Abstract

Reservoir computing has become a predominant member of the unconventional computing paradigm. It is a framework suited for processing of temporal and sequential data, traditionally using recurrent neural network models to incorporate past inputs into an instantaneous readout.

Interestingly, there is no need for the reservoir to be an artificial neural network – any high-dimensional, driven system exhibiting complex dynamic behavior can be used. A wide range of physical substrates have been proposed as reservoir machines, ranging from nanomagnetic assemblies to living cultures of neurons.

A major challenge when realizing physical reservoirs is the physical limitations present in the underlying substrate. This is in contrast to abstract model reservoirs, e.g. echo state networks, which have no physical constraints in regards to dimensionality, spatial layout and observability. In this thesis, we investigate reservoirs with realistic dimensional and spatial properties, constraining the possibility of making structural changes. Initially, we conduct experiments with echo state networks that consist of random geometric graphs, the simplest spatial network model. We then further this work with lattice structures, which are highly regular architectures, and are common in computational physics.

Results show that spatial constraints by default inhibit the NARMA-10 benchmark performance of both models. However, introducing directed edges to the network instead of bidirectional ones restores performance to compete with established models, indicating that the flow of information is an important property in quality reservoirs.

Furthermore, simple square lattice reservoirs with a fixed, global input are found to perform as well as echo state networks on NARMA-10 and Mackey-Glass benchmarks. The value of regular, deterministic structures as a tool for theoretical analysis is evaluated, giving examples of methodology to explore the inner workings of networks when solving specific tasks.

Sammendrag

Reservoarberegning har blitt et fremtredende medlem av paradigmet for ukonvensjonell dataprosessering. Dette er et rammeverk velegnet for prosessering av tidsmessige og sekvensielle data, tradisjonelt ved bruk av rekurrente nevrale nettverk, som gjør tidligere input tilgjengelig som en umiddelbar avlesning.

Det er dog ikke nødvendig at reservoaret er et kunstig nevralt nett – ethvert høydimensjonalt, drevet system som innehar kompleks, dynamisk oppførsel kan brukes. Et bredt spekter av fysiske substrater er foreslått som reservoarmaskiner, fra nanomagnetiske ensembler til levende nevronkulturer.

En stor utfordring under realisering av fysiske reservoarer er substratets fysiske begrensninger. Dette er i kontrast til abstrakte reservoarer, f.eks. tilfeldige rekurrente nevrale nettverk, som ikke har fysiske begrensninger med hensyn til dimensjonalitet, romlig utforming og observerbarhet. I denne oppgaven undersøkes reservoarer med realistiske dimensjonelle og romlige egenskaper, hvor muligheten for å gjøre strukturelle endringer er begrenset. I utgangspunktet gjennomfører vi eksperimenter med tilfeldige rekurrente nevrale nettverk som består av tilfeldige geometriske grafer, som er den enkleste modellen for romlige nettverk. Eksperimentene videreføres med gitterstrukturer, som er høyst regulære arkitekturer, og er vanlige i numerisk fysikk.

Resultater viser at romlige begrensninger hemmer reservoarers ytelse under ytelsestesten NARMA-10 for begge modeller. Dersom rettede kanter introduseres i nettverkene istedenfor toveis kanter, vil ytelsen kunne konkurrere med etablerte modeller, noe som indikerer at informasjonsflyt er en viktig egenskap i gode reservoarer.

Videre viser det seg at reservoarer basert på firkantede gitter med fast, global input yter like godt som tilfeldige rekurrente nettverk på NARMA-10 og Mackey-Glass ytelsestester. Verdien i regulære, deterministiske strukturer som et verktøy for teoretisk analyse evalueres, og det gis eksempler for å utforske hvordan nettverk oppfører seg når de løser spesifikke oppgaver.

Preface

I would like to thank my supervisors, PhD candidate Johannes Høydahl Jensen and professor Gunnar Tufte, for helpful feedback and encouragement. It has been a pleasure.

I would also like to thank PhD candidate Peter Aaser for kindly introducing me to the world of complex systems. It has been a stimulating endeavor, and has become an interest that will last me a lifetime.

Contents

Chapter 1

Introduction

With the inevitable demise of Moore's law, researchers now seek computational methods beyond the traditional transistor-based computer architecture. A wide range of approaches, collectively dubbed *unconventional computing* methodology, aim to exploit the intrinsic computation present in many natural systems. For example, *evolution in-materio* shows that computation can be implemented in physical systems as a hybrid of analogue and traditional computation [1].

Reservoir computing (RC) has become a predominant member of the unconventional computing paradigm [2]. It is a framework suited for processing of temporal and sequential data, exploiting the underlying dynamics of a *reservoir*. Classical RC is derived from recurrent neural network (RNN) models, e.g. echo state networks [3]. Utilizing an RNN as a reservoir, input sequences are projected into a high-dimensional space, incorporating its temporal information in an instantaneous readout. Training is then carried out by adapting the readout layer with supervised linear regression, providing faster and simpler training than traditional gradient descent methods.

Interestingly, there is no need for the reservoir to be an artificial neural network – any high-dimensional, driven system exhibiting complex dynamic behavior can be used [4]. Through a fusion of in-materio computation and the reservoir methodology, *physical* reservoir computing has seen a recent surge of interest [2, 5].

In this work, the goal is to explore inherent limitations faced when exploiting physical substrates as reservoirs. In the project preceding this thesis, noise, equipment accuracy, and system observability were investigated as physical limitations [6]. Herein, we further this work, focusing on constrained topology or physical morphology. Physical reservoirs must necessarily be embedded in physical space, often having completely fixed structural properties. This is in contrast to abstraction

models – physical models are commonly limited in the way we may change its geometry. The role of structure is a relatively unexplored area of RC.

1.1 Research Goals

In this thesis, we seek a better theoretical foundation for how reservoir computing methodology translates to physical substrates. Specifically, we are interested in understanding how spatially restricting the nodes of a reservoir network impacts performance. Ultimately, the goal of the thesis is to answer the following research questions:

RQ1: How does the reservoir computing paradigm translate to the spatially constrained topology setting of a physical medium?

First, we are concerned with investigating practical challenges of realizing physical reservoirs. We are interested in how restrictions, primarily embedding reservoirs in physical space, will affect reservoir quality.

RQ2: How do highly regular, physical structures compare in information processing capability to that of established models such as echo state networks?

Second, we are interested in how reservoirs with regular structures, such as lattice grids, compare to established models. If there are discrepancies in capability, we pursue the reasoning to gain a deeper theoretical insight into why the regularity is disadvantageous.

RQ3: Can we find simple, deterministic reservoir generation methodology, relying less on random weighting schemes?

Designing reservoirs in a highly deterministic manner is desirable, especially due to the fact that it may simplify the physical realization process. Simple schemes to embed nodes in space and establish connectivity are thus beneficial. Additionally, less stochastic elements in reservoir generation may allow us to peer into the "black box" character of reservoir computing.

To answer the posed research questions, we conduct simulations using traditional echo state network methodology. We investigate two types of spatially constrained network models as reservoirs: random geometric graphs and lattices. Thus, since all experiments use echo state networks, we use a higher level abstraction to model physical reservoirs by imposing specific structural properties on the architecture. Generated networks are evaluated using widely used approaches: the nonlinear autoregressive moving average (NARMA) [7], the Mackey-Glass delay differential equation [8], kernel quality and generalization [9], and short-term memory [10].

The goals of this thesis are closely related to the work conducted by the SOCRATES[1] and SpinENGINE[2] projects. Both projects explore the suitability of artificial spin ice for massively parallel data analysis. Artificial spin ice consists of nanomagnets arranged on a two-dimensional lattice, and has shown promise as a physical reservoir system [11, 12].

1.2 Thesis Overview

The thesis is structured as follows. Chapter 1 introduces the research domain, and presents the motivation behind exploring spatial constraints in reservoir computing. Chapter 2 covers relevant background material used throughout the thesis. An overview of previous work on physical reservoir computing is given in 2.5.1, and a discussion of topology and spatial restrictions within reservoir computing is presented in 2.5.2.

Chapter 3 describes the thesis methodology. Section 3.1 presents the parameters used for echo state network generation, while Section 3.2 explains the implementation details of benchmarks and other metrics.

Chapters 4 and 5 present experiments made with random geometric graphs and lattice networks, respectively. Most experiments rely on the knowledge gained in previous sections, and thus follow a chronological order.

Finally, Chapter 6 summarizes the discoveries of the experiments, draws conclusions related to the research goals of the thesis, and suggests areas of future work.

[1] https://www.ntnu.edu/socrates
[2] https://spinengine.eu/

Chapter 2

Background

Related work and the state of the art were reviewed, and identification of relevant background material was carried out in the project preceding this thesis [6]. This background is herein amended with deeper insights into the paradigm of physical reservoir computing, and adapted to fit a thesis rather than an article. Specifically, the main focus is transferred from physical limitations in general, to the narrower scope of spatial limitations relating to physical morphology and topology.

2.1 Complexity: Order, Chaos and Criticality

In deeming a physical system able to *compute*, one implies information storage, retrieval and modification. We are as humans intimately familiar with the continuous, yet spontaneous computation present in our brains – our consciousness. We are less acquainted, however, with the conditions that *caused* the emergence of such a system.

Spanning a wide range of topics and disciplines, the field of *complexity theory* seeks answers to this conundrum. An exact definition of "complexity" is perhaps ever so elusive, but at its core lies an emergence of behavior greater than the sum of its parts. Simple, local interactions give rise to intricate, global patterns. This spontaneous emergence of complex behavior is ubiquitous in nature. Ranging from convection cells in physics, to swarm and flock behavior in biology, there is an abundance of interesting phenomena to study [13].

Langton investigated the emergence of computation in cellular automata (CA) [14]. His findings indicate a *criticality* as a condition to support computation. In essence,

in between *ordered* and *chaotic* dynamics, we find a critical phase transition. It is these systems, intertwining order and chaos, that are of interest.

In systems that are too static, perturbation will fade too quickly. Chaotic systems, on the other hand, are wildly unpredictable, making them excessively sensitive. This *edge of chaos* is a recurring theme in the investigation of the computational capabilities of physical systems [14]. In fact, the edge of chaos has been found to be of significance in predicting the computational performance of neural microcircuit models, consisting of spiking neurons and dynamic synapses [9].

Biologically inspired models, most famously the artificial neural network (ANN), are valuable scientific tools. Oftentimes, finding a suitable set of parameters for a model will amount to much the same as finding the critical phase transition between order and chaos. Reservoir computing (RC), a niche framework within the field of machine learning, is concerned with observing the inherent dynamics of a "reservoir" of local interconnections. Often employing random neural networks, RC exploits the intrinsic computation emerging from these local interconnections to solve practical tasks.

2.2 Reservoir Computing Fundamentals

Training recurrent neural networks (RNN) is an inherently difficult task. Gradient descent methods that incorporate loss information become increasingly inefficient on problems with long-range temporal dependencies. This inefficiency makes the backpropagation algorithm used with feed-forward structures less attractive. Specifically, a continuous search in the parameter space of recurrent networks may cause bifurcations points in the dynamics of the system, causing non-convergent training [15]. To circumvent this complexity, alternative methods which leave internal weights untrained have been proposed [3, 16].

Echo state networks (ESN) [3] and liquid state machines (LSM) [16] independently present supervised learning methods that do not adapt the internal weights of the network. Instead, the output is generated using a simple, memoryless classifier or regressor. This makes the RNN function much like a kernel in kernel method algorithms, which seek features and general relations in datasets to increase separability.

Thus, by projecting into a high-dimensional space, temporal information of an input may be incorporated in an instantaneous readout. This methodology has been unified into the research subfield of RC, in which the focus is on separating the randomly generated *reservoir* from the trained readout layer [4].

Interestingly, there is no need for the reservoir to be an artificial neural network –

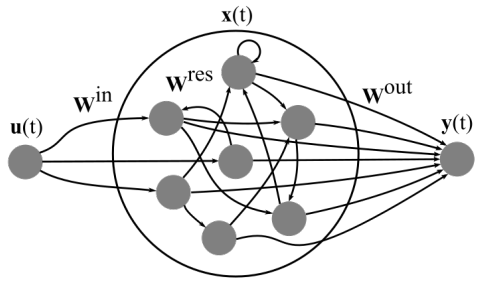


Figure 2.1: Fig. 1: Basic architecture of ESN reservoir systems. The reservoir acts as a high-dimensional kernel, transforming the temporal input sequence into a spatial representation. The readout is trained with supervised linear regression, providing a least squares optimum.

any high-dimensional, driven system exhibiting complex dynamic behavior can be used [4]. As long as the dynamics of the *substrate* can respond suitably to input, it can in theory be used as a reservoir.

A multitude of substrates have shown promise as reservoirs: dynamical systems models such as CA [17], and the more general random Boolean network (RBN) [18], provide a discrete alternative to the analogue ESN. Furthermore, a recent surge in physical reservoir implementations has reinvigorated the field, and is introduced further in Section 2.5.

2.3 Echo State Networks

The ESN is one of the key flavors of RC, and at its core lie untrained, randomly initialized RNNs. Its conception introduced a highly practical approach to training RNNs that is both simple and computationally feasible. In ESNs, inner weights remain fixed, and only the output weights are adapted to construct a linear readout. The basic architecture of the ESN model is illustrated in Figure 2.1, and it consists primarily of real-valued neurons connected by unrestricted synapses, which results in a recurrent network.

2.3.1 Echo State Network Internals

In Figure 2.1, at time-step t the ESN reservoir is defined by its input, internal, and output units, denoted by $\mathbf{u}(t)$, $\mathbf{x}(t)$, and $\mathbf{y}(t)$, respectively. The reservoir dynamics are characterized by three weight matrices, $\mathbf{W}^{in}$, $\mathbf{W}^{res}$, and $\mathbf{W}^{out}$. In

the traditional ESN approach, the reservoir state is evolved according to

$$\mathbf{x}(t+1) = \tanh(\mathbf{W}^{res}\mathbf{x}(t) + \mathbf{W}^{in}\mathbf{u}(t)), \tag{2.1}$$

using tanh as the nonlinear transfer function for internal reservoir nodes. The output of the reservoir is given by

$$\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{x}(t). \tag{2.2}$$

2.3.2 Training

To train an ESN model of size N in a supervised and offline mode, it is run to completion on a training set. The reservoir states are collected row-wise into a matrix $\mathbf{X}$, and the one-dimensional output into a vector $\mathbf{Y}$. The linear readout layer is then trained to minimize the squared output error $E = \|\mathbf{Y} - \hat{\mathbf{Y}}\|$ where $\hat{\mathbf{Y}}$ is the target output, which amounts to finding the $\mathbf{W}^{out}$ that minimizes the error with linear regression. Well-known methods include ridge regression, often called Tikhonov regularization [19], and the Moore-Penrose pseudo-inverse [20].

