Imperial College of Science, Technology and Medicine
Department of Computing

The dynamics of continuous-time recurrent neural networks and their relevance to episodic memory

Kyriacos Nikiforou

June 2019

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London and the Diploma of Imperial College London

Declaration of Originality

This is to certify that this work is my own original research. All assistance received and all collaborations have been acknowledged. This thesis has not been submitted for any other degree or work.

Kyriacos Nikiforou

Abstract

Continuous-time recurrent neural networks (CTRNNs) are both plausible models of cortical circuits and practically relevant for biologically inspired machine learning. To date, their dynamics are not well understood, especially when they are driven by external signals, and are hence typically treated as a "black box". This work builds on previous attempts to better understand the complex dynamics of CTRNNs. A series of computational experiments are presented whose aim is to reveal, through both visualisation and analysis, the dynamical transitions exhibited by these networks when driven by external periodic signals. On the applications front, the suitability of CTRNNs for episodic-like memory is explored, using the reservoir computing paradigm. This paradigm has emerged as a simple approach to overcoming important limitations of the gradient-based methods conventionally used to train recurrent neural networks. To explore their applicability to episodic-like memory, CTRNNs are trained to memorise and reconstruct sequences of high-dimensional images, representing episodic experiences. The various parameters affecting network performance are investigated, revealing a strong dependence on the dynamic regime the networks operate in. Finally, a simple computational experiment is presented using CTRNNs with modular, small-world connectivity. The aim is to visualise information broadcast in complex networks, as well as to provide computational support for the plausibility of the "connective core hypothesis", which posits a particular relationship between connectivity and information flow in the brain. The experiments and findings presented in this work offer a deeper and more intuitive understanding of the dynamics of continuous-time recurrent neural networks, and open up new avenues for future research in both their theory and application.

Acknowledgements

The work in this thesis was funded by the Engineering and Physical Sciences Research Council (EPSRC UK) through the Doctoral training fund (Award reference 1506349), the European Commission who partially funded me through the European Union Future and Emerging Technologies grant (GA:641100) for the TIMESTORM project, as well as the A. G. Leventis Foundation for their support.

Importantly, I would like to express my sincere gratitude to the following people:

- My thesis supervisor, Professor Murray Shanahan, for his never-ending support, guidance and advice over the past four years. I am deeply grateful to him for all the time he has devoted and for everything he has taught me.

- My thesis examiners, Professors Henrik Jensen and Vito Latora, for a memorable discussion and their valuable feedback that helped improve this thesis.

- All members of the Computational Neurodynamics Lab at Imperial College, for making my time there one of the most inspiring and memorable experiences in my life. Namely, I would like to thank Pedro Mediano, who spent countless hours to teach me how to be a good coder and for his contribution to the foundation of much of the work presented in this thesis. I would like to thank everyone with whom I have worked with during the TIMESTORM project, and in particular Zafeirios Fountas, who travelled around the world with me; David Bhowmik, who worked with me on developing the initial model of episodic memory; and Anastasia Sylaidi, from whom I learned many things about predictive coding. I would also like to thank Nat Dilokthanakul, for sharing his knowledge on reinforcement learning and for the great discussions over games of backgammon, as well as Christos Kaplanis and Matthew Crosby for the memorable and stimulating discussions about Artificial Intelligence and consciousness over the past years. Last, and very important, I would like to thank my dearest friend Marta Garnelo, for everything she has done for me over the past few years, and for being *the best*.

- Everyone in the Department of Computing who has made my time here possible, frictionless, and have dealt with all the necessary bureaucracy that allowed me to undertake my studies

here. In particular, I would like to thank Amani El-Kholy, Ann Halford and Hassan Patel, as well as all the members of the CSG, who kindly offered to resolve any issue over the past four years.

- All the people who have supported me at various times and friends from the Department of Computing; specifically Matthew Lee, Kai Arulkumaran, Hugh Salimbeni, Georgia Kouyialis, Marc Deisenroth, Kostantinos Kamnitsas, Amir Alansary, and Justine Courty, for our fruitful discussions and for their support.

- My house mates Charis, Pieris and Sophia, who have tolerated living in the same house as me during the past few years; as well as Kat, for her kindest support.

Last, and most important, I would like to thank my family, my sister Maria, and especially my parents, Toulla and Spyros, who have never stopped supporting me at every stage of my life. To them I owe everything, but for now, I dedicate this work.

Contents

Declaration of Originality **3**

Copyright Declaration **3**

Abstract **3**

Acknowledgements **7**

1 Introduction **21**

1.1 General Introduction 21

1.2 Motivation and Objectives 23

1.2.1 Modelling brain networks as dynamical systems 24

1.2.2 Reservoir computing for episodic-like memory 25

1.2.3 Information broadcast in modular small-world networks 27

1.3 Contributions Summary 28

1.4 Publications 32

1.4.1 Journal Papers 32

1.4.2 Conference Papers 32

1.4.3 Abstracts 32

2 Background and Methods **33**

2.1 Artificial Neural Networks . . . 33

2.1.1 Feedforward architectures . . . 33

2.1.2 Recurrent architectures . . . 36

2.1.3 Continuous-time recurrent neural networks . . . 37

2.2 Learning . . . 40

2.2.1 Training artificial neural networks . . . 42

2.2.2 Reservoir computing . . . 46

2.3 Dynamical Systems Theory . . . 57

2.3.1 States, state space and trajectories . . . 57

2.3.2 Vector fields of dynamical systems . . . 57

2.3.3 Attractors . . . 58

2.3.4 Stationary points of two dimensional systems . . . 59

2.3.5 Bifurcations . . . 69

2.3.6 Stationary points in three dimensions and higher . . . 72

3 Dimensionality of dynamics in driven continuous-time recurrent neural networks **74**

3.1 Precis . . . 74

3.2 Introduction . . . 75

3.3 Methods . . . 78

3.3.1 Network simulations . . . 78

3.3.2 Dimensionality of dynamics . . . 82

3.4 Results . . . 85
3.4.1 Dimensionality of dynamics in stable networks at the input timescale . . 86
3.4.2 Dimensionality of dynamics in stable networks at the network timescale . 91
3.4.3 Dimensionality of dynamics in unstable networks at the input timescale . 96
3.4.4 Dimensionality of dynamics in unstable networks at the network timescale 101
3.5 Discussion . . . 105
3.5.1 Reservoir computing applications . . . 106
3.5.2 Biological significance . . . 107
3.5.3 Closing remarks . . . 108

4 Stationary points in continuous-time recurrent neural networks 109

4.1 Precis . . . 109
4.2 Introduction . . . 110
4.3 Related work . . . 111
4.4 Methods . . . 114
4.4.1 Identifying stationary points . . . 114
4.5 Results and discussion . . . 119
4.5.1 The trajectory of stationary points in stable networks . . . 119
4.5.2 Beyond stable networks . . . 124
4.6 Conclusion . . . 132

5 Reservoir computing for episodic-like memory using continuous-time recurrent neural networks 135

5.1 Precis . . . 135

5.2 Introduction and background . . . 136
5.3 Methods . . . 138
5.3.1 Synthetic images . . . 141
5.3.2 Natural images . . . 144
5.4 Results . . . 147
5.4.1 Synthetic images . . . 147
5.4.2 Natural images . . . 152
5.5 Discussion . . . 161
5.5.1 Data compression using reservoirs . . . 163
5.5.2 Relation to other work on episodic memory . . . 166

6 Identifying information broadcast in complex networks with continuous-time firing-rate neurons 170

6.1 Precis . . . 170
6.2 Introduction . . . 171
6.3 Methods . . . 173
6.3.1 Network . . . 173
6.3.2 Measuring information broadcast . . . 175
6.4 Results . . . 179
6.5 Link with network dynamics . . . 187
6.6 Discussion . . . 191

7 Conclusion 194

7.1 Overview . . . 194

7.2 What have we learned and what still remains 195

7.2.1 Modelling brain networks are dynamical systems 195

7.2.2 Reservoir computing for episodic-like memory 197

7.2.3 Information broadcast in modular small-world networks 198

7.2.4 Concluding remark and outlook 199

Bibliography **199**

Appendix **222**

A **223**

A.1 Copyright Permissions 223

List of Tables

4.1 Parameters of `fsolve` used for finding stationary points. 116

5.1 Parameters of the multilayer perceptron used to classify the Columbia University Image Library dataset[NNM+96]. 142

List of Figures

2.1 Schematic of the architecture of a typical feedforward neural network. 35

2.2 Schematic of the architecture of a simple recurrent neural network. 37

2.3 Schematic of a randomly connected network. 39

2.4 Hebbian-type weight modification diagram. 41

2.5 Differentiation of the training and testing phase in Echo State Network training. 51

2.6 Vector field with stable and unstable stationary points. 62

2.7 Vector field with stable and saddle stationary points. 63

2.8 Vector field with a unstable spiral. 64

2.9 Vector field of the Lotka-Volterra system with a saddle and centre. 65

2.10 Vector field of a system with an improper unstable node. 67

2.11 Vector field of a system with a neutrally stable centre before a bifurcation. . . . 71

2.12 Vector field of a system with an unstable spiral after bifurcation. 72

2.13 Locally linear dynamics of 3-D systems with real eigenvalues. 73

2.14 Locally linear dynamics of 3-D systems with mixed real and complex eigenvalues. 73

3.1 Network connectivity used in experiments of Chapter 3. 78

3.2 Attractor dimensionality of a stable network at the input timescale. 87

3.3 Attractor dimensionality of stable networks at input timescale. 88

3.4 Estimator behaviour for stable networks at the input timescale. 89

3.5 Activity of stable networks at the input timescale. 90

3.6 Attractor dimensionality of stable networks of different sizes at the input timescale. 91

3.7 Attractor dimensionality of stable networks at the network timescale. 92

3.8 Activity of stable networks at the network timescale. 93

3.9 Euclidean distance from the final recorded point of the simulation, at the network timescale. 94

3.10 Time step of appearance of periodicity. 95

3.11 Activity of stable networks at the network timescale. 95

3.12 Average Euclidean distance between stable networks' states at the network timescale. 96

3.13 Activity of an unstable network at the input timescale. 97

3.14 Dimensionality of dynamics of unstable networks with 200 neurons at the input timescale. 98

3.15 Activity of unstable networks at the input timescale. 99

3.16 Estimator behaviour for unstable networks at the input timescale. 100

3.17 Dimensionality of dynamics of unstable networks with 800 neurons at the network timescale. 102

3.18 Dimensionality of dynamics of unstable networks with 200 neurons at the network timescale. 102

3.19 Dimensionality of dynamics of unstable networks with 2000 neurons at the network timescale. 103

3.20 Average Euclidean distance between unstable networks' states at the network timescale. 103

3.21 Activity of unstable networks at the network timescale. 104

4.1 Stationary point visualisation for stable networks. 120

4.2 Activity of stable networks projected on PC-space of the stationary point trajectory. 121

4.3 Radius of smallest enclosing sphere of stable network trajectory and maximum instantaneous distance from stationary point trajectory. 122

4.4 Comparison of stationary point trajectories. 125

4.5 Number of complex eigenvalues as signal value changes. 126

4.6 Number of eigenvalues with a positive real part as signal value changes. 127

4.7 Number of real eigenvalues as signal value changes. 128

4.8 Activity of unstable networks projected on PC-space of the stationary point trajectory. 131

5.1 Example images from the dataset of synthetic images. 141

5.2 Diagram of the training and recall phase for experiments using synthetic images. 144

5.3 Example images from the dataset of natural images. 145

5.4 Diagram of the reservoir connectivity in experiments using natural images. . . . 147

5.5 Example frames from the reconstruction of the five objects for memorising of synthetic images. 148

5.6 PCA on the reservoir trajectories during the recall of the five episodes. 149

5.7 Euclidean distance in PC-space between corresponding frames appearing in each of five episodes. 150

5.8 PCA on the reservoir trajectories during the recall of the two long episodes. . . 151

5.9 Quality of reconstructed videos for reservoirs with different gain. 153

5.10 Example reconstructed frames of two episodes from the dataset of natural images. 154

5.11 Example reconstructed frames of of varying quality. 155

5.12 Quality of reconstructed videos for reservoirs with different gain and number training loops. 156

5.13 Quality of reconstructed videos for reservoirs with varying number of training loops. 157

5.14 Quality of reconstructed videos for reservoirs with different feedback scaling. . . 158

5.15 Quality of reconstructed videos for reservoirs with different gain and probability of feedback connection. 159

5.16 Quality of reconstructed videos for reservoirs with different number of neurons. . 160

6.1 Structural connectivity of the network. 175

6.2 Average temporal mutual information and temporal transfer entropy within each module. 180

6.3 Average temporal mutual information between each module and the rest of the network. 182

6.4 Average temporal transfer entropy *from* each module *to* the rest of the network. 183

6.5 Average temporal transfer entropy *to* each module *from* the rest of the network. 184

6.6 Changes in mutual information between modules of the network around an external perturbation. 186

6.7 Changes in mutual information between modules of the network around an external perturbation. 186

6.8 Activity of the network and stationary points, projected on the PC-space of the network trajectory. 188

6.9 Number of different types of the eigenvalues of the network for different stimulus values. 190

A.1 Permission from IEEE publisher. 224

Chapter 1

Introduction

1.1 General Introduction

Over the past centuries, science has made progress in understanding nature by deconstructing the whole and studying its constituent parts. For example, the differentiation of elements and their classification into the periodic table was an important milestone in making the necessary conceptual breakthroughs that allowed the development of modern day chemistry. Following the same approach, the atom was deconstructed to reveal that it is composed from electrons, protons and neutrons, which in turn were reduced and classified into the standard model of elementary particles. Even though this reductionist approach has proven successful in generating new knowledge throughout history, there are various phenomena that we are not yet able to explain. Some of these phenomena emerge in what researchers have termed *complex systems*, systems composed of a large number of interacting components. What makes these systems interesting to study is the fact that the behaviour of the collective ensemble of the different components cannot be analytically derived solely from the properties and isolated behaviour of each individual component. The reason for this is that these systems are non-linear and often chaotic, meaning that a small difference in their initial conditions can lead to our inability to *a priori* predict the exact temporal evolution of the system. This property was first discovered by Edward Norton Lorenz – after whom the Lorenz attractor [Lor63] was named. Lorenz,

while trying to reproduce a trajectory of the evolution of a system of deterministic non-linear equations used in weather forecasting, realised that after rounding the initial conditions of the system down by a few decimal places, his calculations resulted in a completely different prediction for the weather forecast. From this observation the term *butterfly effect* was coined, which is now used to describe the huge long-term effects that small changes can have in chaotic systems, and gave birth to the mathematical theory of chaos [Gle88, Str14]. Since then, there has been a growing interest in understanding how such properties emerge and in looking for similar patterns in other complex systems.

