Comparison and Benchmarking of Reservoir Computing using Cellular Automata and Random Boolean Networks as Substrates

Ruben Jahren

Thesis submitted for the degree of
Master in Applied Computer and Information Technology (ACIT)
- Phase II
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

OSLO METROPOLITAN UNIVERSITY

Spring 2022

Comparison and Benchmarking of Reservoir Computing using Cellular Automata and Random Boolean Networks as Substrates

Ruben Jahren

Comparison and Benchmarking of Reservoir Computing using Cellular Automata and Random Boolean Networks as Substrates

http://www.oslomet.no/

Printed: Oslo Metropolitan University

Abstract

Reservoir Computing is an emerging concept in artificial intelligence derived from Recurrent Neural Networks, which utilizes an untrained reservoir substrate to memorize and separate input in such a way that it may be linearly separated in a readout layer. Since its origins with RNN, it has shown great flexibility in what can be facilitated as a substrate.

In this thesis, Reservoir Computing models with Cellular Automata and Random Boolean Networks as substrates were implemented. The Lifelike and Elementary CA rulespaces were explored on the X-bit Memory Benchmark with both substrates.

Efforts were made to discover how the innate differences in connectivity between CA and RBN would affect the performance of the reservoirs, and whether further exploration of RBN RC in conjunction with ReCA could bridge the gap between CA and biological neural networks as a substrate.

The RBN reservoirs used in this thesis showed tendencies to perform worse than CA for much of the explored rulespace, for the specific configurations tested on the X-bit Memory Benchmark. However, several rules exhibited interesting behaviour, an prompted further exploration and mapping of a shared rulespace between CA and RBN.

The results suggest that further exploration the parameterspaces and rulespaces of RBN RC in tandem with ReCA could facilitate the use of CA as an interface to biological substrates like BNN.

Acknowledgments

I would like to thank my main supervisor, Tom Glover. A witty and compassionate genius, and a master of Elementary CA. If not for his unwavering will to criticise my models and encourage me, this whole thesis would likely have been a mere shadow of its potential.

My co-supervisor, Professor Stefano Nichele. A true inspirational force in the field of biologically inspired AI, and quite possibly the reason I even wrote this thesis.

My unofficial tertiary supervisor, Francesco Martinuzzi. A humble pioneer in reservoir computing using the Lifelike CA rulespace, and a contributor to the Julia programming language.

A sincere thanks to Sidney Pontes-Filho for adding support for the Lifelike rulespace in his library EvoDynamic just so I could use it in my thesis. Also for granting me remote access to servers I could run experiments on at the OsloMet AI Lab, *while* he was on vacation in Brazil.

I would also like to extend my gratitude to Kristine Heiney, Håkon Weydahl and Jørgen Jensen Farner for being great intellectual sparring partners throughout my time as a master's student.

A big thanks to my family and friends for not bothering me while I was doing my thesis these past five months. I don't know if it was intentional, but it means a lot.

I'm kidding. Thanks for checking in. :)

Last, but not least, thanks to Sarah Ødegaard, my fiancée and best friend, for keeping us both sane during these trying times. ♡

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reservoir computing (RC) is a growing field within artificial intelligence where a non-linear problem is projected into an untrained substrate, propagating the input stream into a higher dimensional space where it's linearly separable by a linear algorithm in the readout layer.

While RC architecture bears resemblance to the various Artificial Neural Network (ANN) architectures of Deep Learning (DL), the training process demands very few resources in comparison. The key point of RC is that the weights[1] inside the reservoir medium are not trained, only the linear readout layer, which can be made up of simpler Machine Learning (ML) techniques like linear regression, or more complex architecture like a Support-Vector Machine (SVM).

The earliest approaches to RC entailed the employment of Recurrent Neural Networks (RNN) as a substrate, but with randomly initialized weights. The pioneers of RC sought to exploit the innate computational capabilities of RNN without the need to go through the resource-intensive training of the internal weights. The traditional layered structure of ANN is removed, and what is left is a *reservoir* of dynamic properties.

Even though RC achieved success in maintaining high accuracy in appropriate tasks while reducing the resource costs of training, RNN still has fundamental drawbacks which must be considered. The weights between the artificial neurons in the network are represented by floating point numbers, which makes the computation cost of inference relatively high.

A consequence of this is that implementing such a reservoir in hardware becomes highly complicated. A common goal of RC researchers as of late has been to further reduce computational complexity and energy consumption by exploring and exploiting other viable substrates.

For a physical or conceptual system to be considered a viable substrate in RC, it must possess the *echo-state* property, as coined by (Jaeger, 2001). Previous state and input must dissipate gradually in the medium without getting amplified, in other words, the system must exhibit a *fading memory* of sorts.

The other integral requirement is the *separation property*, the ability to propagate the input

[1] the "strength" of the connections between the artificial neurons or other similar elements

throughout the reservoir.

Two dynamical systems that have been shown to possess such properties and have been used in modelling of complex systems (Sayama, 2015) are:

- Cellular Automata (CA)
- Random Boolean Networks (RBN)

(1) Having been studied closely since the dawn of computer science, CA is embedded in many a computer scientist's heart. CA is a complex system of cells, typically organized in a lattice[2] structure. Each cell has a discrete position based on its coordinates in the plane, from where it interacts locally with other cells.

(2) RBN exhibit many of the same computational capabilities as CA. The update schemes of classical RBN may be exactly the same as those of CA, however, their systematic structure and interaction between elements are fundamentally different to each other.

If one is familiar with common data structures, CA might be explained as a two-dimensional list, or an array of arrays. RBN are more easily compared to a linked list - a graph of sorts, where the absolute positions of elements do not matter as much, as the interactions are based on edges[3] instead.

As rapid advancements are made in DL, so too do the computational resource requirements and energy consumption associated with both training and inference increase. Taking steps towards more biologically inspired artificial intelligence could result in a major positive impact on the environmental influence of the technological advancements in the field.

Training being a well known cost in the field of DL, and AI in general, where a GPU might be running at capacity to train a model on a specific dataset continuously for days at a time. Running inference in real-time with continuous streams of data is not trivial either, however, and requires state-of-the-art, powerful components like GPUs in many cases.

The Sustainable Development Goals (SDG) of the United Nations (UN) are the product of the sovereign nations of the world coming together to envision a better world for all for the future. A set of 17 measurable goals that cover every essential issue in the globalized world and cultures we live in, which lays the foundation for a more sustainable approach to everything from our basic human needs to prosperity.

It is a bit of a paradox, as AI can arguably aid in the progression of many SDG, including those within Energy (Chui, Lytras and Visvizi (2018); Khan, Paul, Momtahan and Aloqaily (2018); Muhammad, Lloret and Baik (2019), but it's simultaneously consuming a lot of energy.

The environmental implications of conserving energy can be connected to the Sustainable Development Goals (Desa & Nations, 2016), particularly (*Goal 7, Ensure access to affordable, reliable, sustainable and modern energy for all*, n.d.), (*Goal 9, Build resilient infrastructure, promote inclusive*

[2] 2-dimensional regular grid

[3] directional or bi-directional connections between two elements, or from one element to itself

and sustainable industrialization and foster innovation, n.d.) and (*Goal 11, Take urgent action to combat climate change and its impacts*, n.d.).

The massive growth of the Internet of Things (IoT) is in parallel causing computing devices to become ever-smaller and ubiquitous.

However, with Deep neural networks becoming deeper and more computationally demanding, it's becoming less viable to deploy DL models to devices without state-of-the-art GPUs for real-time processing.

Integrating AI with smaller devices is therefore becoming more difficult, and currently requires inference to take place in the cloud, which can be unreliable. Edge computing and 5G are other converging technologies on the horizon that could alleviate some of the problems (Deng et al., 2020; Hassan, Yau & Wu, 2019; Zhou et al., 2019).

CA-based reservoirs can offer a significant reduction in computational demands Yilmaz (2014), and can even be physically implemented using XOR logic gates and shift-registers (Morán, Frasser & Rosselló, 2018).

Biological intelligence is more tolerant to the randomness in nature and more robust to perturbed information. It is better able to handle previously unseen data and situations than engineered forms of intelligence like deep neural networks[4].

And so, the primary motivation to explore the viability of RC built on CA and RBN is precisely the approximation of natural biological computation, and the lower energy consumption associated with it.

Thus, the essence of the thesis lies in exploring the small differences in connectivity between CA and RBN. RBN is after all an abstraction of CA, meaning CA can be seen as a type of RBN where the spatial properties and connectivity are uniform. The intent of the work herein is to design RBN reservoir substrates with CA as the starting point, with as few *steps* between them as possible. It is my hypothesis that merely changing from local to random connectivity will affect how input is propagated through the reservoir. By keeping the different substrates as similar as possible, one can uncover the impact of different connection strategies.

The research questions then become:

1. How do innate differences in connectivity between CA and RBN with similar rules affect RC accuracy?
2. Does the effect of varying connectivity increase between one-dimensional and two-dimensional CA compared to RBN with similar rules?

Finding answers to these questions could potentially contribute to bridging a gap between CA and biological neural networks as substrates in reservoir computing.

[4]It is regarded a common truth in the field of AI, that weak or narrow AI is vulnerable to even the smallest perturbations.

1.1 Ethical Considerations

The field of artificial intelligence has evolved quickly over the last two decades, and is becoming rather powerful, more so than most people might think. At least in the area of specific tasks - advancements in the approximation of general intelligence are far slower.

AI also amplifies many ethical issues regarding how information about individuals is collected and used in society as a whole. AI developers aren't always aware of innate biases in the data that they use to train their models, and thus unwittingly pass on these biases to predictions that the models will make. This is particularly dangerous when that data concerns anything that is used to identify people and, for instance, their behaviour.

While this is not likely to be directly applicable to the benchmarking tasks in this thesis, it's important to highlight issues that we're collectively facing. When working with bit-based memory benchmarks, these considerations are of little consequence. However, as soon as one starts to look at human-made data, like the MNIST digit character dataset, things get more complicated. The model is in effect only trained to discern between different Arabic numerals, and does not give any indication of efficiency with, say, Chinese Mandarin numerals.

AI is not perfect. It is directly shaped and constrained by our own perception, and the means that are at our disposal. As academic researchers in a potent field, we owe it to the general public to write in such a way that we convey both the beneficial and harmful potency of advancements in AI technology.

Although the environmental impacts of this research may be positive, it is highly unlikely that it would be a realistic factor in the nearest future. Most of the research done in the field of reservoir computing is primarily concerned with the long-term vision of AI and computation in general.
This thesis was done in association with the DeepCA project[5][6] at OsloMet, and was proposed by PhD Candidate Tom Glover.

The research goal of DeepCA is to establish the theoretical and experimental foundation for a hybrid Deep Learning paradigm with CA and biological neural networks.

1.2 Thesis Outline

Firstly, the field of research is established in the Background, the problem area is narrowed down by looking at the contributions of recent studies in State-of-the-Art, the steps taken to answer the research questions will be detailed in Methods, the experiments and the results are shown in Results, the results, process and limitations are discussed in Discussion, and finally a conclusive summary and musings about possible future work in Conclusion.

[5] https://www.oslomet.no/en/research/research-projects/deep-ca

[6] https://www.nichele.eu/DeepCA.html

Chapter 2

Background

This chapter explains the fundamental theoretical concepts that are relevant to thesis topics. Firstly, Reservoir Computing (RC) is described in brief, before the knowledge on the two substrates Cellular Automata (CA) and Random Boolean Networks (RBN) is established. Afterwards, RC is explained more thoroughly in the context of using CA and RBN as reservoir substrates, and ways to physically implement these. Lastly, different types of linear readout methods are described.

2.1 Reservoir Computing

Reservoir Computing (RC) emerged from the two independent trajectories of Echo-State Networks (Jaeger, 2001) and Liquid State Machines (Maass, Natschläger & Markram, 2002). The former was an alternative RNN approach for control tasks. The latter focused on biological modeling of cortical microcircuits with spiking integrate-and-fire neurons. RC is a relatively new paradigm in RNN training, outperforming the *classical*, fully trained RNNs.

It involved projecting input through an untrained, dynamical substrate and separating the results in a linear output layer. The input needed to be mapped to the reservoir layer, the reservoir layer then projected the input to a higher dimension, and the output layer read the state of the reservoir and extracted the features. The readout layer in RC architectures comprises a linear regression, or classification model. It is responsible for the classification of input by linearly separating the feature vectors created by the reservoir.

The continued works in RC attempted to use the innate dynamical capabilities of RNN as a medium for computation, by randomly initializing the weights in the network and leaving them untrained. Exploiting the intrinsic dynamics of RNN as an excitable system, and thereby only having to spend resources training the output layer.

RNN are a type of cyclical derivative of feed-forward neural networks, that takes into account the past states of the network when determining the next output. RNNs are turing complete (Siegelmann & Sontag, 1995). The memory capabilities of RNN have made them apt at, and widely popular for, temporal problems like natural language processing and speech

recognition (Jaeger, 2001).

There are at present day several drawbacks of deep learning architectures that persist in RNN-based RC, especially floating point calculations, which could be alleviated by using CA or RBN as substrates instead.

For a reservoir to perform well at computational tasks, it must be able to both forget past perturbations and keep two input streams that have begun converging separated (Bertschinger & Natschläger, 2004). These properties are the *fading memory* and *separation property* mentioned in chapter 1.

2.1.1 Reservoir Computing with Physical Substrates

A great prospect of RC is the versatility of reservoirs. In theory, any medium that possesses *fading memory* and *separation property*, could be successfully applied as a substrate. (Fernando & Sojakka, 2003) is often cited in this regard, as it demonstrates that water, a natural dynamical system, could be used as a substrate in RC to allow sound waves to be linearly separable in a voice recognition task.

The floating point weights in RNN substrates entail that hardware implementations of RC would be quite inefficient with, e.g., Field-Programmable Gate Arrays (FPGA). This drawback turns into and advantage of CA and RBN as discrete, binary systems that can be described with logical gates.

One could argue that CA has an *edge*[1] over RBN in this application because of its regular, local connectivity. CA is also supported natively in FPGA by being fully parallel, while RBN could be asynchronous.

Research of RBN in the context of physical substrates are still very much worthwhile, because it could be a closer approximation to biological neural networks.

Some rules in the two substrates can be very expressive and difficult to implement, but a rulespace such as Elementary Cellular Automata can easily be expressed with boolean logic.

ReCA can be synthetically implemented in FPGA through emulation software and trained (Liang, Hashimoto & Awano, 2021; Morán et al., 2018), before being implemented physically in FPGA hardware (Morán et al., 2018).

2.2 Cellular Automata

Automata (singular: automaton) are theoretical machines that alter their internal states based on external inputs and knowledge of their own previous states (Sayama, 2015). Cellular Automata (CA) consist of a set of automata arranged in a regular grid, which states are updated synchronously (Sayama, 2015). This grid can be a one-dimensional vector as in the ECA rulespace, or a two-dimensional lattice as in the Lifelike rulespace. CA is a discrete

[1] Pun intended.

computational model of complex systems, one of the first to describe self-reproduction and evolution in living systems (Neumann, 1966; Sayama, 2015).

The interaction between cells is local and self-organized - devoid of any form of centralized control. A cell can be in one of a predefined number of states, where two states is most common, like in RBN.

The state is updated in discrete time steps based on local interactions in a structured way. This update scheme is referred to as a rule and is defined in conjuncture with the size of the local neighbourhood and the number of possible cell states.

The number of possible rules and transition tables increase exponentially with the number of states and neighbours. Given the number of possible states S, the number of possible neighbourhood states will be S^N, where N is the number of neighbours each cell has. Each of these possible neighbourhood states can be mapped to any of the S states, resulting in S^{S^N} unique rules in the given rulespace (Babson & Teuscher, 2019).

For Elementary CA, the rulespace is of size $2^{2^3} = 256$, and the Lifelike rulespace is of size $2^{2^8} = 2^{256} = 1.1158 \times 10^{77}$ = *immense*.

Because the interactions are predefined by the rule, and the cells interact in the same local neighbourhoods each time, a CA is deterministic. That means if you apply the same initial conditions to a CA, it will evolve in the exact same way every time.

CA are capable of displaying emergent behaviour with certain rules.

Like RNN, certain rules of CA, like ECA rule 110, are considered Turing Complete (Cook et al., 2004).

2.2.1 Lifelike Cellular Automata

The Lifelike rulespace comprises a subset of binary, totalistic rules with a Moore neighbourhood, spanning the eight cells surrounding the centre cell. The rules sum the states of the neighbours, and state transition is determined by these factors:

- The cell's own binary state (dead or alive)
- A set of state sums that allow a dead cell to be born
- A set of state sums that allow a living cell to survive

(Sayama, 2015)

The rules are expressed simply as rulestrings on the form *B3/S23*, where the B-component expresses the set of state sums that allow a dead cell to be born, and the S-component in turn expresses the set of state sums that allow a living cell to survive. Should the sum of states in the neighbourhood fall outside these defined sets, living cells will die and dead cells remain dead.

The most famous among the Lifelike rules is Conway's Game of Life (Conway et al., 1970), having reached a wide audience of mathematicians and computer scientists since its inception in the 1970s. The mini-universe of Game of Life has been extensively studied throughout the years, and many of the discoveries are now detailed in (Johnston & Greene, 2022). Game of Life

has been considered to belong to Wolfram's Class IV, displaying complex behaviour (Packard & Wolfram, 1985).

There is likely untapped potential in the Lifelike rulespace (Bak, Tang & Wiesenfeld, 1987; *Life-like cellular automata*, n.d.), and researchers have begun applying advances in heuristics to search and classify the Lifelike rulespace for other CA rules with equally complex behaviour as Game of Life (Eppstein, 2010).

Some of the rules explored by (Eppstein, 2010) include:

- B25/S4 (Class III)
- B27/S0 (Class III)
- B35/S236 (Class III)
- B37/S23 (Possible Class IV)
- B36/S245 (Class II or IV)
- B368/S12578 (Possible Class IV)
- Morley (B368/S245) (Class II or IV)

Many of these rules support Lifelike constructs like spaceships, replicators, gliders, etc. The behavioural classifications seemed to vary between them when initial conditions were randomized.

B36/S245 could be characterized by its narrow, yet persisting living and dead regions with fine-grained variations and smaller emerging patterns of dead islands which appeared and disappeared in two to three input cycles.

Figure 2.1 depicts the history of different Lifelike rules when perturbed by the X-bit Memory Benchmark.

The CA size is $10 \times 10 = 100$, but it's flatted for every timestep in the figure so that every x in the top of the $y - axis$ represents the two-dimensional CA in the first timestep. Every subsequent step downwards in the $y - axis$ represents the next state. This representation directly correlates with the reservoir feature vectors, and makes Lifelike rules more easily comparable to ECA than by, e.g., looking at several two-dimensional plots for each rule to determine how they evolved, and is consistent with Lifelike history figures in the rest of the thesis.