When the network is adapted to $\mathbf{W}^{out}$, the ESN is fully trained, thus illustrating the apparent simplicity and low algorithmic complexity of the method. Gauging the performance of a trained network is done by running a test set.

2.3.3 ESN Generation

As with virtually every machine learning technique, the application of ESNs requires some experience. Although a conceptually simple idea, generating adequate reservoir networks is influenced by multiple global parameters. Recommendations to achieve sufficient performance are presented in [21, 22], suggesting parameters such as the scaling of the input weight matrix $\boldsymbol{\iota}$, the spectral radius of the reservoir connection matrix $\boldsymbol{\rho}$, and the model size parameters to be of high importance. However, in practice the evaluation of a reservoir is an endeavor often conducted by training the output and measuring the error, sometimes requiring extensive parameter sweeps.

2.3.4 Improvements to the Traditional ESN

Many improvements have been made to the vanilla ESN, most of which are beyond the scope of this thesis and its focus on physical reservoir computing. Nevertheless, an introduction to the methodology would be incomplete without a mention of some of the alterations that improve upon it.

Jaeger also proposed sigmoid reservoir nodes with memory to learn slow, continuous dynamics [3]. In reservoirs with such *leaky integrator neurons*, the nodes will not discard their previous state entirely, but maintain a memory due to a leaking rate α. Another addition proposed by Jaeger is inserting noise into the training procedure, as it is well known in the field of traditional artificial neural networks that an addition of noise to training data can lead to generalization improvements similar to that of Tikhonov regularization [23].

Other important discoveries include improving reservoirs using intrinsic plasticity [24] and lateral inhibition [25], both inspired by concepts of neurobiology.

Lastly, stacking layers similarly to deep learning methods has been attempted. With the aim of developing and enhancing *hierarchical* dynamics, it is intended to allow for multiple time-scales, and increased *richness* in the reservoir, measured as the entropy of the reservoir states, and it has shown great potential [26].

2.3.5 Real World Applications

The ESN methodology has been applied somewhat successfully to real world tasks. Approaches include equalizing a wireless communication channel [27], and short-term traffic [28], electric load [29], and stock price forecasting [30]. Robot control is also a popular area of research for RC applications, particularly for motor control and event detection [31, 32, 33]. Perhaps less conventionally, RC has also been applied in the context of reinforcement learning [34].

However, as the practicality of the paradigm resides primarily in chaotic time series prediction and classification, this is also its main focus. Furthermore, recent years have seen an increase in the realization of physical reservoirs to accompany existing software simulations. An example is a silicon photonics chip capable of 5-bit header recognition up to 12.5 Gbits^{-1}, and is scalable to even higher bitrates [35]. This surge of optimism has breathed new life into the field of RC, as physical reservoirs pave the way for new types of integrated chips.

2.3.6 Comparison to State of the Art

Few definitive comparisons between the ESN and similar flavors of the RNN have been carried out. The long short-term memory (LSTM) [36], as well as its more recent descendant, the gated recurrent unit (GRU) [37], are mainstays in sequence processing, particularly in natural language processing. Early experiments demonstrated the ESN methodology to outperform previous LSTM methods on learning a chaotic attractor by orders of magnitude [3], but ESNs have largely remained a secondary tool outside of chaotic time series prediction and classification.

A recent study compares the promising deep echo state network architecture (DeepESN) to that of gated RNNs, where an experimental comparison between recurrent models on multivariate time-series prediction tasks is made [38]. It is established that the DeepESN methodology outperforms other RNN approaches in their prediction ability on challenging, real world datasets. The computation time is also lessened by about one order of magnitude compared to fully-trained RNN approaches. Thus, the adoption of LSTM and GRU in practice may not necessarily be based on performance suitability, but rather software availability and popularity.

2.4 Assessing Reservoir Quality

Designing *good* reservoirs, possessing some set of desired properties, naturally requires some metric by which we can evaluate and compare. Parameter sweeps, i.e. our trial and error methods, must be accompanied by sufficient methods of assessing computational performance.

Evaluation of reservoir quality is split into two different approaches. Intuitively, measuring the performance of the model on a given benchmark task is a simple, direct way of assessment. However, to gain an intuition for a more general, expected performance across multiple benchmarks, one may measure independent properties of the system, e.g. the spectral radius of the internal weight matrix. The two approaches are often used in conjunction, combined to propose an overall quality.

2.4.1 Independent Metrics

Kernel Quality and Generalization

Within the RC paradigm we are concerned with producing a complex mapping from the input stream to some spatial, internal representation, such that a memory-less, linear readout map may be employed for classification.

The *linear separation property*, or *kernel quality*, measures ability to separate different inputs [9]. It is an empirical measure of this complex mapping, denoting the potential diversity of nonlinear operations carried out by a reservoir. Kernel quality is evaluated by presenting a reservoir of size n with m different input sequences, and computing the rank of the resulting $m \times n$ matrix consisting of the reservoir states at some time step t for the input streams [39].

Another metric accompanying the kernel quality is the *generalization capability* of the reservoir [9]. This metric addresses ability to generalize already learned functions or filters to new, unseen input, and is used as an estimation of the VC-dimension of the reservoir. Generalization capability is evaluated with the same method as kernel quality, but instead requires input streams that are similar, or belong to the same class [39].

A reservoir in the ordered regime will naturally exhibit low values on both metrics, while both metrics will be high in a network in the chaotic regime. Thus, in general, reservoirs exhibiting a high kernel quality and a low generalization rank are desirable, and the difference between the two is sometimes used as its own metric [39]. Kernel quality, generalization, and their difference have been used to evaluate artificial spin ice as physical reservoirs [12].

Short-Term Memory

Short-term memory capacity was introduced as a quantitative measurement of linear memory capacity in reservoirs [10]. It is a way to examine the fading memory present in the system, and is measured by attaching output units to the reservoir, which each are trained to recall some time delayed version of the input sequence. By measuring how much of the input each output unit can recover, we can estimate the memory capacity MC by summing over all time delays. Jaeger defined this as

$$MC = \sum_{k}^{\infty} MC_k = \frac{cov^2(u(t-k), y_k(t))}{\sigma^2(u(t))\sigma^2(y_k(t))}, \tag{2.3}$$

where MC in general is limited by the reservoir size N, such that $MC \leq N$. High input retention is a desirable property, but an increase in memory capacity through parameter tuning is often met with a decrease in complex information processing, due to a universal trade-off between memory and nonlinearity [40].

Memory-Nonlinearity Trade-off

Experimentation with a wide range of reservoirs has indicated a crucial interplay between the memory and nonlinearity properties in reservoir operation [41]. In fact, the interplay has been uncovered to be a universal trade-off between depth of memory and nonlinear computation performed by a dynamical system [40].

Thus, analyzing the boundary between an ordered, static regime that provides memory, and a chaotic, dynamic regime that provides processing, is of vital importance in the design of reservoirs. Determining the required nonlinearity for a task is not simple, and often benefits from intuition about nonlinear dynamics. Empirically, it has been shown that the input scaling, determining the nonlinearity of reservoir responses, and the spectral radius, scaling the importance of previous states, are the main parameters for optimization in ESNs, illustrating the significance of the trade-off [21].

Further formalization of the trade-off has been conducted, accompanied by a proposition of a mixture reservoir combining both linear and nonlinear dynamics. Adding a "pinch of linearity" is cited to improve performance considerably [42].

Further Metrics

A handful of other methods to assess quality and criticality of reservoirs have been adapted, including the Lyapunov exponent [43], the Jacobian of the reservoir [44], Fisher information [45], and a separation ratio [46].

In summary, given the vast amount of methods for evaluation, choosing a set of suitable metrics is a surmountable task. This is especially so given that few metrics are entirely orthogonal, and can often be found to correlate in prediction of performance [47].

2.4.2 Benchmarks

Employing benchmarks to measure the performance of reservoirs is a means to directly capture performance on specific tasks. Myriads of benchmarks exist within the field of time series prediction, generation, and classification. The benchmark spectrum ranges from simple tasks, to complicated, highly dynamic and autoregressive time series.

Simpler tasks include the XOR problem of which is not linearly separable [48], and n-bit temporal density and parity [49]. More complex tasks may range from

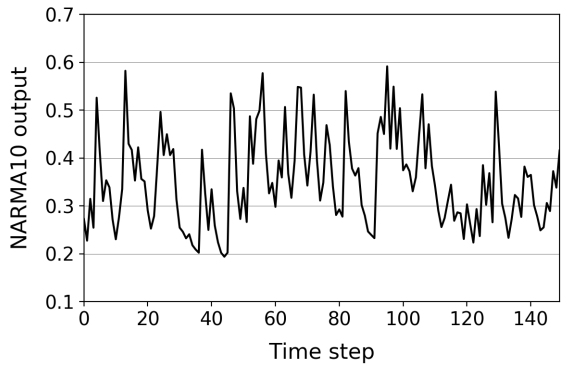


Figure 2.2: Example output generated by a 10th-order NARMA system. The autoregressive moving average nature of the time series is clearly visible.

recognizing isolated digits in speech [50], to predicting time series, of which the most popular are the nonlinear autoregressive moving average, NARMA [7], and the Mackey-Glass time delay differential equation [8]. Further datasets, such as the Santa Fe Laser, Hénon Map, IPIX Radar, and Sunspot series datasets have also been used [51].

NARMA - Nonlinear Autoregressive Moving Average

The class of time series provided by a nonlinear autoregressive moving average, most often simply referred to as *NARMA*, is a model commonly used to benchmark recurrent networks [7]. Its widespread use yields baseline performances for well established models, as well as more novel approaches [43, 52]. A 10th-order NARMA system is depicted in Figure 2.2, showing how the output of the time series is evolved for each time step according to Equation 2.4.

NARMA provides discrete-time temporal tasks, introducing a time-lag of n time steps, and is given by

$$y_t = \alpha y_{t-1} + \beta y_{t-1} \sum_{i=1}^{n} y_{t-i} + \gamma u_{t-1} u_{t-n} + \delta. \tag{2.4}$$

Here, n is the order of the system, and common constant parameters are $\alpha = 0.3$, $\beta = 0.05$, $\gamma = 1.5$ and $\delta = 0.1$. The input u_t is an i.i.d. stream drawn uniformly from the interval [0, 0.5], and the nonlinear product on the input sequence is shown in Figure 2.3, illustrating the nonlinearity of the $u_{t-1}u_{t-n}$ term in Equation 2.4. The time series is unstable, and tasks with higher than a 10th-order time lag

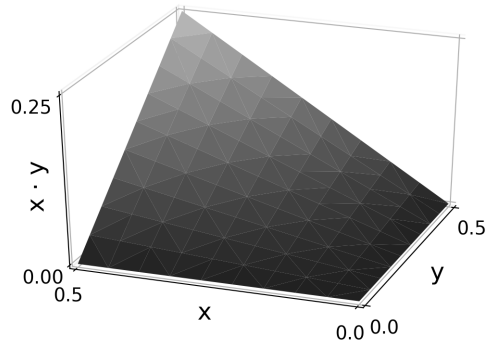


Figure 2.3: Nonlinear mapping of the product $u_{t-1}u_{t-n}$ of inputs in the NARMA time series in Equation 2.4.

introduce a saturation function to produce a bounded sequence:

$$y_t = \tanh(\alpha y_{t-1} + \beta y_{t-1} \sum_{i=1}^{n} y_{t-i} + \gamma u_{t-1} u_{t-n} + \delta). \tag{2.5}$$

Predicting a NARMA time series, given an input sequence u, presents a challenge of both memory and nonlinearity. This makes NARMA well-suited for evaluating both the memory capacity and computational power of reservoirs with a single metric. Reservoirs must necessarily remember input sequences of length n, and should preferably adhere to suitable dynamics on top of this.

Evaluation of ESN performance on the NARMA10 system is a thoroughly explored area in the field of RC. Reported NRMSE performances for traditional ESN reservoirs of size $N = 200$ lie in the range [0.20, 0.25] [43, 51, 53, 54]. For some context, using a shift register containing the input as a reservoir will achieve a minimal NRMSE of 0.4. To achieve NRMSE values below this threshold it is necessary to introduce nonlinearity in the reservoir.

Mackey-Glass Equation

A common benchmark for dynamical systems is chaotic attractor learning. One such benchmark is the Mackey-Glass delay differential equation

$$\dot{y}(t) = \alpha \frac{y(t-\tau)}{1 + y(t-\tau)^{\beta}} - \gamma y(t), \tag{2.6}$$

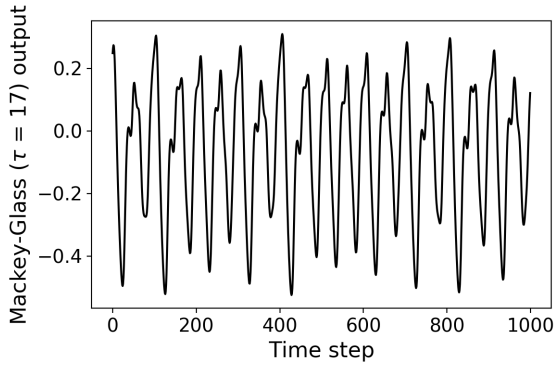


Figure 2.4: Sample sequence from the Mackey-Glass ($\tau = 17$) delay differential equation.

where common constant parameters are $\alpha = 0.2$, $\beta = 10$ and $\gamma = 0.1$. A chaotic attractor appears when $\tau > 16.8$, and in practice $\tau = 17$ is used to generate a mildly chaotic attractor, while $\tau = 30$ yields strongly chaotic behavior. The time series is most often generated with numerical methods such as the Runge-Kutta method. Figure 2.4 shows an example Mackey-Glass sequence with $\tau = 17$, where the output is evolved for each time step according to Equation 2.6.

2.5 Physical Reservoir Computing

Developments in the field of RC have inevitably lead to novel and creative approaches to reservoir design. Previously considered an "exotic" technique, using physical substrates to realize reservoirs has become a common concept, and the number of studies has been increasing rapidly. Physical reservoir computing deviates from the traditional computer architecture in which processing and memory units are separate entities, and into the territory of unconventional computing. The notion of physical RC thus takes part in the evolution *in materio* paradigm [55], encompassing the general idea of exploiting the physical properties of materials.

