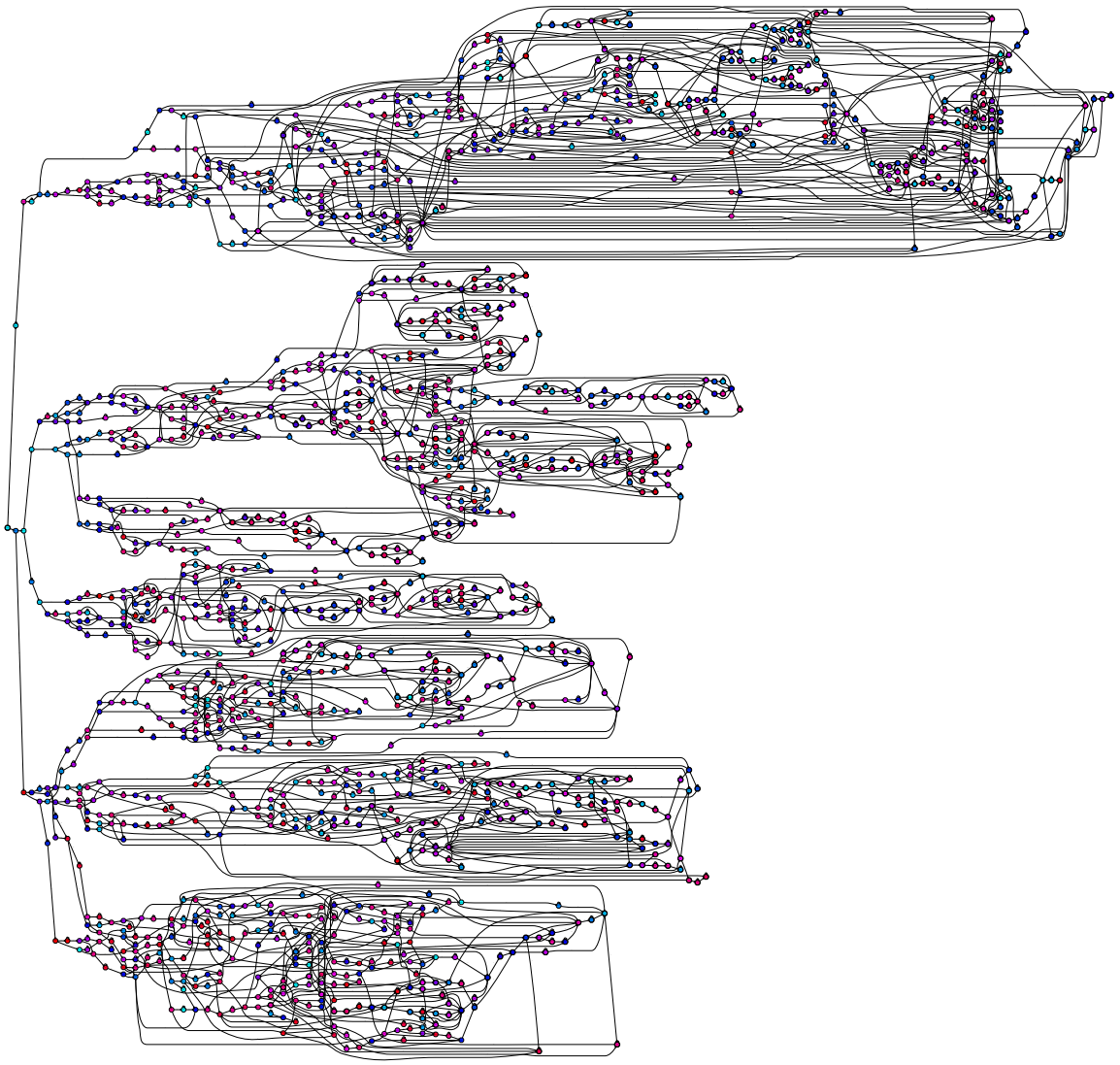


**Figure A.3:** This 20x20 reservoir graph has been coloured based on the classification of the Label k-means strategy discussed in section 5.2.2.

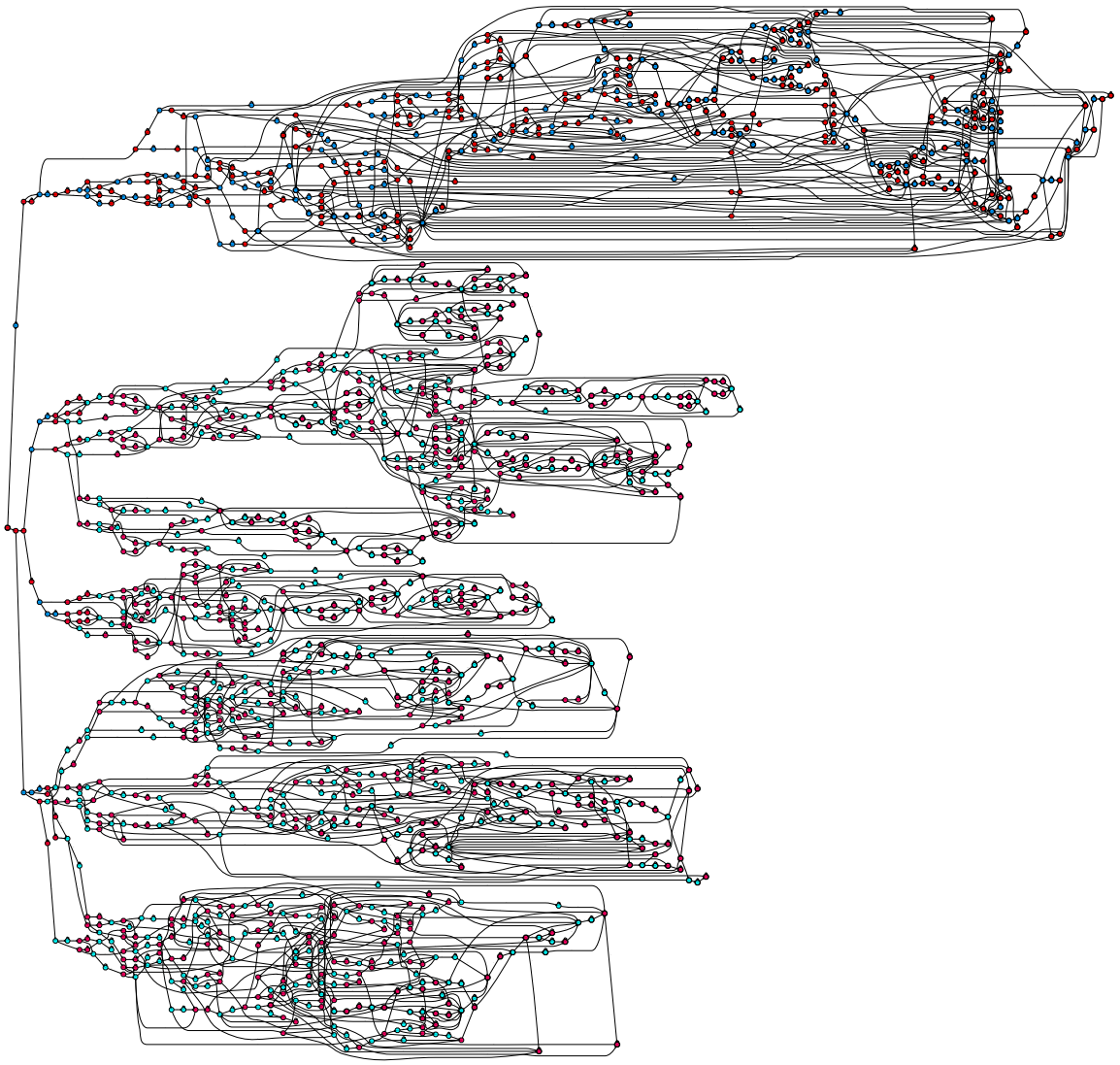


**Figure A.4:** This 20x20 reservoir graph has been coloured based on the classification of the corex classification strategy discussed in section 5.2.2.

## A.5 Additional Reservoir Graphs

Some of the graphs where very large, the biggest being the 20x20 ASI reservoir with over 1500 states. This section will make space for some of them for the interested reader.

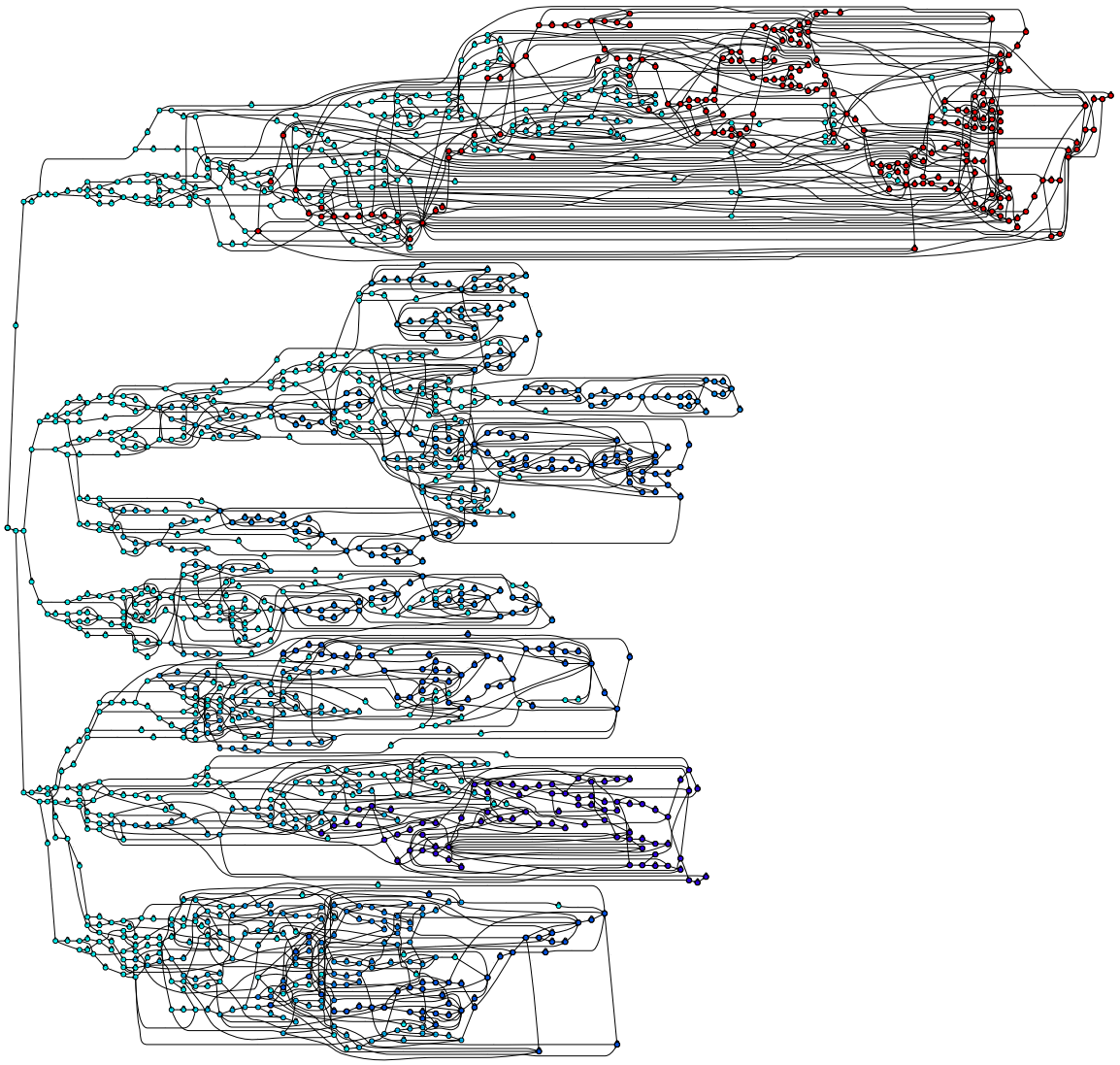


**Figure A.5:** An ASI reservoir with a 20x20 diamond pinwheel configuration as a directed graph. All possible spin states are nodes, with inputs as transitional edges. All SCG subgraphs are coloured by size.