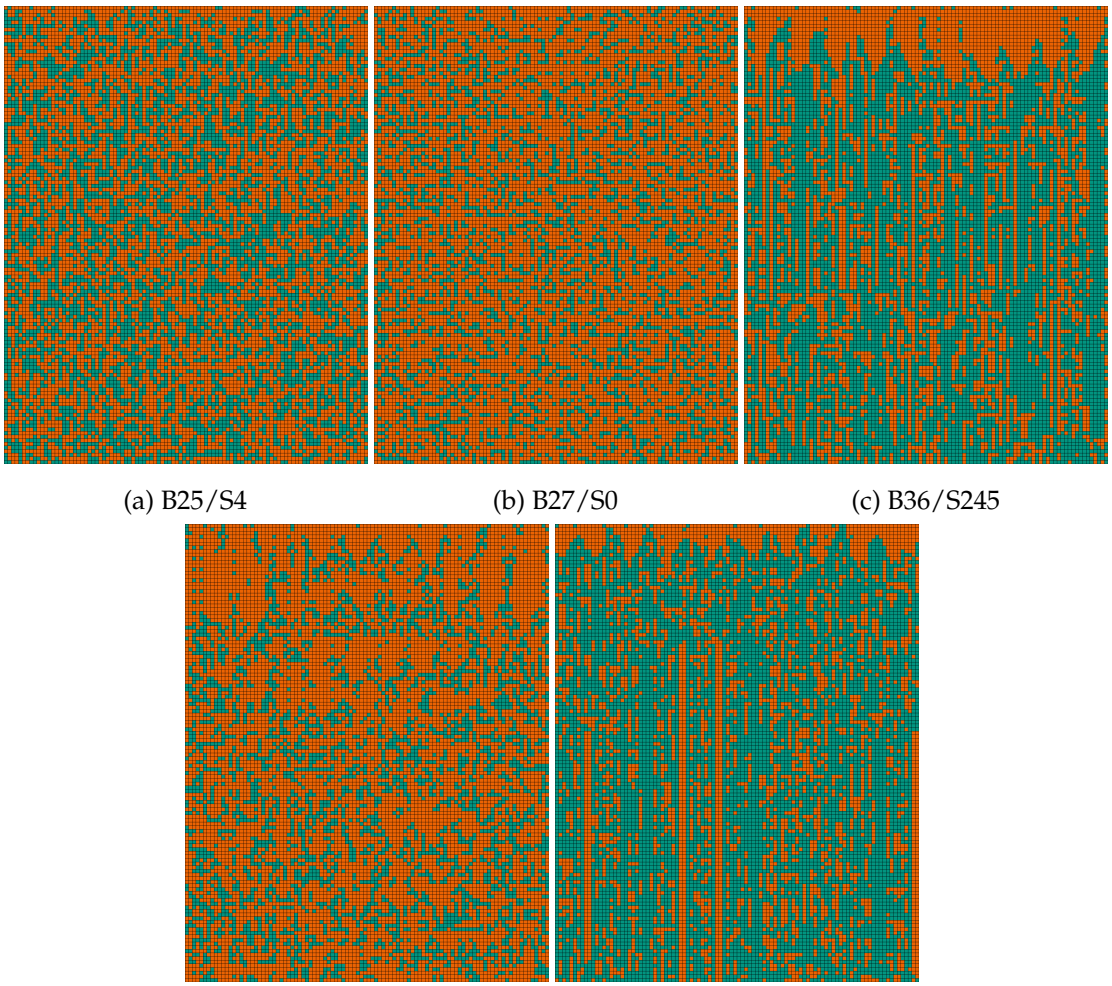
2.2.2 Elementary Ceullular Automata

Elementary Cellular Automata (ECA) is a rulespace defined by having two possible states $S = 2$ and $N = 3$ number of neighbours, thus comprising $S^{S^N} = 256$ unique rules. The order of the neighbours is a factor in the state transition tables, and the centre index neighbour is always a self-connection. Many of the 256 rules have equivalent rules inside the rulespace, either by having complimentary or mirrored versions of the other's state transition tables.

Rule	Equivalent	Rule	Equivalent	Rule	Equivalent
0	255	35	49,59,115	108	201
1	127	36	219	110	124,137,193
2	16,191,247	37	91	122	161
3	17,63,119	38	52,155,211	126	129
4	223	40	96,235,249	128	254
5	95	41	97,107,121	130	144,190,246
6	20,159,215	42	112,171,241	132	222
7	21,31,87	43	113	134	148,158,214
8	64,239,253	44	100,203,217	136	192,238,252
9	65,111,125	45	75,89,101	138	174,208,244
10	80,175,245	46	116,139,209	140	196,206,220
11	47,81,117	50	179	142	212
12	68,207,221	51		146	182
13	69,79,93	54	147	150	
14	84,143,213	56	98,185,227	152	188,194,230
15	85	57	99	154	166,180,210
18	183	58	114,163,177	156	198
19	55	60	102,153,195	160	250
22	151	62	118,131,145	162	176,186,242
23		72	237	164	218
24	66,189,231	73	109	168	224,234,248
25	61,67,103	74	88,173,229	170	240
26	82,167,181	76	205	172	202,216,228
27	39,53,83	77		178	
28	70,157,199	78	92,141,197	184	226
29	71	90	165	200	236
30	86,135,149	94	133	204	
32	251	104	233	232	
33	123	105			
34	48,187,243	106	120,169,225		

Table 2.1: The group of equivalent rules. The primary column is the minimum equivalence rule that represents its group of equivalent rules if any. (Glover et al., 2022)

Figure 2.1: History plots with flattened two-dimensional CA Lifelike rules. The other four Lifelike rules examined in this thesis is examined in figures 5.1a, 5.2a, 5.2b and 5.3a



(a) B25/S4 (b) B27/S0 (c) B36/S245

(d) B37/S23 (e) B368/S245

2.3 Random Boolean Networks

Random Boolean Networks (RBN) were first conceptualized by (Kauffman, 1969) as a method to model a gene regulatory network.

RBN can be considered an abstraction of CA with two possible states, and as such one might consider CA a subtype of RBN with regular and local connectivity.

Defined in graph theory, RBN is a directed graph with N vertices and $E = \langle K \rangle \times N$ directed edges (Snyder, Goudarzi & Teuscher, 2013).

The connections between cells, or nodes, are not uniformly structured, but randomly initialised. Unlike CA with its regular grid, nodes in a network don't necessarily need a determined spatial position.

The connections of a network can be either homogeneous or heterogeneous, where the nodes have a static number of edges or an average number of edges, respectively.

The number of edges in a network, and the neighbourhood size is described by the *in-degree*

or K. $\langle K \rangle = 2$ means that the nodes in the network will have two neighbours *on average*, some more - some fewer, while $K = 2$ means that every node is forced to have two neighbours.

RBN are typically totalistic, which means that the order of the neighbours do not matter, unlike in ECA. The state transition tables are formally defined as boolean operators on the boolean/binary states of the nodes.

Several types of RBN were introduced in (Gershenson, 2004):

1. Classical Random Boolean Networks
2. Asynchronous Random Boolean Networks
3. Deterministic Asynchronous Random Boolean Networks
4. Generalized Asynchronous Random Boolean Networks
5. Deterministic Generalized Asynchronous Random Boolean Networks
6. Mixed-Context Random Boolean Networks

(1) can be considered deterministic on like CA, because they are synchronous. (2) elements in systems of biological intelligence, like neurons in neural networks and genes in DNA, do not update their states in discrete timesteps. A node is picked at random and updated, instead of updating the nodes synchronously. Non-deterministic. Destroys the cycle-attractors of CRBNs, as it's highly unlikely that a sequence of states will be repeated with a non-deterministic updating. (3) introduces to parameters to ARBNs, the *period* and the *translation* of the node updates. If more than one node is to be updated in the same time-step, it is done in an arbitrary order. (4) a group of nodes is randomly selected and updated synchronously. Semi-synchronous, but non-deterministic. (5) if more than one node is to be updated in the same time-step, all of those nodes are updated synchronously. (6) the *period* and the *translation* of the node updates are seen as the *context* of the network.

2.4 Classifications and the Edge of Chaos

Both CA and RBN have well-established behavioural classifications associated with them, often referred to as *classes* and *phases* in literature pertaining to CA and RBN, respectively.

In both paradigms, there is a consensus that optimal conditions for computation of information is found somewhere on the edge between *ordered* and *chaotic* states.

These lower and an upper limits of complex behaviour (Wolfram, 1984) are found to be relatively close together, posing a bit of an observational challenge in locating the phase transition of a given CA or RBN system. (Langton, 1990)

2.4.1 Wolfram's CA categories

Since its inception, several attempts have been made to characterize and classify the behaviour of CA. Best known among those is the Wolfram Classification. (Wolfram, 1984)

- Class I - All cells go towards the same state
- Class II - The system settles into stable or oscillating patterns
- Class III - The chaotic state of the initial pattern is perpetuated for eternity
- Class IV - The rest of the possibilities, complex behaviour that lies somewhere between classes II and III.

This classification has been thoroughly used and tested in past decades, and received criticism for not being entirely applicable to widely explored rulespaces, such as ECA and Lifelike CA. However, it still serves as a baseline for most researchers who endeavour into the field of CA, and many newer sub-classifications use it as a frame of reference.

2.4.2 ECA with Memory (ECAM)

ECA rules have been found to have varying alignments with their original Wolfram Classifications when paired with a memory function (Martinez, Adamatzky & Alonso-Sanz, 2013).

The ECA rules where found to group into 3 classes:

- **Strong:** Most memory functions change the rule to another different class quickly
- **Moderate:** Memory functions can transform to a different class and conserve the same class as well
- **Weak:** Memory functions are unable to transform into another class

These 3 definitions are in line with the interpretations made by (Glover et al., 2022), and not the original publication (Martinez et al., 2013).
The classification tables can be found in the appendix.

2.4.3 Additive Cellular Automaton (ACA)

An additive cellular automaton is a cellular automaton whose rule is compatible with an addition of states (*Additive Cellular Automaton*, n.d.).

ECA Rule	Boolean
0	0
60	$c_{-1} \oplus c_0$
90	$c_{-1} \oplus c_1$
102	$c_0 \oplus c_1$
150	$c_{-1} \oplus c_0 \oplus c_1$
170	c_1
204	c_0
240	c_{-1}

Table 2.2: Additive Cellular Automata and corresponding boolean form. In this notation c_{-1}, c_0 and c_1 is the left, central and right neighbour, respectively (Glover et al., 2022).

2.4.4 Alternitve Four-way Classifcation of Two-dimensional Semi-Totalistic Cellular Automata

A new classification method was proposed for Lifelike rules in (Eppstein, 2010).

1. do there exist patterns that eventually escape any finite bounding box placed around them?

2. do there exist patterns that die out completely?

- If both are true: cellular automaton rule is likely to support spaceships, small patterns that move and that form the building blocks of many of the more complex patterns that are known for Life

- If one or both are not true: there may still be phenomena of interest supported by the given cellular automaton rule, but we will have to look harder for them

2.4.5 Edge of Chaos

The critical point between orderly and chaotic states in complex systems has been dubbed "the Edge of Chaos". Usually referred to in combination with the Wolfram Classification, where Class II represents order, Class III represents chaos and Class IV is somewhere in between - on the edge (Wolfram, 1984).

The λ parameter was introduced by (Langton, 1990) as a method to parameterize the rulespaces of CA. It orders the set of possible state transition tables based on how many states a transition function go towards an arbitrary *quiescent state* S_q.

Lower values of λ represent systems which dynamics quickly die out, and higher values are associated with chaotic and random dynamics. The emergent behaviour in either end of the spectrum is considered *simple* and easily predictable. However, there exists a space in

between, where the phase transitions from ordered to chaotic, vice versa, in which *complex*[2] and unpredictable emergent behaviour can be seen (Langton, 1990).

One can observe that the ordered behaviour displays better memorizing capabilities, *fading memory*, while the chaotic behaviour is more inclined to propagating the input information, the *separation property* - both essential components of computation.

Therefore, it is not impossible for systems leaning more to either side to be capable of computation, however, ordered systems would need more time, while chaotic systems need more redundancy to counteract instability (Gershenson, 2004).

Simply put, a perfect balance of order and chaos is more economical and therefore advantageous in an evolutionary sense.

Criticality in heterogenous RBN has been found with *in-degrees* of $\langle K \rangle = 2$ (Snyder, Goudarzi & Teuscher, 2012; Snyder et al., 2013), while it is reportedly found closer to $K = 3$ in homogeneous RBN (Burkow, 2015, 2016). However, computation is considered possible in RBN with higher in-degrees. However, it is possible that RBN with a higher in-degree are capable of computation.

Criticality in CA doesn't appear to be as limited in terms of connectivity, as computation is seemingly possible in both the ECA and Lifelike rulespaces, where the *in-degree* would be $K = 3$ and $K = 8$, respectively. A reason for this may be the regular, local connectivity, and there is a chance this thesis might prove that the computational properties of RBN are indeed more sensitive to the in-degree.

2.4.6 Rulespaces

Elementary CA (ECA) has the benefit of an exhaustible rule-space of 256 unique transition tables.

RBN has also been found to have an optimal balance of separability and fading memory at critical connectivity, an in-degree of $\langle K \rangle = 2$ in heterogeneous RBN and $K = 3$ in homogeneous RBN, which is similar connectivity to ECA (Burkow, 2015; Snyder et al., 2013).

Game of Life (GoL) and the rest of the Lifelike rulespace would offer more interesting spatial and temporal comparison to a nature-like network model. Conway's GoL (Conway et al., 1970; Johnston & Greene, 2022) is one of the most tried and tested rules for CA, and displays computational properties.

(Babson & Teuscher, 2019) found that more expressive CA rules required a smaller reservoir to achieve comparable accuracy to ECA rules, but it is unclear whether rules in the Lifelike rulespace fall into the category of *more expressive*.

Sampling from the Lifelike rulespace has the potential of reaching beyond the criticality in ECA, with its many rules that might have similar computational potential as GoL, or better.

[2]Behaviour reminiscent of Wolfram's Class IV description.

A major problem is exactly the sheer size of the space to search, though, meaning the selection of rules would have to be done on an empirical basis in this thesis.

Chapter 3

State-of-the-Art

Given the time-constraints of a short thesis, it was deemed implausible to do an exhaustive parameter search for the substrates of interest, while also doing comparative experiments with the models. Instead, a qualitative selection process was chosen to determine which directions were viable, based on relevant literature of experiments with ReCA in both the Lifelike and ECA rulespace, and RBN RC.

There were several important constraints to consider when narrowing down the type of reservoir substrates to implement thesis. This chapter seeks to clarify the grounds for the decisions made in developing the method, while bringing to light the most relevant recent works on the topics of ReCA and RBN RC.

3.1 Reservoir Computing with CA (ReCA)

The first to do RC with CA as substrates in the reservoirs, was (Yilmaz, 2014). Their methods were a bit difficult to interpret. The reservoir architecture utilizing Game of Life was seemingly comprised of several 2-dimensional sub-reservoirs R. For the 5-bit Memory Benchmark, each sub-reservoir was perturbed with the input vector. Based on the wording of the paper, it seemed that 100% of the reservoir was perturbed, and that the total size of the grid was equal to (Input Size) X R. However, taking into account the very specific figures in the paper, it seemed that the sub-reservoirs R were the size of the input squared, and thus around 25% of each sub-reservoir was perturbed. In any case, the size of the CA was not exactly specified and rather left up to interpretation.

Discrepancies between the figures and tables representing ECA data did not help, either. 5-bit experiments with distractor period 200 and ECA Rule 90 were clearly shown in the tables to achieve zero errors with any combination of (R, I) which resulted in size 512 or larger. However, the same rule was shown to require a minimum size of roughly 750 to achieve zero errors with a distractor period of 160. It is likely that that figure is pointing to 768, splitting the difference between 512 (16 x 32, or 32 x 16) and 1024 (32 x 32).

Recent works in ReCA have futher explored the ECA rulespace and beyond, and experimented with variations in the reservoir architecture.

(Nichele & Molund, 2017) implented ReCA with an expansion into a two-layer (deep) reservoir in the ECA rulespace, tested on the 5-bit Memory Benchmark. Deep reservoirs entailed that the output of the first reservoir was used as input in the second.

Outputs from the first layer was compared with the multi-layer outputs. Results from the layered system exhibited improvements to the single reservoir system, particularly rule 165 at ($I = 4$, $R = 4$), where I is the number of iterations and R is the number of separate CA in a single reservoir.

(Nichele & Gundersen, 2017) implemented ReCA with Non-Uniform Binary Cellular Automata in the ECA rulespace, tested on the 5-bit benchmark. Reservoirs with different CA rules were used in parallel, showing that some rules work well in combination with each other. Good combinations of rules seemed often have a relation like being complementary rules.

(Babson & Teuscher, 2019) implemented ReCA in C++ and tested it on the 5-bit Memory Benchmark. Expanded on research on Elementary CA to investigate rule-spaces from more complex CA. Used a genetic/evolutionary algorithm to find "edge of chaos" rules capable of computation in those complex CA.

(McDonald, 2017) implemented ReCA and Extreme Learning machines with pairs of ECA rules, seemingly using one rule for memory and the other for separation. It was a bit unclear how the memory rules affected the benchmarks, or if this part of the method was perhaps not tested.

The paper proposed several novel benchmarks in the field of ReCA:

- Sine and Square Wave Classification
- Non-linear Channel Equalization
- Santa Fe Laser Data
- Iris Classification

The parameter space of ReCA with ECA on the X-bit Benchmark in recent works (Glover et al., 2021, 2022). A ReCA model was implemented using the EvoDynamic library (Pontes-Filho et al., 2020) and tested on the X-bit Memory Benchmark, where the impact of varying the number of bits (N_b), the redundancy R, the length of subreservoirs[1] L_d, reservoir height (I) and distractor period (D_p) was evaluated.

(Glover et al., 2021) systematically examined parameters of ReCA with Elementary CA, namely redundancy, number of bits, size of the vector the input is mapped to, the CA permutation iteration and the CA rules. The parameters reservoir height I and distractor period D_p were static for the experiments. Showed that the parameter spaces in the explored regions

[1] Described as the size of the vector the input is mapped to.

were dynamic, impacting different rules in different ways, and require careful consideration when using CA as a reservoir.

The work was continued in (Glover et al., 2022), where more of the paramerter space was explored: I, which controlled the size of the reservoir feature[2] and D_p, the distractor period of the benchmark. They observed an increase in difficulty with increasing D_p much like (Yilmaz, 2014), who claimed that this increase was logarithmic. Not as impactful as increasing the number of bits N_b (the other benchmark-specific parameter), but (Glover et al., 2022) hypothesised that varying D_p might make either the input or cue signal easier or harder to separate in the reservoir.