2.5.1 Physical Reservoir Requirements

What types of physical properties must be considered when experimenting with novel materials for computation? Dale et al. suggest four key factors of relevance: observability, nonlinear dynamics, methods of modeling the system, and the impact of environmental factors [2]. Tanaka et al. present four requirements

for physical reservoirs to efficiently solve tasks: high dimensionality, nonlinearity, fading memory, and the separation property [5].

Hence, the computational requirements of physical reservoirs are similar to those of conventional reservoirs. Their main difference lie in their operability, as physical reservoirs tend to be harder to interact with. Suitable input and output schemes must exist, and may be hindered by environmental factors. Moreover, physical reservoirs have to be realized in a real, physical space, which is further introduced in Section 2.5.2.

In practice, physical reservoirs have been realized with a broad range of substrates: photonic systems [35], electronic memristor circuits [56], mechanical springs [57], and more biologically oriented reservoirs such as gene regulation networks [58], and the cat primary visual cortex [59]. Consult [5] for a thorough review of investigated physical substrates, their applications, and the general trends in physical RC.

2.5.2 Topology and Spatial Networks

When designing physical reservoirs, there is a chance that the choice of underlying topology is limited by physical factors, such as the type of physical interactions that are present in the substrate. Additionally, physical reservoirs must necessarily be embedded in physical space. The layout of vertices and edges of such systems, however complicated their dynamics may be, are thus *restricted by space*. Topology alone proves insufficient to describe these networks, as these spatial constraints determine freedom in structure, i.e. node position and edge cost.

The real world presents us with plenty of networks that possess spatial components, ranging from the Internet to rail roads. Models have been developed to study the properties of these networks, which in turn may be useful in studying reservoirs that will have spatial constraints imposed. Enticing models include the Watts-Strogatz model [60], with its small-world properties, and the Waxman model [61], where nodes are connected with a probability that depends on the distance between them. A thorough review of spatial networks is presented in [62].

However, in the context of physical reservoir computing, it may turn out to be helpful to take a step back. Before delving for interesting spatial models, it is important to gain insight into the performance penalties one might expect from enforcing a spatial layout in the first place. The foundation of the ESN is the Erdos-Renyi graph [63], a simple model for random geometric graphs in which nodes are connected with a probability p. Imposing a metric space onto the ESN model, in practice a bare minimum, allows us to observe the simplest spatial reservoir.

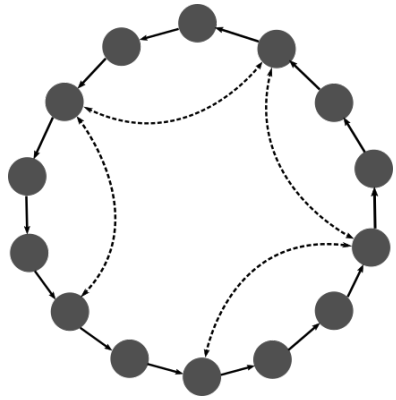


Figure 2.5: Ring topology investigated by Rodan and Tiňo [51, 64]. Dashed lines indicate bidirectional regular jumps of length $\ell = 3$.

Related Work

Rodan and Tiňo discovered that simple, cyclic reservoirs, i.e. ring topologies, perform comparably to the ESN [51]. These cyclic reservoirs were later extended with regular jumps to consistently outperform regular ESNs [64]. The ring topology is depicted in Figure 2.5, where dashed lines are bidirectional regular jumps of length $\ell = 3$. Gallicchio et al. reported similar findings in their work on DeepESN, where structured schemes such as multiple rings lead to the best performances [65].

Dale et al. compared ring, lattice, and fully-connected reservoir topologies [66]. They argue that fully-connected ESNs exhibit the highest substrate quality, given higher memory capacities and a better ability to produce rich nonlinear representation of the input. The discrepancy between performance predicted by these metrics and the benchmarks used by Rodan and Tiňo in their work on ring topologies should be noted.

Lastly, Manevitz et al. have found small world topologies to produce fault tolerant reservoirs [67]. Errors resulting from introducing dead and noisy neurons were remedied by choosing a connectivity scheme pertaining to a power law distribution, illustrating superiority to uniform connectivity in terms of robustness.

Research on reservoir computing with constrained topology or physical morphology is a relatively sparse area. There is much to be discovered about the impact of reservoir construction with structured organization. Moreover, efforts to find useful topologies should also yield valuable insights about why the stochastic ESN works so well, to allow for more deterministic construction of reservoirs with desirable properties.

2.5.3 Artifical Spin Ice

Artificial spin ice (ASI) consists of a large amount of nanomagnets arranged in a two-dimensional lattice, and is a prime candidate for exploiting the intrinsic dynamics of a physical substrate. Each magnet behaves as a binary spin, and spins are coupled via magnetic dipole-dipole interactions. Thus, ASI bears resemblance to previous reservoir methods, where complex dynamics emerge from simple, local interactions between nanomagnets. An ASI reservoir may be perturbed by an external magnetic field to encode input patterns, and the resulting ASI magnetization is used as output.

Recent work has illustrated the promise of ASI as interesting reservoirs through software simulations, indicating a wide range of dynamics when perturbed by an external magnetic field [11]. Furthermore, the development of the flatspin ASI simulator allows fast simulations in coupled spin systems [68], enabling large scale experiments that demonstrate excellent computing capacity in ASI [12].

Chapter 3

Methodology

3.1 ESN Parameters and Sample Sizes

In this section we present the baseline ESN used during experiments. For each experiment described in this thesis, the default setup will be as described in this section, unless otherwise specified. All ESNs are generated according to the architecture presented in Figure 2.1.

We considered discrete-time ESNs with N internal network nodes, a single input, and a single output node. $\mathbf{W}^{in}$ was generated as a random matrix with i.i.d. entries in the interval [-0.5, 0.5], and was fully connected. In experiments with default ESNs, i.e. experiments where the internal network $\mathbf{W}^{res}$ is not replaced, the weights were generated from the same distribution as $\mathbf{W}^{in}$, but with a 10% node connectivity. This method for instantiating $\mathbf{W}^{res}$ and $\mathbf{W}^{in}$ is common practice in RC [21].

In experiments where relevant, the reservoir weight matrix was rescaled such that its spectral radius $\rho(\mathbf{W}^{res}) = 0.9$. The default input scaling used was $\iota = 1.0$. Both parameters could be tuned to provide marginally better results in most cases, but these values were found to give a good baseline for comparisons between models.

$\mathbf{W}^{out}$ was adapted with ridge regression, using single value decomposition for computational routines. This was found to lead to the most stable and precise results.

For all experiment runs, the first 200 states of each run were discarded to provide a washout of the initial reservoir state. Reported performances are the mean across 20 randomizations of each model representative. This sample size was found to be

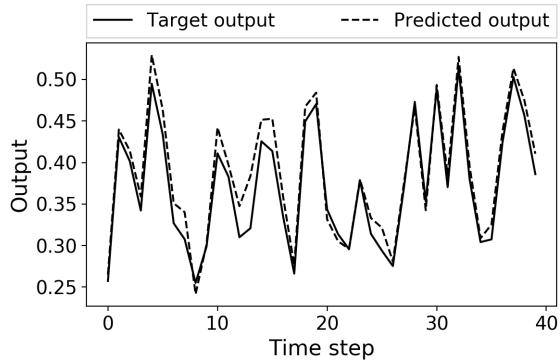


Figure 3.1: Evaluation of a traditional ESN of size $N = 200$ on the NARMA-10 benchmark. The first 40 time steps of the benchmark are shown, comparing target output to the predicted output of the reservoir. The resulting NRMSE was 0.23.

appropriate to pinpoint definite trends in the results. Standard deviations for all experiments are available online[1].

The Python software library implementation is available online, including a Jupyter Notebook for reproducing each experiment[2].

3.2 Benchmarks and Metrics

To evaluate reservoirs, the benchmarks and metrics described in Section 2.4 were used. Note that the our primary evaluation tool was the NARMA-10 benchmark, using the rest as supplements where relevant.

For the NARMA-10 benchmark, We generated the time series according to Equation 2.4. The generated input was split into a training and test set, with $L_{train} = 2000$ and $L_{test} = 3000$, to allow comparisons with results presented in [51].

The Mackey-Glass benchmark has been used in a range of ways, making comparison of models harder than for NARMA. We used the method presented in [69]. We generated a time series from Equation 2.6 using discrete approximation, squashing the output into a range of [-1, 1] with $\tanh(y-1)$. The series was split into a training and test set, with $L_{train} = 6400$ and $L_{test} = 2000$. The task of the reservoir was to predict input 84 time steps ahead, i.e. $\mathbf{y}(t) = \mathbf{u}(t + 84)$.

[1] https://github.com/thomaav/esn-spatial-constraints/blob/master/appendix.md
[2] https://github.com/thomaav/esn-spatial-constraints

To evaluate performance on the benchmark, we used the normalized root mean square error (NRMSE). The NRMSE is a commonly used error metric, providing comparability to performances reported in previous work. Given a predicted signal $\mathbf{y}(t)$ and a target signal $\hat{\mathbf{y}}(t)$

$$\text{NRMSE}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{\langle \|\mathbf{y}(t) - \hat{\mathbf{y}}(t)\|^2 \rangle}{\langle \|\hat{\mathbf{y}}(t) - \langle \hat{\mathbf{y}}(t) \rangle\|^2 \rangle}}. \tag{3.1}$$

An evaluation of a traditional ESN of size $N = 200$ using the NARMA-10 benchmark is shown in Figure 3.1. In this figure, the first 40 time steps of the benchmark are shown, comparing the target output to the predicted output of the reservoir. The resulting NRMSE of the benchmark was 0.23.

Kernel quality was evaluated by drawing N input streams of length 20 uniformly from the interval [-1, 1], where N is the number of hidden nodes. The resulting input streams were then run through the reservoir, and the rank of the resulting $N \times N$ matrix, consisting of the reservoir states, was computed. Generalization ability was evaluated in a similar manner, but differing in that the last four inputs across all input streams were identical. This is a standard way of computing the metrics [39].

Memory capacity was computed according to [70]. An input stream of length 2200 was drawn uniformly from the interval [-1, 1], and the first 200 inputs were discarded to get rid of transients. Next, an input sequence of length 1000 was used for training, and the remaining 1000 inputs for testing. Memory capacity was computed according to Equation 2.3, using $k = 1.4N$ output neurons.

3.3 Experiments

In chapter 4 we are concerned with replacing the generated $\mathbf{W}^{res}$ with one that is generated as a random geometric graph. In chapter 5 we replace the matrix with reservoirs in which the internal connectivity is set to be square, hexagonal and triangular regular tilings. Beyond this, the ESN approach remains entirely the same, unless explicitly stated otherwise.

Chapter 4

Experiments: Random Geometric Graphs

The first idea that springs to mind when considering spatially constrained graphs, is of course to place the vertices in some metric space. When investigating spatial reservoirs, it is thus only natural that we begin with the simplest spatial network model – the random geometric graph (RGG).

To construct a reservoir based on the RGG model, we place its N internal nodes randomly in an underlying metric space $[0, l)^{dim}$, giving each node a position attribute sampled from a uniform distribution. An example distribution of the vertices is shown in Figure 4.1a. For any given pair of nodes x, y, we consider the Euclidean distance between them

$$d(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{dim} (x_i - y_i)^2}. \tag{4.1}$$

Nodes are connected only to other nodes within their neighborhood radius r, i.e. where $d < r$, depicted in Figure 4.1b. However, since we are interested in the behavior in physical materials, we set $r = \infty$ to allow full connectivity in all experiments. Thus, edges are weighted according to several distance functions, $1/d$, $1/d^2$, and $1/d^3$, to model how the interaction strength diminishes with distance in many physical substrates. For example, the spins in artificial spin ice, presented in Section 2.5.3, is subject to a magnetic field from all neighboring spins that diminishes according to $1/d^3$ [11]. Lastly, a dimensionality of 3, i.e. $[0, l)^3$, is used.

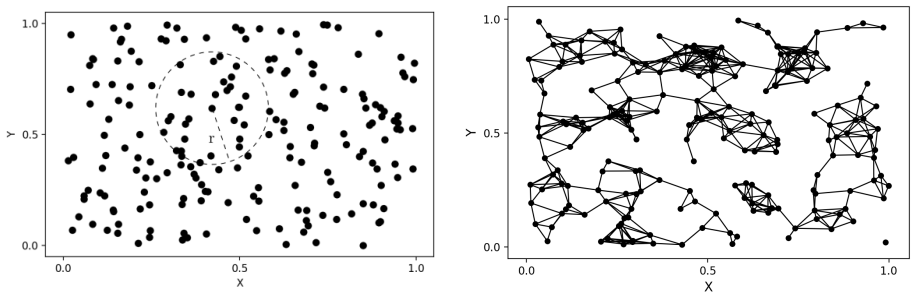


Figure 4.1: (a) Example vertices drawn to generate an RGG embedded in a two-dimensional Euclidean space. The neighborhood radius r of a single node is shown. (b) RGG instance with neighborhood radius $r = 0.1$.

4.1 Size of the Underlying Volume

4.1.1 Synopsis

Our first experiment is concerned with the size of the metric space in which we embed the reservoir, the l of $[0, l)^3$. The behavior of the ESN created will obviously be affected by the magnitude of the weights of the network edges. A space that is too small will result in weights that are too big, and vice versa.

4.1.2 Results and Discussion

The results of tuning the underlying volume size l is shown in Figure 4.2. Shown are the results for the distance functions $1/d$ and $1/d^2$, respectively, where the parameter for network creation is the sweep over l. The y-axis presents both the resulting NRMSE on the NARMA-10 benchmark (left side), and a measurement of the resulting spectral radius of the networks (right side).

First, we see that an underlying volume of size $l = 1$ is near useless. The weights generated by the $1/d$ and $1/d^2$ distance functions are too big, as expected. The consequence of big weights is that the spectral radius of the $\mathbf{W}^{res}$ grows too large as well. In fact, the tanh transfer function of the internal nodes, defined by Equation 2.1, stays completely saturated throughout the entire benchmark run.

Next, we see that as the spectral radius of $\mathbf{W}^{res}$ decreases, so does the benchmark error. It is well-known that a spectral radius that is too small will cause ordered

or fixed-point behavior, and one that is too big may cause unbounded deviation from initial states [44]. Hence, the valleys seen in Figure 4.2, are equivalent to those seen in early explorations of the spectral radius property in reservoirs [43]. However, in our case, the spectral radius is implicitly scaled by stretching the size of the underlying volume in which the graph is embedded.