**Figure A.6:** An ASI reservoir with a 15x15 diamond pinwheel configuration as a directed graph. All possible spin states are nodes, with inputs as transitional edges. All SCG subgraphs are coloured by size.

## A.6 Bonus Material: Balancing 2 Pendulums with the 20x20 Reservoir

The following figures show a detailed plots related to balancing 2 pendulums with the 20x20 reservoir just as featured in table 8.2. When reading the graphs, remember that the bottom node is the cart labelled node 0. The pendulums are the rest of the nodes. E.g. q0 is cart position, while u0 is cart speed.

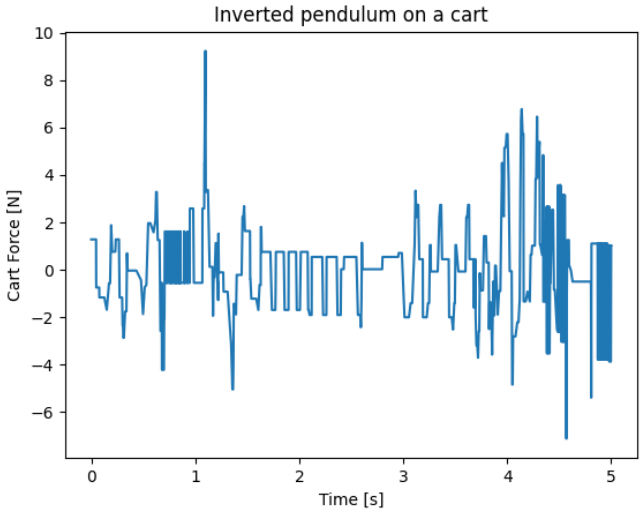



**Figure A.7:** 20x20 balancing: cart force plot.

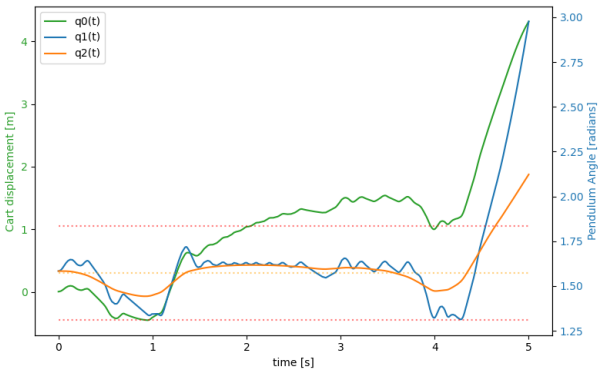



**Figure A.8:** 20x20 balancing: node displacements.

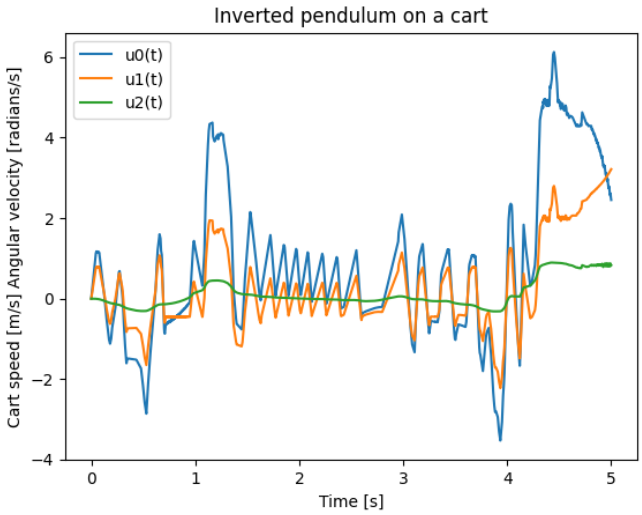



**Figure A.9:** 20x20 balancing: node speeds.

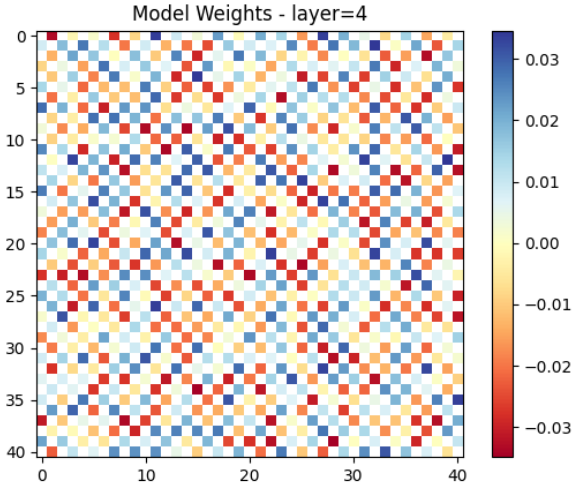



**Figure A.10:** 20x20 balancing: weights plot
20x20 balancing: weights plot at index 4 from layers [0,7].

# Appendix B

# ESP Metrics

## B.1 N-Inverted Pendulum System Parameters

The numbers summarized in figure B.1 were set by intuitive experimentation to get the largest system that was still able to balance.

**Table B.1:** N-inverted pendulum system parameters. The numbering is from cart to top pendulum, so Node0 corresponds to the cart. N is the number of inverted pendulums in the system.

| N | Part | Settings | Value |
|---|---|---|---|
| 1 | Node0 | mass | .10kg |
| 1 | Pendulum1 | length | 3.0m |
| 1 | Node1 | mass | .10kg |
| | | | |
| 2 | Node0 | mass | .05kg |
| 2 | Pendulum1 | length | 1.5m |
| 2 | Node1 | mass | .05kg |
| 2 | Pendulum2 | length | 1.5m |
| 2 | Node2 | mass | .05kg |
| | | | |
| 3 | Node0 | mass | .03kg |
| 3 | Pendulum1 | length | 1.0m |
| 3 | Node1 | mass | .03kg |
| 3 | Pendulum2 | length | 1.0m |
| 3 | Node2 | mass | .03kg |
| 3 | Pendulum3 | length | 1.0m |
| 3 | Node3 | mass | .03kg |

## B.2 N-Inverted Pendulum Initial Conditions

Since the system is naturally perpetually unstable, it was considered a fair test of the reservoirs capabilities, even when the initial conditions were relatively easy during the testing state. Furthermore, the criteria of balancing during training was reduced to not falling for 10 seconds, a number set based on the collapse of the system after roughly 1 second with no balancing controller. Note that all pendulum and cart values in table B.2, are given as a the total range from a reference point. E.g. .1 meter cart displacement means the cart is +- 0.05m from its reference point. All unit are in base units: radians, meters and seconds.

**Table B.2:** N-inverted pendulum initial conditions.

| n | stage | sampling | pendulum (r) | cart (m) | end-condition (s) |
|---|---|---|---|---|---|
| 1 | training | random uniform | $\pi/6$ | .1 | balanced |
| 2 | training | random uniform | $\pi/6$ | .1 | balanced |
| 3 | training | random uniform | $\pi/12$ | .1 | balanced |
| 1 | testing | deterministic fixed | $\pi/3$ | 1 | 10 (not falling) |
| 2 | testing | deterministic fixed | $\pi/128$ | .01 | 5 (not falling) |
| 3 | testing | deterministic fixed | $\pi/256$ | .01 | 5 (not falling) |

## B.3 Balancing Problem Sweep Parameters

In table B.3 are the parameters for the sweep made to train for the balancing benchmark. The parameters were selected based on experience from trial and error. As seen in an example training log in figure 7.5, the loss and accuracy stabilize nicely and are relatively smooth at 64 epochs. The readout sweep was originally [1, 8], but it was found that the sweep could be reduced to just 8 readouts, as the optimal values were mostly this value. Similarly, the bits per reservoir were originally from [1, 100], but the optimal values were usually more in the range of [0, 50]. Note that the ranges were only reduced to save on compute time.

**Table B.3:** Balancing problem sweep parameters

| Parameter | Value/Range |
|---|---|
| Epochs | 64 |
| Readouts | 8 |
| Bits per reservoir | [1-50] |

## B.4 Additional ESB Graphs

In this section addition ESB graphs are included. In each graph the ESB metric is plotted with random and 50% fixed density inputs. The graph is an average over 10k samples. The start nodes are random from within a SCG. On the second axis the loss from the test-dataset was plotted for easier detection of correlations. The graphs are discussed in more detail in section 9.