The work presented in this thesis is inspired by the human brain, which is undoubtedly one of the most interesting natural complex systems. Even though this organ has fascinated scholars and scientists over the centuries, it is only recently that we have developed non-destructive techniques to study it. This has allowed scientists to make considerable progress over the past decades in mapping its structural and functional connectivity and to assign different functional roles to its various regions [Fri11, ZLZK11, Spo13, DCKM14]. Furthermore, a range of neuron models have been developed that aim to mimic the non-linear behaviour of neuron cells. By combining these two elements, the field of computational neuroscience has developed ways of building computer simulations of neuronal tissue, with some researchers even envisioning a full brain simulation in the near future [Mar06, DGSGR10, ESC+12]. The motivation behind this line of computational research is two-fold: First, to understand, by reconstruction, the biological mechanisms that support mental function (both healthy and diseased) for medical purposes, and second, to gain a deeper understanding of the processes that give rise to cognition and intelligent behaviour. While the former often aims to reconstruct the neuronal pathways found in the brain as closely as possible, the latter is more focused on searching for the overarching principles that allow the human brain to demonstrate general intelligence. Many scholars have claimed that such principles depend on the dynamic activity of neurons, which in turn depends on the brain's structural connectivity [Spo10, SCKH04]. For this reason, understanding more about the relationship between connectivity and dynamics will help us understand more about the brain's complex, but energy efficient, information processing and learning capabilities. Moreover, the reason that the human brain remains one of the biggest inspirations in our attempt to

create Artificial General Intelligence (AGI) is because we have still not managed to match these capabilities even with our most advanced computer algorithms and hardware.

1.2 Motivation and Objectives

Motivated by the endeavour to uncover these overarching neuroscientific principles, the present work uses a specific type of neural network model, namely the continuous-time recurrent neural network (CTRNN) [SCS88, Bee95] for computer simulations. These network type has been extensively used to model cortical activity [MSSN13, SSC13, SCKS15, EPQD16] and is thus a suitable choice of mathematical abstraction for the purpose our study, which follows three different but related approaches.

The first approach analyses these simulated networks using a well-founded, classical framework, namely dynamical systems theory, in order to understand their behaviour and explain the dynamical phenomena they exhibit. To this end, the effect of an external input on the network dynamics is explored, by examining the qualitative and quantitative changes in the attractors of the system, as well as the underlying vector field that determines the network's response.

The second one aims to show how these networks can be trained to form reproducible dynamics that can support episodic-like memory. To achieve this, a neural network is used to demonstrate that high dimensional memories, predominantly visual, can be stored in and recalled, using the dynamic activity of these networks. Furthermore, the motivation is to determine how various network parameters determine the performance in this task.

The third, and final approach, uses these networks, with a brain-inspired modular connectivity, in order to explore how information is broadcast in such networks, and provide computational evidence to support two important hypotheses about information broadcast in the brain, namely the Global Workspace Theory [Baa93] and the Connective Core Hypothesis [Sha12]. The aim is to use simple yet mathematically grounded measures from information theory to measure and visualise how different parts of the network use their dynamical activity to share information about an external stimulus. Let us now go into each of the three approaches taken in this work.

1.2.1 Modelling brain networks as dynamical systems

In order to be able to study a system like the brain, we need to first choose the level of abstraction at which it will be considered. A fundamental aspect of the human brain is that it is a dynamical system, meaning that its activity changes over time according to a set of laws. The appropriate mathematical framework for studying this type of systems is dynamical systems theory. This theory defines the properties of systems described by differential equations and provides the necessary tools for their analysis. Examples of informative features of dynamical systems are the stationary points and whether they are stable or not, and the types of attractors present in the state-space of these systems, which can explain a great deal about their behaviour [Str14]. Even though being able to directly analyse the real brain with all the tools of dynamical systems theory would be very helpful in our attempt to understand it, this is not yet possible because we do not have a full description of its dynamics in a tractable mathematical form. What can be done instead, is to simulate networks composed of artificial neurons that have been developed to generate similar dynamics to those of the brain. The model or the generated trajectories of the system can then be analysed to gain a better understanding of these dynamics.

Following this approach, the present work begins by considering continuous-time recurrent neural networks as an abstract model of cortical networks. The dimensionality of the networks' attractors subject to different simple periodic inputs is first investigated, both at the timescale of the input and that of the network. Attractors with greater dimensionality can have higher informational capacity than attractors with lower dimensionality, because the dynamics evolve in more complex ways in higher dimensions [FMR16]. Interestingly, recent work showed that the dynamics of cortical regions responsible for higher-level cognitive functions in the human brain demonstrate higher dimensionality than lower-level sensory regions [TYFT15]. In this way, the lower-dimensional dynamics of sensory areas cause the higher-dimensional dynamics that are responsible for higher-level cognition in cortical areas, a property that can be reproduced by the model used in this study. The networks are then analysed to identify how the distribution of stationary points present in their state-space and their types depend on (a) the strength of the random network connections and (b) the intensity of the input signal to the network.

These results are useful for explaining the networks' behaviour and how it depends on these parameters, which are the two most important parameters to consider when constructing such networks for machine learning applications [SA09, Mic17] or for modelling cortical regions [LB13, MSSN13, SSC13, SCKS15, CSFW17].

Of course, it does not necessarily follow that the insight gained by analysing these mathematical models can be directly applied to the dynamics of the real brain, because these insights are specific to the formulation of the model used, and there are no guarantees that they hold beyond the model. Nevertheless, this does not mean that there is nothing to be gained by analysing these models. For example, some of the techniques used, especially the ones that analyse the neural activities of the simulated network rather than the equations generating them, could be applied directly to brain data once appropriate brain activity recording techniques have been developed. These techniques will be required to provide reliable recordings with both high temporal and spatial resolution. Even though researchers have been able to use calcium imaging to obtain full-brain functional recordings in simple organisms, such as larval zebrafish [AOR$^+$13], it is yet not possible to do this for the human brain. Instead, we still rely on techniques that provide high resolution in either space or time; for example higher spatial resolution for fMRI vs. higher temporal resolution for EEG. Finally, these computational models could one day be used to support some of the cognitive functions of biologically inspired artificial systems. Understanding the dynamic behaviour of such systems will be an important requirement for improving their performance and capacity.

1.2.2 Reservoir computing for episodic-like memory

Following the investigation of the dynamical features of these simulated networks, our focus shifts towards memory, which is one of the most important, yet poorly understood cognitive capability exhibited by humans. Scientists broadly define two high-level types of memory [Squ86, MSGVK97]. One type is procedural memory, which refers to the ability to learn and reproduce sets of motor actions, for example riding a bike. The other type is declarative memory, which is the ability to learn and reproduce propositional knowledge. Declarative memory can

further be categorised in semantic memory, which is the ability to remember facts, for example recalling that Paris is the capital of France, and episodic memory, for example being able to remember and recall my trip to work this morning [TS09]. The second section of this thesis is concerned with whether continuous-time recurrent neural networks can be used as a form of distributed, dynamic memory that can support episodic-like memory in artificial agents.

Episodic memory refers to the capacity of humans to remember past personal experiences, which include information about where, when and what happened [Tul85]. It is a form of *mental time-travel*, where the person is able to re-live the visual, contextual and auditory details, for example, of a particular past event. This knowledge can then be used to take a more informed decision about a present similar situation, or even to simulate a future scenario, in order to explore the outcome of a possible action based on a particular past experience [SAC09]. The role of episodic memory is so important that researchers have tried to incorporate it into classical reinforcement learning systems. Specifically, it has recently been shown that artificial agents can improve their performance in reinforcement learning tasks, such as playing simple Atari games, if equipped with episodic-like memory [BUP$^+$16, PUS$^+$17, HKSB17], thus showing the relevance of this type of memory in performing intelligent tasks.

Many studies have shown that episodic memory formation and recall is supported by a vast distributed network of brain regions that predominantly include the hippocampus, medial temporal lobe and prefrontal cortex [Squ92, SS03, PE13]. A theory that has received increased attention and is trying to understand how the various brain regions are involved in the mechanism of memory formation and recall is the Complementary Learning Systems (CLS) theory [NO03, Nor10, KHM16]. According to the latest revision of this theory, the hippocampus and the cortex are both involved in learning and retrieving memories, with the hippocampus operating on a faster time-scale than the cortex. The presence of recurrent connections in the dentate gyrus in the hippocampus is hypothesised to support the separation of similar input patterns, in this way distinguishing between different, but related, experiences, as well as the rapid learning of new ones. The cortical regions' role on the other hand is to learn the generalised structure of the environment at a more abstract level and slower timescale. This abstract knowledge can then be used in a compositional way during memory recall, in order to reconstruct the

sequence of events taking place in an episode, by blending together smaller pieces of pre-existing knowledge. Finally, recurrent connections from the cortex to the hippocampus provide the necessary coupling between the two regions to support the mechanism of memory recall.

In Chapter 5, we will be concerned with training neural networks to perform a simple episodic-like memory task, in the paradigm of reservoir computing. The reservoir does not incorporate the aforementioned level of complexity, i.e. the interactions between the different brain regions, but rather, it uses a randomly connected network of firing rate neurons and a simple learning rule to show that high-dimensional information can be stored and retrieved through the dynamic activity of the network. To achieve this, a CTRNN is used, trained with FORCE learning [SA09], in the paradigm of reservoir computing [Jae01, MNM02, SVVC07, LJ09, HS12]. The task is to store and recall sequences of frames in a supervised manner. The most important parameters affecting the quality of the memorised and reconstructed episodes are also identified through computational experiments.

1.2.3 Information broadcast in modular small-world networks

In the final section of this work we will focus on gaining some more insight into the processes responsible for intelligent behaviour and how the organisation of the brain supports them. A very important aspect of the brain, one that can reveal a great deal about how it is organised, is its structural connectivity. It has been shown that the brain has modular small-world connectivity at the macro-scale [SCKH04, STK05, SHK07, HCG$^+$08, GHC$^+$09, Spo10, MLB10], meaning that it is made up of different modules, where regions belonging to the same module are densely interconnected, but sparsely connected to regions in other modules. It has been hypothesised that global communication between the different modules is achieved through a small number of cortical hubs, sometimes refer to as the connective core, that act as a *highway* to transmit information between regions in different modules in order to be integrated before making decisions [Baa93, SB05, DC05, Sha12]. Of course, information is not transmitted in a similar way to how conventional computers, using the Von Neumann architecture, move bits of information between the memory and the processor. Instead, information is transferred

between different brain regions in a dynamic way, i.e. through the activities of the regions. According to this view, whole networks of brain regions can interact and influence each other, using their dynamical activity as a means for broadcasting information. Various theories have been proposed along these lines that try to explain how information from multiple modalities can be integrated into a single unified experience [DC05, Sha10a, Sha12], and how relevant information from different brain regions about a potential action can be integrated to make a decision.

A suitable theoretical framework to study information transfer between various brain regions through their dynamics is information theory. Rooted in the pioneering work of Claude Shannon in 1948, "A Mathematical Theory of Communication" [Sha48], it treats information as an abstract mathematical quantity, allowing us to calculate the necessary quantities for inferring how information flows in a network. For example, scientists have successfully used it to identify the flow of information in the cortex of monkeys [SBM15] and even between the brains of different humans [WCJ14].

In this spirit, and to provide additional evidence for the connective core hypothesis [Sha12], a series of computational experiments were performed, where modular, small-world continuous-time recurrent neural networks were subject to external stimulation through one of their modules. The results show that the dynamics of CTRNNs support the connective core hypothesis, as measured by well the well-founded measures of Mutual Information and Transfer Entropy.

1.3 Contributions Summary

At a time when research in neural networks for machine learning and computational neuroscience is growing with an exploding rate, this work is positioned among those trying to decipher the "black magic" taking place inside such networks. The overarching aim of this collective effort, is to understand *how* to build models that learn more accurately from data, that are robust to variations of the data, that are able to generalise, and that can be used to build artificial systems that exhibit intelligent behaviour. Even though the research community is still far away

from its goal, the work here can help to identify complex phenomena exhibited by continuous-time recurrent neural networks and decipher a small portion of all the mysteries that remain unanswered. Following, a list outlines the specific contributions of this work towards achieving the overarching goal.

- The work presented in this thesis contributes towards revealing novel dynamical phenomena in continuous-time recurrent neural networks, exhibited when they are driven by simple periodic signals. These phenomena are:

  - The dimensionality of the network activity changes as the ratio between the timescales of the network and the period signal is varied. In particular, it is shown that a peak in this dimensionality exists over a specific range of this ratio, ρ. This observation is of interest when assessing the suitability of CTRNNs as models of cortical dynamics, since a similar phenomenon is observed when lower-dimensional signals from sensory areas in the brain, cause higher-dimensional activity in cognitive areas [TYFT15]. Hence it is shown that this feature is shared between CTRNNs and brain dynamics.
  - A variety of transient and periodic attractors, that take on distinct geometric forms, appear as the ratio of timescales, ρ, varies. The visualisation of the network attractors is shown in a reduced space, using PCA.
  - The dimensionality of the network dynamics can be different depending on the timescale over which the analysis takes place. This is observed most distinctly when the network timescale is about three orders of magnitude larger than the timescale of the input signal, and is characterised by a higher dimensionality of the network dynamics when analysed at the shorter timescale of the input, compared to when analysed at the longer timescale of the network.
  - Networks with a connectivity matrix that has a spectral radius of less than 1 – commonly referred to as the gain, g, of the network – (1) exhibit more stable dynamics, (2) have a qualitatively different vector field and (3) exhibit a peak in dimensionality over a different range of ρ values, compared to networks with $g > 1$.