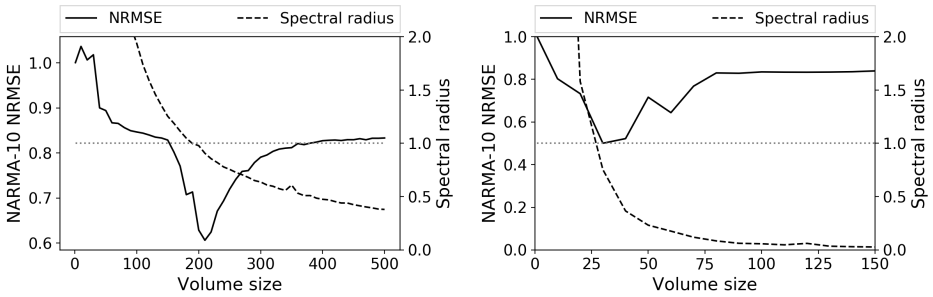


Figure 4.2: Spectral radius and performance of generated random geometric graphs of size $N = 100$, as a function of the size of the underlying volume. Illustrated is node coupling using two different distance functions: $1/d$ (a) and $1/d^2$ (b). For the $1/d^2$ (b) distance function the minimum NRMSE value over 20 runs is used instead of the mean, as it exhibits a higher variability, explained further in Chapter 4.2.

Knowledge that stretching the underlying space is equivalent to scaling the spectral radius of the reservoir is of key interest. In physical contexts, this means that we can scale the degree of dynamical richness in the reservoir by moving nodes, or otherwise strengthening or lessening their connectivity. This conclusion seems obvious, but is of importance for following experiments. We can directly scale the spectral radius $\mathbf{W}^{res}$ with confidence that we could do the same by scaling the metric space, which in practice amounts to scaling all edge weights by a single scalar.

Performance-wise, spatially constraining ESN reservoirs has caused an increase in error compared to the non-constrained approach. The best reservoirs, seen at the bottom of the valley in Figure 4.2b, achieve an NRMSE around 0.5, with a spectral radius $\boldsymbol{\rho}$ around 0.8. This is worse than what is achieved using a shift register with perfect memory. There is thus an indication that spatial restrictions do cause a performance penalty, and figuring out what is causing this is a main theme of the following sections.

4.2 Distance Functions and Memory Capacity

4.2.1 Synopsis

Before exploring changes to improve upon our RGG model, it is reasonable to examine the differences between the distance functions we have available, $1/d$, $1/d^2$, and $1/d^3$. The increasing degree of the inverse of the distance essentially acts to reduce the neighborhood size r. For example, with a distance function of $1/d^3$, nodes that are distant will have little or no impact on each other.

Using the knowledge we gained in Section 4.1 about the equivalence of the size of the underlying space and the resulting spectral radius of the network, we now also directly scale the spectral radius of $\mathbf{W}^{res}$ to 0.9 by multiplying it by the appropriate scalar.

4.2.2 Results and Discussion

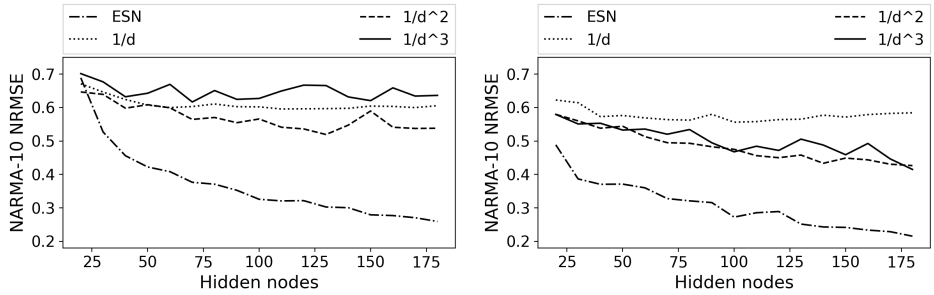
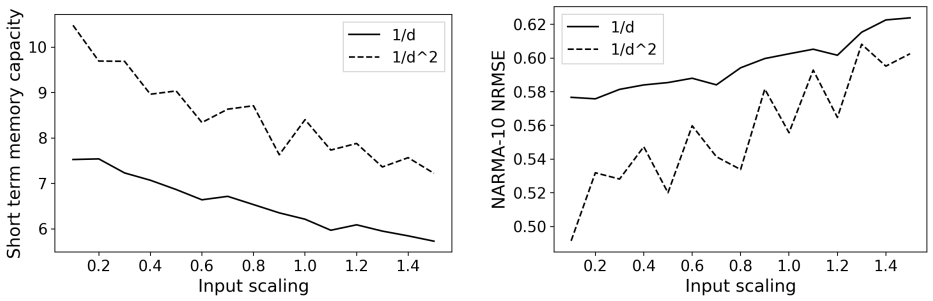


Figure 4.3: NRMSE on the NARMA-10 task with use of different distance functions to generate connection weights. Plots shown are the mean (a) and minimum (b) error aggregations over 20 runs per individual parameter setup. Distance functions are compared to the standard echo state network.

Figure 4.3 compares the performance of the distance functions to that of default ESNs. Plotted is NARMA-10 NRMSE as a function of reservoir size, i.e. the number of hidden nodes. The average over 20 runs, shown in Figure 4.3a, shows little noticeable decrease in error for the distance functions as reservoir size increases. Additionally, it is clear that there is some degree of variability in the error when using the $1/d^2$ and $1/d^3$ functions. This notion is strengthened by Figure 4.3b, where we see best performances for $1/d$ is about the same as the mean, while $1/d^2$ and $1/d^3$ show a slight decrease in error with reservoir size.



(a) **(b)**

Figure 4.4: Effect of input scaling on reservoirs generated as random geometric graphs, size $N = 80$. There is a correlation between the short-term memory capacity of the reservoirs (a), and error rates for the NARMA-10 task (b).

Empirically, we have found that reservoirs that get no significant performance increase on the NARMA-10 task with an increasing reservoir size, tend to be limited by memory capacity. The inability of the reservoirs to reach the performance of a delay line supports this hypothesis. Figure 4.4 highlights this, presenting both the short-term memory capacity and the NARMA-10 NRMSE of reservoirs of size $N = 80$ as a function of input scaling. As we decrease the input scaling, the memory capacity increases, and error decreases. This is expected, as it determines how nonlinear the responses of the reservoir are, and there exists a memory-nonlinearity trade-off, described in Section 2.4.1.

To summarize, we have found that the average performance of our distance functions seem to hover around similar values. A general trend is that the RGG model lacks sufficient memory capacity to solve the NARMA-10 task well, regardless of the distance function used. Memory retention is improved by lowering the input scaling. Further exploration can now be done by choosing a single distance function, as our results suggest that they suffer from problems of a similar nature.

4.3 Restoring Echo State Network Performance

4.3.1 Synopsis

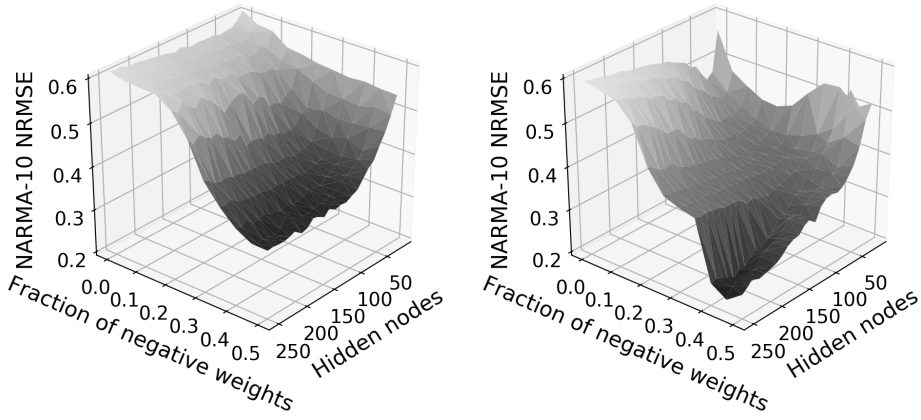
Next, we will make changes to the $\mathbf{W}^{res}$ generated with the RGG model to move its performance close to the ESN model. This may seem counter-intuitive, as we primarily concern ourselves with physical reservoir computing. Making arbitrary changes to the model generation is not a procedure that will necessarily translate well to physical substrates and their restrictive nature. However, we argue that

determining the root cause of the difference in error seen in Figure 4.3 will uncover important properties of reservoir design. In turn, this improves our fundamental understanding of the paradigm.

In this section, we therefore introduce signedness and directedness to the edges of the RGG. The signedness of the reservoir is given by some percentage, e.g. 10%, such that each edge has a corresponding chance of becoming negative. Reservoirs are either directed or undirected, where edges in a directed reservoir have a 50% chance of going in either direction. Note that making an edge negative is not the same as reversing its direction, as it simply means that a node will weight the value of its neighbor negatively.

For these experiments, we use the $1/d$ distance function. In Section 4.2 we found this function to give the most stable results. Although it does not strictly produce the *best* results, we found the stability to easier to work with when looking for definitive trends in performance. Similar results were seen with $1/d^2$, though less pronounced. We also found lowering the input scaling to improve memory capacity, and experiments in this section use an input scaling of 0.1

4.3.2 Results and Discussion



(a) **(b)**

Figure 4.5: Introducing signed weights to RGG reservoirs. Results shown are for undirected (a) and directed (b) reservoirs.

The impact of introducing signed, directed edges to reservoirs is shown in Figure 4.5. The shape of both surfaces indicate that an increased fraction of signed weights (left lower axis) decreases the benchmark error in all cases, both for directed and

undirected reservoirs, except for very small ones. Moreover, making the reservoir directed decreases error drastically when the fraction of negative edges is 0.5, shown by the difference between the lowest points of Figure 4.5a and Figure 4.5b. This is particularly true for the biggest reservoirs.

Firstly, negative weights discard some of the inherent symmetry of the network. Although the weight matrix is still symmetric in magnitude, it now allows for a wider range of node behavior, especially considering that the negative half of tanh becomes available. The addition of directed edges makes it clear that a non-symmetric weight matrix enables richer dynamics, as we move below the performance of a delay line.

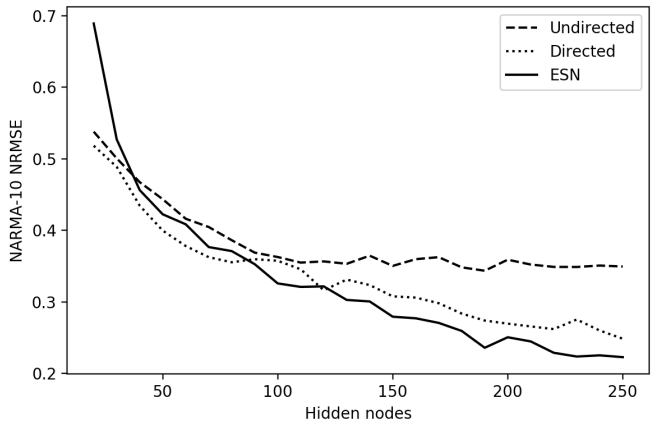


Figure 4.6: Comparing RGG reservoirs with signed and directed edges to traditional ESNs. Both RGG reservoirs have a fraction of negative edges of 0.5.

Figure 4.6 compares the performance of these reservoirs with ESNs. Directed reservoirs with a signed weight fraction of 0.5 perform comparably to ESNs, and scale similarly with reservoir size. Our interpretation of this result is that the importance of *flow of information* in reservoirs should not be understated. It seems that the *structure* of the reservoir network is crucial, and that *where* information flows is as important as its magnitude.

A similar conclusion was reached in [66], where directed networks were shown to cover a bigger behavior space than their undirected counterparts.

4.4 Reservoir Weight Distribution

4.4.1 Synopsis

By reintroducing signed and directed edges to the RGG model, it goes back to resembling traditional ESNs. In fact, the major difference between the RGG model and ESNs was their distributions of internal reservoir weights. In this section we make a short comparison between the two.

4.4.2 Results and Discussion

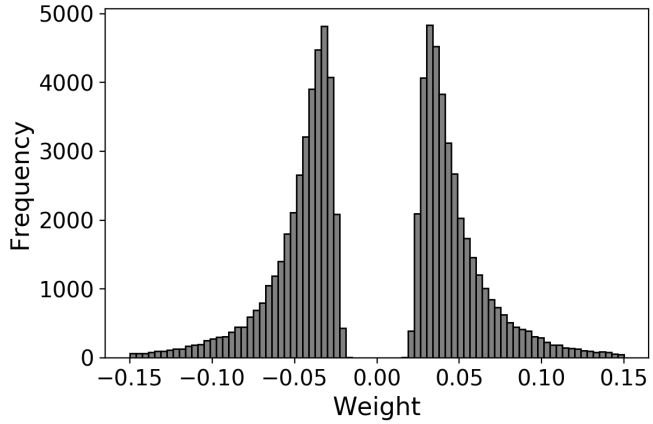


Figure 4.7: Weight distribution of a reservoir with both signed and directed edges. Zero-elements have been removed, as half of the matrix entries contain zeroes. The distribution follows the $1/d$ distance function, which incidentally looks similar to the reciprocal normal distribution.

The distribution of the internal weights of RGG reservoirs with signed and directed edges is shown in Figure 4.7. Three key points are worth mentioning: (i) the distribution resembles the used distance function $1/d$, (ii) the distribution is symmetric around zero, due to signed edges, and (iii) elements of zero magnitude have been removed, as half of $\mathbf{W}^{res}$ contain zero-valued entries due to the directedness of the reservoir.

Distributions commonly proposed to provide ESNs with good performance include a symmetrical uniform distribution, or a normal distribution centered around zero [21]. We end up with a symmetric distribution resulting from the distance function $1/d$, a distribution resembling the inverse Gaussian distribution. This distribution has, to our knowledge, not been used in previous experiments.

Finally, we mention that the dimensionality of the underlying metric space of the RGG plays a small role in the performance of the resulting reservoirs. Changing the dimensionality will simply shift the weight distribution slightly, causing little change in performance.

4.5 Conclusions

The experiments in this chapter demonstrate how inherent structural limitations may impact computational capacity in reservoirs. By employing RGG reservoirs, we have illustrated how reservoir computing translates to a simple, spatially constrained topology, and have reasoned about the degradation in performance.

First, we found that the spacing between nodes in RGGs is vital. We presented a correlation between the spacing and the spectral radius of the resulting internal connectivity matrix of the reservoir, suggesting that scaling of the volume in which the graph is embedded is equivalent to traditional scaling of spectral radius.