A Support Vector Machine was used in the readout layer of (Glover et al., 2021).

From Scikit-Learn's documentation:

> The primal problem can be equivalently formulated as
>
> $$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max(0, 1 - y_i(w^T \phi(x_i) + b)), \tag{3.1}$$
>
> where we make use of the hinge loss (*1.4. Support Vector Machines*, n.d.).

ReCA has been synthetically and physically implemented with Field-Programmable Gate Arrays (FPGA) in recent studies (Liang et al., 2021; Morán et al., 2018).

ReCA with max-pooling, where the readout layer used a softmax regression, aka. Multinomial logistic regression or maximum entropy classifier was implemented in (Morán et al., 2018). Generalisation of logistic regression for use with multiple classes/categories. Trained and tested on MNIST. Synthetic hardware implementation using emulated circuitry for training. Used those saved weights on a physical hardware implementation using FPGA. Elementary CA rule 90 was iterated over rows and columns independently and combined using XOR, with a 3D boolean tensor output.

(Liang et al., 2021) implemented ReCA where the readout layer was an ensemble of Bloom filters, tested on the MNIST benchmark. Synthetic hardware implementation. Achieved 43 times reduction in inference memory cost compared with earlier work with Bloom filters. Elementary CA rules were applied to rows and columns independently and combined with XOR.

Few recent works in ReCA have explored the Lifelike rulespace. However, (Martinuzzi, 2022) has explored Lifelike rules that were highlighted in the works of (Eppstein, 2010; Peña & Sayama, 2021) in CA reservoirs on the 5-bit Memory Benchmark, first endeavoured as a part of Google Summer of Code 2020 (Martinuzzi, 2020).

The input mapping method was inspired by (Yilmaz, 2014) random distribution of binary data directly into the reservoir. Differing from ReCA models, the R parameter defines the full width of the reservoir, instead of the *redundancy* of separate connected subreservoirs. The full reservoir size is simply R^2, so for $R = 28$ there are 784 cells in the reservoir. The input vector

[2]Described as the reservoir height, which makes sense in ECA, but isn't easily transferable to Lifelike or other CA with more than one dimension.

was randomly mapped and projected into a pre-defined percentage of the entire reservoir, by a projection ratio of 60% $P_r = 0.6$.

At $R = 24, 26, 28, 30$ and $I = 6, 8, 10, 12$, this resulted in reservoir feature vectors of size $30 \times 30 \times 12 = 10,800$ for the biggest reservoirs, and $24 \times 24 \times 6 = 3456$ for the smallest. Significantly bigger than the ECA reservoirs previously described in this section. For instance, (Glover et al., 2021) had for $R = 4$, $L_d = 40$ and $I = 2$ reservoir feature vectors of size $4 \times 40 \times 2 = 320$.

Ridge Regression was used in the readout layer of (Martinuzzi, 2022). Ridge regression is very similar to the ordinary least squares of linear regression, but penalizes the size of the coefficients.

Linear Regression problem:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + ... + w_p x_p \tag{3.2}$$

Linear Regression:

$$\min_w ||Xw - y||_2^2 \tag{3.3}$$

Ridge Regression:

$$\min_w ||Xw - y||_2^2 + \alpha ||w||_2^2 \tag{3.4}$$

The complexity parameter $\alpha \geq 0$ determines the strength of this penalty.

The reservoirs achieved good results on the 5-bit Memory Benchmark with many of the reservoir configurations, for most the rules that were tested.

3.2 Reservoir Computing with Boolean Networks (RBN RC)

As this thesis endeavoured to contribute to closing the gap between CA and BNN in hopes of eventually being able to to use the former as a kind of interface to the latter in RC, the research and methods have a profound basis in CA, and not so much RBN.

This is reflected in the amount related work that was reviewed on the topic of RBN RC. This conscious, partial neglect was deemed fair with regard to the intended purpose of producing RBN that would resemble CA as closely as possible, and almost a necessity with respect to the time-constraints of a short thesis.

(Snyder et al., 2012) investigated the optimal connectivity of heterogeneous RBN as reservoirs, and observed a trade-off between *fading memory* and the *separation property* for different average in-degrees $\langle K \rangle$ in RBN.

Continued work in (Snyder et al., 2013) observed that the computational properties in heterogeneous RBN were often strongest at critical connectivity $\langle K \rangle = 2$. However, they warned that the manner in which the reservoir was perturbed should be taken into account when analyzing this correlation as well. They claimed that chaotic RBN may perform well if the length of perturbation is short, but will struggle to differentiate between different inputs when the perturbation lengths increase.

(Burkow, 2016) explored the performance of different RBN reservoir sizes for the Temporal Parity and Temporal Density benchmarks, with the intention of facilitating future implementations of different physical reservoirs. They found that optimal perturbation was found to be at around 50% of the reservoir for homogeneous RBN with in-degree $K = 3$. In their thesis pre-project, it was claimed that $K = 3$ would provide the most optimal conditions for computation (Burkow, 2015), differing from that of heterogeneous RBN at $\langle K \rangle = 2$.

The results of these papers and the master's thesis have provided grounds that RBN is an apt substrate for RC.

Chapter 4

Methods

In this chapter the methods used in pursuit of the research problem are detailed in order of significance. This experimental software setup is briefly explained before going into the model design details. Firstly, the architecture of the implemented RC system with support for both CA and RBN as substrates will be described, including methods used to create RBN with as few abstractions away from CA as possible. Secondly, the design and procedure of the experiments that were conducted are outlined before moving on to the experiment details and results in the next chapter.

For reference, the research questions are repeated here:

1. How do innate differences in connectivity between CA and RBN with similar rules affect RC accuracy?

2. Does the effect of varying connectivity increase between one-dimensional and two-dimensional CA compared to RBN with similar rules?

4.1 Experimental Setup

The software used in these experiments was developed in the Python programming language. EvoDynamic, a library created specifically for complex systems modelling with TensorFlow support was used to set up the reservoirs (Pontes-Filho et al., 2020). The readout layer was powered by Scikit-Learn's out-of-the-box linear classification models, specifically Ridge Regression and Support Vector Machine.

4.1.1 Model Architecture

Herein follows a recount of how the three major parts of this reservoir computing system was implemented. These three are as follows:

1. Input layer

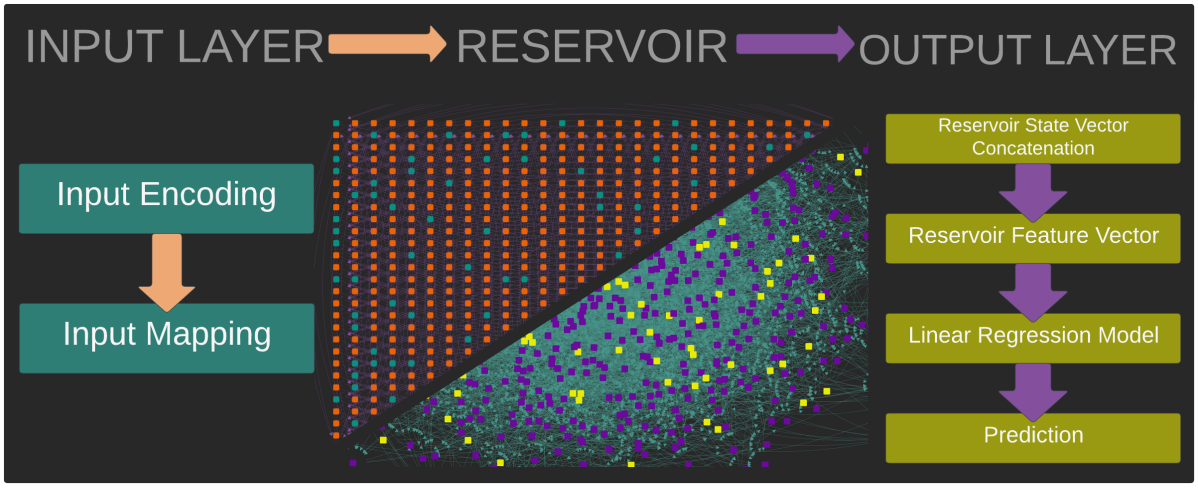


Figure 4.1: Reservoir Computing Architecture

2. Reservoir

3. Output layer

Input Layer

The purpose of the input layer is to encode the input data and map it to the elements of the reservoir itself. The encoding method should reflect the type of data and the mapping strategy will vary depending on the shape and dynamics of the reservoir, relative to the task at hand.

Two main methods of input mapping had been suggested in earlier literature (Yilmaz, 2014). The first option was to apply weighted summation to input data in order to get binary inputs for *every* element in the reservoir. The second option was valid for binary data, and entailed a random mapping of an input vector representation of that data.

Since the benchmarks of this thesis revolved around binary data, the second option was adapted for the Lifelike reservoirs in this work. Depending on the reservoir model used, the input vector was either randomly mapped *once* per sub-reservoir (Glover et al., 2021), or to a certain number of elements in the full reservoir, defined by a *projection ratio* P_r (Martinuzzi, 2022). As a rule, ECA reservoirs used the former, and Lifelike reservoirs the latter.

Reservoir

The reservoir is where the input is separated and propagated into a higher dimensional space, in order to make separation through linear algorithms possible. The reservoir models in this work were implemented with CA and a variation of RBN designed to be as similar to CA as possible. Results from (Glover et al., 2021) suggested that the aptness of different rules could vary greatly with even the smallest changes to reservoir size and sub-division. However, the parameter space is largely left unexplored in this work, with the intention of focusing in on the behavioral differences between CA and RBN in the same rulespace.

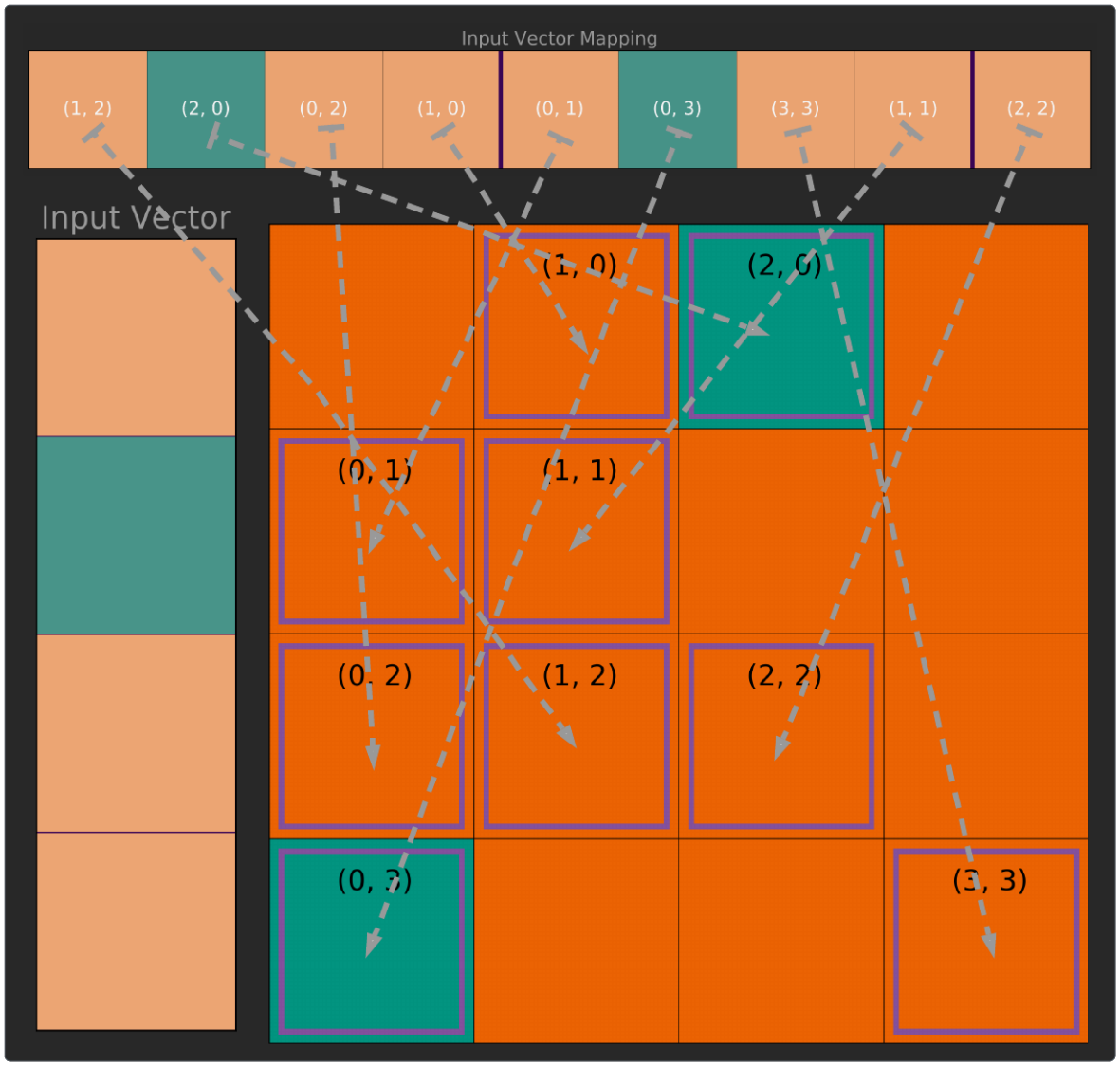


Figure 4.2: Input Mapping Scheme for the Lifelike rulespace

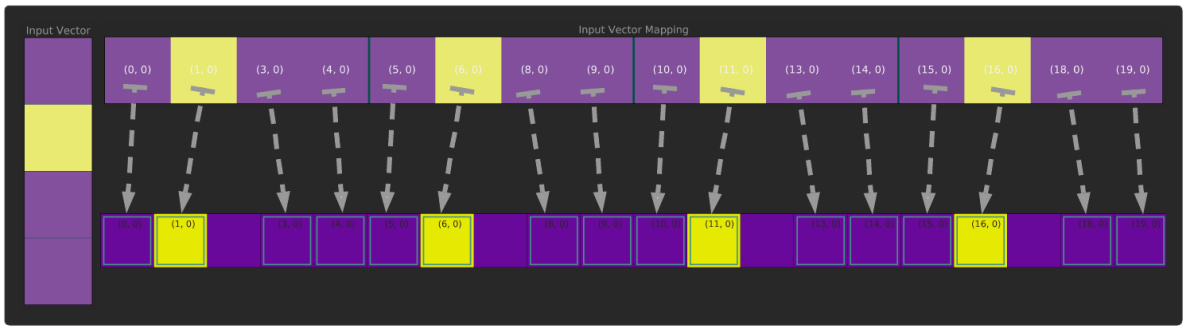
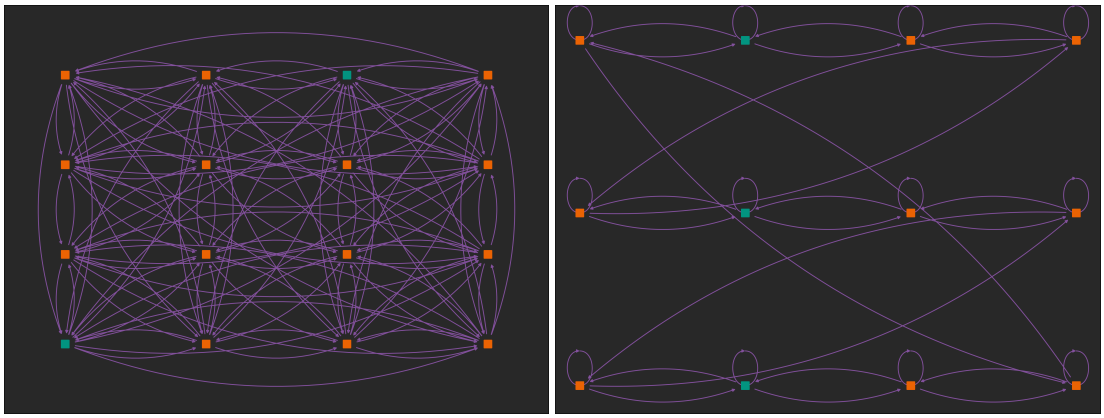


Figure 4.3: Input Mapping Scheme for the ECA rulespace



(a) Network graph depicting connectivity in Lifelike rules.

(b) Network graph depicting connectivity in ECA.

The Lifelike reservoirs were inspired by (Martinuzzi, 2022) and (Yilmaz, 2014), and primarily featured in experiments replicating those of the former. As in (Martinuzzi, 2022), the reservoir was not divided into sub-reservoirs, and R instead represented the width of the full reservoir. The shape of the reservoir was R^2, thus the total number of elements were $R \times R$. A few variations of reservoir shape were implemented, specifically in attempts to interpret (Yilmaz, 2014) architecture, but were not featured in substantial experiments.

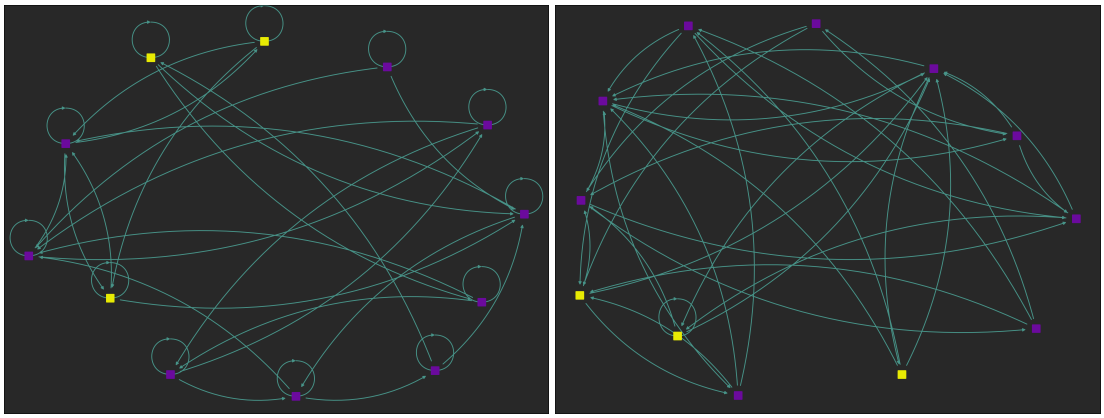
The ECA reservoirs were derived from (Glover et al., 2021) and featured the parameter L_D, which defined the length of the vector the input was mapped to. Essentially the length of each sub-reservoir, which was repeated R times. Therefore the total number of elements in the ECA reservoirs were defined by $R \times L_D$.

The connectivity in the two CA rulespaces that are explored in this work is local and homogeneous, as long as the grid is set up with periodic boundaries[1], which EvoDynamic does by default for CA. The Lifelike rules use the Moore neighbourhood structure and the ECA rules use a neighbourhood scheme consisting of the centre cell and their immediate left and right neighbours.