- Another important contribution of this thesis is that it explains in a simple and intuitive way, the appearance of the aforementioned dynamical phenomena. To achieve this intuitive explanation, it employs a combination of qualitative and quantitative methods, as well as visualisations of the dynamics and the vector field of the network, in a reduced PC-space. More specifically, the work presented here contributes the following:

  - It reveals the structure of the vector fields of randomly connected, continuous-time recurrent neural networks using numerical optimisation, linearisation and dimensionality reduction techniques. In contrast to previous work, this is performed over the whole range of possible input values, thus visualising all the possible forms that the vector field attains, as the input signal is varied.

  - It provides a simple description, from the stand point of classical dynamical systems theory, of how g acts as a bifurcation parameter in CTRNNs, that determines the qualitative behaviour of these networks. This explanation is supported by visualisations of the vector field of networks with different g, and by considering the eigenvalues of the stationary points identified in the dynamics of these networks.

  - It shows that the amplitude of the input signal is also a bifurcation parameter, in particular for the case of networks with $g > 1$, that also causes qualitative changes in the vector field of these networks. Specifically, if the amplitude of the input signal is very high, saddle points in the vector field of CTRNNs disappear, leaving only stable stationary points. Hence, computational evidence, and an explanation of previously reported theoretical results is provided by this work, on the effect of a periodic input on the dynamics of CTRNNs.

  - It explains the mechanism that gives rise to the different forms and types of the attractors, that are reported to appear in driven CTRNNs when ρ is varied. The explanation makes use of the relative speeds with which the underlying vector field changes – as the input signal value changes – and the speed with which the network state evolves, hence, constructing an intuitive understanding of how these effects arise.

  - It uses the insights gained about the dynamics of CTRNNs from the results obtained here, to explain in a more complete – yet simple – way, the operation of simple neural circuits reported in literature [SB13], trained to perform a simple computational task. This explanation is compatible with the operation of the neural circuit, as well as the insights gained in this work, and reveals the simple conditions that need to be met by the network dynamics for solving the task.

- In this work, a novel application of CTRNNs is proposed and assessed, where, in the paradigm of reservoir computing, they are used as a medium for storing and retrieving sequences of high-dimensional data. In particular, it shows that frames of synthetic and natural videos can be memorised and reconstructed by these networks, to implement a simple form of episodic-like memory. Further contributions of this section of the work are:

  - It shows that the dynamic state of the network can maintain a memory trace of the currently reconstructed memory, such that different memories sharing very similar features can be encoded in trajectories that are very close in the network's state space. This allows the network to efficiently represent and store many different memories. Simultaneously, the trajectories used to store and recall similar – but different – memories, remain separated enough, so that each memory can be independently recalled without interference from the other ones.
  - It identifies the most important network parameters affecting their ability to memorise the episodes, and the quality of the reconstruction.
  - It shows that a compression ratio can be defined, which measures by how much can reservoirs compress the incoming, high-dimensional information, without considerable degradation in the reconstruction quality.

- Finally, information broadcast in modular small-world CTRNNs is identified, using temporal versions of Mutual Information and Transfer Entropy. This provides supporting computational evidence for the plausibility of the Connective Core Hypothesis [Sha12], which attempts to explain how the connectivity of the brain supports the sharing of information between its different regions.

1.4 Publications

What follows is a list of publications that are relevant to the work presented in this thesis.

1.4.1 Journal Papers

- **K. Nikiforou**, P. Mediano, M. Shanahan. An investigation of the dynamical transitions in harmonically driven random networks of firing-rate neurons. Cognitive Computation, 2017; 9(3), 351–363 doi:10.1007/s12559-017-9464-6.

- W. Roseboom, Z. Fountas, **K. Nikiforou**, D. Bhowmik, M. Shanahan, A. K. Seth. Activity in perceptual classification networks as a basis for human subjective time perception. Nature Communications, 2019; 10(267). doi:10.1038/s41467-018-08194-7. (*Note: Even though the work in this paper is not included in the thesis, it is an extension of the work presented in Chapter 5 on episodic memory.*)

1.4.2 Conference Papers

- D. Bhowmik, **K. Nikiforou**, M. Shanahan, M. Maniadakis, P. Trahanias. A reservoir computing model of episodic memory. 2016 International Joint Conference on Neural Networks (IJCNN), pp. 5202–5209 doi:10.1109/IJCNN.2016.7727887 (oral presentation).

1.4.3 Abstracts

- **K. Nikiforou**, P. Mediano, M. Shanahan. Identifying Information Broadcast in complex Networks. Annual Conference on Cognitive Computational Neuroscience, 2017. (poster presentation).

- **K. Nikiforou**, P. Mediano, M. Shanahan. Recurrent Neural Networks for Interval Duration Discrimination Task. Time in Tokyo: International Symposium on temporal perception and experience, Tokyo, Japan, 2016. (poster presentation).

Chapter 2

Background and Methods

This chapter covers background material regarding the networks used in this work and the various methods employed, aiming to introduce the reader to the relevant models, concepts and mathematical background. In the first section of this chapter, Section 2.1, artificial neural networks are discussed and the specific neuron model used throughout this work is introduced in detail. What follows in Sections 2.2, is an overview of reservoir computing (RC) and the learning rule used for the Episodic Memory model presented in Chapter 5. The last Section 2.3 introduces the necessary mathematical background for Chapters 3 & 4, where an analysis of randomly connected networks of firing rate neurons, from the stand point of dynamical systems theory, is presented.

2.1 Artificial Neural Networks

2.1.1 Feedforward architectures

Artificial neural networks, originating from the simple perceptron [Ros57] and inspired by the circuitry of the brain, have been extensively studied and used both in research and industry. Their use has been continuously increasing due to the availability of training data and more powerful hardware, which enable their training using previously developed but unused algorithms.

A representative example is the backpropagation algorithm [RHW86a], which even though developed in the '80s, it was not until recently that enough data and processing power were available to allow its implementation. The simple feedforward architecture and its variation known as the convolutional neural network architecture have shown excellent performance, and in many cases surpass human performance in classification tasks, for example image recognition [KSH12], and regression tasks, for example approximating Q-values of state-action pairs for reinforcement learning tasks [KNST16]. These tasks require the network to learn a sequence of transformations of the input data in order to produce the desired output. This is guaranteed by the Universal Approximation Theorem [Hor91, Csá01], which states that neural networks can approximate any continuous function on compact subsets of $\mathcal{R}^n$, under mild assumptions on the activation function and provided there are enough neurons in the hidden layer.

Fig. 2.1 below shows a simple diagrammatic representation of a feedforward neural network, showing the input and output neurons at each end as well as the hidden layer of neurons in the middle. Eq. 2.1a shows the most common update rule for the activation of neuron j in layer l, α_j^l after applying the standard sigmoid function, σ, shown in Eq. 2.1b. Other common choices for the activation function are the rectified linear unit (ReLU) and the hyperbolic tangent function. The weight from neuron k in the previous layer $l-1$ connecting to neuron j in layer l is w_{jk}^l, the bias of neuron j in layer l is b_j^l, and N^l is denotes the total number of neurons in layer l. The main idea in a feedforward network is to sequentially pass the input through the layers of the network, where each neuron in a layer first performs a linear combination of the output of the neurons from the previous layer and then a nonlinear transformation of this combination. The network weights w_{jk}^l are adjusted by the backpropagation algorithm to perform the mapping that minimises the error between prediction and target label.

$$\alpha_j^l = \sigma\Big(\sum_{k=1}^{N^l} w_{jk}^l \alpha_k^{l-1} + b_j^l\Big) \qquad l = 1, 2, 3 \tag{2.1a}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1b}$$

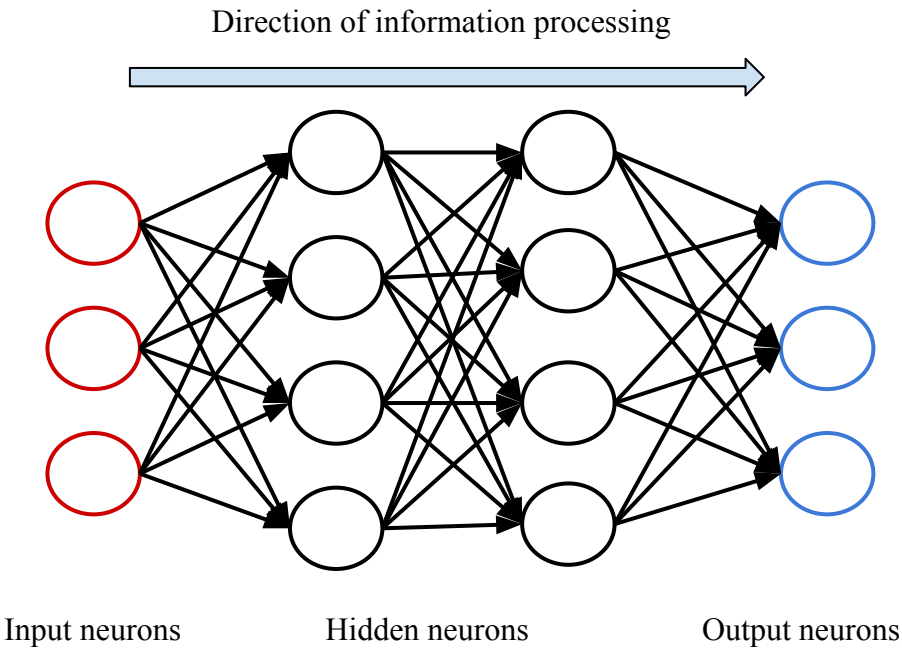


Figure 2.1: Schematic of the architecture of a typical feedforward neural network. Four layers are shown in the picture: The input layer at the far left (red circles), the two middle hidden layers where the sequential linear combinations of the previous layer outputs and nonlinear transformations thereof take place (black circles), and the output layer on the far right (blue circles). The arrow at the top shows the direction of information processing.

Even though this kind of architecture has been shown to be very good at the kind of tasks described above, it has a significant limitation. Due to the fact that each layer receives input only from its previous layer, it processes each network input independently. This means that after training, each time the same input is shown to the network, for example an image, its output will always be the same, no matter what the previous inputs were. This of course is a useful feature for the tasks described above, but not for tasks involving sequential information, which therefore require some form of memory of past inputs.

In order to address this limitation, researchers have extended this feedforward architecture with feedback connections in the hidden layers, and termed this new kind architecture Recurrent Neural Networks (RNNs). This architecture is introduced in the next section.

2.1.2 Recurrent architectures

Many machine learning problems require some type of memory of previous input. In Natural Language Processing for example, strings written in human language are the input and output to a neural network, which tries to predict the next word, given a string. The network performance considerably improves if information about the preceding text is maintained, as it sets the context and limits the possible words that can succeed it. In order to augment neural network architectures with memory, recurrent, or feedback connections are added to the vanilla feedforward architecture, as shown in Fig. 2.2.

These recurrent connections allow information from previous time steps encoded in the activity of a neuron, or other neurons in the same layer, to also be taken into account. This information is combined with the input in the current time step to make a better prediction about the word that will follow. It should be noted, that even though feedback connections make these networks recurrent, they still maintain a strong feedforward organisation. This allows RNNs to be "unfolded" into a deeper, feedforward network which is trained with the backpropagation algorithm. This method of training is known as backpropagation through time (BPTT) [Wer90], but it has been shown to suffer from the vanishing gradient problem if the resulting unfolded networks had a very large number of layers [HBF$^+$01].

The equations describing the vanilla version of RNNs are shown in Eq. 2.2. In the two equations, $\mathbf{h}(t)$, $\mathbf{x}(t)$ and $\mathbf{y}(t)$ are the activation of neurons in the hidden, input and output layer at time step t respectively and f_H and f_O nonlinear transformations for neurons in the hidden layer and output layers. The network weights are incorporated in the connectivity matrices $\mathbf{W}_{IH}$(input to hidden (black) connections in Fig. 2.2), $\mathbf{W}_{HH}$ (hidden to hidden (magenta) and self (green) connections) and $\mathbf{W}_{HO}$ (hidden to output (black) connections).

$$\mathbf{h}(t) = f_H\Big(\mathbf{W}_{IH}\mathbf{x}(t) + \mathbf{W}_{HH}\mathbf{h}(t-1)\Big) \tag{2.2a}$$

$$\mathbf{y}(t) = f_O\Big(\mathbf{W}_{HO}\mathbf{h}(t)\Big) \tag{2.2b}$$

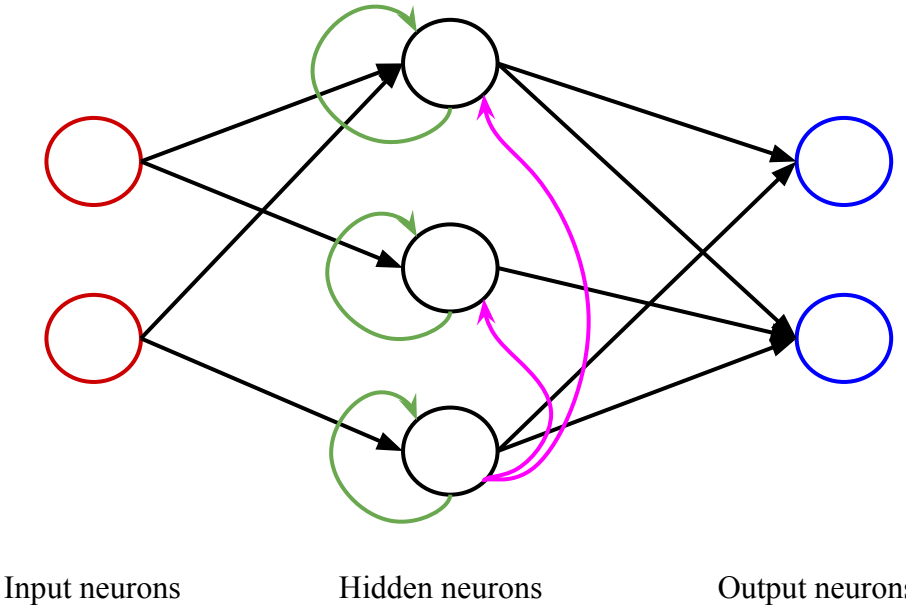


Figure 2.2: Schematic of the architecture of a simple recurrent neural network. Three layers are shown in the picture: The input layer at the far left (red circles), the middle hidden layer (black circles), and the output layer on the far right (blue circles). There are two types of recurrent connections shown in the diagram. Green arrows self-connections of hidden layer neurons and magenta arrows inter-connections between neurons of the hidden layer.