Further, we evaluated multiple distance functions to determine node coupling, finding that the $1/d$, $1/d^2$, and $1/d^3$ functions perform almost the same, suggesting that they produce similar structure and weight distributions. Additionally, we discovered that a major reason for low performance is low memory retention, which can be remedied by lowering input strength.

We discovered that RGG performance becomes equivalent to that of ESNs once we re-introduce signed and directed edges to the reservoir. We interpret this to indicate that the information flow is a key component in reservoirs.

Moreover, we found that the resulting weight distribution of an RGG with directed edges is different from the traditional uniform or normal distributions. We understand this to imply that multiple weight distributions are suitable, given that the resulting *structure* of the reservoir network is suitable.

This chapter thus serves as an introductory evaluation of spatially constrained reservoirs. The main contribution provided is the notion that directedness, i.e. directed information flow, is a crucial property in quality reservoirs. A deeper investigation is conducted in the following chapter on lattice reservoirs.

Chapter 5

Experiments: Lattices

Lattice models are common in computational physics [71]. Understanding important models of computational physics in reservoir contexts is thus crucial to advance physical RC methodology. For example, the Ising model with dipole moments of atomic spins [11], spin-torque oscillator arrays [72], and the Ginzburg-Landau equation [73], describe systems that are employed on a two-dimensional lattice, and have been used in reservoir settings. In this chapter, we therefore investigate lattice networks as more realistic models of physical reservoirs.

We explore the properties that lattice graphs exhibit as reservoirs by structuring internal nodes in this manner. Lattice graphs may be embedded in Euclidean space to form regular tilings, of which there three in two-dimensional space: square, hexagonal and triangular, which are all depicted in Figure 5.1. Other, more complicated tiling schemes exist. Semiregular, often called uniform, tilings are created using two or more faces. However, complicated grids are left outside the scope of this thesis, as our primary focus is the fundamental applicability of lattice layouts, not comparing the performance between them.

Reservoirs are created by replacing the reservoirs of ESN models with the adjacency matrix generated for lattice models. For each experiment, $\mathbf{W}^{res}$ is then scaled to a spectral radius of 0.9, as this is equivalent to scaling the coupling, or spacing, between nodes in a physical system. Beyond this, our reservoir model remains the same as that of the ESN.

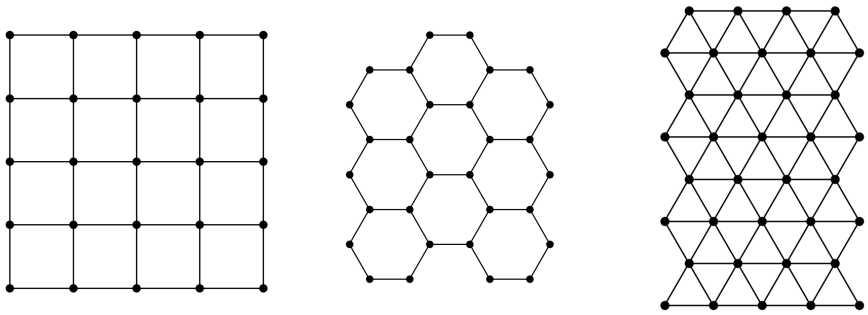


Figure 5.1: Types of lattices investigated for their quality as reservoir topologies. Investigated tilings include square (a), hexagonal (b), and triangular (c).

5.1 Reservoir Quality of Lattices

5.1.1 Synopsis

First, we evaluate the default quality of lattice reservoirs with the NARMA-10 benchmark. Reservoirs are generated by embedding internal nodes in a metric space, much like in Figure 5.1, and connecting neighboring pairs with an edge of unit length. Nodes along the edges are not connected to the opposite side of the lattice, making the lattice aperiodic. Lastly, unit weight of all edges are scaled by the appropriate scalar to allow a spectral radius of 0.9.

5.1.2 Results and Discussion

Figure 5.2 shows how reservoir error scales with reservoir size, depicting NARMA-10 benchmark NRMSE as a function of the amount of hidden nodes N. We see that restricting reservoir topologies to lattice structures results in a significant performance penalty. Additionally, little difference is seen between the three types of tilings.

In Section 4.2, it was discovered that reservoirs modeling random geometric graphs exhibited low memory retention. The symptoms are similar here: the lattice reservoirs perform worse than a delay line would, and only perform marginally better with an increasing size. Figure 5.3 illustrates the effect of scaling the magnitude of the input. Again, we see clearly see reservoirs favoring low input scaling values.

We interpret these results to indicate that the structure imposed by an undirected

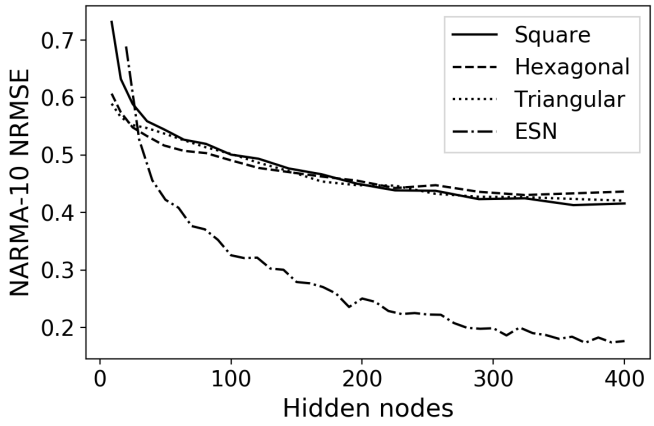


Figure 5.2: NARMA-10 NRMSE of square, hexagonal, and triangular regular tilings as reservoir topologies.

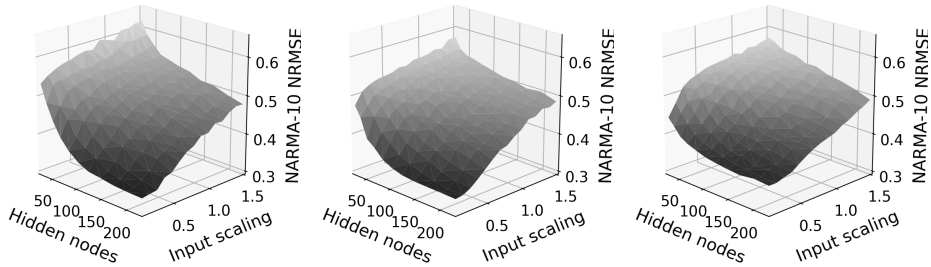
lattice shifts the point of criticality described in 2.1. When input scaling is lessened such that the required memory capacity for the benchmark task is reached, the error diminishes rapidly, and the existing reservoir dynamics work as intended.

Curiously, the NRMSE differs only slightly between the three types of lattice. It seems that it is the overall lattice structure that is important, not the type of tiling implementing it. We therefore argue that the different tilings, which in practice dictate the amount of incident edges per vertex, work mostly as minor tuning parameters. The idea that overall structure is important is in accordance with our findings in 4.3, concluding that *how information flows* in the network is vital.

Input scaling decreased the benchmark error of lattice reservoirs, but the best performing networks of Figure 5.3, i.e. the biggest reservoirs with the lowest input scaling, are not quite comparable to the ESN. For example, square grid reservoirs of size 200 benchmark a mean NRMSE of around 0.35, while corresponding ESNs average around 0.25.

Overall, it is interesting that undirected lattice reservoirs perform as well as they do. On the other hand, the distribution of the input weights are drawn from a uniform distribution in the interval [-0.5, 0.5], letting internal nodes see varying representations of the input signal. As physical substrates may differ in the input schemes they offer, the input scheme will be further investigated in the next section.

To summarize, we have in this section found undirected lattice reservoirs to provide promising results. A key discovery of Chapter 4 is the importance of a directed flow of information, and whether directedness also improves lattice models is the topic of the next section.

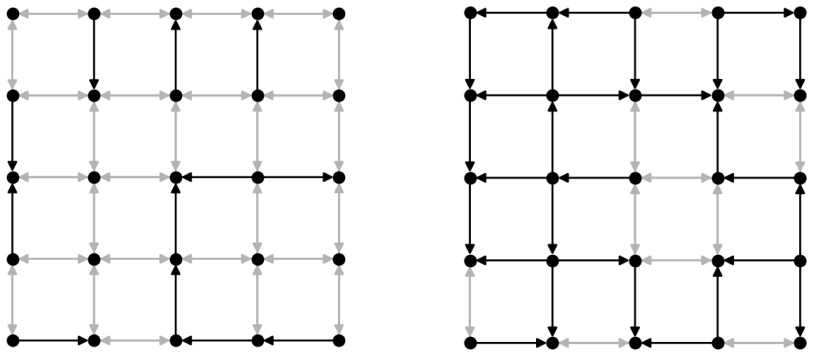


(a) **(b)** **(c)**

Figure 5.3: Regular tilings investigated for their quality as reservoir topologies, here as a function of reservoir size and input scaling. Investigated topologies include square (a), hexagonal (b), and triangular (c) regular tilings.

5.2 Lattices with Directed Edges

5.2.1 Synopsis



(a) **(b)**

Figure 5.4: Example square grids where 25% (a) and 75% (b) of the undirected edges are made directed.

One of the key discoveries of Chapter 4 is that directed edges improve performance significantly in random geometric graph reservoirs. It is of interest to repeat this experiment with lattice reservoirs, especially since information flow is so clearly visible in a lattice structure. We modify the generated lattice graphs generated in previous sections of this chapter to have a fraction of directed edges. Figure 5.4 illustrates the concept for square lattices, where 25% (Figure 5.4a) and 75%

(Figure 5.4b) of the edges have been directed. The directed reservoir edges have a 50% chance of going in either direction.

5.2.2 Results and Discussion

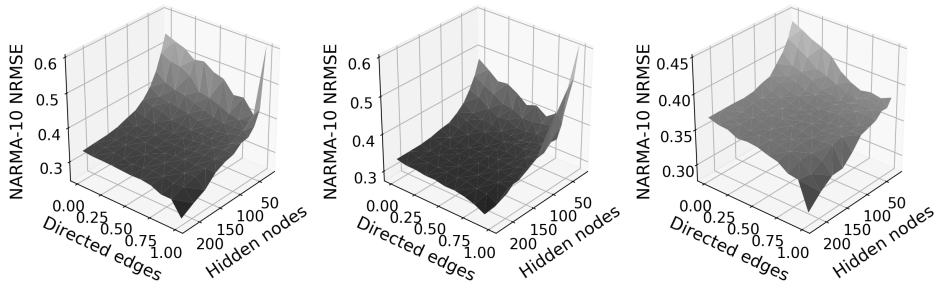


Figure 5.5: Benchmark error as a function of reservoir size and directedness. The fraction of directed edges determines the amount of edges that are left bidirectional, explained by Figure 5.4. Shown are results for square (a), hexagonal (b) and triangular (c) lattices.

Figure 5.5 presents the results of introducing a fraction of directed edges to lattice reservoirs. Most reservoirs show reduced error rates with increasing fractions of directed edges.

A small exception is visible for very small square and hexagonal reservoirs, where a fully directed reservoir may degrade if the edges align in an insufficient manner. Nodes in triangular reservoirs have six incident edges, as opposed to the three and four of square and hexagonal, thus giving a smaller chance of insufficient reservoirs appearing. Note that this problem of generating insufficient directions disappears with bigger reservoirs. This compelling result indicates that our method of generating directed edges guarantees good reservoirs as their sizes increase – one does not need to "get lucky" with directions.

Convincing improvements are exhibited once reservoirs become fully directed and sufficiently big, which is especially visible at the sudden drops at the closest points of the surface areas. The drops are only sudden when compared to reservoirs of a lower fraction of directed edges. We plot the cross section of the surface areas at full directedness in Figure 5.6, showing that the error rates keep decreasing with increased reservoir size. This is again in stark contrast to their undirected counterparts, which in Section 5.1, and also previous in Section 4.3 only perform marginally better as reservoir size is increased.

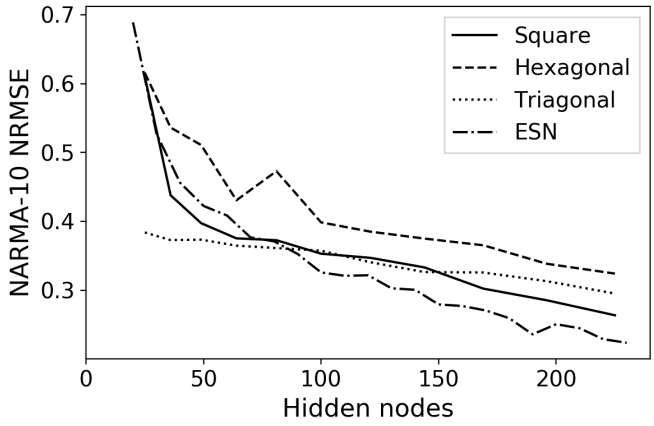


Figure 5.6: Benchmark error as a function of size for directed lattice reservoirs. All reservoirs are fully directed, and the direction of each edge is decided by an unbiased coin toss.

Thus, twice we have found directed edges to improve reservoir performance significantly, once in random geometric graph reservoirs in Section 4.3, and now again in lattices. Despite this, lattice reservoirs still perform slightly worse than ESNs.

In the project preceding this thesis, it was found that a global input scheme, i.e. an input scheme where every input weight is set to 1 and then scaled, works with ESNs given appropriate scaling [6]. This simple input scheme is relevant to physical RC, given physical substrates in which every node is forced to see the same input. Results from using this input scheme with square grids are shown in Figure 5.7.

It is abundantly clear that the global input scheme works well with the square lattice. In fact, it scales even better with reservoir size than the regular ESNs used in our experiments. Additionally, we have also included error rates for a square grid with a sparser input scheme in which only 50% of the hidden nodes see the input. These networks marginally improve even further upon the performance of square lattices.

Table 5.1 shows the simplicity of the square grids used in Figure 5.7. The input of the entire reservoir is decided by a single scalar. Furthermore, there is only a single unique magnitude used for reservoir weights in $\mathbf{W}^{res}$, which is determined by the spacing of nodes. ESNs shown in Table 5.1 contain a large amount of unique reservoir weights, but achieve a worse NRMSE on the NARMA-10 benchmark.