Both structures thus take into account every cell around the centre cell in a radius of one, however, the Lifelike rulespace relates to the self-state (centre cell) differently than ECA, usually by summing the states of the neighbours and checking the self-state after the fact. Where ECA instead factors in self-state in the state transition table as if it is connected to itself. Lifelike rules are totalistic in this sense, where the order of the neighbours doesn't matter, only the sum of the states. As a result, reservoirs with Lifelike rules had an in-degree of eight even though the neighbourhood size was technically $3 \times 3 = 9$, and ECA rules had an in-degree of three, counting the centre index cell.

The Lifelike rulespace was translated to RBN by generating a set of random connections while keeping the in-degree constant in the network. Because the Lifelike rules were totalistic,

[1]Connections wrap around at the edges.



(a) Network graph depicting RBN with a locked centre index.

(b) Network graph depicting RBN with a free centre index.

the order of the neighbours didn't matter, and the randomly generated edges could remain unweighted. However, when translating from the ECA rulespace, the state transition table required knowledge of the order of the neighbours, i.e., which nodes were to be regarded as the left, centre and right indices. This was done through weighting the edges in an *n-ary* fashion based on the number of possible states ($n = 2$ states in these rulespaces), as this was how the ECA state transition tables were implemented in (Pontes-Filho et al., 2020).

The ECA reservoirs were tested with two different variations in connectivity. The first is with a locked centre index, to more closely resemble ECA, and the other is with a free centre index, which is more RBN-like.

Output Layer

The reservoir feature vector consisted of the full list of reservoir states for a given input. This state vector was produced by concatenating the list of states for every reservoir timestep (not to be confused with the benchmark timestep) into a single longer list. The length of the reservoir state vector was in all cases defined by the parameter I, which also corresponds with how many reservoir iterations occur between inputs. The state of the reservoir persisted between inputs, until a new permutation begun.

The ECA models employed a development step introduced by (Glover et al., 2021), which added a reservoir timestep between the input and the recorded state vector. This was done primarily to prevent the output layer from directly accessing the input states, but it also allowed the reservoir dynamics to develop while keeping the features smaller in size. In the lifelike models, the list of states were flattened from two dimensions to one, and they did not employ the development step between inputs and recorded states in the experiments that were conducted.

The models implemented in this work comprised two different classifiers from the scikit-learn library.

1. Ridge Regression Classifier - a linear regression model where the loss function is the linear

least squares function and regularization is given by the l2-norm[2].

2. Support Vector Classifier with a linear kernel[3].

Though implemented in such a way that the two were interchangeable by passing a single parameter, Ridge regression was primarily used for Lifelike reservoirs and SVM was used for ECA reservoirs, for no other reason than an attempt to replicate previous methods.

4.2 X-bit Memory Benchmark

The experimental work in this thesis revolved around replicating known Lifelike and ECA reservoir architectures which had previously been tested on the X-bit benchmark. After validating the setup with CA, comparative runs were executed with RBN connectivity in order to compare the results between the two.

The task is simple by design, but is able to test long-short-term memory in the reservoir. The task is considered very difficult for standard feed-forward neural networks (Hochreiter & Schmidhuber, 1997), and thus serves to evaluate the *fading memory* and *separation property* of the reservoir in conjunction with the chosen linear model. It can be divided into four parts:

1. Input Pattern
2. Distractor Period
3. Cue Signal
4. Output Pattern

(1) The first steps define a pattern of bits according to the number of bits defined. If the bit in the first channel is set to one, the bit in the other channel must be zero, vice versa. The benchmark table shows input steps defined by a length of 5 bits, more specifically the number 25 in binary form. The two other input channels repeat zeros. The output of the reservoir is expected to be neutral during these first steps, i.e., the third channel should return one and the others zero.

(2) During this *neutral state*, a single value input is repeated to the reservoir for a number of timesteps defined as the distractor period. The purpose is to *distract* the reservoir's memorizing properties, thus the longer it is, the harder the task becomes. The output of the reservoir is expected to remain neutral for this period as well, identical to the input stream.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html`
[3] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC`

	Timestep	Input				Output		
		u_1	u_2	u_3	u_4	y_1	y_2	y_3
Input Pattern	1	1	0	0	0	0	0	1
	2	1	0	0	0	0	0	1
	3	0	1	0	0	0	0	1
	4	0	1	0	0	0	0	1
	5	1	0	0	0	0	0	1
Distractor Period	6	0	0	1	0	0	0	1
	7	0	0	1	0	0	0	1
	⋮			⋮			⋮	
	$T-7$	0	0	1	0	0	0	1
	$T-6$	0	0	1	0	0	0	1
Cue Signal	$T-5$	0	0	0	1	0	0	1
Output Pattern	$T-4$	0	0	1	0	1	0	0
	$T-3$	0	0	1	0	1	0	0
	$T-2$	0	0	1	0	0	1	0
	$T-1$	0	0	1	0	0	1	0
	T	0	0	1	0	1	0	0

Table 4.1: Benchmark timetable for inputs the size of 5 bits.

(3) On the final step of the distractor period, a cue signal is input to the reservoir, i.e., the fourth input channel is set to one, while the others input zero. The output is still expected to be neutral at this step as it marks the end of the distractor period.

(4) After the cue signal, the input is once again neutral, but the output channels should return the pattern defined in the first five steps.

Having defined the objectives of the task, one can see that it is analogous to a one-hot encoded classification task. The linear readout layer takes the reservoir state vectors and attempts to classify the feature into one of the three *classes* in the benchmark output channels.

Depending on the number of bits to solve, there are a set number of input permutations available. For five bits, the most popular choice, that number is $2^5 = 32$. For the benchmark to succeed, i.e., get a perfect score, it must correctly classify the state of the reservoir for every benchmark step and for every possible permutation. For $N_b = 5$ and $D_p = 200$, there are $2 \times 5 + 200 = 210$ timesteps per permutation, and $210 \times 32 = 6720$ steps to correctly classify in one full run of the 5-bit benchmark.

When considering that most of the benchmark is dominated by the neutral state in the output

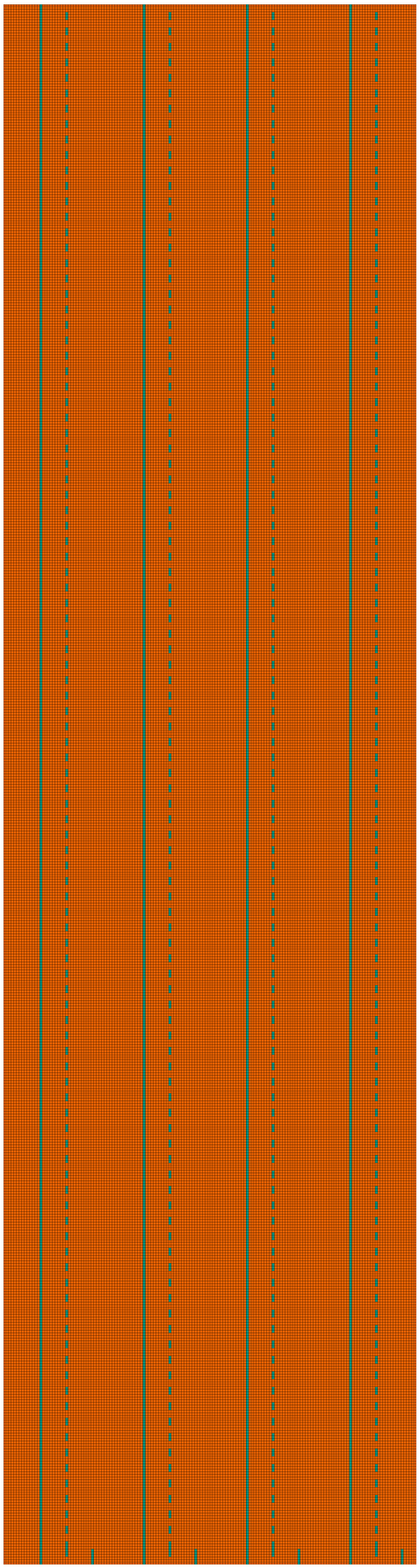


Figure 4.6: History plots of ECA Rule 204 with $N_b = 1$ and $D_p = 200$ clearly depict the input flow of an X-bit Memory Benchmark.

channels, it is evident that the crux of the challenge is retrieving the input pattern in the final X steps. Take for instance the 6720 total steps to correctly classify all permutations for $N_b = 5$ and $D_p = 200$. Of those, $6720 - (5 \times 32) = 6560$ steps might be considered trivial for most reservoirs.

If a reservoir manages to correctly classify the neutral state in the first five steps and the distractor period for all permutations, but fails to correctly retrieve *any* steps of the input pattern in the final five, the final score will be deceiving. $\frac{6560}{6720} \approx 0.9762$ is a seemingly high score with 1.0 being perfect.

(Glover et al., 2021) argued that this *perfect run* scoring system gave very little information about the reservoirs that only came close to solving the tasks. This work adapts the *Weighted Average* scoring system proposed by them, in order to compare accuracy between CA and RBN beyond the perfect runs. The weight in question is defined by the score fraction mentioned above, generalized in the following equation:

$$W = \frac{D_p + (N_b)}{D_p + (2 * N_b)} \tag{4.1}$$

Given that the fraction of correctly classified states is $\bar{S}_c$ the $\bar{W}$ can then be found using Equation 4.2.

$$\bar{W} = \frac{\bar{S}_c - W}{1 - W} \tag{4.2}$$

One might observe that if $\bar{S}_c$ is equal to W, the weighted average score is zero, while a perfect score is still 1.0.

Chapter 5

Results

This chapter describes the experiments that were executed, their configurations, and the following quantitative results, while shedding light on what qualitative observations could be made with respect to the research questions.

For all experiments recorded in this work, the distractor period D_p was set to 200. Every configuration of the benchmark was run 100 times to get a reliable percentage score. The weighted average takes the mean score from all 100 runs with that configuration as its $\bar{S}_c$ value.

5.1 Reservoir Computing in the Lifelike Rulespace

The pilot experiment with Lifelike reservoirs was executed with a reservoir architecture modelled after (Martinuzzi, 2022). The first objective was to validate the behaviour of a selection of rules tested by them with Lifelike CA, and run the same experiment with an equivalent RBN reservoir.

The input mapping strategy used was inspired by the concept of mapping binary data to random cells in the reservoir in (Yilmaz, 2014), however, the amount of cells that were mapped to was determined by an input projection ratio $P_r = 0.6$. This entailed 60% of all the cells in the reservoir would have their states overwritten by the input vector every I_{th} reservoir timestep.

It was observed in preliminary visual tests that the dynamics in Lifelike reservoirs which received more sparse input, quickly dissipated, and often died out completely before the next input arrived. Thus warranting the dense input for this particular architecture.

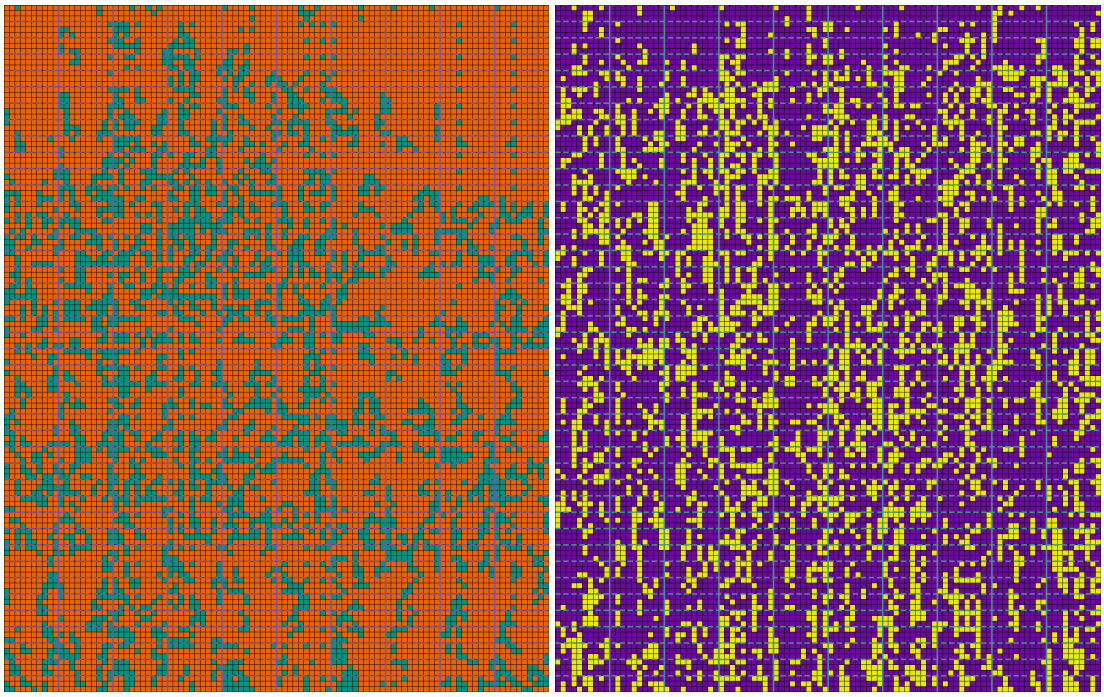
Due to the way EvoDynamic was integrated in the experiment code at the time, new random edges were generated for the RBN reservoirs between every permutation of the inputs. This may have affected the accuracy of the RBN reservoirs, and was rectified for the ECA reservoir and continued Lifelike reservoir experiments.

The rules were selected with validation in mind, with hopes of seeing clearly whether the models were the same, or at least similar enough, since their implementation was in the Julia programming language. The number of rules were a bit on the low side compared to later experiments, with respect to the computing power that was available at the time.

Model	CA		
(R, I)	(24, 8)	(28, 8)	(30, 6)
B3/S23	0	100%	63%
B35/S236	1%	100%	94%
B368/S12578	2%	100%	92%

Table 5.1: Results of the selected rules and reservoir configurations from (Martinuzzi, 2022).

Figure 5.1: Game of Life reservoir with $N_b = 1$ and $D_p = 10$. Flattened Lifelike reservoir representation like in figure 2.1



(a) CA (b) RBN

One thing to note in this experiment, is that *scikit-learn's* Ridge Classifier was used with $\alpha = 1.0$, whereas (Martinuzzi, 2022) had used a Ridge Regression model implemented in Julia with $\alpha = 0.001$, resulting in my model having a much stronger regularization parameter. This reduction in variance could explain the improvements in accuracy in my replication of their experiments. This work was based on an early access version of the paper, where not every single detail was explained, and the primary goal was to compare the following results to my RBN reservoirs.

In the CA validation test, which results can be seen in table 5.2, Game of Life (B3/S23) achieved 2% perfect runs with $R = 24$ and $I = 8$, marginally better than (Martinuzzi, 2022) with 0, seen in table 5.1.

B35/S236 and B368/S12578 both achieved much higher scores with 72% and 60%, respectively, which raised some concerns about discrepancies between the two reservoir

Model (R, I)	CA (24, 8)	RBN	CA (28, 8)	RBN	CA (30, 6)	RBN
B3/S23	2%	0	100%	0	91%	0
B35/S236	72%	0	100%	0	100%	0
B368/S12578	60%	48%	100%	100%	100%	100%
B356/S23	67%	0	100%	0	100%	0

Table 5.2: Results from the replicated reservoir architecture of (Martinuzzi, 2022) with the addition of Dynamic Life (B356/S23) (Peña & Sayama, 2021). Cross-referenced with the RBN reservoir architecture. (R, I) where R is the *reservoir width* and I is both the length of the reservoir feature vector and the number of iterations between inputs.

architectures. However, it became evident that the general behaviour of the rules were consistent when looking at the results for $R = 28$.

All three rules performed better with $R = 30$ and $I = 6$ than in (Martinuzzi, 2022) as well, with Game of Life achieving a score of 92%, up from 63%, the highest visible increase among the three rules.

Dynamic Life (B356/S23), put forward as a candidate of computational prowess in (Peña & Sayama, 2021), was tested, and achieved very similar scores to B35/S236. The two are quite close in the Lifelike rulespace, both allowing births on 3 or 5, and survival on 2 or 3, with the 6 switching from survival to birth in Dynamic Life.

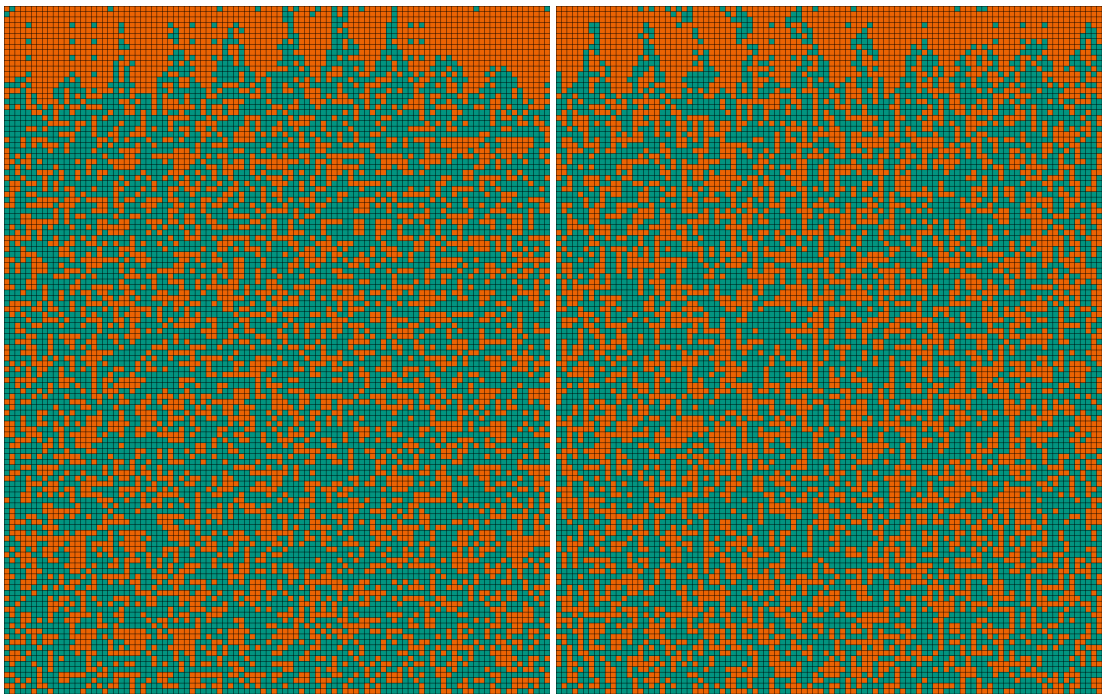
Figure 5.2 depicts the evolution of the two rules side-by-side. The dynamics of both rules fully enveloped in around the same time, and exhibited similar chaotic patterns, with Dynamic Life looking a little bit denser overall. The ramp-up time of reservoir activity took a lot longer for the RBN connectivity in figures 5.2c and 5.2d.