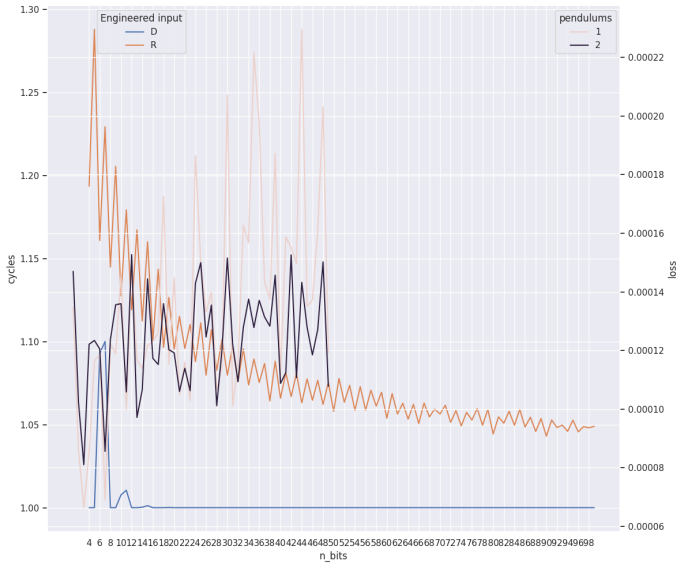



**Figure B.1:** ESB metric and test training loss for the 10x10 reservoir.

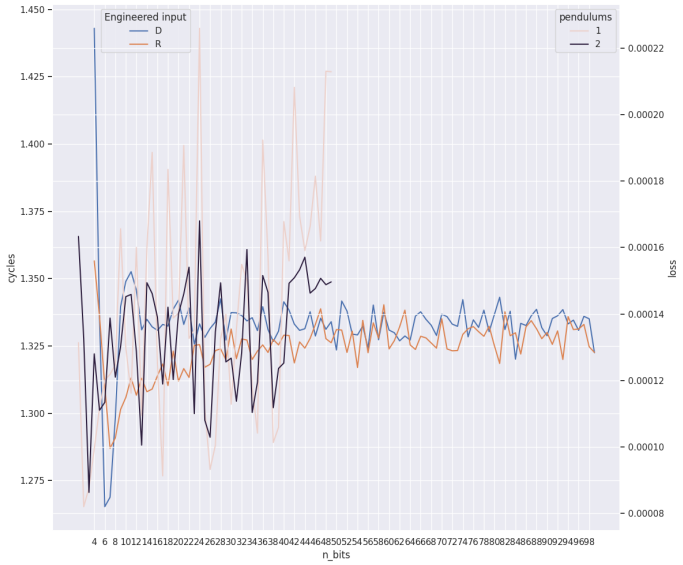



**Figure B.2:** ESB metric and test training loss for the 11x11 reservoir.

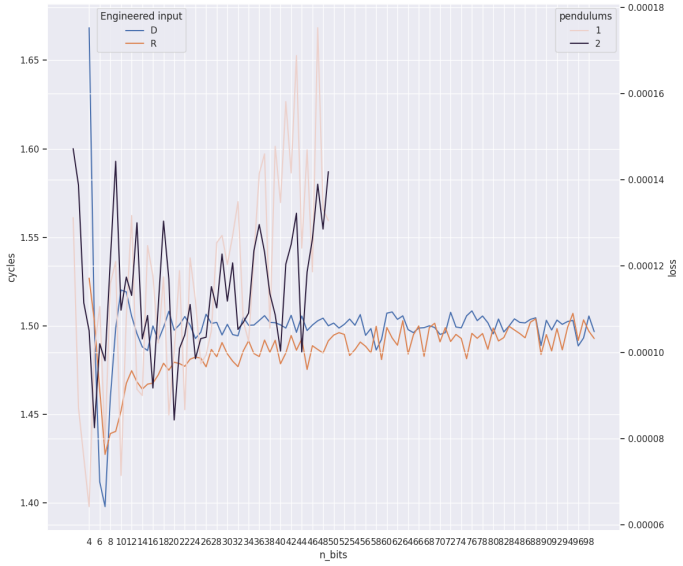



**Figure B.3:** ESB metric and test training loss for the 12x12 reservoir.

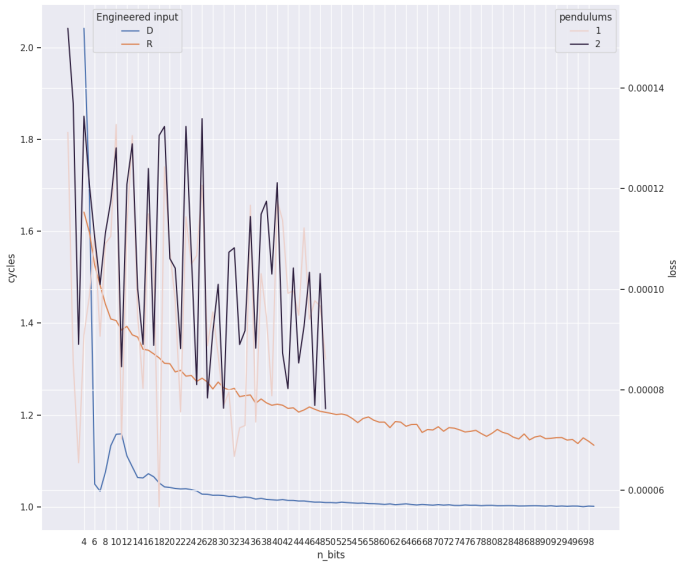



**Figure B.4:** ESB metric and test training loss for the 13x13 reservoir.

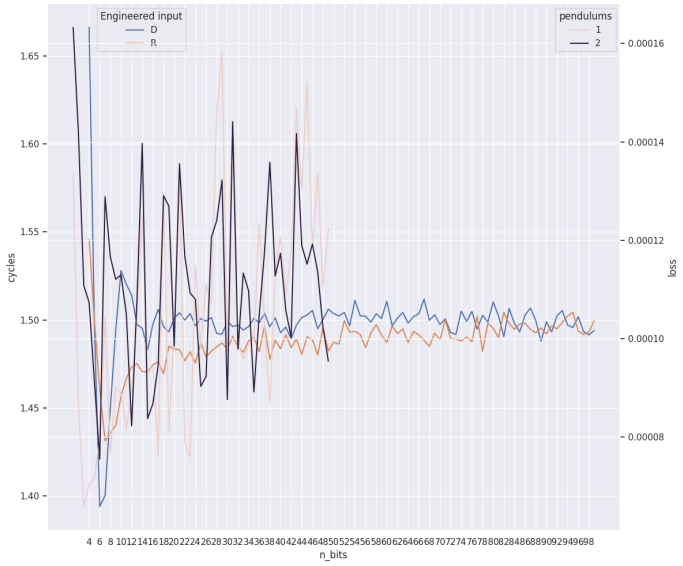



**Figure B.5:** ESB metric and test training loss for the 14x14 reservoir.

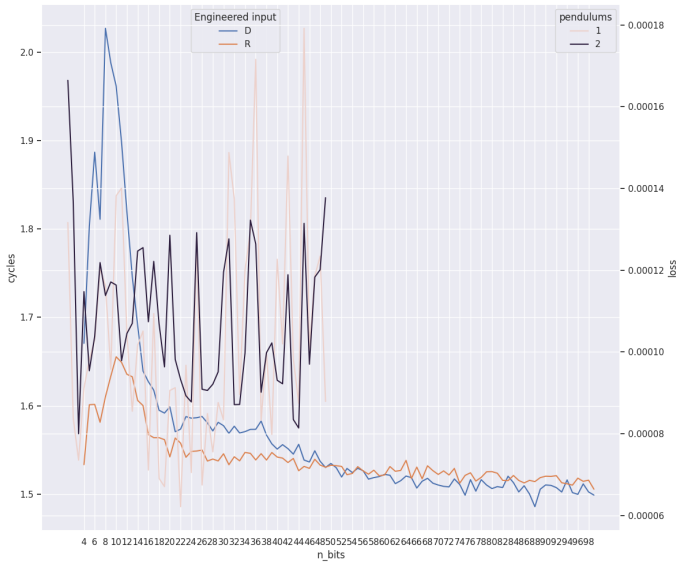



**Figure B.6:** ESB metric and test training loss for the 15x15 reservoir.

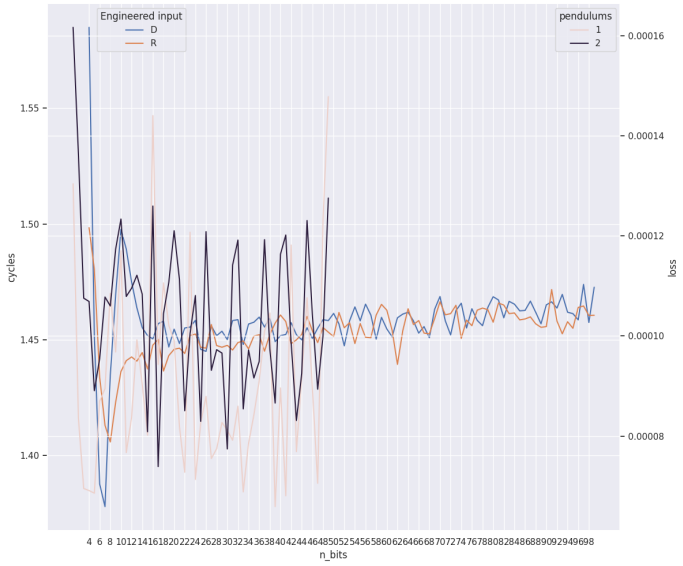



**Figure B.7:** ESB metric and test training loss for the 16x16 reservoir.

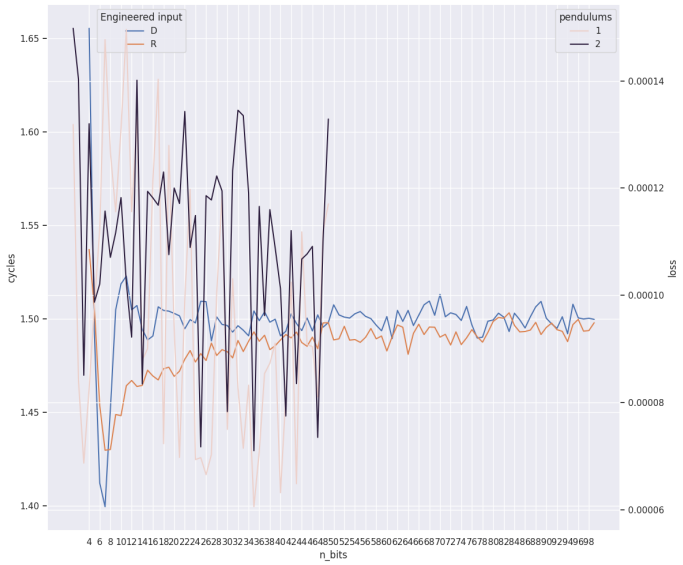



**Figure B.8:** ESB metric and test training loss for the 17x17 reservoir.

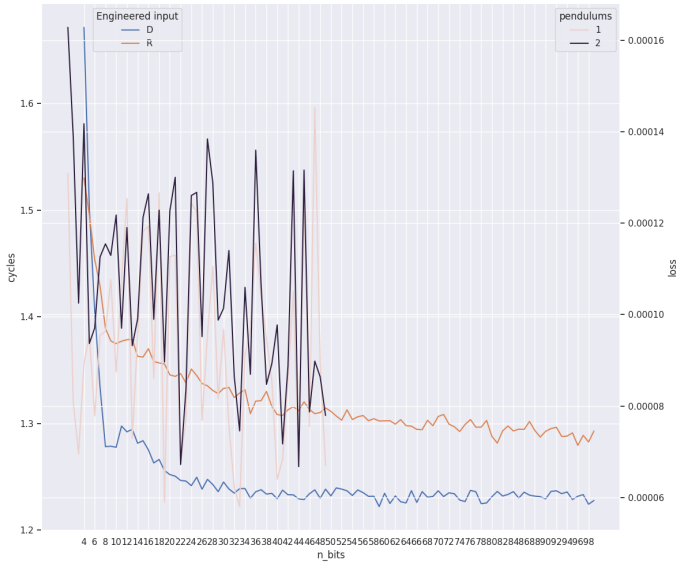



**Figure B.9:** ESB metric and test training loss for the 18x18 reservoir.

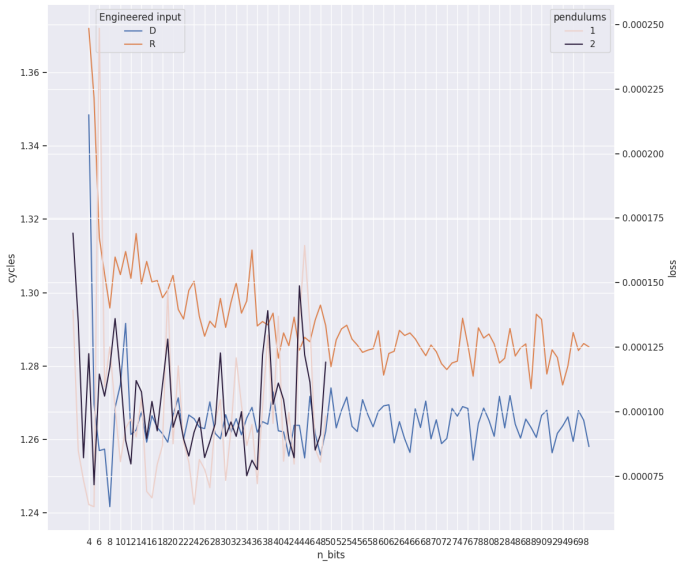



**Figure B.10:** ESB metric and test training loss for the 19x19 reservoir.

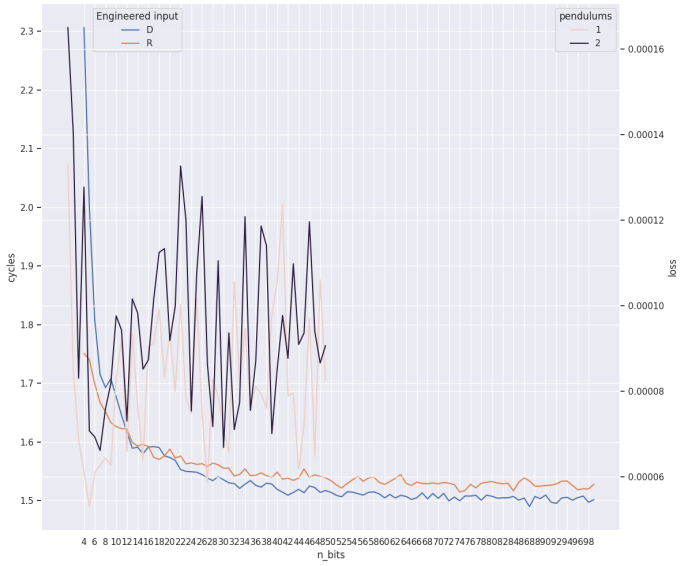



**Figure B.11:** ESB metric and test training loss for the 20x20 reservoir.

# Appendix C

# Master's project

This is the master's project which introduced foundations for the thesis like the concept of the ESB. The work has been cited as [1] throughout the report, in an effort to avoid self-plagiarism.