Simple cyclic reservoirs (SCR) use topology in a similar manner to use a deterministic weighting scheme [51]. Units in SCRs are organized in a cycle, with a single unique weight magnitude. All input connections have the same absolute weight

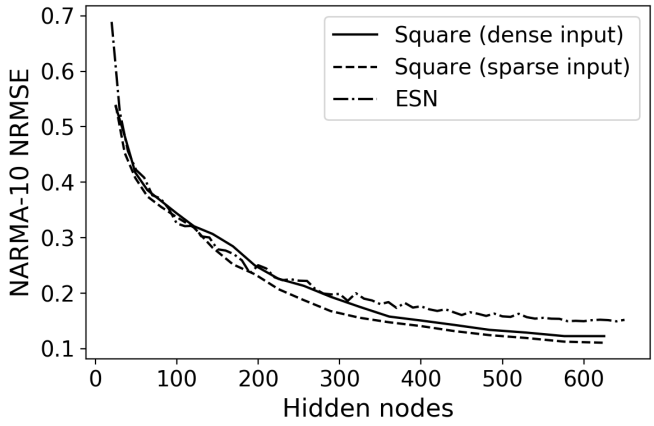


Figure 5.7: NRMSE of square lattice reservoirs with global, fixed input schemes compared to standard ESNs. With the sparse input scheme only 50% of the hidden nodes see the input.

value, but the sign is determined by means of an unbiased coin. The intention with SCR is to remove stochasticity in reservoir generation. With square lattices a similar strategy is employed. However, we do not use a stochastic scheme of generating signed input values, but instead generate the flow of information in this manner. This leaves an opportunity to study the structures that are generated, allowing us to peer into the apparent black box of the ESN, as we shall see in Section 5.4.

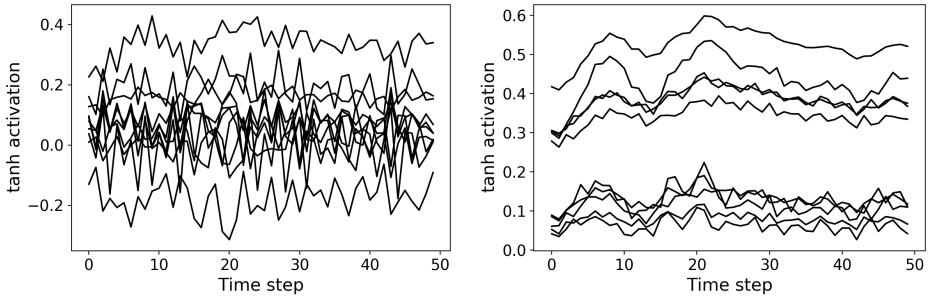
The time evolution of internal node activations proves to be of interest. Figure 5.8 depicts an example run, comparing the standard ESN to square grids with a fixed, global input scheme. It is clearly visible that nodes in Figure 5.8b receive the same input, while the activations of the ESN in 5.8a seems more sporadic due to its uniform input distribution in the interval [-0.5, 0.5].

Note also that the activations of nodes in a square lattice reservoir are strictly positive, as the NARMA input sequence is strictly positive. This may warrant a change of activation function to other sigmoid functions, as half of *tanh* remains unused, but this has not been investigated further.

In summary, our experimental results demonstrate the potential of designing reservoirs in a non-stochastic manner. By introducing directed edges to spatially constrained lattice reservoirs, we have stumbled upon reservoirs that perform exceptionally well on the NARMA-10 benchmark. These results suggest that, in a physical RC context, physical substrates will show degraded performance unless there is a directed flow of information. Additionally, we propose directed lattice reservoirs as a means to explore the impact of information flow further.

Table 5.1: Simplicity of the weighting scheme of square grids. Square grid reservoirs contain a single unique magnitude for input and reservoir weights. Displayed values are given as an average across 20 experiment runs (std. dev.).

Reservoir type	Hidden nodes	Unique input weights	Unique reservoir weights	NARMA-10 NRMSE
Square grid	100	1	1	0.346 (0.019)
Square grid	225	1	1	0.245 (0.022)
Square grid	400	1	1	0.168 (0.009)
ESN	100	100	997 (26)	0.388 (0.019)
ESN	225	225	5098 (74)	0.282 (0.019)
ESN	400	400	16070 (113)	0.215 (0.021)



(a) **(b)**

Figure 5.8: Evolution of internal node activations for ESN (a) and square lattice with global input (b). Reservoirs contain $N = 144$ nodes, but only a subset of 10 is shown to avoid clutter.

5.3 Nonlinear Dynamics in Square Grids

5.3.1 Synopsis

In this section we look at the potential diversity of nonlinear operations in square lattice reservoirs. Although the NARMA-10 presents a task of nonlinear operation, we herein conduct experiments to determine the kernel quality and generalization capabilities of the square lattice reservoirs, and run benchmarks with the Mackey-Glass benchmark to generate a mildly chaotic attractor ($\tau = 17$).

5.3.2 Results and Discussion

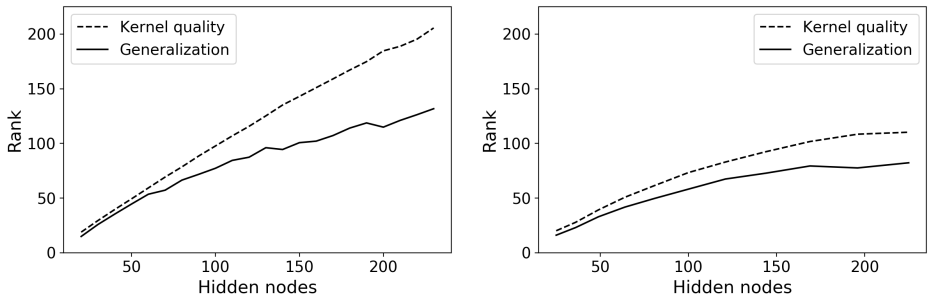


Figure 5.9: Kernel quality and generalization capability as a function of reservoir size for ESN (a) and square lattice reservoirs (b).

Consider Figure 5.9, comparing the kernel quality of the default ESN to square lattice reservoirs. The parameters of the reservoirs remain the same as in Section 5.2, except a scaling of the ESN spectral radius to 0.7, as to avoid complete saturation of the respective ranks. We discover that square lattice reservoirs attain lower kernel qualities. Additionally the difference between kernel quality and generalization is also higher for ESNs. This difference is often used as a metric of reservoir quality, as a high kernel quality and low generalization rank is desirable.

If we examine kernel quality (KQ) and generalization rank (G) as a behavior space $KQ : G$, sweeping parameter spaces will reveal the flexibility of reservoir types. For all reservoirs, $KQ < N$ and $G < N$, given reservoir size N. It is known that the standard, fully-connected ESN may be tuned to have almost any $KQ : G$, while lattice reservoirs cover a smaller area of the $KQ : G$ behavior space, as they are unable to achieve a high KQ [66], which is a conclusion that is partly reproduced in Figure 5.9.

In Section 2.5.2, we noted a discrepancy between performance predicted by kernel quality and benchmarks used by Rodan and Tiňo in their work on ring topologies. Ring topologies cover a smaller area of the $KQ : G$ behavior space than fully-connected ESNs. Nonetheless, cyclic reservoirs with regular jumps (CRJ) consistently outperform ESNs across multiple benchmarks [64]. We see a similar outcome in our experiments with square lattices, where reservoirs perform comparably on the benchmark, but exhibit a lower kernel quality. This strengthens a conclusion that deterministically constructed reservoirs can perform well, given appropriate tasks.

Figure 5.10 shows NRMSE achieved for square lattice reservoirs on the Mackey-Glass delay differential equation with mild chaos, $\tau = 17$. Again, see a comparable

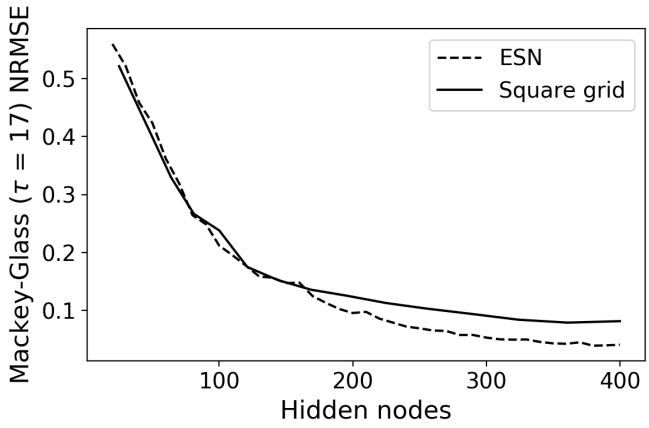


Figure 5.10: NRMSE as a function of reservoir size. Note that this benchmark is the Mackey-Glass chaotic attractor with $\tau = 17$.

performance, although in this case the ESN model scales slightly better. We would like to stress that previous work commonly compares reservoirs up to a size of $N = 200$, in which case both Figure 5.7 and Figure 5.10 show ESN and square lattice reservoirs to perform equivalently. Thus, from these benchmarks we may gather that there is potential in lattice reservoirs as simulation models, most pressingly as a tool for theoretical analysis rather than an ESN competitor.

5.4 Shrinking and Growing Square Grids

5.4.1 Synopsis

In Section 5.2 we asserted that the directedness of square lattice reservoirs will allow us to peer into the "black box" character of reservoirs. Traditional reservoir generation is driven by ad-hoc methodology, resulting in reservoirs with internals that are difficult to interpret. Hence, constructing simpler, more deterministic reservoirs will allow us a clearer view of where and how input will flow in the network. Lattice reservoirs follow a straightforward connectivity scheme and are embedded in space, making analysis easier.

In this section we attempt to gain a deeper understanding into what makes a square lattice reservoir perform well on the NARMA-10 benchmark. First we remove nodes from the lattice in an incremental manner, where each iteration removes the node that results in the lowest increase in error. Then we take the opposite route, adding nodes along the frontier of the lattice, always adding the node and directing the

edges in the manner causing the largest decrease in error. Both shrinking and growing of square lattice reservoirs thus follow a simplistic, exhaustive approach.

5.4.2 Results and Discussion

Shrinking Reservoirs

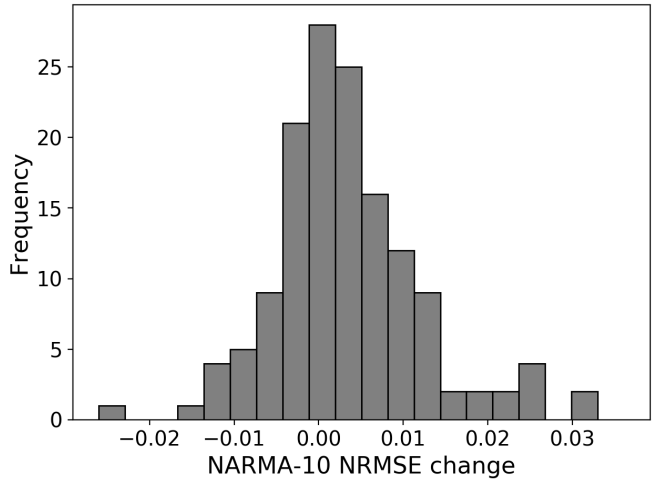


Figure 5.11: Impact of node removal from a 12×12 square lattice reservoir. The original NRMSE of the reservoir is 0.28.

We begin the experiment by creating a single 12×12 square lattice reservoir and evaluating the impact of removing singular nodes in individual copies. Figure 5.11 shows the distribution of the impact the removal of a node has on a reservoir with an original benchmark NRMSE of 0.28. Few node removals make a difference, and the few that do only change the NRMSE marginally. This speaks to an inherent robustness in the reservoir, as dead nodes do not degenerate performance entirely. Interestingly, a minority of nodes provide a decrease in error upon removal, indicating a noisiness that causes a hinderance.

When node removals have been exhausted, the node causing the smallest increase in benchmark error is removed. This is then repeated until there is a single node left in the reservoir. Figure 5.12 details the development of reservoir error for each iteration. We also conduct the same experiment with default ESNs, comparing both models to randomly generated ESNs of the same size.

Results show that both square lattice reservoirs and ESN reservoirs benefit from removal of a few noisy nodes. Furthermore, almost half the reservoirs may be

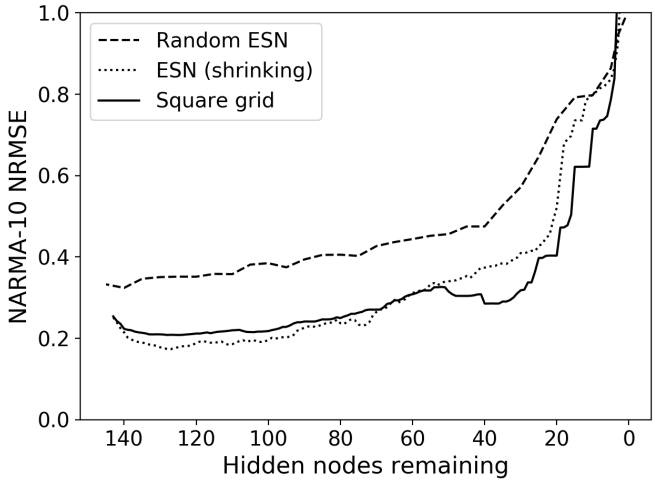
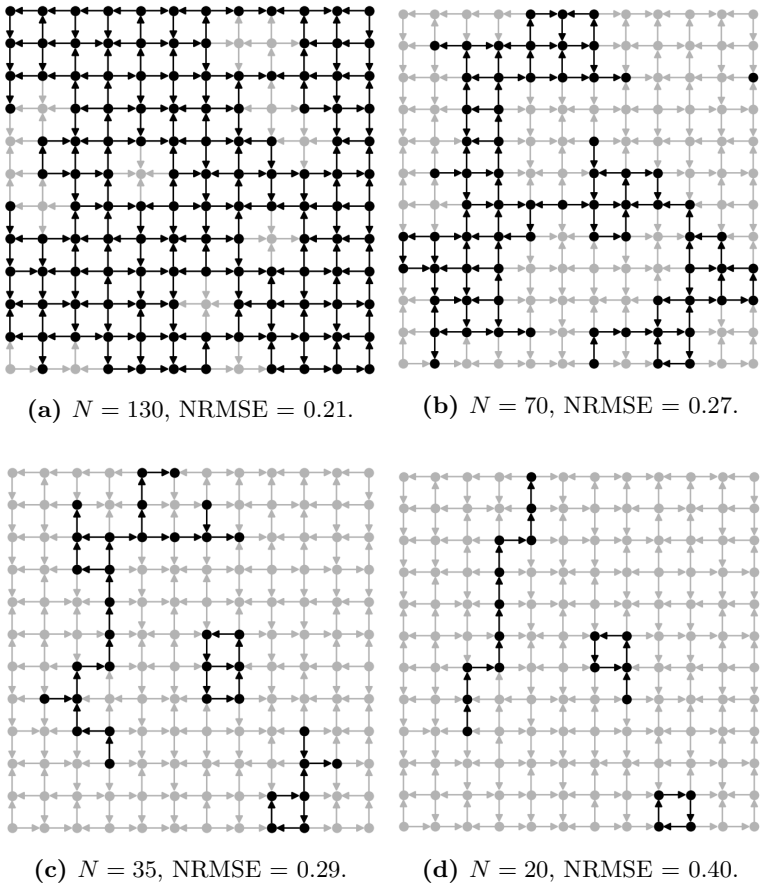


Figure 5.12: NRMSE as a function of nodes removed from the original reservoir. Both traditional ESN and square lattice reservoirs are investigated, and compared to randomly a randomly generated ESN of the same size.

removed in this manner before performance degrades to values below that of the initial geometry. Clearly, numerous of the hidden nodes in the reservoirs that were generated originally serve little purpose. Removing such nodes result in a marginal increase in benchmark error, or even a decrease in some cases. Pruning ESNs has been attempted before. For example, by pruning output connections as a regularization method [74], or by using the concept of attack tolerance to remove nodes which output weights $\mathbf{W}^{out}$ correspond to large or small values [75]. Results that indicate optimization potential in network pruning are interesting in a physical RC context, as the resources required to realize reservoirs naturally increases with size.