There were certain narrow regions in Dynamic Life that *stayed dead* for the duration of the experiment. B35/S236 showed similar tendencies, but had more variation in those narrow regions. The patterns looked similar overall. However, when compared to the behaviour with CA connectivity, the patterns in RBN seemed more *edgy* and rounded off like many CA patterns. Likely due to not enveloping locally, but rather popping up more sporadically.

Results for RBN reservoirs were poor across the board, with the exception of B368/S12578, which achieved results similar to its CA connectivity. With random initial conditions, observations indicated an intersection between Class II and Class III behaviour of this rule (Eppstein, 2010), which made it an apt candidate. It would seem like those traits carry over in part to the RBN connectivity (see figure 5.3).

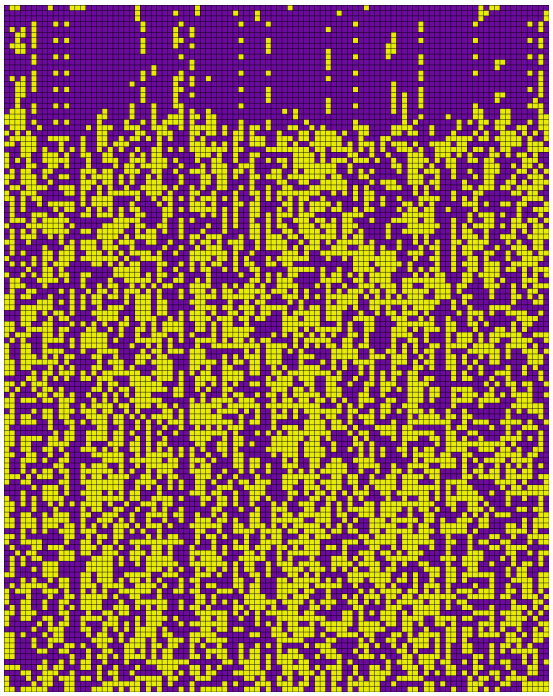
The poor accuracy of the Lifelike RBN reservoirs compared to CA may be correlated with the way patterns envelop with random connectivity, making it difficult to either detect the cue signal, or separate between different inputs. From visual observations, it was also evident that RBN reservoirs were generally more sensitive to the ratio of input projection P_r. Too few input

Figure 5.2: Comparison of reservoir dynamics with three similar rules for $N_b = 1$ and $D_p = 40$. $InputVector = [0, 1, 0, 0]$ and the mapping is inconsistent between the reservoirs. Flattened Lifelike reservoir representation like in figure 2.1

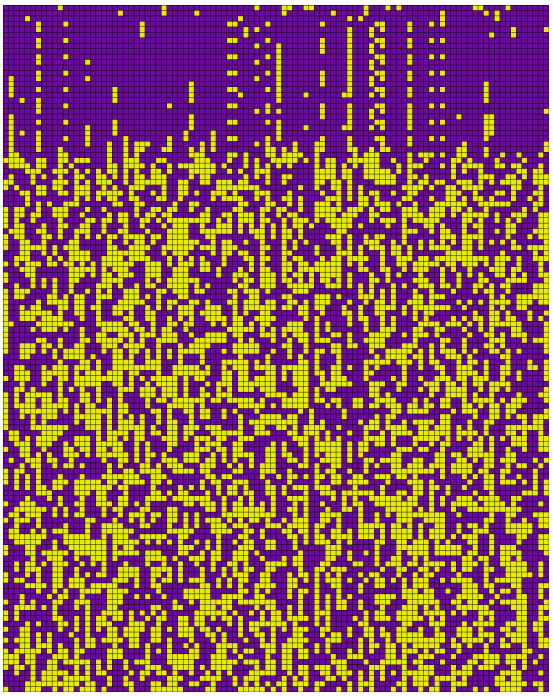


(a) Dynamic Life (B356/S23)

(b) B35S236

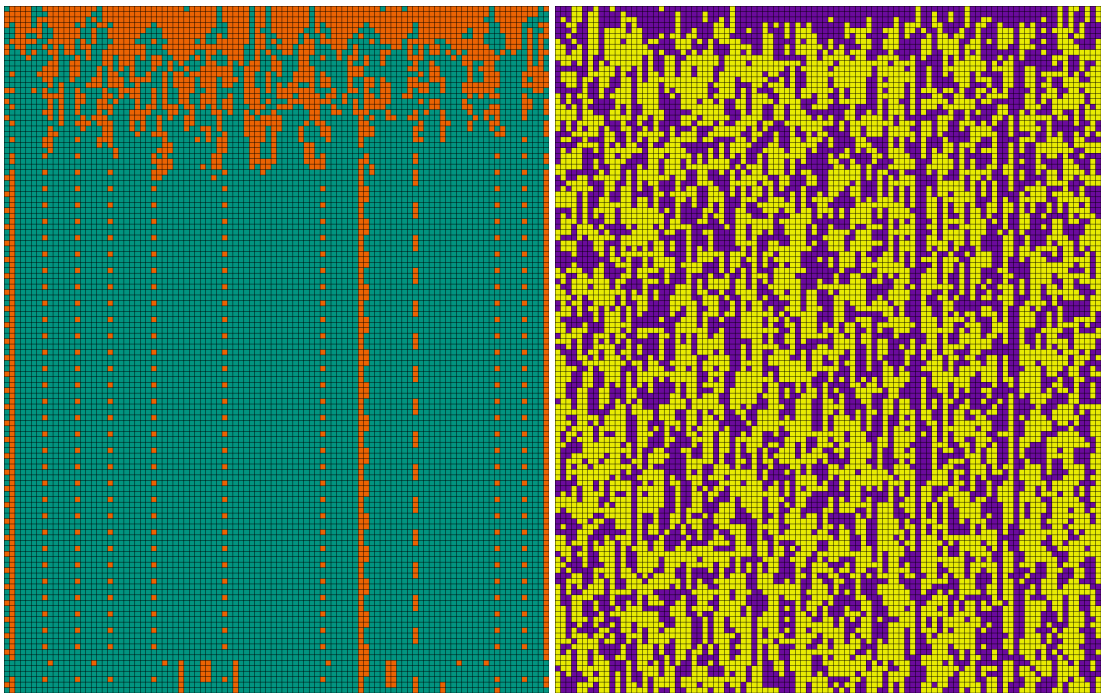


(c) Dynamic Life RBN



(d) B35S236 RBN

Figure 5.3: B368/S12578 reservoir with $N_b = 1$ and $D_p = 10$. Flattened Lifelike reservoir representation like in figure 2.1



(a) CA (b) RBN

mappings resulted in completely dissipated reservoir dynamics, or no build-up at all (like in figure 5.4), more often with RBN connectivity than CA.

5.2 Reservoir Computing in the ECA Rulespace

The second experimental phase involved running the Minimum Equivalence subset of the ECA rulespace with RBN reservoirs, and comparing the results to (Glover et al., 2021). The architecture for ECA reservoirs in these experiments were therefore based on theirs.

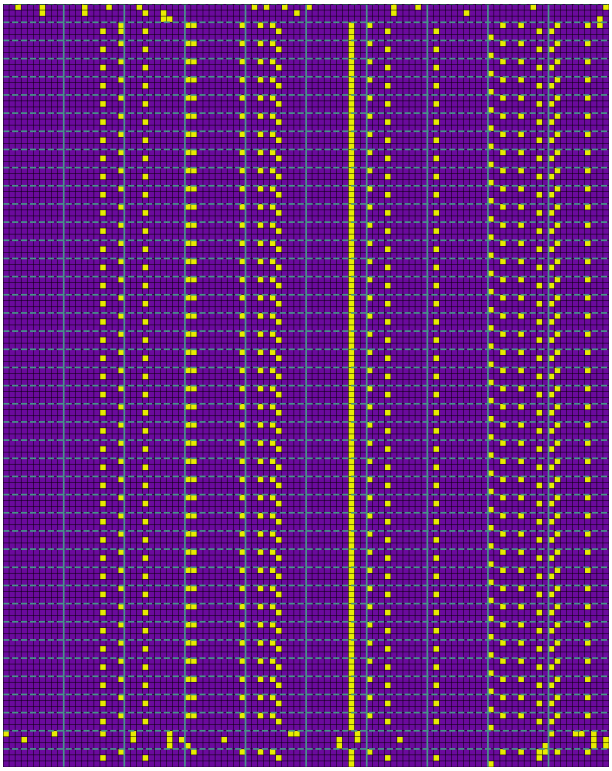
Notable differences from the Lifelike reservoirs were as follows:

- subreservoirs and the L_d parameter
- the input mapping strategy
- size of the reservoir and the reservoir feature

(Martinuzzi, 2022) moved away from subreservoirs in their work, but it was a part of (Yilmaz, 2014) ECA (and presumably Game of Life) reservoirs, and defined by the parameter L_d in many recent works with ReCA (Babson & Teuscher, 2019; Glover et al., 2021, 2022; Nichele & Gundersen, 2017; Nichele & Molund, 2017).

The total reservoir size in these experiments was defined by $R \times L_d$, which formed a long vector. For all experiments in the ECA rulespace, the reservoirs are defined by $R = 4$ and $L_d = 40$ resulting in 160 total length of the reservoir vector.

Figure 5.4: The reservoirs dynamics can quickly dissipate in Lifelike RBN reservoirs.



This was more in line with the traditional method of defining redundancy in the reservoir, by mapping the input vector once to each subreservoir. With an input vector of size four, the total number of input projections was 16, where four random cells in each subreservoir were projected on-to.

The resulting input projection ratio was 10%, significantly lower than 60% ($P_r = 0.6$) in the Lifelike reservoir experiments.

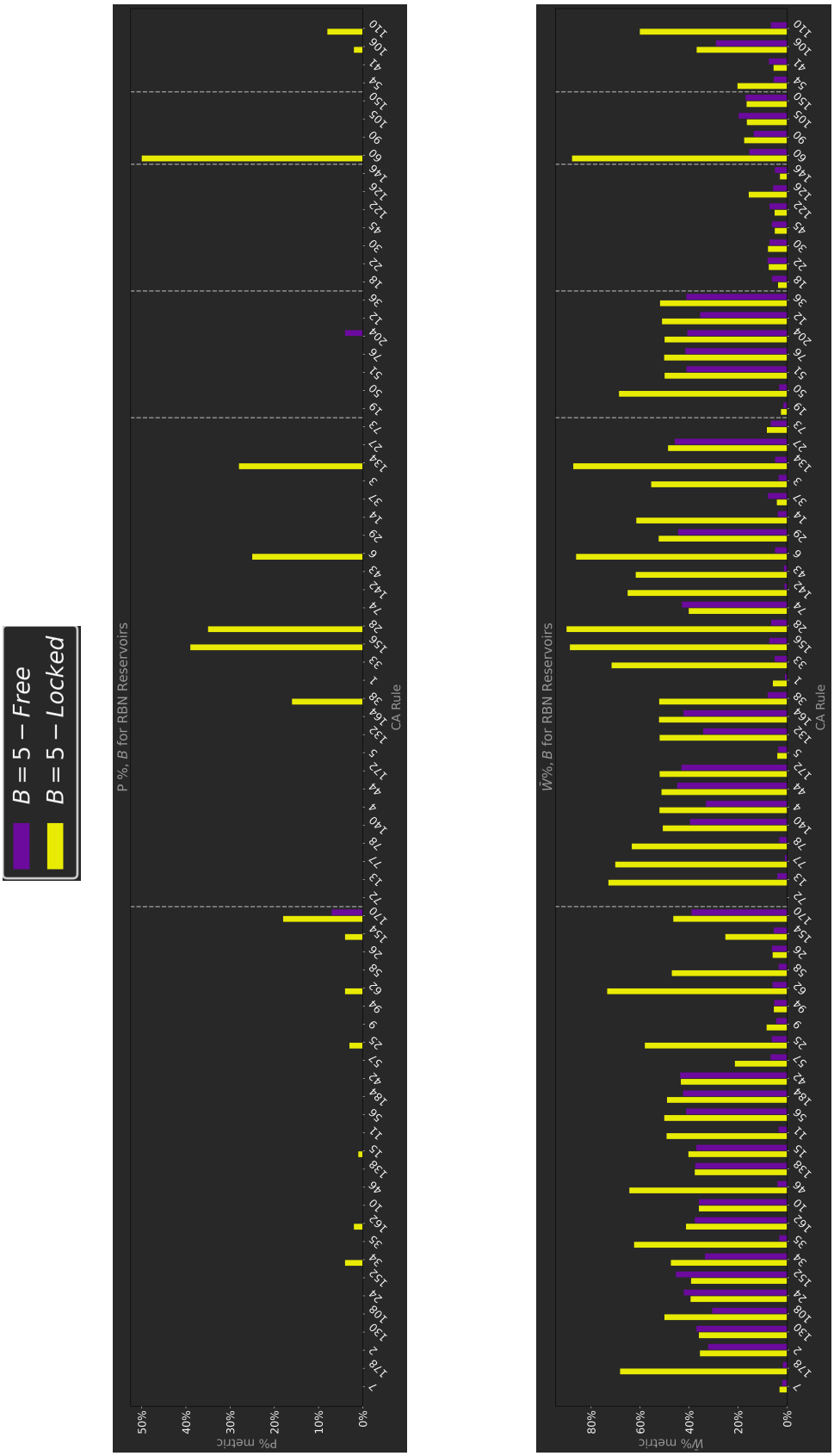
Instead of directly overwriting the current states of the reservoir with the input, and input projection method where the input was XOR-ed with the current cell state was used.

Both the size of the reservoir feature vector and the length of the benchmark in timesteps were much shorter with $I = 2$.

See the Methods chapter for a more detailed explanation of model differences.

In the validation tests, the results were initially slightly different to (Glover et al., 2021), seen in table 5.3, marked with an asterisk at the bottom. This was due to the development step between the input and the recorded state vector missing in the model. The results matched exactly those in the original paper once the development step was implemented. It might be interesting to note that rule 54 performed better without the development step, perhaps due to the SVM accessing the input directly in the first of the two state vectors that make up the reservoir feature. Then again, the very same phenomenon made rule 90 perform worse.

Figure 5.5: Comparison of results for $N_b = 5$ with different RBN connectivity.



(a) Perfect Score metric

(b) Weighted Average metric

Rule	170	204	30	90	54	110
CA	99	0	0	100	0	0
RBN Free	7	4	0	0	0	0
RBN Locked	18	0	0	0	0	8
CA*	99	0	0	83	5	0

Table 5.3: Results from the replicated reservoir architecture of (Glover et al., 2021). Cross-referenced with the RBN reservoir architecture. (R, I) where R is the *redundancy* parameter and I is the number of generations.
*Tests performed without the development step.

After validating the performance of the model with CA connectivity, the Minimum Equivalence subset of the ECA rulespace, with the exception of the rules that were left out by (Glover et al., 2021), was tested on the benchmark with $N_b = 2, 3, 4, 5$.

For the 5-bit benchmark, among the rules selected for validation, 170 and 204 were somewhat successful with the free RBN connectivity. In fact, those were the only successful rules among the 76 ECA rules that were run on the 5-bit benchmark with that connectivity. The behaviour of the two rules is analyzed further down in this section. Rule 170 achieved higher accuracy in the locked RBN connectivity, where rule 110 also had better results than the CA.

Rules that had any number of perfect runs on the 5-bit benchmark with RBN connectivity include 34, 15, 25, 62, 154, 170, 38, 156, 28, 6, 134, 204, 60, 106 and 110, in the order that they appear in figure 5.5a.

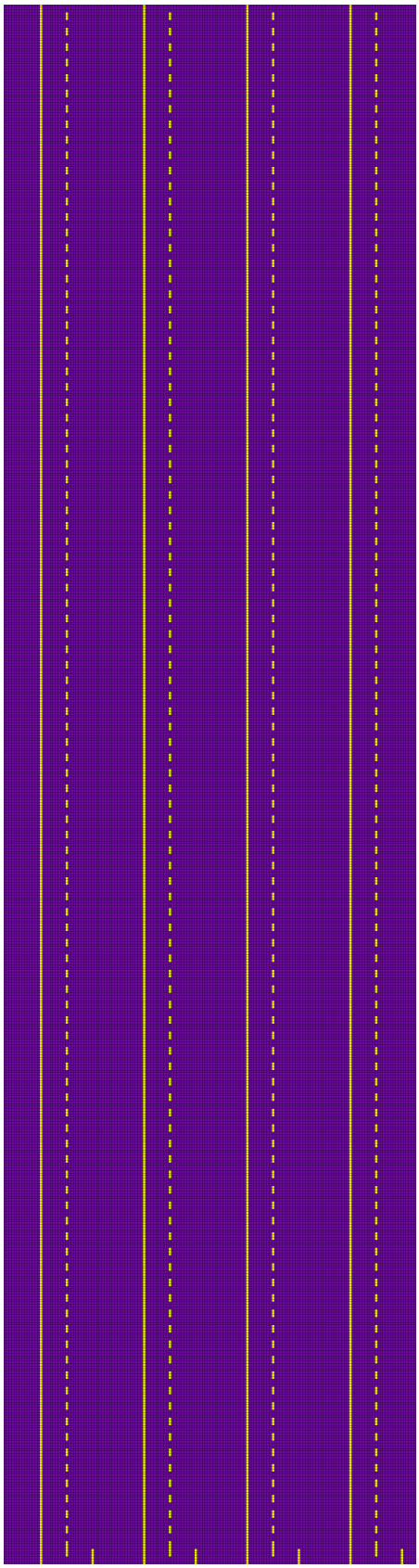
The only rule that utterly and completely failed on the 5-bit benchmark was rule 72. Every other rule managed to get at least some percentages with the weighted average scoring method. Moderately performing rules with the free RBN connectivity, with a weighted average score of 40% or more, include 24, 152, 56, 184, 42, 44, 172, 164, 74, 29, 27, 51, 76, 204 and 36, where none of them achieved 50% or better (see figure 5.5b). Rules performed better with the locked RBN connectivity across the board, with a few exceptions. Rules achieving a 70% weighted average or higher include 62, 13, 77, 33, 156, 28, 6, 134 and 60.

Rule 204 was explained to possess a certain *guessing* stochastic property in (Glover et al., 2021), by clearly indicating the cue signal. It was never able to produce a perfect run for any N_b between 2 and 5, and the weighted average was always a perfect 50% out of 100 runs.