Beyond this vanilla version of RNNs, many other architectures for recurrent neural networks have been developed and used for different machine learning tasks. The most well-known ones are the Restricted Boltzmann Machine (RBM) [HS06], Hopfield networks [Hop82], Long Short-Term Memory (LSTM) networks [HS97b] and Echo State Networks (ESN) [Jae01]. We will not go into further detail about each type here as not all of them are closely related to this work. Instead, the next section focuses on continuous-time recurrent neural networks (CTRNN), which are the networks of interest here.

2.1.3 Continuous-time recurrent neural networks

The continuous-time recurrent neural networks we will be considering in this work are not organised in layers. Rather, input can be provided to any neuron and every neuron can be

connected to any other as illustrated in Fig. 2.3, which can lead to an arbitrary number of cycles in the network. Furthermore, the output can be produced using any combination of neural activities. In contrast to the type of networks described above, which are constructed in a way that makes them convenient for training on machine learning tasks, these networks are more focused on being biologically relevant. More precisely, their equations are developed to capture the effect that an incoming spike train has on the membrane potential of a neuron. These equations are of the following form:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} r_j + \sum_{k=1}^{N_{In}} W_{ik}^{In} I_k(t) + \sum_{o=1}^{N_{Out}} W_{io}^{Fb} z_o \qquad i = 1, 2, .., N \quad (2.3a)$$

$$r_i = \tanh(x_i) \quad (2.3b)$$

$$z_o = \sum_{j=1}^{N} W_{oj}^{Out} r_j \qquad o = 1, 2, ..., N_{Out}. \quad (2.3c)$$

In these set of equations, N is the number of neurons in the network, N_{Out} is the number of outputs, N_{In} is the number of inputs, x_i is the membrane potential of neuron i and represents the internal state of each neuron in the network. The time constant of the network, τ, determines the timescale of the neuron's activity. A larger value of τ results in a slower evolution of the state of the network. The matrices $\mathbf{W}^{Res}$, $\mathbf{W}^{In}$, $\mathbf{W}^{Out}$ and $\mathbf{W}^{Fb}$ respectively hold the connection weights any neuron in the network to every other, from multiple time-varying inputs, $I(t)$, to neurons in the network, from neurons to multiple outputs, z, and from outputs back into the network neurons. Note that the notation W_{mn} corresponds to connections from n to m. The activation of neuron i, r_i, is calculated by applying the hyperbolic tangent to its membrane potential, as shown in Eq. 2.3b and the output o from the set of outputs, z_o, is calculated according to Eq. 2.3c.

The networks that we will be predominantly focusing on in this study have recurrent connections, stored in the matrix $\mathbf{W}^{Res}$, which are given by a random Erdös-Renyi graph with connection

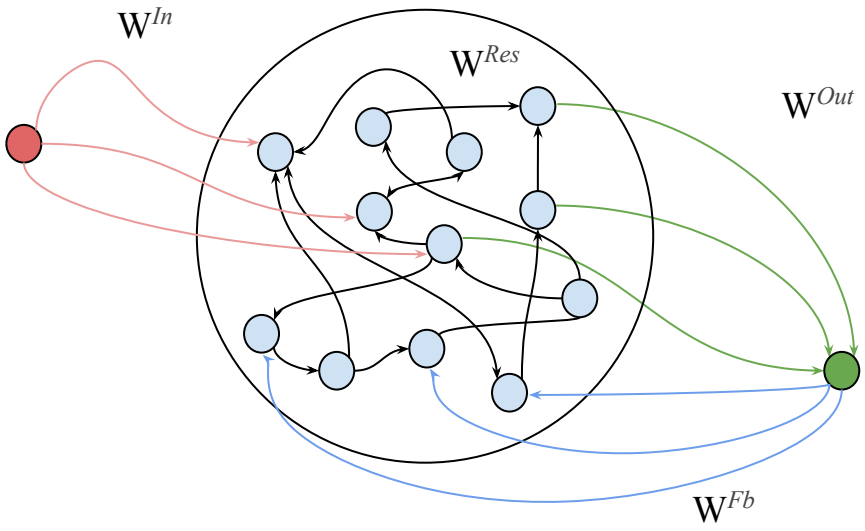


Figure 2.3: Schematic of a randomly connected network, showing all the connectivity matrices used in Eq. 2.3a–2.3c. For simplicity, only one input (red) and one output (green) are shown in the diagram.

probability $p = 0.1$, no self-connections and weights $W_{ij}^{Res} \sim \mathcal{N}(0, g_G^2)$, where $g_G = g/\sqrt{pN}$ acts as a global scaling factor of the network weights. The parameter g is usually referred to as the *gain* of the system, which is an important parameter that strongly affects the dynamic behaviour of the network. Specifically, it has long been established [SCS88] that autonomous networks with $g < 1$ exhibit attracting dynamics towards a globally stable stationary point, whereas networks with $g \geqslant 1$ can exhibit complex periodic or even chaotic behaviour. On the other hand, the activity of input driven networks is also strongly dependent on the input signal. In some cases, the input is also able to suppress chaotic activity of networks with g greater than 1 [RAS10]. For convenience, we will refer to networks with $g < 1$ as stable networks and those with $g \geqslant 1$ as unstable.

The equivalent of the Universal Approximation Theorem [Hor91, Csá01] that holds for feedforward networks, also holds for continuous-time recurrent neural networks of the form shown in Eq. 2.3a–2.3c. In the recurrent case, the networks can approximate any dynamical system to arbitrary precision provided there are enough neurons in the network [FN93, CL00].

In order to simulate these networks, Eq. 2.3a was numerically integrated using the Euler method

with an integration step of $\delta = 0.01$. The corresponding discrete form of Eq. 2.3a is shown in Eq. 2.4 below, with time varying quantities calculated in the indicated time step. Even though other more stable methods for numerical integration of ordinary differential equations exist, for example the Runge-Kutta method, this work uses the Euler integration method, which is the method used in the relevant literature [LB13, MSSN13] and is preferred when implementing these networks in applications. This way, the analysis and results presented here are relevant to the existing body of work. Nonetheless, future work should aim to assess whether the Euler method introduces artefacts in the dynamics, by running a comparative study with other, more robust, numerical integration methods.

$$x_i(t) = x_i(t-1) + \frac{\delta}{\tau}\Big(- x_i(t-1) + \sum_{j=1}^{N} W_{ij}^{Res} r_j(t-1) + \sum_{k=1}^{N_{In}} W_{ik}^{In} I_k(t) + \sum_{o=1}^{N_{Out}} W_{io}^{Fb} z_o(t-1)\Big) \quad (2.4)$$

2.2 Learning

Memory formation and learning in the brain is a very active field of research in neuroscience and has led to the formulation of mathematical models that rely on synaptic plasticity as the mechanism that supports learning in the brain [Sho07, Heb49, Oja82, BCM82, IC92, Izh07b]. These models are biologically plausible, in the sense that the only information required to change the strength of the synapse is local to the synapse, and are usually applied to models of spiking neurons [HH52, Izh03]. This means that only the relative timing of the pre- and post-synaptic neurons' activations are required. In addition, these models are inspired by the biochemistry of synapses and the way physical changes in these relate to changes in the effective synaptic strength.

For example, let us consider spike-time dependent plasticity (STDP), which is a form of Hebbian learning [Heb49] and the most well-known synaptic plasticity rule. According to STDP, the synaptic connection between a pre-synaptic neuron and the post-synaptic neuron is strengthened if the post-synaptic neuron fires shortly after the pre-synaptic neuron, whereas the connection

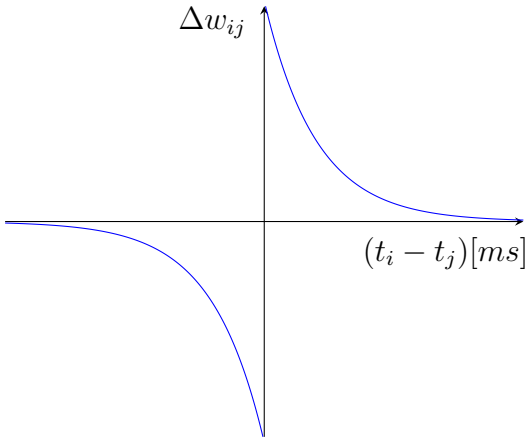


Figure 2.4: Plot showing the change of synaptic strength according to Hebbian-type learning described by Eq. 2.5, as a function of the timing difference between the spikes of the pre-synaptic neuron j and post-synaptic neuron i.

is weakened if it fires shortly before. This is captured by the following simple equation, where the Δw_{ij} is the change in synaptic strength between pre-synaptic neuron j and post-synaptic neuron i, A is a pre- exponential factor, τ is a decay time constant and t_k the firing time of neuron k. The change in synaptic strength is shown in Fig. 2.4.

$$\Delta w_{ij} = \begin{cases} Ae^{-\frac{(t_i - t_j)}{\tau}}, & \text{if } (t_i - t_j) > 0 \\ -Ae^{\frac{(t_i - t_j)}{\tau}}, & \text{if } (t_i - t_j) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

As we have mentioned previously, this is a biologically plausible rule because it only uses local information about the neurons around the synapse. Despite their simplicity, plasticity rules based on Hebbian learning have been successfully used to enable simulated networks of spiking neurons to learn a variety of simple tasks [Izh07b, BD15, Mic17, MS16]. Furthermore, it has been also shown that STDP promotes criticality in networks of spiking neurons [TS14] which is related to promoting global synchronisation between different sub-modules of modular networks of spiking neurons [TS15].

Even though these plasticity models have done much to increase our understanding about learning in the brain, artificial neural networks used in machine learning applications are trained using a different approach to the Hebbian-type learning described in this section. For this

reason, our focus will shift in the following sections towards the approaches used for adjusting the weights of these networks to achieve good performance in their tasks.

2.2.1 Training artificial neural networks

For neural networks to perform a task, for example classifying images or determining the best set of actions in a game, adaptation of the weights between neurons must take place through the process of training. Supervised learning is the most common type of training for neural networks used to approximate some function. During the training phase, the network is presented with an input, which it uses to compute an output. The network's output is compared with the correct output, which is already known, and the network weights are adjusted so that they minimise some cost function that is usually designed to minimise the difference between the network output and the correct output. This training phase is followed by an evaluation phase, where no more adjustment of weights takes place, to assess the network performance on unseen inputs and outputs [FHT01].

At this point we should note the conceptual difference between supervised learning approaches described here and the unsupervised Hebbian-type learning described in the previous subsection. In the supervised approach sets of inputs and correct outputs are used during training, whereas in Hebbian learning, no such information is required. Instead, the spike timing between the two neurons sharing a synapse is the only information taken into account when adjusting the synaptic weight between them.

The backpropagation algorithm

The most widely used algorithm for training perceptron-like feedforward neural networks is backpropagation [RHW86b]. The main motivation behind this algorithm is to answer the following question: given an input to the network, and a cost function that returns an error between the network output and the correct label, how should each weight in the network change such that this error is reduced? The backpropagation algorithm solves this problem by using

the chain rule of differentiation to find the derivatives of this loss function (gradients) with respect to the weights of each of the layers, which are used to compute the required direction of change in each of the network weights. Given these gradients, the backpropagation algorithm with stochastic gradient descent and a batch size of one sample proceeds as follows:

1. Given the current input, compute the activations in each of the network's layers and the output.
2. Compute the error between the network output and the correct label.
3. Compute the gradients, multiply them by an appropriate learning rate and apply corresponding weight updates.

Stochastic gradient descent is an approach to training with backpropagation, where, instead of the true gradient of the loss – that can be obtained if the whole dataset is used for each gradient update – only a small batch of samples is used to calculate an estimate of the gradient. The main benefit of stochastic gradient descent is an enormous convergence speed-up, since in many cases the single or mini-batch sample gradient is a good estimate of the true gradient, and hence only a small part of the dataset is used for every weight update. This of course requires that the samples in a mini-batch are independent and identically distributed, something that can be achieved by shuffling all data points prior to training.

Even though training neural networks with the backpropagation algorithm has proven to be successful in many tasks, it still has inherent limitations. The main limitation is the fact the the loss function that the training procedure tries to minimise is not convex. This means that the local optimisation method employed, namely gradient descent, is heavily dependent on its initialisation and does not guarantee convergence to the global optimum solution, meaning that some inputs will be misclassified. This means that in order to achieve better performance, the training must be repeated for as many times as possible, subject to different initialisations, so that the trained network with the best performance can be chosen. A common set of tricks to overcome this limitation and help the networks converge to better minima with reduced computational cost include adaptive learning rates, conjugate gradient methods, normalisation

and transformation of the input into the network [LBOM12]. All these tricks can help to considerably speed-up learning in feedforward neural neural networks and have hence become common practice. The simplicity and success of the backpropagation method for feedforward neural networks has led to its adaptation even in the case of recurrent networks, as we will demonstrate in the following section.

Backpropagation through time

The simplicity of gradient descent for adjusting the weights of feedforward networks has inspired researchers to adapt it for the case of recurrent networks. This has led to the development of the backpropagation through time (BPTT) method [Moz89, Wer88], where a recurrent neural network similar to the one depicted in Fig. 2.2 is unfolded over time. The unfolding consists of recasting the network in a feedforward form, where each layer is a repetition of a specific building block – constructed to capture the recurrent connectivity – and the number of repetitions of this block represents the number of time steps the network is unfolded over. Further, since layers of repeating blocks represent different processing time steps, the weights of all layers in the unfolded network are shared.

Unfortunately, these constraints make training recurrent neural networks with backpropagation through time less stable, with the most common problem observed being that gradients flowing backwards in the network become either very small or unstable [BSF94, HBF+01]. This problem has led to the development of Long-Short Term Memory (LSTM) networks by the machine learning community [HS97a, GSC99], which can overcome this problem by using a "forget gate" in each LSTM unit and by clipping the gradients, thus preventing them to vanish or explode.