Pruning reservoir size is relevant not only to reduce costs, it is also valuable for theoretical analysis. Figure 5.13 depicts a snapshot of reservoir geometry at four different points during shrinking. Performance degrades to that of a delay line when there is around $N = 20$ nodes left in the reservoir, shown Figure 5.13d. This is also exactly what is remaining: a delay line of length 11, and two cycles of length 4. Moreover, an NRMSE of 0.29 is achieved with just $N = 35$ nodes in Figure 5.13c. We see a "core" providing the required short-term memory, with nodes along this stem for processing purposes. This core is also present in the original 144-node network, but is now easier to spot.

We have thus illustrated the value of square lattice reservoirs as an analytical tool. This rather simple experiment has deduced that heavy lifting to solve the NARMA-10 benchmark is done by a core stem of memory augmented by surrounding nodes.



(a) $N = 130$, NRMSE $= 0.21$. **(b)** $N = 70$, NRMSE $= 0.27$.

(c) $N = 35$, NRMSE $= 0.29$. **(d)** $N = 20$, NRMSE $= 0.40$.

Figure 5.13: Evaluation during incremental removal of nodes from a 12×12 square lattice reservoir.

A comparable analysis is harder to make for the corresponding shrunk ESN reservoir from Figure 5.12, as even just embedding the ESN network in a metric space requires complicated spring models such as force-directed graph drawing.

In summary, in this section we have deviated slightly from the original goal of the thesis relating spatial constraints to physical substrates to investigate the spatial structures that emerge in lattice reservoirs that solve the NARMA-10 benchmark. We suggest the analysis herein to be only the tip of the iceberg, as heuristics such as average node degree, investigation of the cyclic structures, and adding skip edges to square lattices to improve our understanding of the ESN black box.

Growing Reservoirs

Growing reservoirs in the same manner as shrinking them is an enticing approach. There is a finite amount of positions where new nodes may be inserted into the network, and for each such position there is only a handful of ways to direct the incident edges. By exhausting all possibilities, we may attach the node that causes the largest decrease in benchmark error.

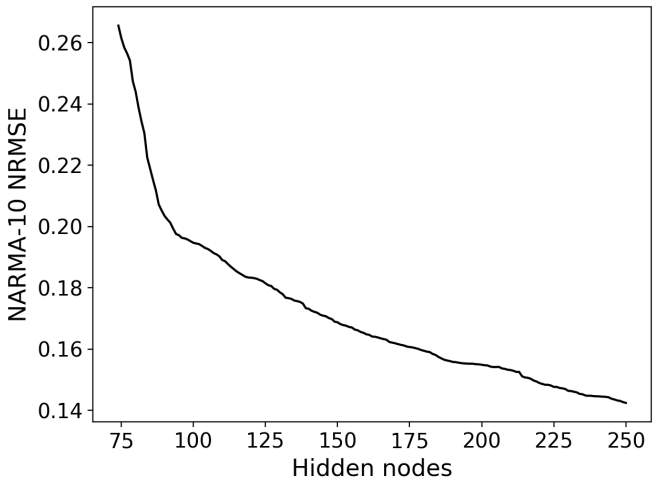


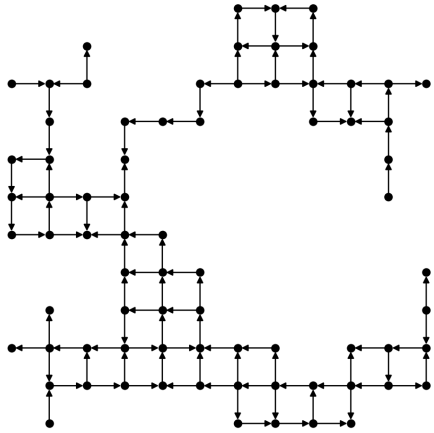
Figure 5.14: NRMSE as a function of hidden nodes when growing a square lattice reservoir. The process is begun from a reservoir of size $N = 74$, depicted in Figure 5.15a.

Figure 5.14 shows the NRMSE development of a reservoir grown with this method. We begin with a node of $N = 74$ hidden nodes, found to be the best performing reservoir when shrinking a 12×12 square lattice reservoir in the previous section. We see a steady decrease in benchmark error for each node addition. Achieving a great benchmark score is of course desirable, as the results clearly show that building reservoirs specialized for specific tasks will outperform randomly generated ones.

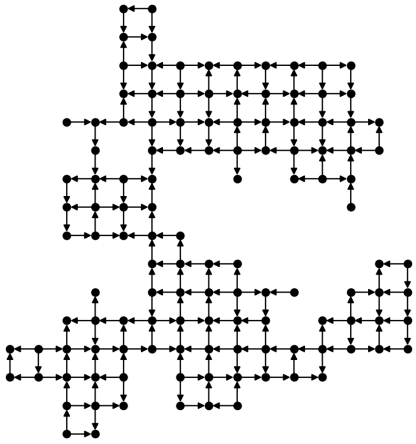
However, as previously mentioned, our intention with square lattice reservoirs is not to provide yet another ESN competitor, but to gain insights into their inner workings. Figure 5.15 depicts the reservoir at three different points during its growth in Figure 5.14. Curiously, holes appear in the lattice in Figure 5.15c, demonstrating that there are parts of the lattice where adding nodes, irrelevant of the directions of its incident edges, will cause little performance gain.

Figure 5.15c also illustrates the apparent difficulty of analyzing networks without good heuristics. Consider for example the difference between Figure 5.15c, and the

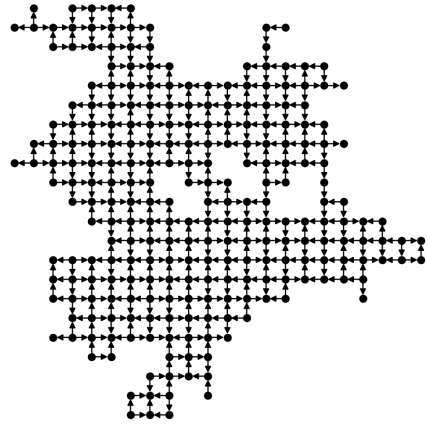
much smaller Figure 5.13c. The latter provides a much clearer picture of what it is *doing* during the NARMA-10 benchmark. We thus refer to the approaches suggested in the summary of the previous section on shrinking reservoirs – there seems to be tremendous potential in researching the creation of structured, deterministic reservoirs.



(a) $N = 74$, NRMSE $= 0.26$.



(b) $N = 124$, NRMSE $= 0.18$.



(c) $N = 250$, NRMSE $= 0.15$.

Figure 5.15: Evaluation during incremental addition of nodes to a square lattice reservoir. The process starts from the reservoir in (a), resulting in a decrease in benchmark error according to Figure 5.14.

5.5 Restoring Bidirectional Edges

5.5.1 Synopsis

Among the discoveries of our previous experiments, the importance of directed edges to create a flow of information is of major relevancy to physical RC settings. In this section we analyze the effect of restoring bidirectional edges by following the greedy approach in Section 5.4. A base 12×12 square lattice reservoir is generated, and its 264 edges are incrementally made bidirectional by choosing the edge causing the least increase in error for each iteration.

5.5.2 Results and Discussion

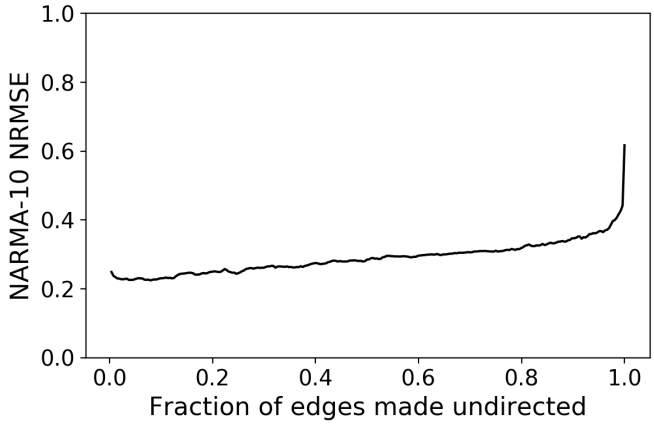


Figure 5.16: NRMSE as a function of the fraction of edges of a square lattice reservoir that are changed to be undirected. A 12×12 lattice is used, starting from 264 directed edges.

Results are shown in Figure 5.16. First, there is a small dip in error for the initial 20 or so edges. When an edge goes from directed to undirected, its entry in $\mathbf{W}^{out}$ is in practice mirrored along the diagonal axis, essentially *adding* an incident edge in the opposite direction. When greedily choosing which edges to change, it is unsurprising that there are instances where an additional edge is beneficial.

As an increasing fraction of edges made undirected, the reservoir benchmark error rises steadily. Performance degrades drastically after about 90% of edges have been changed. Again we see that a non-symmetric weight matrix enables richer dynamics, here as a direct consequence of the restored symmetry.

5.6 Conclusions

The goal of this chapter was to investigate the computational feasibility of lattice models as reservoirs. We have investigated how the fixed geometry can cause degradation in reservoir quality, and evaluated methods to deterministically construct spatially restricted reservoirs.

Analysis of bidirectional lattice reservoirs revealed that there is little difference between square, hexagonal and triangular tilings. The marginal difference between the different tilings again suggest that the *overall structure* of the reservoir determines its feasibility, as the difference between the tilings account for little more than the number of edges incident each node. This simple, sparse lattice architecture was shown to provide decent results overall.

When edges of the lattice were changed to be directed, reservoir quality increased to values almost comparable to traditional ESNs. Changing the input scheme to a fixed, global input saw an additional gain in performance, outperforming traditional ESNs on the NARMA-10 benchmark. The importance of directedness found in Chapter 4 thus resurfaced in lattice experiments, strengthening previous conclusions.

Interestingly, there is a discrepancy between the reservoir quality predicted by the kernel quality, and the resulting benchmark evaluation. We found that lattice reservoirs in general may attain a narrower range of kernel qualities than ESNs, but perform comparably on benchmark tasks. Equivalent conclusions have been drawn for ring topologies previously, illustrating that deterministically constructed reservoirs may perform well on suitable tasks.

By deterministically removing and adding nodes to lattice reservoirs, we illustrated methodology to explore their inner workings. For example, for the NARMA-10 benchmark we found that lattice reservoirs seem work by augmenting a "core" stem of nodes which make up the short-term memory, with augmentative nodes around it. We reasoned that constructing reservoirs in deterministic ways may pave the way to a deeper understanding of ESN internals.

Hence, we argue that there are three main contributions in this chapter. (i) We found physical reservoirs with lattice structures to be feasible, but may require imposed directedness for scalability. (ii) We presented a lattice model suitable for theoretical analysis of ESN internals. (iii) We showed example analyses, suggesting further methods and heuristics.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we have explored physical reservoir computing with spatial constraints. Our goal was to provide a better theoretical foundation for how reservoir computing methodology translates to physical substrates. We conducted software simulations using echo state network methodology to investigate the feasibility of reservoirs with two different spatial restrictions: random geometric graphs and lattices. Furthermore, we have developed lattice reservoirs that can be used for theoretical analysis of ESN internals, accompanied with example analyses of deterministic reservoir construction.

RQ1: How does the reservoir computing paradigm translate to the spatially constrained topology setting of a physical medium?

The difference from abstraction models such as ESNs to physical reservoirs is primarily concerned with the *limitations* posed when interacting with an actual physical substrate. In this work we have investigated spatial limitations, and in Section 4.4 we emphasized that the weight distribution resulting from a spatially constrained reservoir is different from unrestricted architectures. The translation of the RC paradigm to a spatial, physical setting is thus primarily limited by the (possibly fixed) geometries of the underlying substrate and how its internal units are connected. Other limitations than spatial constraints, such as noise and system observability, were explored in preliminary work [6].

More specifically, we have shown that ESNs with imposed spatial limitations by default show a decrease in performance compared to their abstract counterparts.

However, both RGG and lattice reservoirs achieved improved performance once the symmetry in the resulting internal reservoir matrix was broken by directed and signed edges.

RQ2: How do highly regular, physical structures compare in information processing capability to that of established models such as echo state networks?

Highly regular lattice structures with *bidirectional* edges perform worse than ESNs. However, our investigations also demonstrated that introducing directedness to lattice reservoirs restores the ESN performance, indicating that structures that allow a definitive flow of information may be sufficient. We also showed that a fixed, global input betters performance compared to a standard uniform distribution, making lattice reservoirs scale better than traditional ESNs on the NARMA-10 benchmark. We thus observe that regular structures may perform just as well as established models, as both previous work on ring topology as well as the work in this thesis suggests that deterministically constructing regular reservoirs reveals great potential.

RQ3: Can we find simple, deterministic reservoir generation methodology, relying less on random weighting schemes?

It is previously established that ring topology models may be constructed deterministically to serve as quality reservoirs [51]. In this thesis we introduced lattice reservoir models, which are constructed by deterministically placing nodes on a two-dimensional grid. Both ring and lattice reservoirs, however, require some stochastic element to work well. In the case of rings the sign of the input seen by each node is determined by an unbiased coin, while in the case of lattices it is the direction of the edge between each node that is decided by a coin flip. We therefore argue that there is untapped potential in creating reservoirs of deterministic, regular natures.