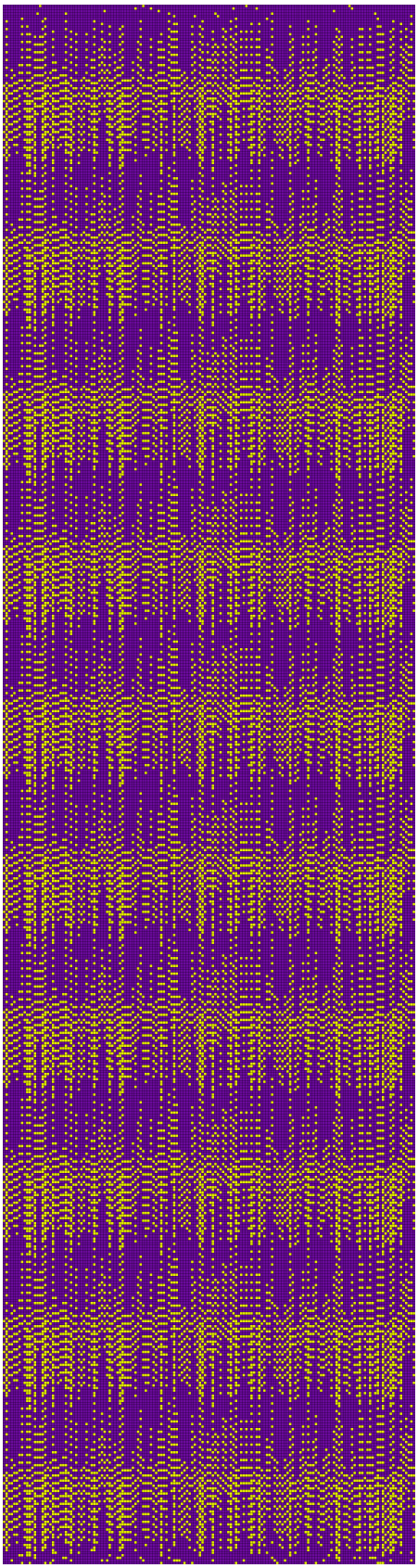
It was expected that the locked RBN connectivity could reproduce this behaviour exactly due to the centre index remaining fixed. The centre index is the only factor in the rule's state transition table, perhaps best explained by Additive CA in table 2.2. (This very fact was used in validating the method of creating RBN in the ECA rulespace.)

Figures 5.9a and 5.9b show for every N_b that the number of perfect runs was zero and the

Figure 5.6: Comparison between ECA Rule 204 in RBN with locked and free centre indices. See figure 4.6 for a reference to the behaviour with ECA connectivity.



(a) Locked RBN connectivity.



(b) Free RBN connectivity.

weighted average was 50%, confirming that the rule's behaviour with locked RBN connectivity was consistent with that of CA. However, looking at figures 5.8a and 5.8b, one may observe that the rule behaved very differently with a free RBN connectivity. For $N_b = 5$, it achieved some perfect runs with a lower weighted average, meaning it lost its stochastic property.

Both the number of perfect runs and weighted average score increased when N_b decreased. In fact, its behaviour was consistent with that of rule 170. As seen in table 2.2, the state transition table of rule 170, much like that of rule 204, is only affected by one of the connected cells. When the centre index was freed, and no longer forced to be a self-connection, the two rules in effect became equivalent.

However, it is interesting to note that by moving away from the local connectivity of the CA, the sideways projecting behaviour of rule 170 was lost, and it went from being a top performer on the benchmark (Glover et al., 2021) to medium at best.

Rule 60 got 100% perfect runs with $N_b = 5$ in (Glover et al., 2021). As seen in figure 5.5a, the performance of the rule jumped down closer to 50% perfect runs with a locked RBN connectivity. Furthermore, the performance of the rule dipped down to 0 perfect runs for the free RBN connectivity. Figure 5.5b shows that the weighted average score of rule 60 was above 80% and below 20% for free and locked RBN, respectively. Figure 5.9a shows that the rule's performance varied greatly with N_b with locked RBN connectivity, as it achieved a higher percentage of perfect runs with $N_b = 4$ than both $N_b = 3$ and $N_b = 5$. Simultaneously, one can observe in figure 5.8a that the rule performed very well for all N_b except $N_b = 5$ with the free RBN connectivity. See figure 5.7 for a comparison of the different reservoir behaviours with rule 60.

The bar plots in figure 5.8 and 5.9 show that many rules that didn't have any success with $N_b = 5$, immediately managed a lot better with $N_b = 4$. A small portion of the rules that didn't do better with $N_b = 4$ found traction by reducing N_b further.

(Yilmaz, 2014) obeserved that there was a polynomial increase in the minimum required reservoir size with respect to N_b. (Glover et al., 2021) also pointed out that rules could be highly sensitive to a number of parameters, especially the likes of L_d.

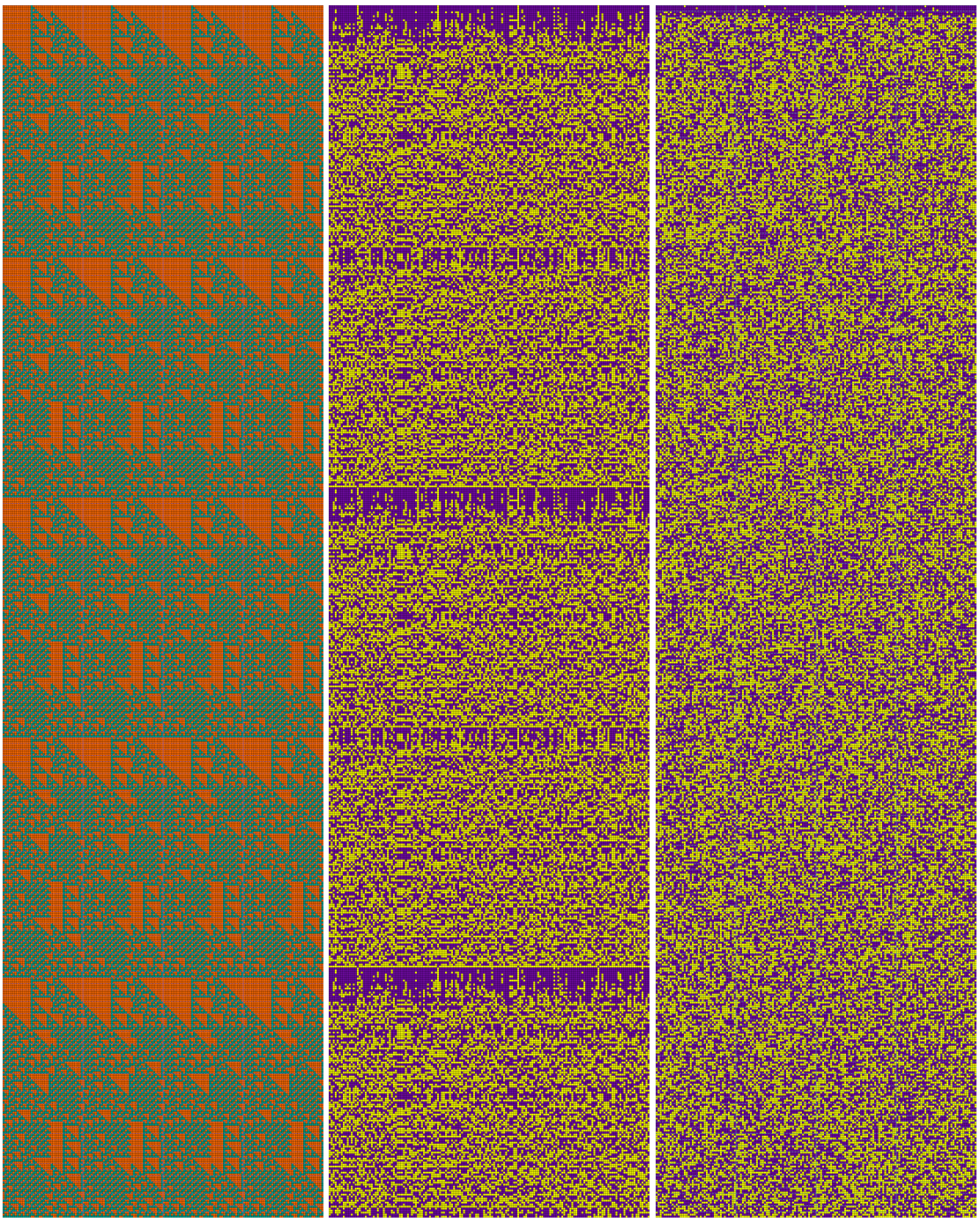
This indicates that there exists larger reservoirs with combinations of R and L_d that could make up a perfect set conditions for many of the ECA rules, even with RBN connectivity.

However, there were a number of rules that didn't even do well with $N_b = 2$, falling below 50% perfect runs. With the free RBN connectivity, these include 7, 2, 130, 24, 152, 34, 162, 10, 138, 15, 56, 184, 42, 170, 140, 4, 44, 172, 5, 132, 164, 1, 74, 29, 3, 27, 19, 51, 76, 204, 12, and 36.

With the locked RBN connectivity, they were 7, 2, 130, 108, 24, 152, 34, 162, 10, 138, 15, 11, 56, 184, 42, 170, 72, 13, 77, 78, 140, 4, 44, 172, 5, 132, 164, 1, 33, 74, 142, 43, 29, 14, 3, 27, 19, 50, 51, 76, 204, 12, 36, and 106.

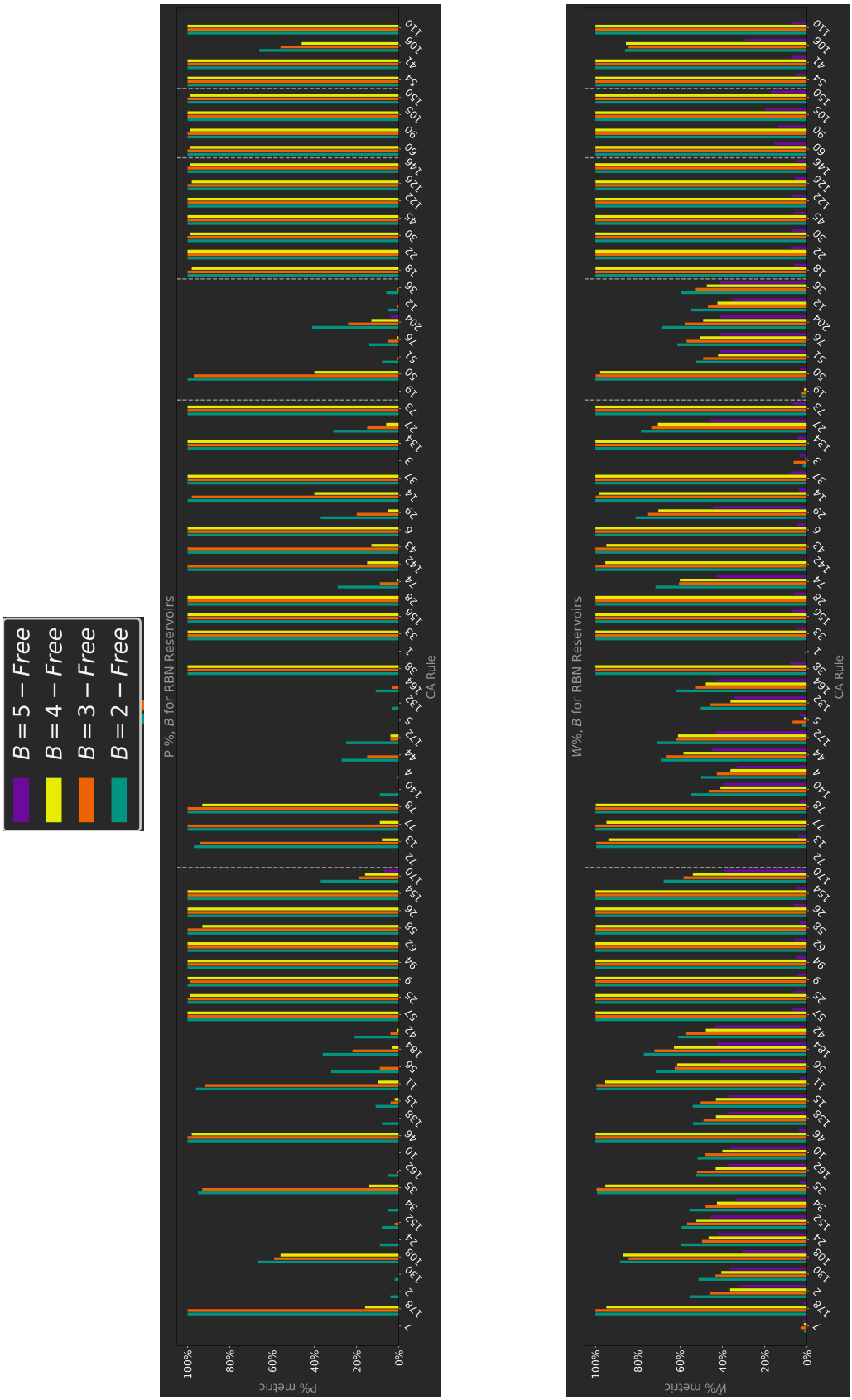
The 12 Minimum Equivalence rules that were left out in (Glover et al., 2021) were also tested on the 5-bit benchmark with this experiment configuration, and achieved abysmal results for both CA and RBN. None of the rules 0, 8, 23, 32, 40, 104, 128, 136, 160, 168, 200 or 232 managed to

Figure 5.7: History plots of rule 60 with $N_b = 1$ and $D_p = 200$ with different connectivity.



(a) CA (b) RBN Locked (c) RBN Free

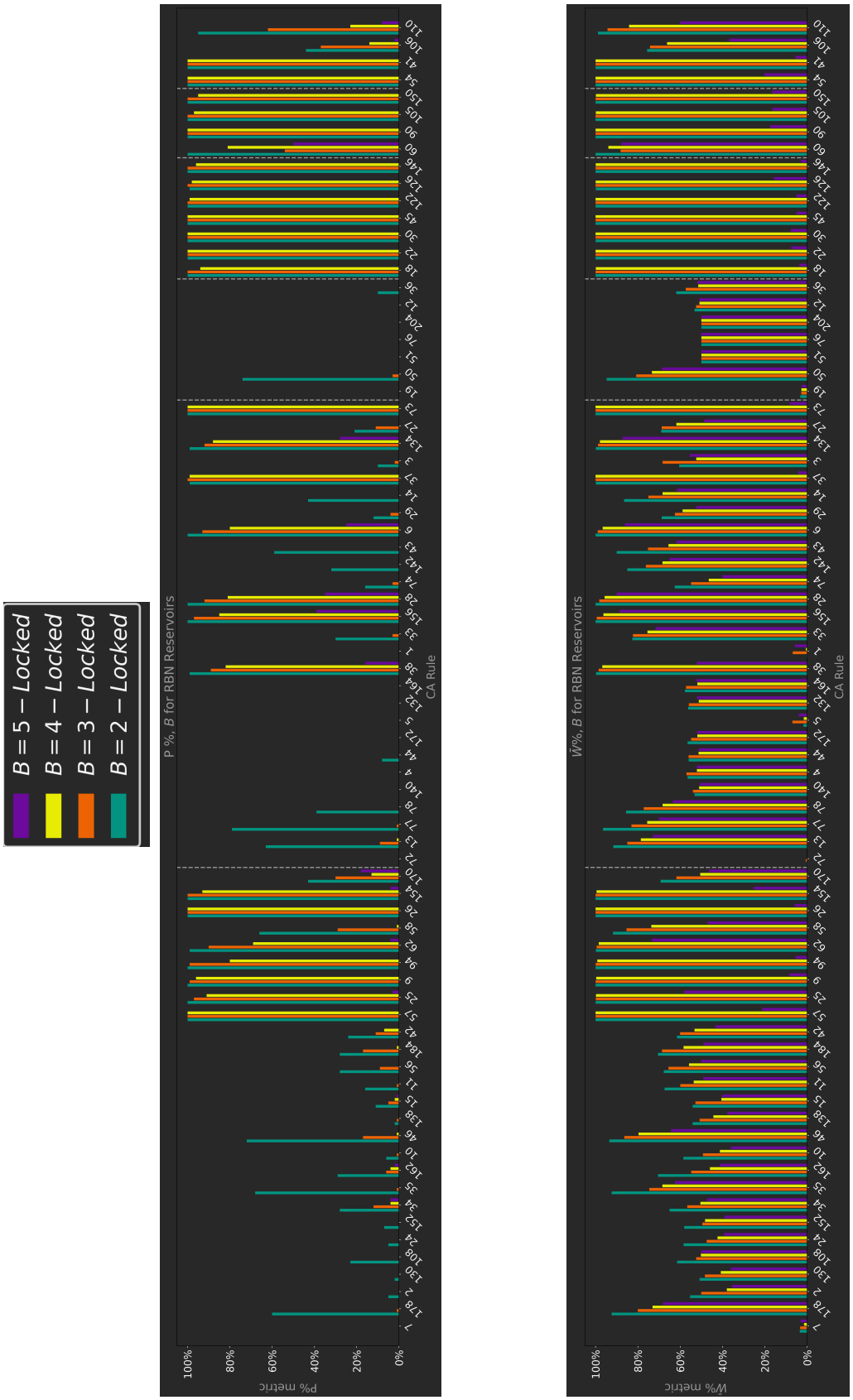
Figure 5.8: Comparison of results for $N_b = 2, 3, 4, 5$ with free RBN connectivity.



(a) Perfect Score metric

(b) Weighted Average metric

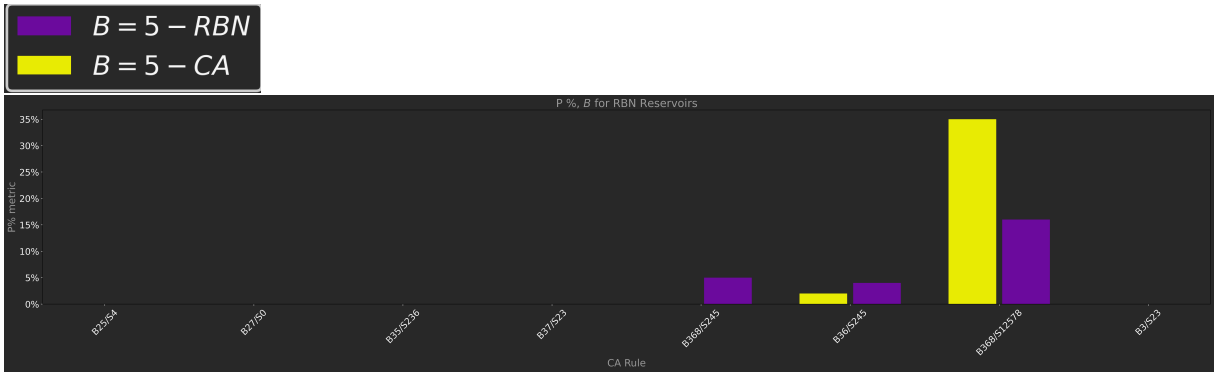
Figure 5.9: Comparison of results for $N_b = 2, 3, 4, 5$ with locked RBN connectivity.