Another way to overcome these difficulties is by replacing normal gradient descent with a Hessian-free optimisation technique [Mar10, MS11] based on the Newton method. This method takes into account the curvature of the parameter space and is able to find better solutions compared to normal gradient descent for cases with "pathological" curvature, such as the weight optimisation problem of unfolded RNNs. The only limitation of the Newton method is that the Hessian of the network is usually very expensive to evaluate, but in the Hessian-free optimisation

method, the product of the Gauss-Newton approximation to the Hessian and the vector of gradients can be evaluated using finite differences. This allows the optimisation to be carried out with reasonable computational and memory costs, which has lead to its use in training RNNs to discriminate between periodic inputs with different frequencies [SB13].

Biological plausibility of backpropagation

Although the development of these algorithms has offered new and efficient ways of training artificial neural networks for machine learning tasks, they have long been criticised by the neuroscientific community as incompatible with neurobiology [Cri89]. One of the most important long-standing arguments against the biological infeasibility of these methods is that synaptic weight updates should only use local information. This is of course in contrast to the backpropagation algorithm and its variants, which require that information about downstream error is propagated backwards in the network in order to change the weights in earlier layers.

In recent attempts to answer to this criticism, researchers have argued that backpropagation could be implemented in biologically plausible ways in the brain. For example, Bengio *et al.* claim that STDP can be interpreted as "[...] gradient descent on some objective function, provided that the neuronal dynamics push firing rates towards better values of the objective function [...]" [BLB+15, BMF+15]. Another recent work by Guergiuev *et al.* has shown that cortical pyramidal neurons, due to their multi-compartmental physiology, can provide higher-level feedback to lower-level neurons that allows them to perform a synaptic update based on STDP, that is approximately equivalent to the update as computed by normal backpropagation [GLR16]. Furthermore, Scellier and Bengio developed an energy-based model, named Equilibrium Propagation, where leaky integrate-and-fire neurons perform computations that are equivalent to both inference and error backpropagation [SB17]. Their work provides computational evidence for the idea that there are biologically plausible mechanisms that only make use of realistic neurons and STDP, and can be used as the means for realising an equivalent form of backpropagation in the brain.

2.2.2 Reservoir computing

Even though the backpropagation algorithm and its variants have been very successful for many tasks, and there is recent work on how they might be realised in the real brain, they still have not been succesfully implemented for networks with arbitrary cycles in their recurrent connectivity, such as for networks of the type shown in Fig. 2.3. A promising new paradigm for training such recurrent neural networks, that is also very fast and biologically plausible, is reservoir computing [Jae01, MNM02, SVVC07, LJ09, HS12]. Historically, the core concept defining reservoir computing was first introduced by Kirby [Kir91] but received little attention until a more recent revival that was supported by related but independent efforts from different people [LJ09]. This lead to a variety of reservoir computing *flavours*, namely Temporal Recurrent Networks [Dom98], Echo State Networks (ESN) [Jae01], Liquid State Machines (LSM) [MNM02] and Backpropagation-Decorrelation (BPDC) [Ste04].

In this paradigm, not all connections in the network need to be plastic. Rather, only the read-out connections, $\mathbf{W}^{Out}$ in Fig. 2.3, are plastic. The goal of the learning procedure is to use the rich and varied dynamic activity of the network to select an appropriate set of readout weights, so that a desirable output is produced. In this sense, the activity of the network already contains the necessary information needed to be recalled and it is through the learning of a linear decoder by the read-out neurons that it is accessed. Furthermore, reservoir computing is considered to be a biologically plausible paradigm for learning, because it solves the problem of learning in networks with arbitrary cycles and rich complex dynamics, similar to the connectivity and activity of the cortex [EPQD16].

It is instructive to contrast the motivation and approach of reservoir computing with that of generative models, a class of models developed in the field of machine learning. When training generative models, the goal is to learn a mapping from a low dimensional "embedding" space, to a higher dimensional space, often images. In this way, the embedding space represents a generalised encoding for the input data, which can be used to either classify unseen data using the encoder, or generate new data using the decoder. This can be done using different models, the most common ones being Generative Adversarial Networks (GANs) [GPAM$^+$14], Variational

Auto Encoders (VAEs) [KW13] and Pixel Recurrent Neural Networks (Pixel RNNs) [OKK16], which make use of LSTM units. All these techniques are computationally much more expensive to train compared to the reservoir computing method, as they need to be provided with a very large number of examples before they can generalise over the data. For the correct generalisation to be implemented, the numerous parameters in the encoding and decoding layers need to be slowly adjusted to correctly encode and decode the dataset.

On the contrary, in reservoir computing, only the cheaper linear readout vector needs to be adjusted to produce the desired output, as the necessary variation is already present in the "rich" dynamic activity of the network. This provides the possibility for memorising temporally extended data points, for example a whole period of a sinusoidal wave, upon a single presentation, and later reproducing it. This property is related to short term and episodic memory in humans, which is the ability to memorise and later recall specific, temporally extended experiences that occur only once. It is interesting to note that even though episodic memory is thought to be realised in a much more complex way in the brain, compared to how simple learning is in reservoir computing, and primarily incorporating the cortex and the hippocampus [Nor10, KHM16], they both share two characteristics. The first is that there is only one instance of the experience available, and the second is that the recalled experience appears to be very rich in terms of sensory information. In fact, this property is very relevant for creating artificial agents [HKSB17], as it has been shown that reinforcement learning agents perform much better if they replay specific instances of their experience [BUP$^+$16, PUS$^+$17].

In the paradigm of reservoir computing, the neural network is differentiated into two parts, the reservoir and the readout, similar to Fig. 2.3. In general, the function of the reservoir is to perform the non-linear mapping of the input signal onto a high dimensional space, i.e. the state of the network, while the readouts are trained to map the high dimensional state to the low dimensional desired output, using a linear projection of the state, as shown in Eq. 2.3c [BM09, Jae07]. After training, the readout vector $\mathbf{w}^{Out}$ remains fixed and maps the activation state of the network at time step t, $\mathbf{r}(t)$, to a scalar output $z(t)$, which we want to be close to the target output $y(t)$. We can then represent the time series of the network's activity up to time step t as the matrix $\mathbf{R}_t$ and the network's output and target time series as the vectors $\mathbf{z}_t$

and $\mathbf{y}_t$ respectively. These will have the following form:

$$\mathbf{R}_t = \begin{bmatrix} \mathbf{r}^T(0) \\ \mathbf{r}^T(1) \\ \cdots \\ \cdots \\ \mathbf{r}^T(t-1) \\ \mathbf{r}^T(t) \end{bmatrix}, \qquad \mathbf{z}_t = \begin{bmatrix} z(0) \\ z(1) \\ \cdots \\ \cdots \\ z(t-1) \\ z(t) \end{bmatrix}, \qquad \mathbf{y}_t = \begin{bmatrix} y(0) \\ y(1) \\ \cdots \\ \cdots \\ y(t-1) \\ y(t) \end{bmatrix}.$$

Following directly from Eq. 2.3c, training the network's readouts to reproduce the target is equivalent to solving the linear system

$$\mathbf{R}_t \mathbf{w}^{Out} = \mathbf{z}_t = \mathbf{y}_t \tag{2.6}$$

by inverting $\mathbf{R}$, such that

$$\mathbf{w}^{Out} = \mathbf{R}_t^{-1} \mathbf{y}_t. \tag{2.7}$$

It is often the case that the time series of the network state results in a the matrix $\mathbf{R}_t$ that is not of full rank and hence has no proper inverse, meaning that there is no unique vector $\mathbf{w}^{Out}$ that exactly satisfies the linear equation system. One way to overcome this is to use the 'least-squares' method to find $\mathbf{w}^{Out}$ that results in a $\mathbf{z}_t$ that is as close to $\mathbf{y}_t$ as possible. In fact, the learning algorithm that was used in this study, namely FORCE learning [SA09], is based on the recursive least-squares approach [Hay02], but before we look at it more closely in the following section, let's first introduce some important aspects of the network's activity that are related to the performance during training.

In order to successfully train the network's readouts to reproduce the desired output, the dynamic activity of the network needs to be at least as "rich" as the signal it is trying to reproduce. By "rich" here, we mean that the network must generate activity with a large enough

variation, otherwise the linear system of equation described in Eq. 2.6 will have no solutions. Let's consider, for example, the case where a network is at steady-state with the activations of all the neurons being equal to 1 at all time steps, resulting in $\mathbf{R}_t$ being a $T \times N$ matrix of 1s, where T is the number of time steps and N the number of neurons. If $\mathbf{y}_t$ is a $T \times 1$ vector of the time-varying signal that the network is trying to learn, for example a sine wave, then there exists no linear readout $\mathbf{w}^{Out}$ that can produce a time-varying output $\mathbf{z}_t = \mathbf{y}_t$ because Eq. 2.6 has no solutions. Furthermore, and using our intuition, we can easily see that the same network state cannot be mapped to different output values using the same linear readout and hence, in order to be able to reproduce a time-varying output signal, the combined activity of the network needs to, in some sense, be at least as "complex" as the output signal that needs to be learned.

As we have just seen, the dynamic activity of the network is a very important factor for successful training CTRNNs in the paradigm of reservoir computing and a large part of the work presented later on in this thesis studies the dynamics of CTRNNs. Before we begin our exploration of the world of dynamical systems though, we will first have a look at the learning algorithm used to train CTRNNs in this work.

FORCE Learning

Throughout this work, the CTRNNs were trained using FORCE learning [SA09], a variant of the recursive least square algorithm (RLS) [Hay02] that adjusts the readout weights of the network to generate a target output time series. There are a number of reason to prefer FORCE learning over other methods to train CTRNN networks with complex, or even chaotic activity.

The first reason is that it can be used to adjust a linear readout $\mathbf{w}^{Out}$ in an iterative, on-line manner, meaning that there is no need to first experience the whole time series $\mathbf{y}_t$ before learning can begin. On-line, sequential learning is a desirable property of any biologically relevant learning algorithm, as natural temporal patterns are extended in time and are hence accessible sequentially to the brain.

The second desirable feature of FORCE learning is that the error $e(t)$ between the network

output $z(t)$ and the target $y(t)$ is kept small right from the start of training. Beyond being a generally desirable feature of every learning algorithm, it is of particular importance when dealing with chaotic networks that have feedback connections $\mathbf{W}^{Fb}$ from the network output back into the network, as shown in Fig. 2.3. The main reasons to include feedback connections in the networks are to make them 1) more stable – and hence reproducible – and 2) to *inject* a signal containing target- or task-specific information back into the network.

The main problem arising when training networks with feedback connections is that if the output $z(t)$ deviates significantly from $y(t)$ during the early stages of training, the feedback to the network is different from what it would have been if it was well-trained and close to $y(t)$. This difference in the feedback, combined with the chaotic dynamics of the network, cause the activity of the network during subsequent time steps to be very different from what it would have been if the correct output $z(t)$ was fed to the network, hence resulting in learning the wrong readout weights.

One way to alleviate this problem, often used when training Echo State Networks [Jae01], is to feed the target signal $y(t)$ back to the network, rather than the output of the network $z(t)$, to ensure that the desired dynamics are realised in the network during the training phase. Once training has been completed, and $z(t)$ is very close to $y(t)$, then the output can be fed back to the network to test the performance. This approach can be seen in Fig. 2.5.

In FORCE learning on the other hand, the error is small throughout training, which enables the network output to be fed back into the network during the training phase without destabilising the network dynamics. It is in fact because of this property that the algorithm was named FORCE, which stands for First Order Reduced and Controlled Error. Moreover, unlike traditional training methods, the aim of FORCE learning is not to iteratively decrease an initially large error, but to iteratively reduce the magnitude of modifications needed to maintain a small error. This is achieved by assigning a time-varying individual learning rate to each of the weights in the readout matrix. This learning rate is updated at every time step of training, based solely on the activity of the network and receives no information about the error or the target time series. To better understand how the algorithm works, let us first define the necessary quantities we

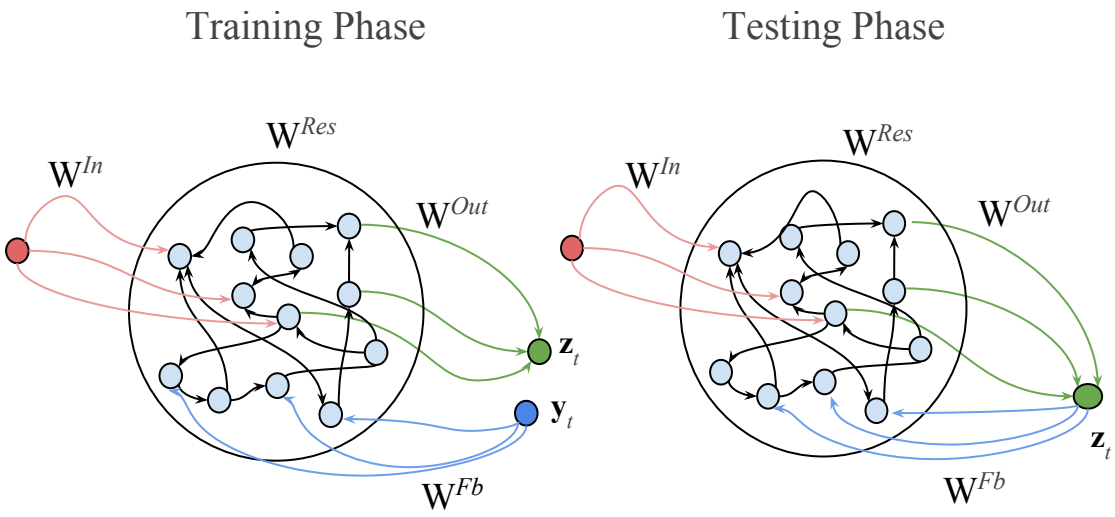


Figure 2.5: Differentiation of the training and testing phase in Echo State Network training [Jae01]. During the training phase the output $\mathbf{z}_t$ is disconnected from the feedback and the target $\mathbf{y}_t$ is injected to the network through the feedback connections $\mathbf{W}^{Fb}$ instead. During the testing phase, the network output $\mathbf{z}_t$ is fed back to the network.

will be considering, adopted from the original work on FORCE learning [SA09].