Finally, by removing and adding nodes to lattice reservoirs, we illustrated methodology for theoretical analysis of the inner workings of such networks. Firstly, we found that a removal of a few nodes can be a convenient measure to reduce network size and improve prediction performance. More importantly, we also argue that understanding the behavior of networks when solving specific benchmarks is a stepping stone to looking into the general black box behavior of ESNs, and that the regular structure of lattice reservoirs is easier to observe directly than the more stochastic and abstract structure of ESNs. Hence, the intention of lattice reservoirs is not to provide yet another ESN competitor, but to open up for further analysis in future work.

6.2 Future Work

As there is always an abundance of future work to conduct, we here limit suggestions to two main categories of experiments we deem the most interesting: physical realization of reservoirs to determine if directed exchange and flow of information is sufficient to create quality reservoirs, and deeper theoretical analysis using lattice models to find interesting heuristics in good reservoirs.

First, as ASI has proven to be a promising substrate for reservoir computing, it is a good candidate for study of real physical limitations [12]. In this thesis it is suggested that directedness is a key component in good reservoirs, and how this translates to an actual physical medium would be interesting to explore. For example, imposing shift register structures onto ASI to allow for directed flows may hold a potential for creating quality reservoirs.

Second, further theoretical analysis using lattice models should result in a deeper understanding of why the traditional, stochastic ESN model works so well. By modifying lattice models to find good heuristics such as average node degree, investigating cyclic structures, adding skip edges, and so on, we may discover what it is that makes ESN reservoirs "tick". Specifically, lattice reservoirs are inherently embedded in space, making them easier to visualize and understand. This is valuable when attempting to understand how specific tasks are solved by an ESN.

Bibliography

[1] J. F. Miller, S. L. Harding, and G. Tufte, "Evolution-in-materio: evolving computation in materials," *Evolutionary Intelligence*, vol. 7, pp. 49–67, Apr. 2014.

[2] M. Dale, J. F. Miller, and S. Stepney, "Reservoir Computing as a Model for In-Materio Computing," in *Advances in Unconventional Computing* (A. Adamatzky, ed.), vol. 22, pp. 533–571, Cham: Springer International Publishing, 2017.

[3] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," *GMD-Report 148, German National Research Institute for Computer Science*, 2001.

[4] B. Schrauwen, "An overview of reservoir computing: theory, applications and implementations," *Proceedings of the 15th European Sympsosium on Artificial Neural Networks*, pp. 471–482, Apr. 2007.

[5] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent Advances in Physical Reservoir Computing: A Review," *arXiv:1808.04962 [cs]*, Aug. 2018. arXiv: 1808.04962.

[6] T. Aven, "Exploring Reservoir Computing with physical limits using Echo State Networks," project report in TDT4501, Department of Computer Science, NTNU – Norwegian University of Science and Technology, Dec. 2019.

[7] A. Atiya and A. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," *IEEE Transactions on Neural Networks*, vol. 11, pp. 697–709, May 2000.

[8] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, July 1977.

[9] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural Networks*, vol. 20, pp. 323–334, Apr. 2007.

[10] H. Jaeger, "Short Term Memory in Echo State Networks," *GMD - German National Research Institute for Computer Science*, 2002.

[11] J. H. Jensen, E. Folven, and G. Tufte, "Computation in artificial spin ice," in *The 2018 Conference on Artificial Life*, (Tokyo, Japan), pp. 15–22, MIT Press, 2018.

[12] J. H. Jensen and G. Tufte, "Reservoir Computing in Artificial Spin Ice," p. 8, 2020.

[13] F. Heylighen, "The Science of Self-Organization and Adaptivity," p. 27, 1999.

[14] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Physica D: Nonlinear Phenomena*, vol. 42, pp. 12–37, June 1990.

[15] K. Doya, "Bifurcations of Recurrent Neural Networks in Gradient Descent Learning 3," p. 11.

[16] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, pp. 2531–2560, Nov. 2002.

[17] O. Yilmaz, "Reservoir Computing using Cellular Automata," *arXiv:1410.0162 [cs]*, Oct. 2014. arXiv: 1410.0162.

[18] D. Snyder, A. Goudarzi, and C. Teuscher, "Computational Capabilities of Random Automata Networks for Reservoir Computing," *Physical Review E*, vol. 87, Apr. 2013. arXiv: 1212.1744.

[19] A. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*. Winston and Sons, 1977.

[20] R. Penrose, "A generalized inverse for matrices," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406–413, July 1955.

[21] M. Lukoševičius, "A Practical Guide to Applying Echo State Networks," in *Neural Networks: Tricks of the Trade* (G. Montavon, G. B. Orr, and K.-R. Müller, eds.), vol. 7700, pp. 659–686, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[22] H. Jaeger, "A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach," *GMD-Forschungszentrum Informationstechnik, 2002*, p. 46, 2002.

[23] C. M. Bishop, "Training with Noise is Equivalent to Tikhonov Regularization," *Neural Computation*, vol. 7, pp. 108–116, Jan. 1995.

[24] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt, "Improving reservoirs using intrinsic plasticity," *Neurocomputing*, vol. 71, pp. 1159–1171, Mar. 2008.

[25] Y. Xue, L. Yang, and S. Haykin, "Decoupled echo state networks with lateral inhibition," *Neural Networks*, vol. 20, pp. 365–376, Apr. 2007.

[26] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, Dec. 2017.

[27] H. Jaeger, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, pp. 78–80, Apr. 2004.

[28] Y. An, Q. Song, and X. Zhao, "Short-term traffic flow forecasting via echo state neural networks," in *2011 Seventh International Conference on Natural Computation*, (Shanghai, China), pp. 844–847, IEEE, July 2011.

[29] Q. Song, X. Zhao, Z. Feng, Y. An, and B. Song, "Hourly electric load forecasting algorithm based on echo state neural network," in *2011 Chinese Control and Decision Conference (CCDC)*, (Mianyang, China), pp. 3893–3897, IEEE, May 2011.

[30] X. Lin, Z. Yang, and Y. Song, "Short-term stock price prediction based on echo state networks," *Expert Systems with Applications*, vol. 36, pp. 7313–7317, Apr. 2009.

[31] E. Aislan Antonelo and B. Schrauwen, "On Learning Navigation Behaviors for Small Mobile Robots With Reservoir Computing Architectures," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 763–780, Apr. 2015.

[32] S. Harding and J. Miller, "Evolution In Materio : A Real-Time Robot Controller in Liquid Crystal," in *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, (Washington, DC, USA), pp. 229–238, IEEE, 2005.

[33] P. Joshi and W. Maass, "Movement Generation and Control with Generic Neural Microcircuits," in *Biologically Inspired Approaches to Advanced Information Technology* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. J. Ijspeert, M. Murata, and N. Wakamiya, eds.), vol. 3141, pp. 258–273, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[34] K. Bush and C. Anderson, "Modeling reward functions for incomplete state representations via echo state networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, (Montreal, Que., Canada), pp. 2995–3000 vol. 5, IEEE, 2005.

[35] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Communications*, vol. 5, Dec. 2014.

[36] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[37] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv:1406.1078 [cs, stat]*, Sept. 2014. arXiv: 1406.1078.

[38] C. Gallicchio, A. Micheli, and L. Pedrelli, "Comparison between DeepESNs and gated RNNs on multivariate time-series prediction," *arXiv:1812.11527 [cs, stat]*, Nov. 2019. arXiv: 1812.11527.

[39] L. Büsing, B. Schrauwen, and R. Legenstein, "Connectivity, Dynamics, and Memory in Reservoir Computing with Binary and Analog Neurons," *Neural Computation*, vol. 22, pp. 1272–1311, May 2010.

[40] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, "Information Processing Capacity of Dynamical Systems," *Scientific Reports*, vol. 2, Dec. 2012.

[41] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen, "Memory versus non-linearity in reservoirs," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, (Barcelona, Spain), pp. 1–8, IEEE, July 2010.

[42] M. Inubushi and K. Yoshimura, "Reservoir Computing Beyond Memory-Nonlinearity Trade-off," *Scientific Reports*, vol. 7, p. 10199, Dec. 2017.

[43] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, pp. 391–403, Apr. 2007.

[44] D. Verstraeten and B. Schrauwen, "On the Quantification of Dynamics in Reservoir Computing," in *Artificial Neural Networks – ICANN 2009* (C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, eds.), vol. 5768, pp. 985–994, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[45] L. Livi, F. M. Bianchi, and C. Alippi, "Determination of the edge of criticality in echo state networks through Fisher information maximization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 706–717, Mar. 2018. arXiv: 1603.03685.

[46] T. E. Gibbons, "Unifying quality metrics for reservoir networks," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, (Barcelona, Spain), pp. 1–7, IEEE, July 2010.

[47] J. Chrol-Cannon and Y. Jin, "On the Correlation between Reservoir Metrics and Performance for Time Series Classification under the Influence of Synaptic Plasticity," *PLoS ONE*, vol. 9, p. e101792, July 2014.

[48] C. Fernando and S. Sojakka, "Pattern Recognition in a Bucket," in *Advances in Artificial Life* (G. Goos, J. Hartmanis, J. van Leeuwen, W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, eds.), vol. 2801, pp. 588–597, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[49] N. Bertschinger and T. Natschläger, "Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks," *Neural Computation*, vol. 16, pp. 1413–1436, July 2004.

[50] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the Liquid State Machine: a case study," *Information Processing Letters*, vol. 95, pp. 521–528, Sept. 2005.

[51] A. Rodan and P. Tino, "Minimum Complexity Echo State Network," *IEEE Transactions on Neural Networks*, vol. 22, pp. 131–144, Jan. 2011.

[52] L. Appeltant, M. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature Communications*, vol. 2, Sept. 2011.

[53] A. Goudarzi, P. Banda, M. R. Lakin, C. Teuscher, and D. Stefanovic, "A Comparative Study of Reservoir Computing for Temporal Signal Processing," *arXiv:1401.2224 [cs]*, Jan. 2014. arXiv: 1401.2224.

[54] H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," *NIPS'02 Proceedings of the 15th International Conference on Neural Information Processing Systems*, pp. 609–616, 2003.

[55] J. Miller and K. Downing, "Evolution in materio: looking beyond the silicon box," in *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, (Alexandria, VA, USA), pp. 167–176, IEEE Comput. Soc, 2002.

[56] M. S. Kulkarni and C. Teuscher, "Memristor-based reservoir computing," in *Proceedings of the 2012 IEEE/ACM International Symposium on Nanoscale Architectures - NANOARCH '12*, (Amsterdam, The Netherlands), pp. 226–232, ACM Press, 2012.

[57] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass, "Towards a theoretical foundation for morphological computation with compliant bodies," *Biological Cybernetics*, vol. 105, pp. 355–370, Dec. 2011.

[58] B. Jones, D. Stekel, J. Rowe, and C. Fernando, "Is there a Liquid State Machine in the Bacterium Escherichia Coli?," in *2007 IEEE Symposium on Artificial Life*, (Honolulu, HI, USA), pp. 187–191, IEEE, Apr. 2007.

[59] D. Nikolic, S. Haeusler, W. Singer, and W. Maass, "Temporal dynamics of information content carried by neurons in the primary visual cortex," in *Advances in Neural Information Processing Systems 19* (B. Schölkopf, J. Platt, and T. Hofmann, eds.), The MIT Press, 2007.

[60] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," vol. 393, p. 3, 1998.

[61] B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, Dec. 1988.

[62] M. Barthelemy, "Spatial Networks," *Physics Reports*, vol. 499, pp. 1–101, Feb. 2011. arXiv: 1010.0302.

[63] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, pp. 290–297, 1959.

[64] A. Rodan and P. Tiňo, "Simple Deterministically Constructed Cycle Reservoirs with Regular Jumps," *Neural Computation*, vol. 24, pp. 1822–1852, July 2012.

[65] C. Gallicchio and A. Micheli, "Reservoir Topology in Deep Echo State Networks," *arXiv:1909.11022 [cs, stat]*, vol. 11731, pp. 62–75, 2019. arXiv: 1909.11022.

[66] M. Dale, J. Dewhirst, S. O'Keefe, A. Sebald, S. Stepney, and M. A. Trefzer, "The Role of Structure and Complexity on Reservoir Computing Quality," in *Unconventional Computation and Natural Computation* (I. McQuillan and S. Seki, eds.), vol. 11493, pp. 52–64, Cham: Springer International Publishing, 2019.

[67] L. Manevitz and H. Hazan, "Stability and Topology in Reservoir Computing," in *Advances in Soft Computing* (G. Sidorov, A. Hernández Aguirre, and C. A. Reyes García, eds.), vol. 6438, pp. 245–256, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[68] J. H. Jensen, A. Strømberg, O. R. Lykkebø, A. Penty, M. Själander, E. Folven, and G. Tufte, "flatspin: A Large-Scale Artificial Spin Ice Simulator," *arXiv:2002.11401 [cond-mat, physics:physics]*, Feb. 2020. arXiv: 2002.11401.

[69] Q. Ma, L. Shen, and G. W. Cottrell, "Deep-ESN: A Multiple Projection-encoding Hierarchical Reservoir Computing Framework," *arXiv:1711.05255 [cs]*, Nov. 2017. arXiv: 1711.05255.

[70] I. Farkaš, R. Bosák, and P. Gergeľ, "Computational analysis of memory capacity in echo state networks," *Neural Networks*, vol. 83, pp. 109–120, Nov. 2016.

[71] D. A. Lavis, *Equilibrium Statistical Mechanics of Lattice Models*. Theoretical and Mathematical Physics, Dordrecht: Springer Netherlands, 2015.

[72] S. Tsunegi, T. Taniguchi, K. Nakajima, S. Miwa, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, "Physical reservoir computing based on spin torque oscillator with forced synchronization," *Applied Physics Letters*, vol. 114, p. 164101, Apr. 2019.

[73] A. Opala, S. Ghosh, T. C. Liew, and M. Matuszewski, "Neuromorphic Computing in Ginzburg-Landau Polariton-Lattice Systems," *Physical Review Applied*, vol. 11, p. 064029, June 2019.

[74] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin, "Pruning and regularization in reservoir computing," *Neurocomputing*, vol. 72, pp. 1534–1546, Mar. 2009.

[75] A. Haluszczynski, J. Aumeier, J. Herteux, and C. Räth, "Reservoir computing: Reducing network size and improving prediction stability," *arXiv:2003.03178 [physics]*, Mar. 2020. arXiv: 2003.03178.