# Echo state metrics in Reservoir Computing

Christopher Vibe
Department of Computer and Information Science
Norwegian University of Science and Technology
Sorgenfriveien 30A, 7031 Trondheim, Norway
vibechris@gmail.com

***Abstract*—Reservoir computing is a lucrative field with great potential in comparison to the the stagnation seen in more traditional computing as exemplified by the slowing of Moore's law due to the so called power-wall. It is also a field enabling the study of intangible phenomena in natural systems, such as emergence. Although the field is young it is already clear that the approach has several advantages, such as providing useful computation with relatively low energy. The key properties for a suitable reservoir are understood on a high-level, but there is a need for metrics to assess the potential of a given reservoir beyond common metrics such as generality, and kernel quality. This paper focuses on metrics for the echo-state property, which dictates the amount of memory available for computation. Reservoir computing can be realized with an infinite number parts or media, but for ease of analysis we employ a realistic simulated reservoir of a dynamic nano-magnet array. Presented is a two-stage method for tuning a reservoir to increase its computing potential, followed by proposed metrics for measuring the echo-state property. This is the foundation for a thesis, which will be written in the next spring term, 2022.**

## I. Introduction

THE Reservoir Computing (RC) framework presents a method for computation inspired by natural systems with emergent properties. The paradigm is a sub-field of recursive neural networks (RNN's), and can be observed in many forms from analog computers used to solve differential equations [3] to learning melodies in echo state networks (ESN) [9]. RC has been re-discovered in various fields, but has been unified to RC in the early 2000's [13]. The RC field is closer to how biological systems perform computation through the emergent properties of their environment, as opposed to the modular formula for computation in classical computers, such as the von Neuman architecture [2].

The RC reservoir is naturally exemplified in the material computing sub-field, which focuses on leveraging the natural properties of materials for the use of computation. A classical example of RC with a material reservoir, is a literal bucket of water that has been perturbed such that water waves form on its surface [6]. The input in this system is the encoded perturbations on the surface of the water and is hence mapped from a low dimensional to high dimensional representation. The reservoirs higher dimensional representation is then normally passed through a machine learning readout layer, to produce more interpretative results. RNN's are hard to train due to their feedback properties, so back-propagation based training is limited to this layer. The idea is that the bulk of computation is from the mapping to the new representation but this is merely a natural process for the reservoir; free computational power. As a natural extension of evolution RC can provide solutions to problems with properties like low energy, parallelism, or fault tolerance, intrinsic to the approach [5]. Figure 1 summarises the classical RC model described above as a three layer process.

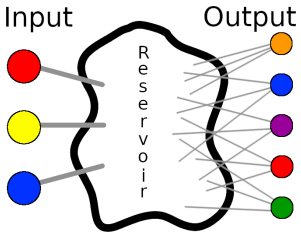



Fig. 1: Classical RC model: Input layer (left), reservoir (middle), and output layer (right). Note that the input has been mapped to a higher dimensional representation, artistically shown by the mixing of prime colours resulting in a rich palette in the output layer. The reservoir is flexible; in this case the inputs are spatially multiplexed with three input nodes, but it can be also be temporally multiplexed to a sequence of vectors. The output nodes are called the readout layer, and typically consists of a single linear machine learning layer.

As a close cousin to RC, extreme machine learning (EML) [8] inspires an intuitive example of a reservoir. Extreme machine learning uses random weights that are largely frozen to constrain costly back-propagation except in the final layers. In EML the system acts as a feed forward network, which is not characteristic of RC. However, one can introduce feedback connections to the system, and the setup becomes suitable as a reservoir in RC. The EML setup can then be thought of as a million small circuits wired up randomly to each other, without worrying about short-circuiting the system. Somewhere in the system it is highly probable to find useful computation, and all that remains is to synthesize the useful parts.

In order for a reservoir to be suitable for computation there are a couple of properties that have been identified as desirable. The reservoir must have a large number of components, with non-linear, local interactions, resulting in a highly dynamic,

Manuscript received December 15, 2021.

but not random, response when perturbed. Informally, the system must also forget inputs over time. This last idea was introduced as the echo state property (ESP) in the ESN [9].

Kernel quality and generality are existing tangible metrics useful for evaluating a reservoir [11]. Kernel quality measures how well the reservoir can separate temporal input by measuring the rank of a *nxm* matrix. The *n* represents the dimension of the output layer nodes after *m* unique input sequences. Generality measures the reservoirs ability to react similarly to similar input. The measurement process is identical to kernel quality, with the difference being *m* similar inputs. A reservoir should then have a large kernel quality with low generality for non-trivial computation. The aforementioned properties needed for computation are well represented by these two metrics, but the idea of fading memory, or the ESP, is not directly captured.

The intention of this thesis is to provide metrics for the echo-state property in a reservoir, being the retention of information in a reservoir used for RC. We will refer to this informational retention as the echo-state buffer (ESB), or buffer for short, defining the term in this context as the amount of information, in units of bits, transiently stored in the reservoir. In order to make a fair measurement the reservoir in the reservoir is first tuned to increase its computing potential. The need for tuning is due to energy topologies which may be mutually exclusive in the reservoir, and is further explained in section II.

As a metaphor for the ESB, if one sings in a room, the degree to which sounds from different temporal origins can be mixed will be limited by the length of the rooms echo. The ability to create such a rich representation of input is the motivation behind measuring the buffer size of a reservoir, and is closely linked to space complexity in computational theory. The echo-state property is therefore an analogy to conventional computers with different working memory. Hence, measuring the buffer size is fundamentally important to successfully uncovering the potential of a given RC system.

## II. Background

Spin ice (SI) have attracted considerable attention in physics due phenomena like: emergence, quantum tunnelling, magnetic mono-poles, and more [1]. In fact in 2021, the nobel prize in physics was awarded to Giorgio Parisi, who utilized a close cousin in the spin ice family, spin glass; a testament to the interest of the scientific community in meta-stable materials.

Commonly, it is not known that all materials have particles with magnetic poles, as only some of them display their magnetic properties on a macro scale. This is due to the internal orientation of their magnetic constituents or spins. The spins can either align, as is the case for a fridge magnet, or cancel each other out like in a potato. SI are materials with semi-stable magnetic particles whose spins exhibit geometrical frustration. In other words, physical circumstances prevent the magnetic particle orientations from configuring in the lowest energy level possible. Physically this means that the material may exhibit many steady states with a number of non-trivial "frustrated" spin configurations; a north-north or south-south pole clash for a number of particles in the substrate. See figure 2 for a picture of a SI based on nanomagnets.

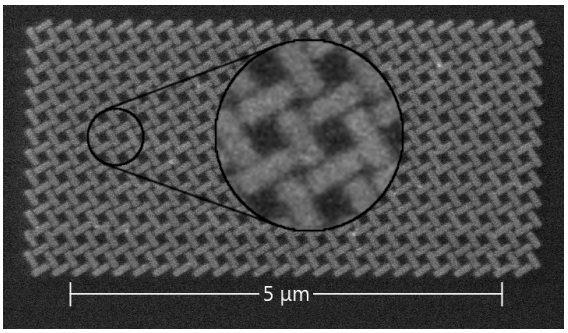



Fig. 2: A physical reservoir: SI as a magnet array in a diamond pinwheel formation. Courtesy of the SOCRATES project at the Norwegian University of Science and Technology (NTNU). In the zoomed circle one can see individual white rectangles, each a magnet at the nanoscale.

This works reservoir is a simulated 2D nanomagnet array, an artificial spin ice (ASI) model [10]. It has been shown, that When tuned correctly, this practical reservoir has properties needed for RC [11]. The employed reservoir stores its information using concepts from mechanical computing [14] with magnet spins to encode 1's and 0's. If one magnet in the array is perturbed, all its neighbors will be perturbed, causing a cascading dynamic effect until the system reaches a steady state. See figure 3 for a visual representation of an ASI reservoir.

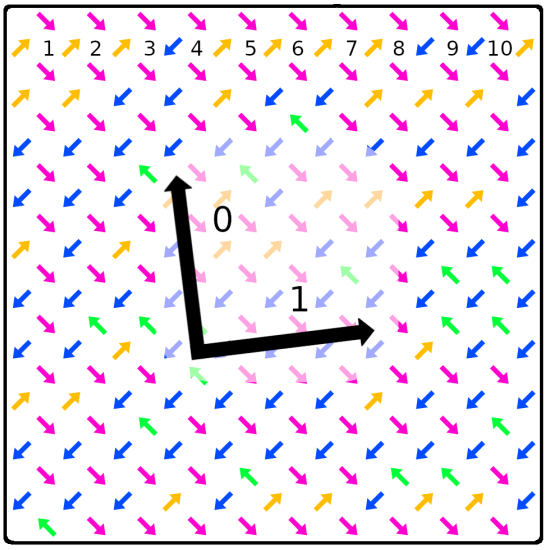



Fig. 3: A virtual reservoir: ASI with a 10x10 diamond pinwheel configuration. Note that the small arrows represent magnets and point in the direction of their magnets north-pole field or spin. This spin is also colourized based on the North-East-South-West orientation. The small coloured arrows reveal regions of frustration, as seen by the non-trivial colouring. The large black arrows represent the global external magnetic field used to perturb the reservoir.

The impression of a given input or perturbation of a reservoir in RC can be interpreted as traversing a 3D search space. The search space in the ASI has two dimensions, one for each input, and a third dimension for frustration energy. The origin will be defined as the polarized state, which is a ferromagnetic ordering. This same ordering is then also the net zero energy state, while other more frustrated states have a higher net-energy. This paper uses directed graphs to enable the traversal of such landscapes. The 2D plan of this topology is then representative of all possible inputs and are exemplified by figure 4 and 12.

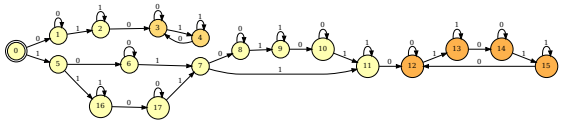


Fig. 4: An ASI reservoir with a 11x11 diamond pinwheel configuration as a directed graph. All possible spin states are nodes, with inputs as transitional edges. The strongly regions are coloured by size.