Let's first rewrite Eq. 2.3c in vector notation, for the case of a single network output and target:

$$z(t) = \left(\mathbf{w}^{Out}\right)^{\mathrm{T}} \mathbf{r}(t). \tag{2.8}$$

The error at time step t is then defined as

$$e(t) = z(t) - y(t). \tag{2.9}$$

$$e(t) = \left(\mathbf{w}^{Out}\right)^{\mathrm{T}} \mathbf{r}(t) - y(t). \tag{2.10}$$

The notation used for $\mathbf{w}^{Out}$ assumes that the readout is time-independent, but since the learning rule will be changing this readout, we can rewrite it as a time-varying quantity $\mathbf{w}^{Out}(t)$. This allows us to define the error before the weight update at time step t as

$$e_{-}(t) = \left(\mathbf{w}^{Out}(t - \Delta t)\right)^{\mathrm{T}} \mathbf{r}(t) - y(t), \tag{2.11}$$

where Δt denotes the previous weights update time step and is usually 1, and the error after the update as

$$e_{+}(t) = \left(\mathbf{w}^{Out}(t)\right)^{\mathrm{T}} \mathbf{r}(t) - y(t). \tag{2.12}$$

Furthermore, the weight update in FORCE learning follows a delta-type rule, involving the product of the error and the firing rate, multiplied by a learning rate $\eta(t)$, which can be expressed as

$$\mathbf{w}^{Out}(t) = \mathbf{w}^{Out}(t - \Delta t) - \eta(t) e_{-}(t) \mathbf{r}(t). \tag{2.13}$$

In the ideal case, we would like the network output to equal the target at every time step, i.e. $z(t) = y(t)$, which can be guaranteed if the readout is a time-varying quantity of the form of Eq. 2.14. It is straightforward to see this by substituting Eq. 2.14 in Eq. 2.8, assuming the readout is time-varying.

$$\mathbf{w}^{Out}(t) = \frac{y(t)\mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)}. \tag{2.14}$$

Of course, Eq. 2.14 is not a unique solution to the time varying readout, which can be expressed in a more general form as

$$\mathbf{w}^{Out}(t) = \mathbf{q}(t) + \frac{\left(y(t) - \left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t)\right) \mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \tag{2.15}$$

for any vector $\mathbf{q}(t)$ with the same size as $\mathbf{w}^{Out}(t)$. This can be shown by substituting Eq. 2.15

into Eq. 2.8 and simplifying as follows:

$$z(t) = \left(\mathbf{q}(t) + \frac{\left(y(t) - \left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t) \right) \mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \right)^{\mathrm{T}} \mathbf{r}(t) \tag{2.16a}$$

$$z(t) = \left(\mathbf{r}(t)\right)^{\mathrm{T}} \left(\frac{\mathbf{q}(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) + y(t)\mathbf{r}(t) - \left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t)\mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \right) \tag{2.16b}$$

$$z(t) = \frac{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{q}(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) + \left(\mathbf{r}(t)\right)^{\mathrm{T}} y(t)\mathbf{r}(t) - \left(\mathbf{r}(t)\right)^{\mathrm{T}} \left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t)\mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \tag{2.16c}$$

$$z(t) = \frac{\left(\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{q}(t) \right) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) + \left(\mathbf{r}(t)\right)^{\mathrm{T}} y(t)\mathbf{r}(t) - \left(\mathbf{r}(t)\right)^{\mathrm{T}} \left(\left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t) \right) \mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \tag{2.16d}$$

$$z(t) = \frac{\left(\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{q}(t) \right) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) + y(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) - \left(\left(\mathbf{q}(t)\right)^{\mathrm{T}} \mathbf{r}(t) \right) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \tag{2.16e}$$

$$z(t) = \frac{\left(\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{q}(t) \right) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) + y(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) - \left(\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{q}(t) \right) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} \tag{2.16f}$$

$$z(t) = \frac{y(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}}{\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t)} = y(t). \tag{2.16g}$$

Hence, Eq. 2.15 is equivalent to Eq. 2.13 if we set $\mathbf{q}(t) = \mathbf{w}^{Out}(t - \Delta t)$ with $\eta(t) = \left(\left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{r}(t) \right)^{-1}$.

The only problem with this learning rate is that it does not lead to a time-independent set of readout weights. In order to overcome this, FORCE learning takes a slightly different approach and instead of a global learning rate $\eta(t)$, it uses a running estimate of the inverse of the regularised correlation matrix of the network

$$\mathbf{P}(t) = \left(\mathbf{R}_t^{\mathrm{T}}\mathbf{R}_t + \alpha\mathbf{I}\right)^{-1}, \tag{2.17}$$

to assign individual learning rates to each pre-synaptic neuron. α here is the regularising constant and $\mathbf{I}$ the identity matrix. Using this learning rate, the weight update Eq. 2.13 then becomes

$$\mathbf{w}^{Out}(t) = \mathbf{w}^{Out}(t - \Delta t) - e_{-}(t)\mathbf{P}(t)\mathbf{r}(t). \tag{2.18}$$

Hence, $\mathbf{P}(t)$ can act as the time varying learning rate $\eta(t)$. Furthermore, rather than computing Eq. 2.17 at every time step, which is computationally expensive as it involves inverting an $N \times N$ matrix, a running estimate of $\mathbf{P}(t)$ is updated before the weight update instead, according to the RLS algorithm mentioned previously [Hay02]

$$\mathbf{P}(t) = \mathbf{P}(t - \Delta t) - \frac{\mathbf{P}(t - \Delta t)\mathbf{r}(t)\left(\mathbf{r}(t)\right)^{\mathrm{T}}\mathbf{P}(t - \Delta t)}{1 + \left(\mathbf{r}(t)\right)^{\mathrm{T}}\mathbf{P}(t - \Delta t)\mathbf{r}(t)}, \tag{2.19}$$

which is cheaper, as it only involves matrix-vector multiplications. Finally, an initial value of $\mathbf{P}(0) = \alpha^{-1}\mathbf{I}$ is required for the algorithm to work, with the recommendation that $\alpha << N$.

Even though the update in Eq. 2.18 produces an output signal $z(t)$ with non-zero error, Sussillo and Abbott [SA09] claim that learning can successfully take place because the error is so small that it acts as noise to the system, which, the authors further claim, helps by making the outcome of the learning process more robust. Furthermore, the ratio between the error before and after the updates approaches 1 as training progresses ($e_{+}(t)/e_{-}(t) \to 1$), meaning that the magnitude of the weight updates becomes smaller, resulting in a time-independent readout, that is able to regenerate the target without further modifications. It is easy to show that this follows from the above equations, if we use the fact that

$$\left(\mathbf{r}(t)\right)^{\mathrm{T}}\mathbf{w}^{Out}(t) = \left(\mathbf{w}^{Out}(t)\right)^{\mathrm{T}}\mathbf{r}(t). \tag{2.20}$$

Using Eq. 2.20, we can rewrite Eq. 2.12 as

$$e_+(t) = (\mathbf{r}(t))^{\mathrm{T}} \mathbf{w}^{Out}(t) - y(t) \tag{2.21}$$

and Eq. 2.11 as

$$(\mathbf{r}(t))^{\mathrm{T}} \mathbf{w}^{Out}(t - \Delta t) = e_-(t) + y(t) \tag{2.22}$$

We can then left-multiply Eq. 2.18 by $(\mathbf{r}(t))^{\mathrm{T}}$ to obtain

$$(\mathbf{r}(t))^{\mathrm{T}} \mathbf{w}^{Out}(t) = (\mathbf{r}(t))^{\mathrm{T}} \mathbf{w}^{Out}(t - \Delta t) - e_-(t) (\mathbf{r}(t))^{\mathrm{T}} \mathbf{P}(t)\mathbf{r}(t). \tag{2.23}$$

By then substituting Eq. 2.22 into Eq. 2.23 and then this into Eq. 2.21 we arrive at

$$e_+(t) = e_-(t) + y(t) - e_-(t) (\mathbf{r}(t))^{\mathrm{T}} \mathbf{P}(t)\mathbf{r}(t) - y(t) \tag{2.24a}$$

$$e_+(t) = e_-(t) \left(1 - (\mathbf{r}(t))^{\mathrm{T}} \mathbf{P}(t)\mathbf{r}(t)\right). \tag{2.24b}$$

What is important, the authors further claim, is that the only condition that needs to hold for $e_+(t)/e_-(t) \to 1$ is that $(\mathbf{r}(t))^{\mathrm{T}} \mathbf{P}(t)\mathbf{r}(t) \to 0$ as $\mathbf{P}(t)$ goes from $\alpha^{-1}\mathbf{I}$ to $\left(\mathbf{R}_t^{\mathrm{T}}\mathbf{R}_t + \alpha\mathbf{I}\right)^{-1}$, which is indeed the case [SA09]. This ensures that the modifications in the linear readout become smaller as training progresses, resulting in a time-independent set of readout weights.

Another benefit of FORCE learning is that the computation of $\mathbf{P}(t)$ is independent of the number of outputs, meaning that for multiple outputs, the weight update Eq. 2.18 simply becomes

$$\mathbf{W}^{Out}(t) = \mathbf{W}^{Out}(t - \Delta t) - (\mathbf{P}(t)\mathbf{r}(t)) \left(\mathbf{e}_{-}(t)\right)^{\mathrm{T}}, \tag{2.25}$$

and hence the computational cost scales linearly with the number of readouts. This makes FORCE learning suitable for using the same network to learn multiple outputs without the need of another reservoir network.

FORCE learning has recently been extended to full-FORCE learning [DCR^{+}18], where the authors use a secondary target network to generate an activity that is relevant to the task at hand and that is guaranteed to solve the task. The activity of the secondary network is then used as the target of the task-performing network that tries to autonomously generate the activity of the secondary network by adjust its internal connectivity. The RLS approached described above can be used to adjust the internal connectivity of the secondary network, by replacing $\mathbf{W}^{Out}$ in Eq. 2.25 with $\mathbf{W}^{Res}$ of the task performing network and $\mathbf{y}_t$ – in the same equation – with $\mathbf{R}_t$ of the secondary network. This approach allows the network to adjust its internal connectivity in such a way as to ensure that its dynamics are relevant for solving the task and can hence achieve good performance with a smaller number of neurons in the reservoir. Finally, we should note that FORCE learning is not biologically plausible learning rule, since it explicitly uses global information about the rates of all neurons in the network to determine the weight changes, unlike STDP for example, which uses only local information to a synapse to change its weight.

In this work, randomly connected networks have been used and hence the focus will be to understand the effect of the scaling of the network connectivity and the input on the dynamics of randomly connected networks, for networks without feedback connections. Furthermore, these are equivalent to networks with feedback connections, provided that the network output can reproduce the input. This means that by studying the case of networks with no feedback, we can extract useful conclusions about the effect of the target on the dynamic behaviour of the network trying to learn them. In the following section, the necessary tools from dynamical systems theory are introduced. These will be used to analyse the response of the networks to various inputs, as well as study the role of the different parameters of the network.

2.3 Dynamical Systems Theory

Continuous-time recurrent neural networks (CTRNN) are described by Eq. 2.3a–2.3c, which is a set of first order differential equations. Systems described by sets of differential equations are called *flows* and can be studied using tools from the theory of dynamical systems, to which I will provide a brief introduction in this section, following Strogatz [Str14].

2.3.1 States, state space and trajectories

Each equation of the system describes how the state x_i of neuron i in the network changes for the current network configuration $\mathbf{x}$. In general, a particular configuration of a system is called the *state* and the set of all possible configurations the *state space*. As the system, in our case the network, evolves in time from an initial state $\mathbf{x}_0$, it follows a set of states $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_t\}$ that we will be referring to as the *trajectory* of the system.

2.3.2 Vector fields of dynamical systems

To more intuitively understand why a deterministic system follows a particular trajectory, we will make use of the concept of the *vector field* over the state space of the system. This means that for every possible configuration of the system, we can assign a vector of the same dimensionality as our system, and an infinitesimal magnitude, which gives the direction of change for the state of the system. Generating a trajectory for the system is equivalent to starting from an arbitrary point in the state space and following the direction of change corresponding to each of the states visited.

The vector field of a system is a way to graphically represent and understand the long term behaviour of a system without having to simulate it. By constructing the vector field of low dimensional systems, for example for a 2-dimensional oscillator, we can fully determine the long dynamic behaviour of the system, but since the method relies on our ability to visualise it, it

becomes cumbersome for systems with more than 3 variables. We will return to the vector fields of dynamical system later, but let us first introduce the concept of an *attractor*.

2.3.3 Attractors

As a system evolves in time, it may be the case that at some time t it reaches a particular state and remains in this state for all subsequent time steps $t+1, t+2, t+3, ..., \infty$, even under small perturbations. Such a state is a *stable* attracting state because once the system enters the vicinity, or the *basin of attraction*, of this stable attracting state, it will be attracted to and thereafter remain in this state.

In the more general case, a system can enter attracting sets of states, meaning that the system no longer settles on a single state, but rather evolves over a fixed set of states. The evolution of the system over this set may be repeating and periodic, in which case this set of states is called a periodic orbit, or limit cycle for the case that the periodic orbit lies on a 2-dimensional plane, and is stable if it will return to them under small perturbations. In a periodic orbit, the same sequence of states will be revisited in every cycle of the system's "oscillation". An example of a system which exhibit a periodic orbit on a plane, therefore a limit cycle, is the Van der Pol oscillator [Kan07].

Finally, beyond the simple and well-understood types of attractors described above, there is another type of attractor that is usually referred to as a *strange* attractor. Attractors of this type have non-periodic orbits and appear to have a fractal structure. Fractals represent a whole field of study on their own, but the simply property that usually characterises them is that they are similar to themselves at all scales. Even though not always [GOPY84], strange attractors are usually chaotic, meaning that any two trajectories that are initially close together, will be arbitrarily far after the system evolves for some time, within the confinement of the attracting set, and will again be close to each other after an arbitrary amount of time. Strange chaotic attractors are considered to be locally unstable, because trajectories that are at some point close together will move away from each other, but globally stable, as no trajectory already in

the attractor will ever leave the attractor. The most widely studied 3-dimensional systems with strange attractors are the Lorenz [Lor63] and the Rössler systems [LR06].