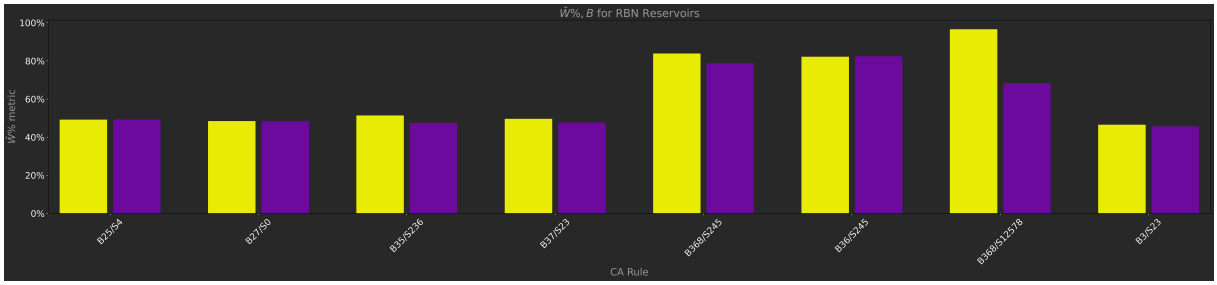
(a) Perfect Score metric

(b) Weighted Average metric

Figure 5.10: Comparison of Continued Lifelike results for $N_b = 5$.



(a) Perfect Score metric



(b) Weighted Average metric

get a weighted average score of 1% with neither CA nor RBN connectivity, hence they were not pursued further.

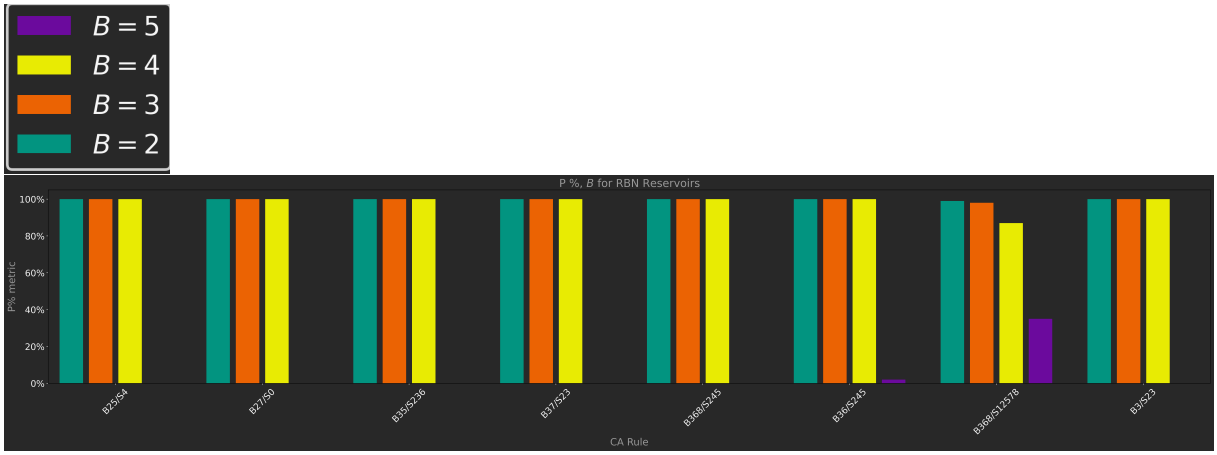
5.3 Continued Lifelike Experiments

Following the exploration of the ECA rulespace, a more comprehensive experiment configuration with Lifelike reservoirs was designed to more closely resemble the ECA reservoirs in size and computation. It was intended to have approximately as many total elements as the ECA reservoirs with reservoir the shape being R^2 and $R = 14$, resulting in 196 to ECA's 160. $R = 12$ would have been a bit short at 144, and $R = 13$ would have resulted in an odd number of elements at 169.

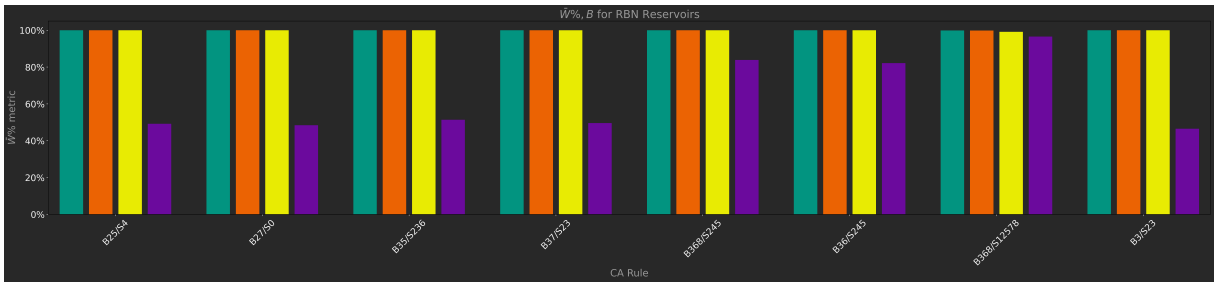
The rules featured in (Martinuzzi, 2022) were further explored with a fixed reservoir configuration. $I = 2$ and the development step was retained from the ECA experiments, and he XOR input projection method was incorporated. The projection ratio was increased to 90% to counteract quickly dissipating dynamics ($P_r = 0.9$). Subreservoirs were not included and input projections were randomly mapped to the whole reservoir. Support Vector Machine with a linear kernel was used in place of the Ridge Classifier. The reservoir connectivity (the edges) was retained between each input permutation of one run, contrary to the Lifelike experiments in which new edges were generated between each permutation.

The results for $N_b = 5$, seen in figure 5.10, showed promise with three rules managing some perfect runs. Rules B368/S245 and B36/S245 notably achieved more perfect runs with the RBN

Figure 5.11: Comparison of results for $N_b = 2, 3, 4, 5$ with CA connectivity.



(a) Perfect Score metric



(b) Weighted Average metric

connectivity than with CA. 5% and 4%, respectively for RBN. 0 and 2% with CA. However, the weighted average was even between the two, slightly favouring CA.

B368/S12578 repeated its success from section 5.1, achieving over 15% perfect runs with RBN connectivity and over 35% with CA. From figure 5.10b, one can discern that the rules was close to solving solving the 5-bit benchmark every run with CA connectivity, closing in on 100% weighted average. The score was around 70% with RBN connectivity.

The figure also shows that five other rules (that haven't been mentioned) got close to 50% weighted average score. One could not safely rule out that these reservoirs were somehow wildly guessing the bit patterns after recognizing the cue signal, but overall the results seemed promising enough to warrant further experiments with $N_b = 4, 3, 2$.

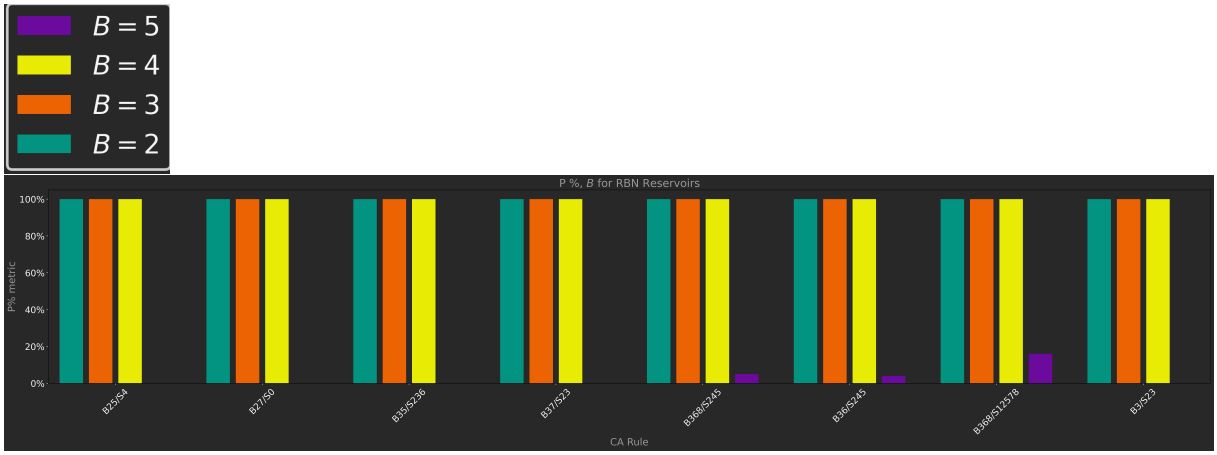
Figures 5.11 and 5.12 show that all rules for both connectivities achieved perfect or near-perfect results across the board for $N_b = 4, 3, 2$.

Ironically, only B368/S12578 displayed a performance weaker than average on the perfect run metric with CA connectivity, $N_b = 4$ and $N_b = 3$, perhaps demonstrating a more generalisable behaviour.

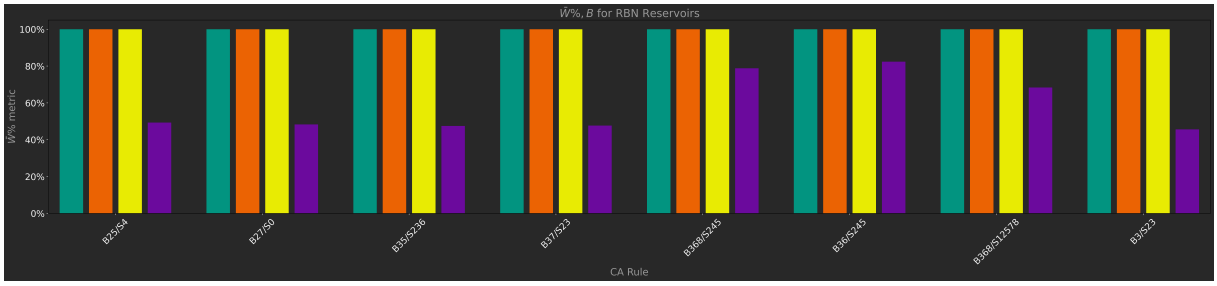
On the back of these results, an experiment with a bigger reservoir was executed for $N_b = 5$, comparable to that of the experiments in section 5.1, with $R = 28$. All other parameters were the same as above.

The results - nothing short of perfect, shown in figure 5.13, were surprisingly good,

Figure 5.12: Comparison of results for $N_b = 2, 3, 4, 5$ with RBN connectivity.

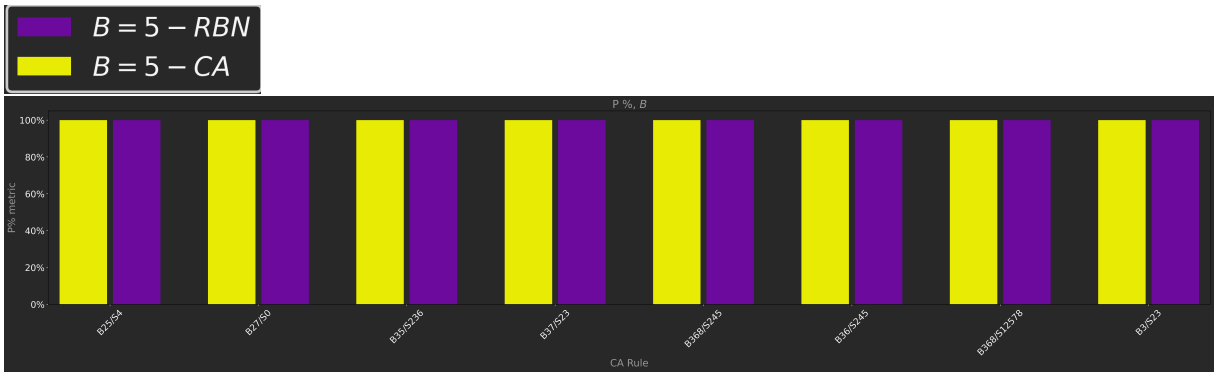


(a) Perfect Score metric

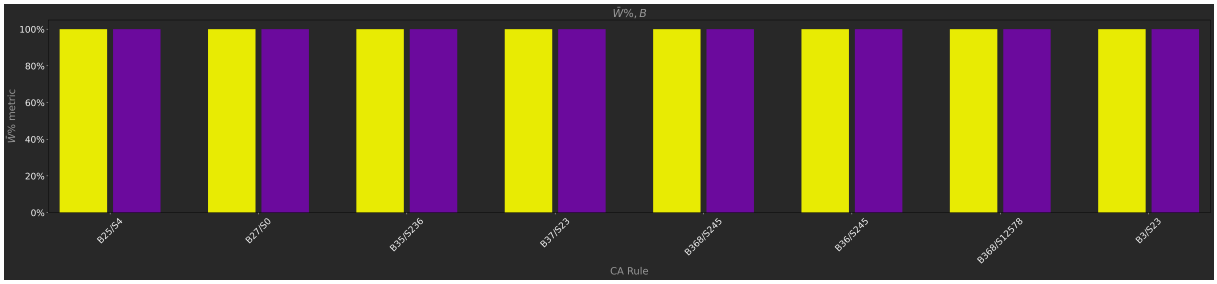


(b) Weighted Average metric

Figure 5.13: Comparison of CA and RBN Lifelike reservoirs with $R = 28$.



(a) Perfect Score metric



(b) Weighted Average metric

considering most of the rules had struggled in bigger reservoirs and with bigger reservoir feature vectors in (Martinuzzi, 2022).

Major reservoir factors could include the increased input projection ratio $P_r = 0.9$ and the presence of the development step. Using SVM with a linear kernel instead of Ridge Regression in the output layer could also have improved the accuracy directly.

$I = 2$ could also be a sweet spot for both the reservoir feature and the number of iterations between inputs. Results of experiments with the I parameter in (Glover et al., 2022) indicated clear dynamical behaviour in certain rules when varying the parameter, and that certain rules might have a *preference* of sorts towards even or odd numbered I.

Perhaps a bit of a stretch in this particular case, but it appears that $I = 2$ might be preferred over $I = 8$ regardless.

5.4 Yilmaz tests

Assumptions were made regarding the architecture required to replicate some of the experiments in (Yilmaz, 2014). The reservoir architecture in this thesis was modified to be able to create a reservoir of a size that was proportionate to the input vector. From the wording in the paper alone it seemed like creating a reservoir with gridsize $InputVector \times R$ would suffice, however, when taking into account the figures describing the architecture $InputVector^2 \times R$ appeared more likely, where each sub-reservoir would be the size of size $InputVector^2$.

It first seemed like 100% of the reservoir was perturbed and it was unclear whether the author used any other method than strictly overwriting. The method of XOR-ing the input with the state of the reservoir was implemented at the time in order to comply with the experimental setup in (Glover et al., 2021) regardless, and was tested with this input mapping strategy. Taking the figures describing the input projection and reservoir architecture into account on this issue as well, it was possible that a mere 25% of the reservoir was perturbed, given an $InputVector$ randomly projected into R number of squared sub-reservoirs. However, brief testing showed that such low perturbation caused reservoir dynamics to quickly die out when using the Game of Life rule.

The exact combination of parameters remained unclear after careful consideration and implementation, but a few experiments were conducted to verify results from certain (R, I) combinations and compare with a RBN reservoir using the same rules. The results of those experiments were nowhere near what was expected, and I concluded that the method was not reproducible in a realistic timeframe.

Chapter 6

Discussion

This chapter attempts to tie together the work that's been done, from establishing the problem area and researching the state-of-the-art, to implementing the method and generating the results.

6.1 The Results

The results of the X-bit Memory Benchmark can hopefully help us answer the research questions:

1. How do innate differences in connectivity between CA and RBN with similar rules affect RC accuracy?

2. Does the effect of varying connectivity increase between one-dimensional and two-dimensional CA compared to RBN with similar rules?

Overall, the results speak largely in favour of CA over RBN when measuring the accuracy of the reservoirs on the X-bit Memory Benchmark. Beneath follows a discussion on some of the reasons why this might be.

The Minimum Equivalence subset of ECA was likely not an optimal choice of rules to use for RBN, but it's what was available to compare with. The original intent was to search the entire ECA rulespace, because it was understood that with a different connectivity, new and interesting behaviour could occur in any of the rules. However, bugs persisted in the ECA reservoir model long enough that it was no longer viable with respect to the time-constraints to run the full experiment for CA as well as the two RBN variations.

The Wolfram Classification of the rules did in general not seem to extend to the RBN connectivity. Especially noticeable in the ECA rulespace and for the free RBN, which was somewhat expected. However, it seems like there could be an overweight of chaotic (III) and homogeneous (I) behaviour. Clearly distinguishable cyclical patterns (II) and complex (IV) behaviour were seldom observed in history plots.

The in-degree of RBN in the ECA rulespace was $K = 3$ for all rules where all three indices were a factor in the state transition table. However, there exists a set of ECA rules that disregard one and only one of the three indices, thereby effectively giving the RBN an in-degree of $K = 2$, essentially becoming elementary versions of elementary rules, $Elementary^2$ if you will. As seen in table 2.2, rules 60, 90 and 102 are among those rules.

This also means that many of the rules in the ME subset that was tested realistically have in-degrees of 1. Ironically, rules 170 and 204 are among those rules, and were the only ones to achieve success among ECA rules on the 5-bit Memory Benchmark.

$Elementary^2$	Rules
Left Exclusive	17, 34, 68, 102, 119, 136, 153, 187, 221, 238
Right Exclusive	3, 12, 48, 60, 63, 192, 195, 207, 243, 252
Middle Exclusive	5, 10, 80, 90, 95, 160, 165, 175, 245, 250

Table 6.1: Rules that exclude one and only one cell in their state transition tables. Filtered by listing the left, right and center exclusive rules found in this calculation Glover (n.d.), and extracting the rules that are exclusive to each list.

The full list of $Elementary^2$ rules can be seen in 6.1

The XOR input projection method was not scrutinized before being applied to the experiments in this thesis. It didn't matter that much when replicating an existing model, but it may have had unknown effects on the final Lifelike experiments.

Consider table 6.2, and that as per the code used in (Glover et al., 2021), the current state in cells that are chosen to projected on are given as argument A and the input bit from the input vector is given as argument B.

The next state in the cell is set set to zero if the input and the current state are the same. If the input differs from the current state in either way, the state is set to one.

The edges being updated between every permutation in the first Lifelike experiments essentially made the RBN reservoir lose its deterministic character inside each experiment. Even though the connections were random, as long as they persisted, the behaviour would be deterministic in the synchronous RBN implemented in this work.

A		B	Result
0	$\oplus$	0	0
0	$\oplus$	1	1
1	$\oplus$	0	1
1	$\oplus$	1	0

Table 6.2: XOR Truth Table

It would be unfair to use these results to say that reservoirs with RBN connectivity perform worse than CA, even though that's seemingly true for the X-bit Memory Benchmark. The accuracy scores in the experiments weren't intended as a competition between CA and RBN, but rather as a tool to discern something about the difference in behaviour and aptness. It would be safe to say that the behaviour changes between the connectivities, and more so between CA and free RBN than between CA and locked RBN. Perhaps with changed behaviour comes different applications - and needs.

(Burkow, 2015) claimed that the optimal perturbation of homogeneous RBN with in-degree $K = 3$ was around 50%.