In a ”good” reservoir, the energy topology is characterized by many attractor states, or regions in the search space. In the ASI model, energy is required from an external perturbation to traverse this topology. Input is then an external dependency which may cause the ASI to confine to a region in the search space. Thus, the effective size of the reservoir may vary greatly depending not only on the encoding, but unfortunately also on the input sequence, as seen in figure 5.

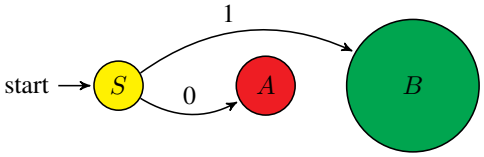



Fig. 5: Input energy bias: the first input bits may confine the effective reservoir size by traversing to one of two regions; A or B. Note how this phenomena relates to figures 4 and 12.

Although a reservoir should be tuned prior to measurement of an ESB this paper proposes a more robust two-stage approach where the ASI is “warmed up” through input to guide the effective search space to a more representative energy region, and then measured. The result is a measurement with reduced input energy bias as the input is more decoupled from the reservoir’s computing potential. To address this issue, the paper covers two approaches to reduce this input energy bias. The first is designing a set of engineered inputs defined in section III-A, and the second is a A* search algorithm with frustration as a heuristic in section IV-B.

The engineered inputs are inspired by Kolmogorov complexity, to avoid compressed representations of the input [7]. Kolmogorov complexity is not a tangible metric, but an abstract concept of how complicated a program needs to be in order to produce a series of bits. Take a set of nodes at a given attractor saturation state as an example, and let each attractor be an edge in a graph and each node a memory cell for storing a single bit. In this scenario, one could represent infinitely long input patterns with just a handful of states. However, if the Kolmogorov complexity is high, the number of nodes in the graph should be forced to organize in sparely connected linear edge networks. See figure 6 for an illustration of how a poorly chosen input can easily be represented by a small number of states. In essence the engineered inputs attempt to increase the Kolmogorov complexity, so that the reservoir transitions involve as many states as possible. By having this property the traversal should also mitigate getting stuck in certain regions of the search space.

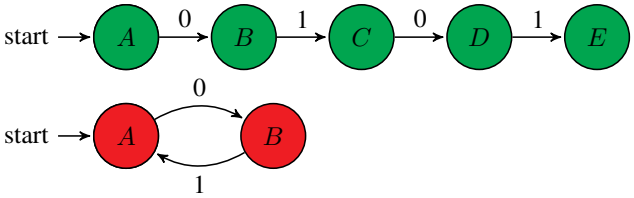



Fig. 6: Two graph traversals with input 0101, leading to highly variable state discoveries.

Once an ideal reservoir region has been identified, it is time for the actual measurement. The ESB measurement process can be summarized by applying persistent input patterns, until the system spin states cyclically transition between a set of attractors. This condition will be called cyclical attractor saturation or saturation for short. The measured ESB will then be the number of transitions between two such saturations. An example of a persistent input pattern is “01” which becomes “010101...”. In Jensen’s work exploring RC in ASI [4] it is demonstrated that ASI can, if tuned correctly, represent complex finite state machines, where each set of spin state orientations define a unique state. This result means that even if a input pattern where to be persistently repeated, it does not necessarily lead to an easily predictable state change. The spin configuration might converge to a point attractor end state, or cycle between many attractors. The perturbation mechanism is further detailed in section III-B and IV-C and the measurement is artistically illustrated in figure 7.

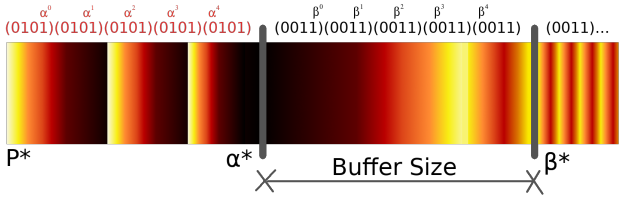



Fig. 7: Measuring the ESB by difference of saturation.

In the context of a 3D search space with a frustration energy topology it is natural that some regions are more interconnected than others. Furthermore, given that input translates to a limited energy budget for traversal, it is predicted that traversal may get irreversibly constrained to certain regions. This idea will be followed up in sections IV-A and IV-B.

## III. Method

The method below is void of numerical values for clarity, see appendix B if this is of interest.

### A. Generating the engineered input:

The method outlined below is a proposed way to engineer the inputs sequences so that they are ideal for probing the buffer. We create a number of input sets categorized by both length and properties. The properties are: bracelet, mirror, internal repetition, and density:

- *Bracelet, B:* Having the property of a mathematical bracelet, is that no two strings from our input set can be equivalent if rotated n times f.ex: $R(1100) = R(1001)$. Since input patterns are cyclically applied, a rotation may cause a pattern to be similar to another pattern in the input set.
- *Mirror, M:* Patterns should not be mirrors of each other. Combined with other properties this may lead to unintended effects, f.ex with the bracelet property: $B(001101) \neq B(001011)$ but combined with mirroring: $M(001101) = 101100 \implies B(101100) = B(001011)$.
- *Internal repetition, I:* This property is to avoid having similar inputs across the sets of various lengths. With internal repetition, the pattern may be indistinguishable to a new cycle. Using '*' as a repetition symbol, f.ex: $(01)^* = (0101)^*$
- *Density, D:* We balance the number of 0's and 1's in a pattern. We argue this may reduce energy biases from the input encoding, producing a more normalized pattern selection. The concept is experimentally explored in section IV.

See appendix C for an example of chosen inputs for an eight bit pattern.

### B. Defining and perturbing the reservoir:

Efforts have been made to find optimal computing parameters in ASI [11]. The 1's and 0's are encoded in the same fashion as this previous work; diamond pinwheel geometry, time extrapolated with a triangle wave, with two orthogonal vectors to represent each binary state, vector magnitude H, and a rotation of $H_r$ degrees to break symmetry. The input is externally applied as a global magnetic field. See figure 3 for a visualization of the reservoir.

### C. Control and generality:

The configuration of the reservoir is referred to as the geometry. The experiment geometry was varied by size from n to N, where the numbers represents the dimension of a square magnet array. Note that instead of dimensioning the square by the number of magnets, it is expressed in units of diamond pinwheel instances. Conducting the experiment with several reservoir sizes was to investigate if the method could generalize over many types of topologies. See figure 3 for a visualization of the geometry.

### D. Surrogate models:

The experiments place the reservoirs in a graph theory context, with nodes as the unique spin states a reservoir can express, and edges as the input needed to transition between these states. Furthermore, in order to decrease the compute time, the reservoirs where fully mapped out in flatspin, and then converted to surrogate models as easy to traverse graphs. See figure 4 and 12. The parameters in section B were chosen with care to get a deterministic system, that allows for conversion from reservoir to static graph; a reservoir graph.

## IV. Experiments and Results

The overarching method proposed in this paper is a two-stage process of reservoir calibration followed by measurements for the calculation of metrics. The intention of experiment IV-A, and IV-B, are to calibrate a given reservoir before measuring the ESB in experiment IV-C.

Calibration success was evaluated by its ability to maximize the number of nodes and or edges, as they both are indicators of a reservoirs upper limit potential. The number of nodes is an upper limit for the reservoirs memory, while the edges are an indicator of connectivity. The number of nodes and edges are therefore indicative of ideal reservoirs, as mentioned in section I.

### A. Calibration with engineered inputs

From qualitative observation of various reservoir graphs, it was noted that there were many strongly connected graphs (SCG) appearing as sub-graphs in the reservoir graph. Furthermore, the largest SCG's in the reservoir graphs tended to be at the extremities or leaves. This observation was verified in all reservoirs graphs used for the experiments but is yet to be rigorously verified in a larger topology search. Based on this tendency, we examined these leaf SCG regions, characterised by cyclic attractor state changes, or saturation. The SGC's can be calculated from the reservoir graph with algorithms like Tarjan's algorithm [12]. From the SCG node sets the leaf SGC's were identified by checking if it was impossible to traverse out of a given SCG. It is important to note that the traversal of the reservoir graph tends to be irreversible over time; the directed edges of the graph tend to move away from the original starting node. Since the larger SGC's seem to congregate near the leaf, one can expect the reservoir to naturally stabilize after warming up, and confine traversal to a certain region. As the first step of the proposed process, calibration is meant to guide the reservoir to a superior SGC.

In this trial the engineered inputs, as defined in section III-A, were combined in all possible property combinations, with random strings representing the empty set combination. The experiment was comprised of traversing the reservoir graphs for each combination, and comparing the length of the path before reaching one of the leaf SGC node sets. The input length was chosen over a range to cover the various sizes of the reservoir graphs. In the case where the inputs were too short to reach the leaf SGC's, a new sample was drawn with identical properties, until any leaf SGC was reached. In the cases where the traversal required more unique samples that

there exists in a set, the samples were randomly re-calculated, and therefore indirectly re-used.

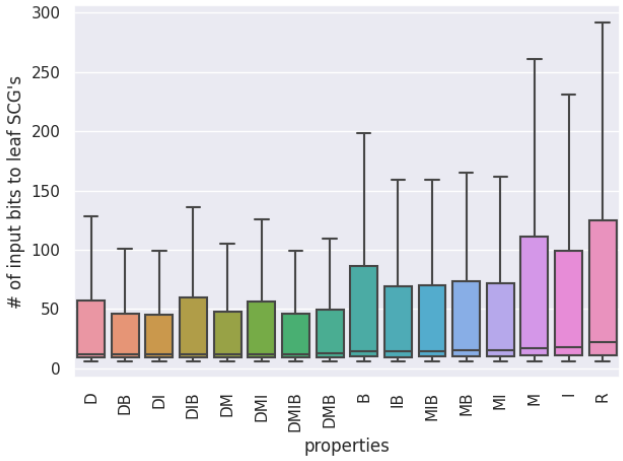



Fig. 8: Preliminary results: Traversing to leaf SCG's (the bottom of the reservoir graph) using 12 bit concatenated engineering inputs on a 15x15 ASI over 1000 trials. The boxplot are sorted by median and have outlier suppression.

Over the various topology sizes it was evident that the density (D) property in the engineered inputs tends to reach the leaf SGC's quickly compared to random inputs. From figure 8 this is clearly illustrated by a significantly lower median and variance for all property combinations with the density property. This tendency was invariant to the reservoir size each tested with over 1000 trials, and is therefore both statistically significant and generalized. Within the groups that have the density property, there is not a statistically significant difference across the topologies, further indicating that density is what contributes to their performance, and not their other properties. Note that all combinations outperform random inputs, regardless of density, with lower variance and median values. This supports the original idea of inputs with greater Kolmogorov complexity relative to random inputs with regard to minimizing the pattern size needed for calibration.

The quality of the leaf SCG's was measured in the number of nodes or edges. From figure 9 there seems to be an advantage to using the MI and DMIB combination of properties to find larger SCG's. This pattern was not clearly established across the various topologies. The number of edges or nodes in the SCG's produces similar plots. The rest of the combinations seem similar to the random category, more or less ending up in either SCG evenly.