2.3.4 Stationary points of two dimensional systems

We will now pin down some of the ideas we have just introduced, using techniques from dynamical systems theory. The focus here will be on the two simplest types of attractors, namely stable/unstable point attractors and periodic orbits, which can be easily studied by finding the *stationary points* of the system. Stationary points are points in the state space of the system, where the gradient in all directions is zero and the system will therefore remain there, if initialised at these points, in the absence of noise or external perturbations. The dynamics around the stationary points are of special interest, as they determine their stability, and the distribution and type of stationary points determine the *dynamical backbone* of the system. For simplicity, we will here consider the case of 2-dimensional systems and determine the possible types of stationary points that can arise. The general 2-dimensional system $F(\mathbf{x})$ can be expressed as

$$F(\mathbf{x}) \triangleq \dot{\mathbf{x}} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} \tag{2.26}$$

Therefore, stationary points correspond to the solutions $\mathbf{x}^* = [x_1^*, x_2^*]^{\mathrm{T}}$of

$$\begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.27}$$

To reveal the type of stationary point, the Jacobian matrix $\mathbf{J}(\mathbf{x})$ is evaluated at the stationary points according to

$$\mathbf{J}(\mathbf{x}^*) = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{(x_1^*, x_2^*)} & \left.\frac{\partial f_1}{\partial x_2}\right|_{(x_1^*, x_2^*)} \\ \left.\frac{\partial f_2}{\partial x_1}\right|_{(x_1^*, x_2^*)} & \left.\frac{\partial f_2}{\partial x_2}\right|_{(x_1^*, x_2^*)} \end{bmatrix} \tag{2.28}$$

The eigenvalues λ_1 and λ_2 of $\mathbf{J}(\mathbf{x}^*)$ can be obtained by

$$\det(\mathbf{J}(\mathbf{x}^*) - \lambda \mathbf{I}) = 0 \tag{2.29}$$

and the eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ by solving

$$(\mathbf{J}(\mathbf{x}^*) - \lambda_i \mathbf{I})\mathbf{e}_i = \mathbf{0}. \tag{2.30}$$

for each of the two eigenvalues. Each eigenvector corresponds to the direction around the stationary point, along which the system will be attracted or repelled from that point. The sign of each eigenvalue determines the direction that the system will move along the corresponding eigenvector. If the eigenvalue is positive, the system will move away from the stationary point along the corresponding eigenvector, and if it is negative it will be attracted to it. The combination of the two eigenvalues determine the type of stationary point. It can also be the case that the eigenvalues are either complex or purely imaginary and if this is the case they always come in conjugate pairs $\lambda_{1,2} = Re(\lambda) \pm Im(\lambda)i$ or $\lambda_{1,2} = \pm Im(\lambda)i$. If the eigenvalue has a non-zero imaginary part, then the dynamics the system around the stationary point are rotational. To illustrate each of these cases, let us consider the following examples.

Stable and unstable nodes – Let's consider the following example of a 2 dimensional system:

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} (1-2x_1^2)e^{-(x_1^2+x_2^2)} \\ (-2x_1x_2)e^{-(x_1^2+x_2^2)} \end{bmatrix} \tag{2.31}$$

Setting the derivatives to 0 and solving the simultaneous equations for x_1 and x_2, we obtain the 2 stationary points $\mathbf{x^*}_1 = (\frac{1}{\sqrt{2}}, 0)$ and $\mathbf{x^*}_2 = (-\frac{1}{\sqrt{2}}, 0)$. The Jacobian of the system is then

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} 2x_1(2x_1^2-3)e^{-(x_1^2+x_2^2)} & 2(2x_1^2-1)x_2e^{-(x_1^2+x_2^2)} \\ 2(2x_1^2-1)x_2e^{-(x_1^2+x_2^2)} & 2x_1(2x_2^2-1)e^{-(x_1^2+x_2^2)} \end{bmatrix} \tag{2.32}$$

and evaluating this at the two stationary points we obtain

$$\mathbf{J}(\mathbf{x^*}_1) = \begin{bmatrix} -\frac{4}{\sqrt{2e}} & 0 \\ 0 & -\frac{2}{\sqrt{2e}} \end{bmatrix}, \quad \mathbf{J}(\mathbf{x^*}_2) = \begin{bmatrix} \frac{4}{\sqrt{2e}} & 0 \\ 0 & \frac{2}{\sqrt{2e}} \end{bmatrix}. \tag{2.33}$$

Then, the eigenvalues of the linearised system around $\mathbf{x^*}_1 = (\frac{1}{\sqrt{2}}, 0)$ are $\lambda_1 = -2\sqrt{\frac{2}{e}}$ and $\lambda_2 = -\sqrt{\frac{2}{e}}$, with corresponding eigenvectors $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$. Since both eigenvalues of the Jacobian are negative, any small perturbation near the stationary point will end up being attracted back to the stationary point. Hence, stationary points corresponding to linearised systems with only negative real eigenvalues are called *stable nodes*. This is the simplest type of attractor, which is also known as a point attractor, because the system will always return to this single point under small perturbations.

On the other hand, stationary points corresponding to linearised systems with only positive real eigenvalues are called *unstable nodes* and, even though a system will remain at the point if initialised there, any small perturbation away from the unstable node will cause the state of the system to deviate even further away. The second stationary point of Eq. 2.31, $\mathbf{x^*}_2 = (-\frac{1}{\sqrt{2}}, 0)$,

is one such unstable node since both its eigenvalues are positive, i.e. $\lambda_1 = 2\sqrt{\frac{2}{e}}$ and $\lambda_2 = \sqrt{\frac{2}{e}}$, with corresponding eigenvectors $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$.

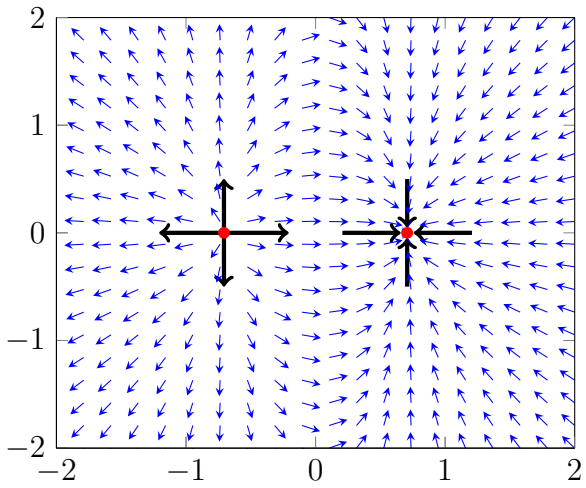


Figure 2.6: The vector field of the system described by Eq. 2.31. The two stationary points, $\mathbf{x^*}_1 = (\frac{1}{\sqrt{2}}, 0)$ and $\mathbf{x^*}_2 = (-\frac{1}{\sqrt{2}}, 0)$, are shown with red marks and correspond to an unstable and a stable node respectively. The black lines correspond to the directions along the eigenvectors of the linearised system around these stationary points.

The vector field of the system plotted in Fig. 2.6 illustrates the directions that trajectories of the system will follow in a small portion of the state space around the two stationary points. By constructing the vector field, we can see that the system will deviate away from $\mathbf{x^*}_2$ and be attracted to $\mathbf{x^*}_1$ if initialised at any point $\mathbf{x} = (\beta - \frac{1}{\sqrt{2}}, \gamma)$, such that $\beta, \gamma \in \mathbb{R}$ and $\beta > 0$.

Saddle nodes – Stationary points that have real eigenvalues with a different sign are called *saddle nodes* and exhibit the interesting property of attracting dynamics along one of the eigenvectors and repelling dynamics along the other. Even though saddle nodes are not attractors in the strict sense that the system will be attracted to them if in their near vicinity, they have an important effect in the overall structure of the vector field, as they can separate the state space of the system into separate basins that are associated with qualitative different types of attractors.

The system shown in Fig. 2.7 exhibits a combination of stable and saddle nodes and is defined as follows:

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} x_1^2 - 1 \\ -x_2 \end{bmatrix}. \tag{2.34}$$

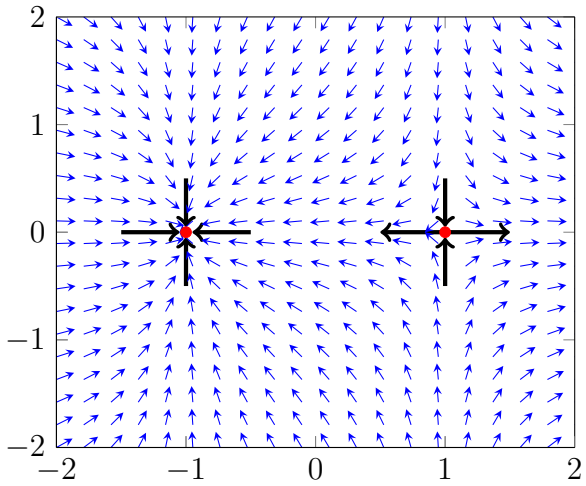


Figure 2.7: The vector field of the system described by Eq. 2.34. The two stationary points, $\mathbf{x^*}_1 = (-1, 0)$ and $\mathbf{x^*}_2 = (1, 0)$, are shown with red marks and correspond to a stable and a saddle node respectively. The black lines correspond to the directions along the eigenvectors of the linearised system around these stationary points.

Following a similar procedure as before for calculating the two stationary points and evaluating the Jacobean at each of them, we obtain the following:

- $\mathbf{x^*}_1 = (-1, 0)$ with $\lambda_1 = -2$, $\lambda_2 = -1$ and corresponding eigenvectors $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$
- $\mathbf{x^*}_2 = (1, 0)$ with $\lambda_1 = 2$, $\lambda_2 = -1$ and corresponding eigenvectors $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$.

Hence, we see that $\mathbf{x^*}_2$ is a saddle node, around which, the local dynamics are attracted along $\mathbf{e}_2$ and repelled along $\mathbf{e}_1$.

Stable and unstable spirals – A different dynamic behaviour can arise in two dimensional systems when the eigenvalues of the Jacobian matrix corresponding to the linearised system

around a stationary point are complex. As briefly mentioned earlier, the local dynamics around such stationary points are rotational, and the sign of the real component of the eigenvalue indicates whether the rotation is attracting or repelling.

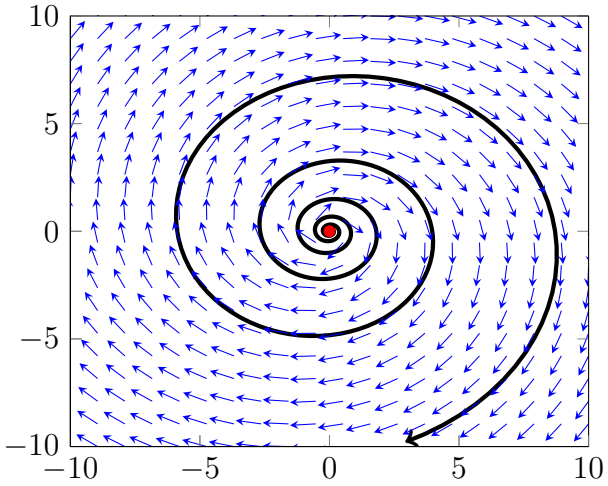


Figure 2.8: The vector field of the system described by Eq. 2.35. The single stationary point, $\mathbf{x}^* = (0, 0)$ shown in red, has complex eigenvalues with positive real part and corresponds to an unstable spiral. The black line corresponds to a trajectory of the system with initial conditions $\mathbf{x}_0 = (0.05, 0.05)$.

The system shown in Fig. 2.8 is described by

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} x_1 + 8x_2 \\ -8x_1 + x_2 \end{bmatrix} \tag{2.35}$$

and has a single stationary point, $\mathbf{x}^* = (0, 0)$, with eigenvalues $\lambda_1 = 1 + 8i$, $\lambda_2 = 1 - 8i$, indicating that 1) since the eigenvalues are complex, the dynamics around the stationary point are rotational, and 2) since the real component of the eigenvalues is positive, the spiral is a source (i.e. repelling dynamics). It is also important to note here that eigenvectors corresponding to complex eigenvalues will themselves be complex, but they are guaranteed to span a two dimensional real subspace, on which the rotation lies. For the case of two dimensional systems, the plane spanned by the two complex eigenvectors corresponds to the whole state space (i.e. the $(x_1 - x_2)$ plane), but for higher dimensional systems, an additional step is required to find

this plane and this will be discussed later on.

Neutrally stable centres – Some systems contain stationary points with purely imaginary eigenvalues. These systems have attractors that are of the second of the three attractor types mentioned earlier, i.e. attracting sets. One such system is the two dimensional Lotka-Volterra system, often used to study the dynamics of predator-prey populations. For a particular choice of parameters that give rise to a *neutrally stable centre*, the system of equations is the following:

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} x_1 - x_1 x_2 \\ -x_2 + x_1 x_2 \end{bmatrix}. \tag{2.36}$$

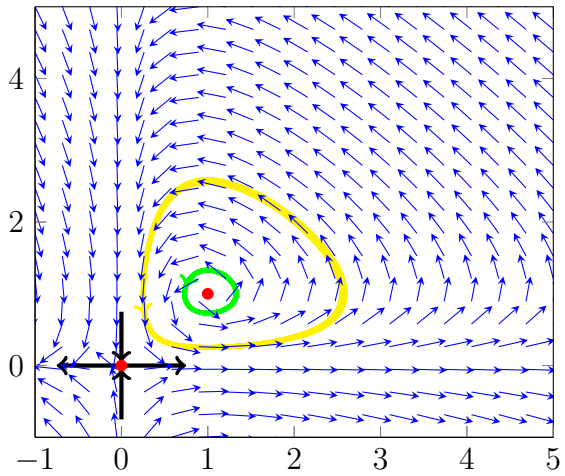


Figure 2.9: The vector field of the system described by Eq. 2.36. The two stationary points, $\mathbf{x^*}_1 = (0, 0)$ and $\mathbf{x^*}_2 = (1, 1)$, are shown with red marks and correspond to a saddle node and centre respectively. The two trajectories of the system are shown in green (initial conditions $\mathbf{x}_0 = (0.8, 0.8)$) and yellow (initial conditions $\mathbf{x}_0 = (2, 2)$). The black lines around $\mathbf{x^*}_1 = (0, 0)$ show the direction of the eigenvectors around the saddle point.