However, for the Lifelike RBN where $K = 8$, even 60% perturbation seemed to be on the low side for the 5-bit Memory Benchmark, while a perturbation of 90% may have contributed to the good results in the final experiments.

In the ECA experiments, the perturbation was static at one mapping of the input vector to each subreservoir, resulting in a perturbation of $16/160 = 10\%$.

Seeing as (Snyder et al., 2012) warned that the level perturbation should be taken into account when finding the optimal values of $\langle K \rangle$ in heterogeneous RBN, perturbation might also be a factor in affecting the behaviour of rules, much like ECA with Memory, making them more prone to either order or chaos.

In the case of the ECA rulespace, it could also be that non-totalistic rules create an environment in RBN in which one cannot rely on discoveries from totalistic RBN.

The final Lifelike experiments achieved good results across the board. Better than many of the configurations in (Martinuzzi, 2022). A full table of their results is included in the appendix.

This success may be caused by the lower value of I or the higher projection ratio, or perhaps a combination of the two. I has presented a bit of a paradox in the Lifelike rulespace. Higher values increase the size of the reservoir features, and may provide better grounds for separating different input patterns. Simultaneously, higher values of I mean more iterations between inputs, potentially allowing reservoir dynamics dissipate, especially if the perturbation is low. This could in turn make *fading memory* into no memory at all.

A potential way to curb these effects is to let reservoir substrates mature before perturbing them, i.e., let the CA or RBN run for a set amount of time with random initial conditions before starting the benchmark timestep zero.

6.2 The Process and Limitations

There were perhaps few papers on RBN researched in this thesis. Hence, it is important to emphasize that this work is done with the basis of CA approximating RBN with as few steps away from CA as possible. Graph theory and boolean state transition tables, along with broader recent works in RBN RC, weren't deemed important enough to this end.

The I parameter in (Glover et al., 2021) experimental setup was misunderstood at first. They used the input step as a CA development step making $I = 2$ give 3 CA steps per input, whereas my original implementation had 2 CA steps per input. I also determined the length of the state vector for each input of course, which was correctly implemented. One bug persisted for the re-booted RBN run, where the input delay in the EvoDynamic experiment was out of sync with the number of CA iterations between inputs. Fixing this further delayed the experiments by approximately nine hours.

(Glover et al., 2021) ran the X-bit benchmark with $N_b = 3, 4, 5, 6$, whereas this work ran it with $N_b = 2, 3, 4, 5$. After starting with $N_b = 5$, and seeing the unconvincing performance of RBN for the reservoir configuration, I decided not to attempt a higher number of bits, but rather see if any interesting discoveries could be made searching down to $N_b = 2$.

EvoDynamic features a method of generating RBN adjacency matrix for use with ECA state transition tables. This method couldn't be used with the Lifelike activation functions in EvoDynamic, and the edge generation method in this work had to be created.

At first this was done by generating a regular grid adjacency matrix for a Lifelike/Game of Life CA of the same size, and shuffling the indices of the resulting sparse matrix. It was discovered that the method to create RBN connections by shuffling this way often produced duplicate edges, which were ignored when inspecting the connections as graphs in NetworkX.

Thus a method of generating the edges from scratch was created.

When moving on to the ECA rulespace, the RBN method inside EvoDynamic could probably have been employed. However, based on the in-degree detection function in NetworkX, it seemed to generate a heterogeneous network[1] with an average in-degree of 3.

It was later discovered that this perceived heterogeneous connectivity might have been due to the weighted edges employed by EvoDynamic when using the ECA transition table. Nevertheless, in order to compare to ECA as closely as possible, I chose instead to use the edge generation method that was created for the Lifelike rulespace to create a homogeneous network[2].

It was unfortunately not properly adapted to the way ECA is implemented in EvoDynamic, and caused a number of bugs and frustration for several weeks. The maintainer of the library eventually clarified some of the necessary properties of the adjacency matrix and the problems were resolved just over a week before the submission deadline. The solution was evident in (Pontes-Filho et al., 2020) and, with a bit of explaining, seemed clear as day.

Essentially the RBN that had been created previously in the ECA rulespace (Lifelike was correctly implemented) treated every connection and as its left neighbour, *and* the direction of the edges was reversed (correctly directed for Lifelike). Minor confusing inconsistencies between the implementation of the two rulespaces in EvoDynamic, a few misunderstandings and too few assertion tests when switching from one rulespace to the other, caused a seemingly

[1] The connectivity varies between nodes.

[2] The connectivity is of the same degree between nodes.

endless stream of invalid data.

Another choice to make in designing the RBN connectivity for the ECA rulespace was whether to force self-connections in the node's centre index akin to the CA, or let it be formed by an edge between any other node in the reservoir. The fundamental intention was to have the RBN model resemble CA closely as possible, as a small step towards biological neural networks. At first, the free RBN connectivity was chosen, because of misunderstanding how ECA was implemented in EvoDynamic. Therefore, after understanding the different approaches to the Lifelike and ECA rulespaces in EvoDynamic, the locked centre index approach was chosen as the baseline - for being more closely aligned with the CA. However, due to a recent increase in computational power, the same experiments were executed for RBN with a free centre index in order to compare across the different variations.

Covid-19 should by all accounts be mentioned as a limiting factor in the thesis work, especially since the author contracted the disease in the middle of the process and work was delayed on all fronts for around a month.

Chapter 7

Conclusion

7.1 Summary

In this thesis, Reservoir Computing models with Cellular Automata and Random Boolean Networks as substrates were implemented. The Lifelike and Elementary CA rulespaces were explored on the X-bit Memory Benchmark with both substrates.

Efforts were made to discover how the innate differences in connectivity between CA and RBN would affect the performance of the reservoirs, and whether further exploration of RBN RC in conjunction with ReCA could bridge the gap between CA and biological neural networks as a substrate.

The RBN reservoirs used in this thesis showed tendencies to perform worse than CA for much of the explored rulespace, and the specific reservoir configurations tested on the X-bit Memory Benchmark. Though it's worth noting that we've only scratched the surface of rulespaces that can be compared between the models, and barely even searched the parameterspace of either the models or the benchmark.

However, some rules showed interesting new behaviour, and some achieved results on par with CA. The results suggest that further exploration the parameterspaces and rulespaces of RBN RC in tandem with ReCA could facilitate the use of CA as an interface to biological substrates like BNN.

7.2 Future Work

Herein follows some closing thoughts on which paths that ought to be tread further down the line.

The rest of the ECA rulespace should be explored in a similar fashion to what has been done in this thesis.

Efforts should be made to group the ECA rulespace with RBN connectivity by Wolfram Classifications and the Lambda parameter. One might see a very different distribution of classes for the rulespace with either connectivity. Based on observations made in this thesis, one could

theorize that the Edge of Chaos is more narrow than in the equivalent rulespace with CA.

Attempts should be made to discern what rules make up the Minimum Equivalence set in the ECA rulespace with RBN connectivity.

It would be interesting to explore Non-Uniform CA and RBN - where rules vary between reservoirs, or perhaps even cells. By combining different rules in the ECA rulespace, this would in effect result in heterogeneous connectivity with average in-degrees between one and three if the rules varied between cells.

Explore totalistic CA rules with neighbourhoods of $N = 2, 3$, where the order neighbours doesn't matter, which are more RBN-like.

Since an in-degree between two and three is considered to be within the realm of criticality for heterogeneous RBN, it could prove valuable to compare with a non-discrete average number of connections between 2 and 3 in CA as well. These musings raise a particular curiosity about e, the *euler* number, or the natural number.

Explore "deep", layered reservoirs. If the ultimate goal is to use CA as an interface to biological neural networks as reservoirs, starting with a layered reservoir with CA -> RBN might be a good idea.

In this regard, one might also explore asynchronous RBN as one step further away from CA towards a biological substrate, and compare the differences between a synchronous CA and an asynchronous RBN.

Synchronous state update is an important characteristic of CA, however nodes of networks in instances of biological intelligence, like neural networks and DNA, update their states asynchronously (Gershenson, 2004).

Neurons develop connections over time, trained by a plethora of different tasks in the real world. An idea could be that substrates are aged or developed in several areas of use, and paired with different, similar, or even the same, readout layers.

References

1.4. support vector machines. (n.d.). scikit-learn developers. Retrieved from https://scikit-learn.org/stable/modules/svm.html#svm-kernels

Additive cellular automaton. (n.d.). mathworld.wolfram.com. Retrieved from https://mathworld.wolfram.com/AdditiveCellularAutomaton.html (Accessed: 2022-05-10)

Babson, N. & Teuscher, C. (2019). Reservoir computing with complex cellular automata. *Complex Systems*, *28*(3). Retrieved from https://doi.org/10.25088/ComplexSystems.28.4.433 doi: doi:10.25088/ComplexSystems.28.4.433

Bak, P., Tang, C. & Wiesenfeld, K. (1987). Self-organized criticality: An explanation of the 1/f noise. *Physical review letters*, *59*(4), 381.

Bertschinger, N. & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, *16*(7), 1413–1436.

Burkow, A. V. (2015). *Evolving functionally equivalent reservoirs for rbn reservoir computing systems.* Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway. Retrieved from https://burkow.no/uploads/forprosjekt-report.pdf

Burkow, A. V. (2016). *Exploring physical reservoir computing using random boolean networks.* (Master's thesis, NTNU, Trondheim, Norway). Retrieved from http://hdl.handle.net/11250/2417596

Chui, K. T., Lytras, M. D. & Visvizi, A. (2018). Energy sustainability in smart cities: Artificial intelligence, smart monitoring, and optimization of energy consumption. *Energies*, *11*(11), 2869. doi: doi:10.1016/j.giq.2018.05.002

Conway, J. et al. (1970). The game of life. *Scientific American*, *223*(4), 4.

Cook, M. et al. (2004). Universality in elementary cellular automata. *Complex systems*, *15*(1), 1–40.

Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S. & Zomaya, A. Y. (2020). Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal*, *7*(8), 7457–7469.

Desa & Nations, U. (2016). Transforming our world, the 2030 agenda for sustainable development.

Eppstein, D. (2010). Growth and decay in life-like cellular automata. In *Game of life cellular automata* (pp. 71–97). Springer.

Fernando, C. & Sojakka, S. (2003). Pattern recognition in a bucket. In W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich & J. T. Kim (Eds.), *Advances in artificial life* (pp. 588–597). Berlin, Heidelberg: Springer Berlin Heidelberg.

Gershenson, C. (2004). *Introduction to random boolean networks.*

Glover, T. E. (n.d.). *Elementary elementary ca proof.* GitHub. Retrieved from `https://github.com/ethol/DeepCA---Hybrid-Deep-Learning-Cellular-Automata-Reservoir/blob/master/ReCA%20parameter%20space/even_more_elementary_elementary_CA.py`

Glover, T. E., Lind, P., Yazidi, A., Osipov, E. & Nichele, S. (2021, 07). *The Dynamical Landscape of Reservoir Computing with Elementary Cellular Automata* (Vol. ALIFE 2021: The 2021 Conference on Artificial Life). Retrieved from `https://doi.org/10.1162/isal_a_00440` (102) doi: doi:10.1162/isal_a_00440

Glover, T. E., Lind, P., Yazidi, A., Osipov, E. & Nichele, S. (2022). *Patterns within patterns: A deeper look at the dynamical landscape of reservoir computing with cellular automata.* Department of Computer Science, OsloMet. (Early access to a pre-print version of the paper.)

Goal 11, take urgent action to combat climate change and its impacts. (n.d.). United Nations. Retrieved from `https://sdgs.un.org/goals/goal11`

Goal 7, ensure access to affordable, reliable, sustainable and modern energy for all. (n.d.). United Nations. Retrieved from `https://sdgs.un.org/goals/goal7`

Goal 9, build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation. (n.d.). United Nations. Retrieved from `https://sdgs.un.org/goals/goal9`

Hassan, N., Yau, K.-L. A. & Wu, C. (2019). Edge computing in 5g: A review. *IEEE Access*, *7*, 127276–127289.

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

Jaeger, H. (2001). 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*, *148*(34), 13.

Johnston, N. & Greene, D. (2022). *Conway's game of life: Mathematics and construction*. Self-published. Retrieved from `https://conwaylife.com/book` doi: doi:10.5281/zenodo.6097284

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, *22*(3), 437–467.

Khan, S., Paul, D., Momtahan, P. & Aloqaily, M. (2018). Artificial intelligence framework for smart city microgrids: State of the art, challenges, and opportunities. In *2018 third international conference on fog and mobile edge computing (fmec)* (p. 283-288). doi: doi:10.1109/FMEC.2018.8364080

Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, *42*(1), 12-37. Retrieved from `https://www.sciencedirect.com/science/article/pii/016727899090064V` doi: doi:https://doi.org/10.1016/0167-2789(90)90064-V

Liang, D., Hashimoto, M. & Awano, H. (2021). Bloomca: A memory efficient reservoir computing hardware implementation using cellular automata and ensemble bloom filter. In *2021 design, automation test in europe conference exhibition (date)* (p. 587-590). doi: doi:10.23919/DATE51398.2021.9474047

Life-like cellular automata. (n.d.). conwaylife.com. Retrieved from `https://conwaylife.com/wiki/Cellular_automaton#Life-like_cellular_automata`

Maass, W., Natschläger, T. & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

Martinez, G. J., Adamatzky, A. & Alonso-Sanz, R. (2013). Designing complex dynamics in cellular automata with memory. *International Journal of Bifurcation and Chaos*, *23*(10), 1330035.

Martinuzzi, F. (2020). *Gsoc week 8: Reservoir computing with cellular automata part 2.* Retrieved from `https://martinuzzifrancesco.github.io/posts/08_gsoc_week/`

Martinuzzi, F. (2022). *Life-like cellular automata as a substrate for computation.* Remote Sensing Centre for Earth System Research, and Center for Scalable Data Analytics and Artificial Intelligence, Leipzig University. (Early access to a pre-print version of the paper.)

McDonald, N. (2017). Reservoir computing and extreme learning machines using pairs of cellular automata rules. In *2017 international joint conference on neural networks (ijcnn)* (p. 2429-2436). doi: doi:10.1109/IJCNN.2017.7966151

Morán, A., Frasser, C. F. & Rosselló, J. L. (2018). *Reservoir computing hardware with cellular automata.*

Muhammad, K., Lloret, J. & Baik, S. W. (2019). Intelligent and energy-efficient data prioritization in green smart cities: Current challenges and future directions. *IEEE Communications Magazine*, *57*(2), 60-65. doi: doi:10.1109/MCOM.2018.1800371

Neumann, J. v. (1966). Theory of self-reproducing automata. *Edited by Arthur W. Burks*.

Nichele, S. & Gundersen, M. S. (2017). Reservoir computing using non-uniform binary cellular automata. *arXiv preprint arXiv:1702.03812*.

Nichele, S. & Molund, A. (2017). Deep learning with cellular automaton-based reservoir computing. *Complex Systems*, *26*(2). Retrieved from `https://doi.org/10.25088/ComplexSystems.26.4.319` doi: doi:10.25088/ComplexSystems.26.4.319

Packard, N. H. & Wolfram, S. (1985). Two-dimensional cellular automata. *Journal of Statistical physics*, *38*(5), 901–946.

Peña, E. & Sayama, H. (2021). Life worth mentioning: Complexity in life-like cellular automata. *Artificial Life*, *27*(2), 105–112.

Pontes-Filho, S., Lind, P., Yazidi, A., Zhang, J., Hammer, H., Mello, G. B., ... Nichele, S. (2020). A neuro-inspired general framework for the evolution of stochastic dynamical systems: Cellular automata, random boolean networks and echo state networks towards criticality. *Cognitive Neurodynamics*, 1–18.

Sayama, H. (2015). *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks.

Siegelmann, H. T. & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of computer and system sciences*, *50*(1), 132–150.

Snyder, D., Goudarzi, A. & Teuscher, C. (2012). Finding optimal random boolean networks for reservoir computing. In *Alife 2012: The thirteenth international conference on the synthesis and simulation of living systems* (pp. 259–266).

Snyder, D., Goudarzi, A. & Teuscher, C. (2013, Apr). Computational capabilities of random automata networks for reservoir computing. *Physical Review E*, *87*(4). Retrieved from `http://dx.doi.org/10.1103/PhysRevE.87.042808` doi: doi:10.1103/physreve.87.042808

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, *10*(1-2), 1–35.

Yilmaz, Ö. (2014). Reservoir computing using cellular automata. *CoRR*, *abs/1410.0162*. Retrieved from `http://arxiv.org/abs/1410.0162`

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K. & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, *107*(8), 1738–1762.

Chapter 8

Appendix

Class I	Rules
Strong	128
Moderate	8, 32, 40, 136, 160, 168
Weak	0

Table 8.1: Class 1 Elementary CA with Memory classifications

Class II	Rules
Strong	2, 7, 9, 10, 11, 15, 24, 25, 26, 34, 35, 42, 46, 56, 57, 58, 62, 94, 108, 130, 138, 152, 154, 162, 170, 178, 184
Moderate	1, 3, 4, 5, 6, 13, 14, 27, 28, 29, 33, 37, 38, 43, 44, 72, 73, 74, 77, 78, 104, 132, 134, 140, 142, 156, 164, 172
Weak	12, 19, 23, 36, 50, 51, 76, 200, 204, 232

Table 8.2: Class 2 Elementary CA with Memory classifications

Class III	Rules
Strong	18, 22, 30, 45, 122, 126, 146
Moderate	
Weak	60, 90 ,105, 150

Table 8.3: Class 3 Elementary CA with Memory classifications

Class IV	Rules
Strong	41, 54, 106, 110
Moderate	
Weak	

Table 8.4: Class 4 Elementary CA with Memory classifications

Table 8.5: Results (Martinuzzi, 2022) Lifelike CA on the 5-bit Memory Benchmark. The results are reported as percentage of perfect runs out of 100 runs.

	(24,6)	(24,8)	(24,10)	(24,12)	(26,6)	(26,8)	(26,10)	(26,12)	(28,6)	(28,8)	(28,10)	(28,12)	(30,6)	(30,8)	(30,10)	(30,12)
B25/S4	0	0	100	100	0	97	100	100	0	100	100	100	82	100	100	100
B27/S	0	0	100	100	0	96	100	100	0	100	100	100	83	100	100	100
B35/S236	0	1	100	100	0	100	100	100	4	100	100	100	94	100	100	100
B37/S23	0	0	89	99	0	65	100	100	2	100	100	100	72	100	100	100
B368/S245	0	2	0	1	0	2	3	2	2	19	15	6	17	51	31	6
B36/S245	0	0	1	2	0	2	10	3	2	25	22	5	20	61	41	13
B368/S12578	0	2	100	100	0	97	100	100	3	100	100	100	92	100	100	100
B3/S23	0	0	85	99	0	51	100	100	3	100	100	100	63	100	100	100