The length of the concatenated components can be considered large or small relative to the average depth of the reservoir graph. Note that their length in figure 8 and 9 was 12 bits. This was chosen based on a compromise between a larger sample size and preserving the properties of the engineered inputs. If the bit size was small, then there would only exists a few instances per set, leading to indirect re-use of a set to reach a leaf SGC. With small bits the differences between the engineered combinations and random tended to increase, indicating that there is a scaling issue. If the bit size was too

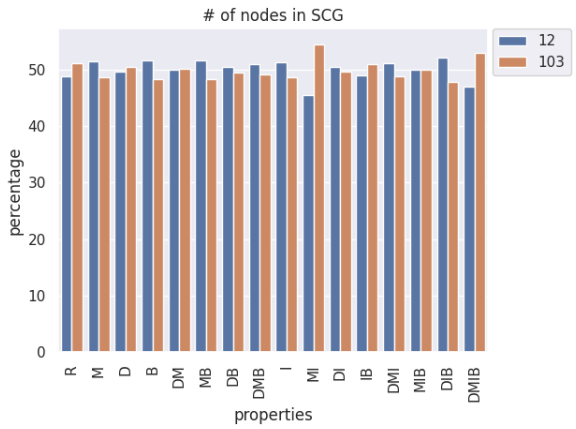



Fig. 9: Preliminary results: Number of nodes at SCG after traversal, using 12 bit concatenated engineering inputs on a 15x15 ASI over 1000 trials. The graph shows the probability split of which SCG the traversal gets confined to.

big, then the all engineered inputs performed roughly equally to the random inputs, with the exception of the combinations that had the density property, which remained invariant to this change.

### *B. Calibration with A* search:*

The underlying hypothesis for this experiment is that highly frustrated reservoirs have more possible spin states. If this is true one could calibrate the reservoir with a less computationally expensive calibration relative to experiment IV-A. The idea is to use the A* algorithm with frustration as the guiding heuristic. This is composed of first investigating if larger SGC's have more frustration, followed by an evaluating of the algorithm if the hypothesis is true. The heuristic will be calculated as the sum of norms of the di-polar field experienced by each magnet on the array. The di-polar field is defined in [10] and represents the net force that neighboring magnets exert on a magnet.

The first part of this experiment will be to produce a graph similar to figures 4 and 12, but coloured by frustration levels instead of SCG size.

### *C. Measuring the ESB with engineered inputs:*

To measure the ESB, we first had to establish two attractor saturation states to take the difference from. The first attractor saturation state will be referred to as state $\alpha^*$ and the second attractor saturation state will be referred to as state $\beta^*$.

We go on to define the termination condition for the persistent input, as when the spin state, after a full pattern has been applied, is part of a set of previously seen starting states. This approach is valid because the model is deterministic, so if the model starts from a state A, it will always end in a state B after the same input pattern. Thus, the input termination condition guarantees saturation; the model will never discover additional states or break the transition patterns between thee cyclic attractors. The average number of transitions between

two saturations will thus be a measure for the ESB, and is artistically shown in figure 7. In these preliminary stage, the beta pattern will be (01)*, just as in figure 7, so as to ensure that the SCG measured will always be the same. This also decouples this experiment from the other calibration experiments while the approach is still under development.

It is predicted that the reservoir will respond differently to various length inputs, implying that a given reservoir system also has an optimal input size if the intention is to maximize useful computation. The engineered inputs will therefore be a sweep of lengths so that this prediction can be verified. Figure 10 illustrates how the number of input bits may influence the ESB.

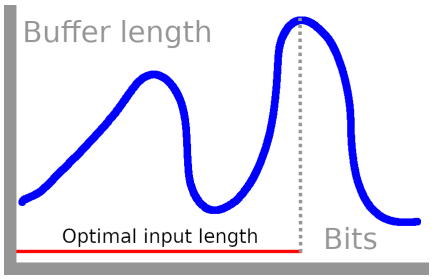



Fig. 10: Placeholder graph: Finding the input bit size that produces the largest ESB

Furthermore, it is predicted that for each set of probing bit lengths the ESB measurement will exhibit properties similar to resonant frequencies in standing waves. Like a musical pipe resonating with wind currents with certain velocities, or harmonics, the bit length is predicted to resonate at certain base frequencies, with higher frequencies being a multiple of this base. Leveraging this behaviour, the buffer length that best represents the system per bit length can then be identified based on this standing wave behaviour. Specifically through resonance at predictable intervals, seen as dips in the number of cycles needed where the probe length of the input bits are a multiple of the length of the ESB. These intervals and may be mathematically described by modulo(p, b), where p is the bit length of the probe and b is the bit length of the ESB. This is artistically shown in figure 10 as two spikes, where the first is half the ESB length, and the second spike is the ESB length. Other fractional patterns are not accurately shown in the figure, such as when the probe is one third of the ESB etc.

Part of the experiment is also to see if the results are invariant with regard to engineered input properties. It is predicted that the ESB will be more optimistic and representative than random inputs. This is because the persistently applied engineered inputs are predicted to take graph traversals visiting more nodes than random input will. The same logic applies to edges. This is based on the idea that the Kolmogorov complexity will be higher, and deter the reservoir from re-using edges and nodes. The point is that larger network coverage should lead to a smaller probability of saturation, and hence, a larger ESB. This idea is further discussed in section II and illustrated in figure 6.

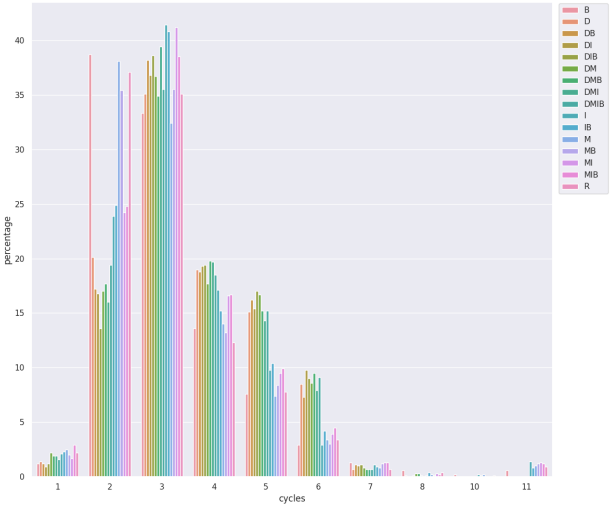



Fig. 11: Preliminary results: Probing the reservoir graph with 12 bit engineered inputs on a 15x15 ASI over 1000 trials. The cycles are the number of times the pattern was repeated, while the percentage is the probability of a pattern meeting the termination condition at a given cycle (defined in section IV-C)

The results in figure 11 are preliminary, but show how there are higher intensity around certain cycle values. These intensities will be placed in the context of resonant frequencies in standing waves to identify an ESB value for each bit length. This is visualized in figure 10.

## V. Discussion

Preliminary results from calibration experiment IV-A indicate that a balanced density property leads to finding SCG's quickly. On one hand, considering the high number of samples and agreement in all the topologies, the result seems to generalize. On the other hand, the results stem from one ASI geometry, with the exception of the size variable. For instance, in figure 8, it may be that this effect from the density property is merely a consequence of the square geometry.

There is no clear indication as to the quality of the SCG across the topologies in terms of nodes and edges, but for the 15x15 size setup the MI and DMIB combinations seemed to help calibrate the reservoir to a region with more nodes and edges. Generality is again a concern, take figure 9, the MI and DMIB combinations may excel due to some other topology-specific property as opposed to a generalized result. In the case that this experiment shows that engineered inputs are not an ideal tool for calibration, it is still useful, as it justifies the use of random inputs in the calibration phase or other techniques like the one explored in experiment IV-B.

Note that the density property in the engineered inputs is not invariant to scaling like the other properties and random inputs. For example a long string may have many repeated zeros or ones in a region, compared to a short string. By forcing a fixed density, one is also setting the maximum distance between each zero or one in a string. With random inputs however,

two samples may not necessarily constrain this maximum distance. The length of the bit strings may therefore have properties that are unaccounted for. Furthermore, the use of the engineered properties are not rigorously justified beyond attempting to emulate Kolmogorov complexity, but should suffice for a preliminary investigation.

Calibration experiment IV-B, has been defined, but has not yet been implemented. It will be interesting to see if the method is suitable for finding SCG's with high node or edge counts.

In experiment IV-C, the ESB measurements are predicted to reveal an optimal input length to perturb the reservoir. This is only possible however if an exact method for identifying an ESB per input length is defined. From fig 11 the chances of getting 3 cycles is the highest. The next step is to see if there are resonant frequencies for a sweep of bits, to see if a standing wave model is valid for predictions, and ultimately identifying an ESB. This is artistically illustrated as a placeholder in graph 10.

Note that method outlined in experiment IV-C for capturing the ESP might not be the best way considering that the ESB is a probabilistic distribution and outlines a metric with dependency on the inputs. For example, the ESB is a function of the engineered input, and assumes inputs have a certain discrete length. Another route would be to focus on the SCG's with graph focused metrics such as connectivity, or von Neumann entropy. Such an approach could decouple the metrics from inputs, but seems like a less direct measurement. Going forward, both of these approaches will be employed as part of the investigation.

## VI. Future Work

One of this thesis hypothesis are that the the largest SCG's are at the leaf nodes of the reservoir graphs, based on empirical observations. A more extensive investigation is required to see if this is a generalized trend, or what conditions are needed for this to be true.

Once a robust method has been developed for measuring the ESP, it will also be necessary to experimentally verify if the metrics can be used to solve real problems with varying spacial complexity demands. This includes checking if the ESB metric, or other metric for measuring the ESP, can predict success.

Furthermore, it is equally important to understand the underlying mechanisms that determines the size of a SCG, both in the spirit of science and being able to manipulate the buffer to fit the needs of an application.

The most obvious way to tune a reservoir in an application focused context, is through its geometry. Note that there is an indirect, but practical, outcome of the calibration approach; The work may be used to guide research focusing on the tuning of reservoirs with input. Once the foundations for calibration with input have been better understood, they may lead to novel methods for reservoir optimization.

Most importantly, for the credibility of future results for this thesis, the proposed two step method for measuring the ESP will need to be applied to a survey of geometries in ASI. In the long term it would also be natural to expand these test to reservoirs in other mediums.

## VII. Conclusion

The proposed two stage method involves calibration, followed by the measurement of the ESB as a metric for the ESP. Two stages are proposed, because through calibration, the metrics can capture the ESP in a more objective way, considering that the reservoir graph can confine traversal to regions of the graph with greatly varying properties.

Preliminary results reveal that balanced density inputs may lead to rapid calibration, but that engineered properties for calibration for achieving large node or edge counts in the SCG's is still unclear. The frustration based calibration approach with A* has not been implemented. The actual measurement of the ESB after calibration has also not been completed, but early experiments have been conducted, which will be used to verify if the method has potential.

A strong theoretical foundation has been established, as well as the development of tools needed for exploring the proposed approach. Most importantly this includes the literature search grounding the theoretical discussion, well defined experiments, and an understanding of the ASI model with flatspin.

## References


[1] S. T. Bramwell and M. J. Harris, "The history of spin ice," *Journal of Physics: Condensed Matter*, vol. 32, no. 37, p. 374010, jun 2020. [Online]. Available: https://doi.org/10.1088/1361-648x/ab8423

[2] A. W. Burks, H. H. Goldstine, and J. von Neumann, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1982, pp. 399–413. [Online]. Available: https://doi.org/10.1007/978-3-642-61812-3_32

[3] V. Bush, "The differential analyzer. a new machine for solving differential equations," *Journal of The Franklin Institute-engineering and Applied Mathematics*, vol. 212, pp. 447–488, 1931.

[4] *Computation in artificial spin ice*, ser. ALIFE 2021: The 2021 Conference on Artificial Life, vol. ALIFE 2018: The 2018 Conference on Artificial Life, 07 2018. [Online]. Available: https://doi.org/10.1162/isal_a_00011

[5] M. Dale, J. Miller, and S. Stepney, *Reservoir Computing as a Model for In-Materio Computing*, 01 2017, vol. 22, pp. 533–571.

[6] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *Advances in Artificial Life*, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 588–597.

[7] A. Gammerman and V. Vovk, "Kolmogorov complexity: Sources, theory and applications." *Computer Journal*, vol. 42, no. 4, 1999.

[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006, neural Networks. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385

[9] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[10] J. H. Jensen, A. Strømberg, O. R. Lykkebø, A. Penty, M. Själander, E. Folven, and G. Tufte, "flatspin: A large-scale artificial spin ice simulator," 2020.

[11] J. H. Jensen and G. Tufte, "Reservoir computing in artificial spin ice," in *Reservoir Computing in Artificial Spin Ice*, 2020.

[12] R. TARJAN R, "Depth- first search and linear graph algorithms," Jan. 1971, pp. 114–121, iEEE Conference Record of 1971 12th Annual Symposium on Switching and Automata Theory; Conference date: 13-10-1971 Through 15-10-1971.

[13] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007, echo State Networks and Liquid State Machines. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360800700038X

[14] H. Yasuda, P. Buskohl, A. Gillman, T. Murphey, S. Stepney, R. Vaia, and J. Raney, "Mechanical computing," *Nature*, vol. 598, no. 7879, pp. 39–48, Oct. 2021, funding Information: Acknowledgements H.Y. and J.R.R. acknowledge support from Army Research Office award number W911NF-1710147, Air Force Office of Scientific Research award number FA9550-19-1-0285 and DARPA Young Faculty Award W911NF2010278. P.R.B., A.G. and R.A.V. acknowledge support from the Materials and Manufacturing Directorate and the Air Force Office of Scientific Research of the Air Force Research Laboratory. T.D.M. acknowledges support from NSF 1837515 and ARO MURI award W911NF-19-1-0233. S.S. acknowledges support from the SpInspired project, EPSRC grant number EP/R032823/1. Publisher Copyright: © 2021, Springer Nature Limited.

## APPENDIX A
## DEFINITIONS AND GLOSSARY

ASI - Artificial SI (simulated material)
A* - The heuristic search algorithm
Bracelet - A string of 0's and 1's connected in a circle
Cycle - A full impression of a pattern on the reservoir
SI - Spin Ice (material)
SCG - Strongly connected graph
EML - Extreme Machine Learning
ESB - Echo state buffer
ESP - Echo state property
ESN - Echo State Network
Pattern - A unique string of 0's and 1's
Saturation - Cyclic attractor state condition

## APPENDIX B
## EXPERIMENT PARAMETERS

The magnet geometry parameters are tuned to produce a rectangular stadium-shaped magnet, as used in [11]. Their geometrical meaning is defined in [10]. Note that the grid size is the number of geometry instances on the grid, not the number of magnets; f.ex. number of pinwheel instances. Consult flatspin's documentation for more parameters default to the PinwheelSpinIceDiamond class, which were used for all experiments, and not listed below. To ensure the system was deterministic for experiment control a random seed was set as well as a deterministic flipping mechanism for the magnets.

| Parameter | Value | Comment |
|---|---|---|
| $G_d$ | [10:15] | dimension of grid (geometry instances) |
| $G_m$ | 220 | magnet count on array |
| $H_i$ | 78mT | input field strength |
| $H_c$ | 200mT | coercive field strength |
| $H_r$ | 7 | anti-clockwise rotation of global field vectors |
| $M_l$ | 220nm | length of magnets |
| $M_w$ | 80nm | width of magnets |
| $M_h$ | 20nm | height of magnets |
| $M_\alpha$ | 1.02e-3 | spacing of magnets (dipolar coupling strength) |
| $b$ | 0.41 | magnet geometry parameter |
| $c$ | 1.0 | magnet geometry parameter |
| $\beta$ | 1.5 | magnet geometry parameter |
| $\gamma$ | 3.9 | magnet geometry parameter |
| $random\_seed$ | 0 | random seed for flatspin++ |
| $flip\_mode$ | max | flipping mode in flatspin |

## APPENDIX C
## INPUT GENERATION

Below is an example of generating inputs with the properties DMIB. In column $B$ are the unique bracelets for 8 bits. The balanced density patterns are in column $B_{50}$ and, in some cases, have been shortened due to internal repetition. Only the inputs of length 8 in column $B_{50}$ are valid for use as part of the 8 bit subset of the engineered inputs due to the I property. The M property has in this case been covered by the stricter B property.

| $B$ | $B_{50}$ |
|---|---|
| 11111111 | - |
| 01111111 | - |
| 00111111 | - |
| 01011111 | - |
| 01101111 | - |
| 01110111 | - |
| 00011111 | - |
| 00101111 | - |
| 00110111 | - |
| 01010111 | - |
| 01011011 | - |
| 00001111 | 00001111 |
| 00010111 | 00010111 |
| 00011011 | 00011011 |
| 00100111 | 00100111 |
| 00101011 | 00101011 |
| 00101101 | 00101101 |
| 00110011 | 0011 |
| 01010101 | 01 |
| 00000111 | - |
| 00001011 | - |
| 00010011 | - |
| 00010101 | - |
| 00100101 | - |
| 00000011 | - |
| 00000101 | - |
| 00001001 | - |
| 00010001 | - |
| 00000001 | - |
| 00000000 | - |

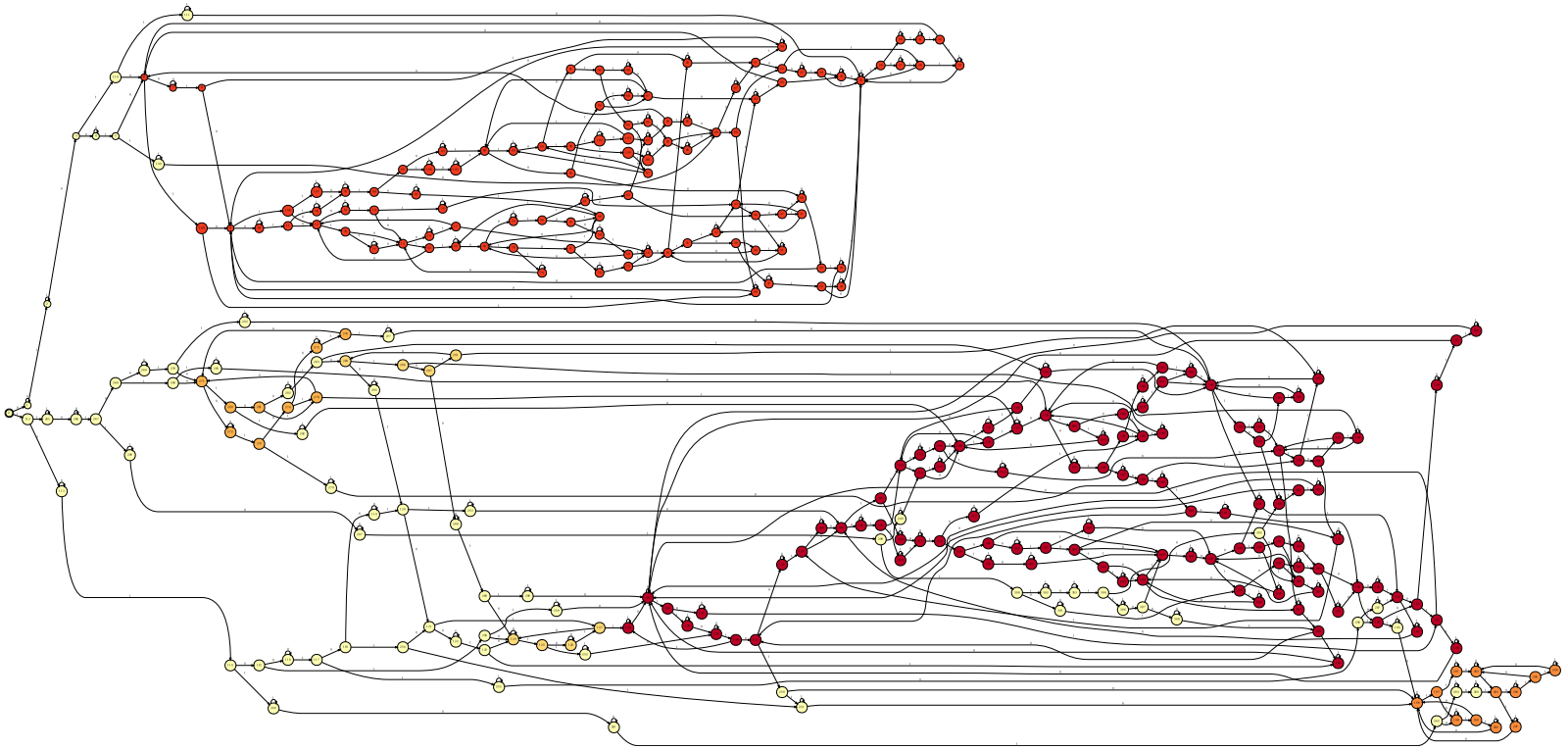


Fig. 12: An ASI reservoir with a 15x15 diamond pinwheel configuration as a directed graph. All possible spin states are nodes, with inputs as transitional edges. The strongly connected regions are coloured by size.

# Appendix D

# Master's thesis agreement

The master's thesis agreement is in English and Norwegian below.

# Masteravtale/hovedoppgaveavtale

*Sist oppdatert 11. november 2020*

| | |
|---|---|
| **Fakultet** | **Fakultet for informasjonsteknologi og elektroteknikk** |
| **Institutt** | **Institutt for datateknologi og informatikk** |
| **Studieprogram** | **MTDT** |
| **Emnekode** | **TDT4900** |

## Studenten

| | |
|---|---|
| **Etternavn, fornavn** | **Vibe, Christopher Michael** |
| **Fødselsdato** | **12.12.1993** |
| **E-postadresse ved NTNU** | **chrismvi@stud.ntnu.no** |

## Tilknyttede ressurser

| | |
|---|---|
| **Veileder** | **Gunnar Tufte** |
| **Eventuelle medveiledere** | |
| **Eventuelle medstudenter** | |

## Oppgaven

| | |
|---|---|
| **Oppstartsdato** | **01.02.2022** |
| **Leveringsfrist** | **27.06.2022** |
| **Oppgavens arbeidstittel** | **Echo state metrics in Reservoir Computing** |
| **Problembeskrivelse** | **Reservoir computing is a lucrative field with great potential in comparison to the the stagnation seen in more traditional computing as exemplified by the slowing of Moore's law due to the so called power-wall. It is also a field enabling the study of intangible phenomena in natural systems, such as emergence. Although the field is young, it is already clear that the approach has several advantages, such as providing useful computation with relatively low energy. The key properties for a suitable reservoir are understood on a high-level, but there is a need for metrics to assess the potential of a given reservoir beyond common metrics such as generality, and kernel quality. This paper focuses on metrics for the echo-state property, which dictates the amount of memory available for computation. Reservoir computing can be realized with an infinite number parts or media, but for ease of analysis we will employ a realistic simulated reservoir of a dynamic nano-magnet array. The research will begin with a theoretical approach with the aim to develop a two-stage method for tuning a reservoir to increase its computing potential, followed by proposed metrics for measuring the echo-state property. Finally, the echo-state property metrics will be used to predict the reservoir's success on a practical control system problem, f.ex. balancing n-inverted pendulum on a cart.** |

## Risikovurdering og datahåndtering

| | |
|---|---|
| **Skal det gjennomføres risikovurdering?**<br><br>**Dersom «ja», har det blitt gjennomført?** | **Nei**<br><br>**Nei** |
| **Skal det søkes om godkjenninger? (REK*, NSD**)** | **Nei** |
| **Skal det skrives en konfidensialitetsavtale i forbindelse med oppgaven?**<br><br>**Hvis «ja», har det blitt gjort?** | **Nei**<br><br>**Nei** |

* Regionale komiteer for medisinsk og helsefaglig forskningsetikk (https://rekportalen.no)

** Norsk senter for forskningsdata (https://nsd.no/)

## Eventuelle emner som skal inngå i mastergraden

**Emner fra 4 og 5 året: 5 ---------- T2: TDT4900 Datateknologi, masteroppgave T1: TDT4501 Datateknologi, fordypningsprosjekt TDT4506 Datateknologi, fordypningsemne TDT04 Avanserte Bioinspirerte Metoder TDT05 Modern Machine Learning in Practice Construction Engineering and Management (innpasning K-emne UBC*) Engineering Economic Analysis (innpasning K-emne UBC*) 4 ---------- T1: TDT4295 Datamaskinprosjekt TDT4255 Datamaskinkonstruksjon University writing (innpasning K-emne UBC*) Technical Communication (innpasning K-emne UBC*) T2: TDT4125 Algoritmekonstruksjon TDT4265 Datasyn og dyp læring TIØ4852 EiT TMA4105 Matematikk 2 *UBC: University of British Columbia**

# Retningslinjer - rettigheter og plikter

## Formål

Avtale om veiledning av masteroppgaven/hovedoppgaven er en samarbeidsavtale mellom student, veileder og institutt. Avtalen regulerer veiledningsforholdet, omfang, art og ansvarsfordeling.

Studieprogrammet og arbeidet med oppgaven er regulert av Universitets- og høgskoleloven, NTNUs studieforskrift og gjeldende studieplan. Informasjon om emnet, som oppgaven inngår i, finner du i emnebeskrivelsen.

## Veiledning

### Studenten har ansvar for å

- Avtale veiledningstimer med veileder innenfor rammene master-/hovedoppgaveavtalen gir.
- Utarbeide framdriftsplan for arbeidet i samråd med veileder, inkludert veiledningsplan.
- Holde oversikt over antall brukte veiledningstimer sammen med veileder.
- Gi veileder nødvendig skriftlig materiale i rimelig tid før veiledning.
- Holde instituttet og veileder orientert om eventuelle forsinkelser.
- Inkludere eventuell(e) medstudent(er) i avtalen.

### Veileder har ansvar for å

- Avklare forventninger om veiledningsforholdet.
- Sørge for at det søkes om eventuelle nødvendige godkjenninger (etikk, personvernhensyn).
- Gi råd om formulering og avgrensning av tema og problemstilling, slik at arbeidet er gjennomførbart innenfor normert eller avtalt studietid.
- Drøfte og vurdere hypoteser og metoder.
- Gi råd vedrørende faglitteratur, kildemateriale, datagrunnlag, dokumentasjon og eventuelt ressursbehov.
- Drøfte framstillingsform (eksempelvis disposisjon og språklig form).
- Drøfte resultater og tolkninger.
- Holde seg orientert om progresjonen i studentens arbeid i henhold til avtalt tids- og arbeidsplan, og følge opp studenten ved behov.
- Sammen med studenten holde oversikt over antall brukte veiledningstimer.

### Instituttet har ansvar for å

- Sørge for at avtalen blir inngått.
- Finne og oppnevne veileder(e).
- Inngå avtale med annet institutt/ fakultet/institusjon dersom det er oppnevnt ekstern medveileder.
- I samarbeid med veileder holde oversikt over studentens framdrift, antall brukte veiledningstimer, og følge opp dersom studenten er forsinket i henhold til avtalen.
- Oppnevne ny veileder og sørge for inngåelse av ny avtale dersom:
    - Veileder blir fraværende på grunn av eksempelvis forskningstermin, sykdom, eller reiser.
    - Student eller veileder ber om å få avslutte avtalen fordi en av partene ikke følger den.
    - Andre forhold gjør at partene finner det hensiktsmessig med ny veileder.
- Gi studenten beskjed når veiledningsforholdet opphører.
- Informere veileder(e) om ansvaret for å ivareta forskningsetiske forhold, personvernhensyn og veiledningsetiske forhold.
- Ønsker student, eller veileder, å bli løst fra avtalen må det søkes til instituttet. Instituttet må i et slikt tilfelle oppnevne ny veileder.

*Avtaleskjemaet skal godkjennes når retningslinjene er gjennomgått.*

## Godkjent av

Christopher Michael Vibe
**Student**

01.02.2022
*Digitalt godkjent*

Gunnar Tufte
**Veileder**

08.02.2022
*Digitalt godkjent*

Berit Hellan
**Institutt**

10.02.2022
*Digitalt godkjent*

# Master`s Agreement / Main Thesis Agreement

| | |
|---|---|
| **Faculty** | **Faculty of Information Technology and Electrical Engineering** |
| **Institute** | **Department of Computer Science** |
| **Programme Code** | **MTDT** |
| **Course Code** | **TDT4900** |

## Personal Information

| | |
|---|---|
| **Surname, First Name** | **Vibe, Christopher Michael** |
| **Date of Birth** | **12.12.1993** |
| **Email** | **chrismvi@stud.ntnu.no** |

## Supervision and Co-authors

| | |
|---|---|
| **Supervisor** | **Gunnar Tufte** |
| **Co-supervisors (if applicable)** | |
| **Co-authors (if applicable)** | |

## The Master`s thesis

| | |
|---|---|
| **Starting Date** | **01.02.2022** |
| **Submission Deadline** | **27.06.2022** |
| **Thesis Working Title** | **Echo state metrics in Reservoir Computing** |
| **Problem Description** | **Reservoir computing is a lucrative field with great potential in comparison to the the stagnation seen in more traditional computing as exemplified by the slowing of Moore's law due to the so called power-wall. It is also a field enabling the study of intangible phenomena in natural systems, such as emergence. Although the field is young, it is already clear that the approach has several advantages, such as providing useful computation with relatively low energy. The key properties for a suitable reservoir are understood on a high-level, but there is a need for metrics to assess the potential of a given reservoir beyond common metrics such as generality, and kernel quality. This paper focuses on metrics for the echo-state property, which dictates the amount of memory available for computation. Reservoir computing can be realized with an infinite number parts or media, but for ease of analysis we will employ a realistic simulated reservoir of a dynamic nano-magnet array. The research will begin with a theoretical approach with the aim to develop a two-stage method for tuning a reservoir to increase its** |

| | |
|---|---|
| | **computing potential, followed by proposed metrics for measuring the echo-state property. Finally, the echo-state property metrics will be used to predict the reservoir's success on a practical control system problem, f.ex. balancing n-inverted pendulum on a cart.** |

## Risk Assessment and Data Management

| | |
|---|---|
| **Will you conduct a Risk Assessment?**<br><br>**If "Yes", Is the Risk Assessment Conducted?** | **No**<br><br>**No** |
| **Will you Apply for Data Management?**<br>**(REK*, NSD**)** | **No** |
| **Will You Write a Confidentiality Agreement?**<br><br>**If "Yes", Is the Confidentiality Agreement Conducted?** | **No**<br><br>**No** |

* REK -- https://rekportalen.no/

** Norwegian Centre for Research Data (https://nsd.no/nsd/english/index.html )

## Topics to be included in the Master`s Degree (if applicable)

**Emner fra 4 og 5 året: 5 ---------- T2: TDT4900 Datateknologi, masteroppgave T1: TDT4501 Datateknologi, fordypningsprosjekt TDT4506 Datateknologi, fordypningsemne TDT04 Avanserte Bioinspirerte Metoder TDT05 Modern Machine Learning in Practice Construction Engineering and Management (innpasning K-emne UBC*) Engineering Economic Analysis (innpasning K-emne UBC*) 4 ---------- T1: TDT4295 Datamaskinprosjekt TDT4255 Datamaskinkonstruksjon University writing (innpasning K-emne UBC*) Technical Communication (innpasning K-emne UBC*) T2: TDT4125 Algoritmekonstruksjon TDT4265 Datasyn og dyp læring TIØ4852 EiT TMA4105 Matematikk 2 *UBC: University of British Columbia**

# Guidelines – Rights and Obligations

## Purpose

The Master's Agreement/ Main Thesis Agreement is an agreement between the student, supervisor, and department. The agreement regulates supervision conditions, scope, nature, and responsibilities concerning the thesis.

The study programme and the thesis are regulated by the Universities and University Colleges Act, NTNU's study regulations, and the current curriculum for the study programme.

## Supervision

### The student is responsible for

- Arranging the supervision within the framework provided by the agreement.
- Preparing a plan of progress in cooperation with the supervisor, including a supervision schedule.
- Keeping track of the counselling hours.
- Providing the supervisor with the necessary written material in a timely manner before the supervision.
- Keeping the institute and supervisor informed of any delays.
- Adding fellow student(s) to the agreement, if the thesis has more than one author.

### The supervisor is responsible for

- Clarifying expectations and how the supervision should take place.
- Ensuring that any necessary approvals are acquired (REC, ethics, privacy).
- Advising on the demarcation of the topic and the thesis statement to ensure that the work is feasible within agreed upon time frame.
- Discussing and evaluating hypotheses and methods.
- Advising on literature, source material, data, documentation, and resource requirements.
- Discussing the layout of the thesis with the student (disposition, linguistic form, etcetera).
- Discussing the results and the interpretation of them.
- Staying informed about the work progress and assist the student if necessary.
- Together with the student, keeping track of supervision hours spent.

### The institute is responsible for

- Ensuring that the agreement is entered into.
- Find and appoint supervisor(s).
- Enter into an agreement with another department / faculty / institution if there is an external co-supervisor.
- In cooperation with the supervisor, keep an overview of the student's progress, the number of supervision hours. spent, and assist if the student is delayed by appointment.
- Appoint a new supervisor and arrange for a new agreement if:
  - The supervisor will be absent due to research term, illness, travel, etcetera.
  - The student or supervisor requests to terminate the agreement due to lack of adherence from either party.
  - Other circumstances where it is appropriate with a new supervisor.
- Notify the student when the agreement terminates.
- Inform supervisors about the responsibility for safeguarding ethical issues, privacy and guidance ethics
- Should the cooperation between student and supervisor become problematic, either party may apply to the department to be freed from the agreement. In such occurrence, the department must appoint a new supervisor

*This Master`s agreement must be signed when the guidelines have been reviewed.*

## Signatures

Christopher Michael Vibe
**Student**

01.02.2022
*Digitally approved*

Gunnar Tufte
**Supervisor**

08.02.2022
*Digitally approved*

Berit Hellan
**Department**

10.02.2022
*Digitally approved*

Christopher M. Vibe

Practical RC & ESP Metrics

NTNU
Norwegian University of
Science and Technology