This system has two stationary points:

- A saddle point at $\mathbf{x^*}_1 = (0, 0)$ with $\lambda_1 = -1$, $\lambda_2 = 1$ and corresponding eigenvectors $\mathbf{e}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{e}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$
- A neutrally stable centre at $\mathbf{x^*}_2 = (1, 1)$ with $\lambda_1 = i$, $\lambda_2 = -i$.

Centres correspond to periodic orbits of the system, meaning that the system follows the same repeating trajectory with a constant period, without either moving closer or away from the central stationary point and is hence considered to be *neutrally stable*. This behaviour can be seen by the green and yellow trajectories superimposed on Fig. 2.9, which correspond to the evolution of the system for multiple periods around $\mathbf{x^*}_2 = (1, 1)$.

In general, stationary points with zero real components in their eigenvalues are considered to be *non-hyperbolic*. This is important for the case of non-linear dynamical systems, because the linearisation around a non-hyperbolic stationary points is not guaranteed, among other nice properties, 1) to have equivalent local dynamics to those of the original non-linear system and 2) to be structurally stable [Ott02], and hence care should be taken when considering such points.

Improper nodes – The final type of stationary points we will consider for the case of two dimensional systems are *improper nodes*. These are stationary points whose Jacobian matrix has repeated eigenvalues and hence repeated eigenvectors. If we consider, for example, the linear system

$$\begin{bmatrix} \dfrac{dx_1}{dt} \\ \dfrac{dx_2}{dt} \end{bmatrix} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 7x_1 + x_2 \\ \\ -4x_1 + 3x_2 \end{bmatrix}, \tag{2.37}$$

we can see that there is a single stationary point $\mathbf{x}^* = (0, 0)$ with Jacobian

$$\mathbf{J}(\mathbf{x}^*) = \mathbf{A} = \begin{bmatrix} 7 & 1 \\ \\ -4 & 3 \end{bmatrix}, \tag{2.38}$$

which has repeated eigenvalues $\lambda_{1,2} = 5$, with associated eigenvector $\mathbf{e} = \begin{bmatrix} 1 & -2 \end{bmatrix}^{\mathrm{T}}$, as shown in Fig. 2.37. Since the eigenvalues are repeated and positive, the stationary point is classified as an

unstable improper node, meaning that the system will deviate away from the stationary point as it evolves in time.

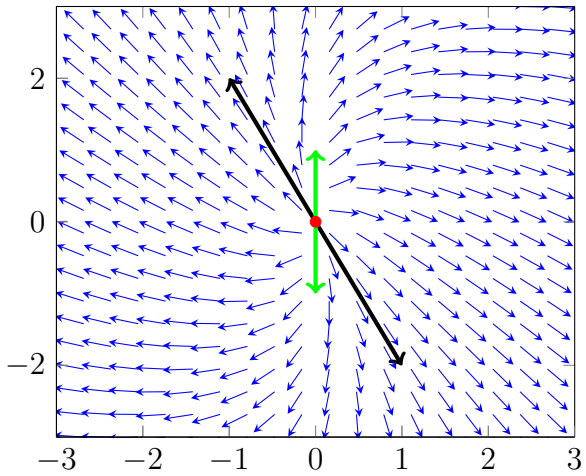


Figure 2.10: The vector field of the system described by Eq. 2.37. The single stationary point, $\mathbf{x}^* = (0, 0)$ is shown in red and corresponds to an improper unstable node. The black line around $\mathbf{x^*}_1 = (0, 0)$ shows the direction of the eigenvector corresponding to the repeated eigenvalue, while the green line shows the direction of a second, generalised eigenvector, obtained when trying to find the general solution of the system.

Since the system is linear, we could write down a closed form solution of the system, which in the general case is of the form

$$\mathbf{x}(t) = \sum_{k=1}^{D} c_k e^{\lambda_k t} \mathbf{e}_k, \tag{2.39}$$

where D is the dimensionality of the system, and $c_1, ..., c_k$ are coefficients that can be found by applying the initial condition of the system at $t = 0$, but for simplicity, they will be ignored in the following equations until the final step, when the final form of the solution will be specified. For systems with improper nodes though, because the eigenvalues are repeated, we can only get one eigenvector and hence the solution of the linear system described by Eq. 2.37 becomes

$$\mathbf{x}(t) = e^{5t} \mathbf{e}. \tag{2.40}$$

This is a particular solution that is valid for the case that the system is initialised along the

eigenvector $\mathbf{e} = \begin{bmatrix} 1 & -2 \end{bmatrix}^{\mathrm{T}}$. In order to find the general solution that encompasses trajectories of the system resulting from all possible initialisations, we need to come up with another term that involves a generalised eigenvector, let's call this vector $\mathbf{p}$, such that our general solution is not restricted solely along the eigenvector $\mathbf{e}$. In order to find such a vector, $\mathbf{p}$, we can assume that a possible solution of the system involves time as a variable, t, leading to the form

$$\mathbf{x}(t) = te^{5t}\mathbf{e} + e^{5t}\mathbf{p}, \tag{2.41}$$

which can be differentiated with respect to t as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \implies e^{5t}\mathbf{e} + 5te^{5t}\mathbf{e} + 5e^{5t}\mathbf{p} = \mathbf{A}(te^{5t}\mathbf{e} + e^{5t}\mathbf{p}). \tag{2.42}$$

After rearranging and comparing coefficients, we arrive at the two following conditions that must hold in order for Eq. 2.41 to be a valid solution.

$$\mathbf{A}\mathbf{e} = \lambda\mathbf{e} \implies (\mathbf{A} - \lambda\mathbf{I})\mathbf{e} = \mathbf{0} \quad (1) \tag{2.43}$$

$$\mathbf{A}\mathbf{p} = \lambda\mathbf{p} + \mathbf{e} \implies (\mathbf{A} - \lambda\mathbf{I})\mathbf{p} = \mathbf{e} \quad (2), \tag{2.44}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{0}$ is a vector with zeros with the same size as $\mathbf{e}$. From condition (1) we recover the fact that $\mathbf{e}$ is an eigenvector of the the linear system, and hence provides no new information, and from (2) we can find another vector, $\mathbf{e} = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}$ that will satisfy the proposed solution (Eq. 2.41). By adding together the two solutions shown in Eqs. 2.40 & 2.41 we arrive at the general solution of the system

$$\mathbf{x}(t) = c_1 e^{5t} \begin{bmatrix} 1 \\ -2 \end{bmatrix} + c_2 \left(te^{5t} \begin{bmatrix} 1 \\ -2 \end{bmatrix} + e^{5t} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right). \tag{2.45}$$

Hence we can find a set of linearly independent vectors that span the two-dimensional state

space of system. If we are further interested in finding the closed form solution of a particular initialisation of the system, we need to apply this initial condition in order to find the coefficients c_1, c_2. For example, if the system is initialised at $\mathbf{x}(0) = (1, 3)$, the coefficients are $c_1 = 1, c_2 = 5$ and hence, after simplification, the solution of the system becomes

$$\mathbf{x}(t) = e^{5t} \left(\begin{bmatrix} 1 \\ 3 \end{bmatrix} + 5t \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right). \tag{2.46}$$

What is further interesting about improper nodes is that they are structurally non-stable, in the sense that a small perturbation in the parameter values of the system, in this case the coefficients of the linear equations, leads to a different type of fixed point. When such a property holds, it is said that the system undergoes a *bifurcation* at the particular parameter value, leading to a sharp transition in the dynamics that is characterised by a qualitatively different behaviour of the system. Identifying bifurcations is useful because they can be used to partition the parameter space of the system and associate each region with a qualitatively distinct dynamic behaviour.

2.3.5 Bifurcations

In order to illustrate the concept of a bifurcation, we will more closely examine the case of the linear dynamical system described by Eq. 2.37. Writing the system in the general form

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \mathbf{A}\mathbf{x} = \begin{bmatrix} \alpha x_1 + \beta x_2 \\ \gamma x_1 + \delta x_2 \end{bmatrix}, \tag{2.47}$$

we can see that the stability of the stationary points will depend on the eigenvalues of the general Jacobian

$$\mathbf{J}(\mathbf{x}^*) = \mathbf{A} = \begin{bmatrix} \alpha & \beta \\ \\ \gamma & \delta \end{bmatrix}, \tag{2.48}$$

which in tern depend of the choice of the parameters α, β, γ and δ. In fact, solving for the characteristic polynomial, as it follows from Eq. 2.29, we arrive at the following general solution for the case of two dimensional linear systems:

$$\lambda_{1,2} = \frac{1}{2}\left((\alpha + \delta) \pm \sqrt{4\beta\gamma + (\alpha - \delta)^2}\right), \tag{2.49}$$

which solely depends on the four parameters. From Eq. 2.49 we can recover all types of stationary points described in Section 2.3.4, by simply choosing appropriate values for the parameters. What is even more instructive though, is to examine the stability of the solutions to the eigenvalue problem. To do this, let's consider the last example, described by Eq. 2.37 and perturb one of the parameters of the system by a small amount, such that the system becomes

$$\begin{bmatrix} \dfrac{dx_1}{dt} \\ \\ \dfrac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} 7.01x_1 + x_2 \\ \\ -4x_1 + 3x_2 \end{bmatrix}. \tag{2.50}$$

Using Eq. 2.49, we can see that the stationary point is still $\mathbf{x}^* = (0, 0)$, but now we get two distinct eigenvalues, $\lambda_1 \approx 5.1465$ and $\lambda_2 \approx 4.8635$, meaning that the system no longer has an improper unstable node, but rather an unstable node. Even more, if we instead use a small perturbation in the opposite direction, such that

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} 6.99x_1 + x_2 \\ -4x_1 + 3x_2 \end{bmatrix}, \tag{2.51}$$

the eigenvalues at the stationary point $\mathbf{x}^* = (0, 0)$ are now $\lambda_1 \approx 4.995 + 0.14133i$ and $\lambda_2 \approx 4.995 - 0.14133i$, hence leading to an unstable spiral.

Another case involving this general two dimensional linear system is when $\alpha = -\beta$, in which case the system has purely imaginary eigenvalues and hence the dynamics are characterised by a neutrally stable centre. Any perturbation in either α or β will lead to a system that has eigenvalues with non-zero real components and hence dynamics described by spiral nodes. Figs. 2.11-2.12 show the qualitative change in the vector field of the system under such small perturbations.

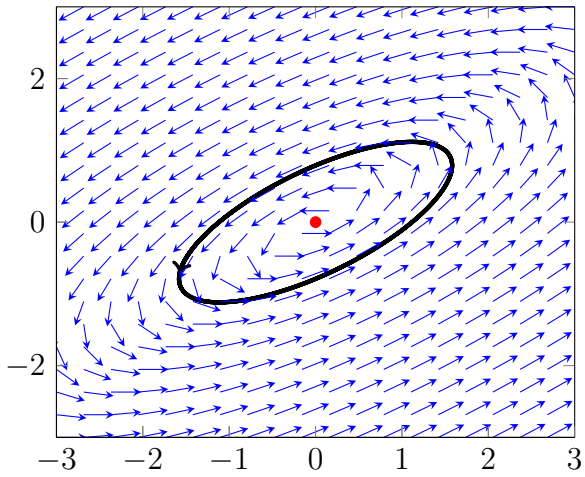


Figure 2.11: The vector field of the linear dynamical system described by Eq. 2.47 and parameter values $\alpha = 1, \beta = -2, \gamma = 1$ and $\delta = -1$. The single stationary point, $\mathbf{x}^* = (0, 0)$ is shown in red and corresponds to a centre. The black line around $\mathbf{x}^* = (0, 0)$ corresponds to a trajectory of the system, with initial conditions $\mathbf{x}_0 = (-1.7, 1.7)$.

Even though we analysed the behaviour of a linear, two dimensional system, bifurcations occur in other types of systems, such as the one dimensional logistic map [May76] and the non-linear Lotka-Volterra system with time delay [YL10]. In Chapters 3 & 4 we will investigate the dynamical transitions and bifurcations of the CTRNN model.

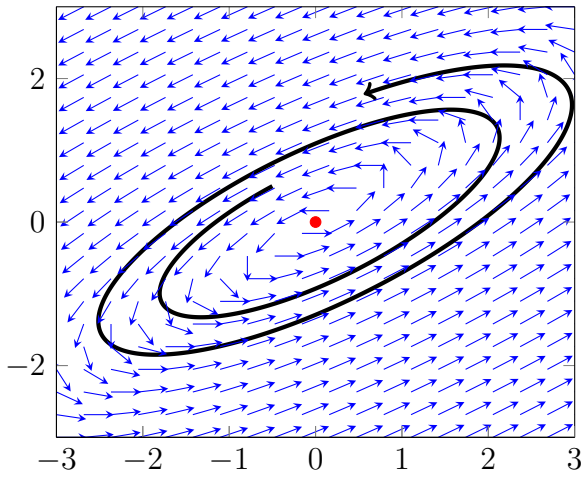


Figure 2.12: The vector field of the linear dynamical system described by Eq. 2.47 and parameter values $\alpha = 1.1, \beta = -2, \gamma = 1$ and $\delta = -1$. The single stationary point, $\mathbf{x}^* = (0, 0)$ is shown in red and corresponds to an unstable spiral. The black line around $\mathbf{x}^* = (0, 0)$ corresponds to a trajectory of the system, with initial conditions $\mathbf{x}_0 = (-0.1, 0.1)$.

2.3.6 Stationary points in three dimensions and higher

So far, we have only looked at linear and non-linear systems in two dimensions. Extending these results to more dimensions is straightforward because by linearising around fixed points, and finding linearly independent eigenvectors along which eigenvalues operate, we can treat each of the eigenvalue-eigenvector pairs as independent. This way, we can simply add more terms in Eq. 2.39, which describes the approximate linear system around the stationary point. For example, the diagrams in Fig. 2.13 & 2.14 illustrate how the dynamics around stationary points with non-zero eigenvalues extend if we add a third dimension to the system.

Even though results from two dimensional systems generalise to more dimensions in a straightforward way, the analysis of very high dimensional systems using algebraic manipulations becomes very challenging. Chapters 3 & 4 present an adaptation of the simple approaches presented in the previous subsections to higher dimensional systems, in order to reveal the structure of vector fields and transitions that occur in CTRNNs with hundreds of neurons.

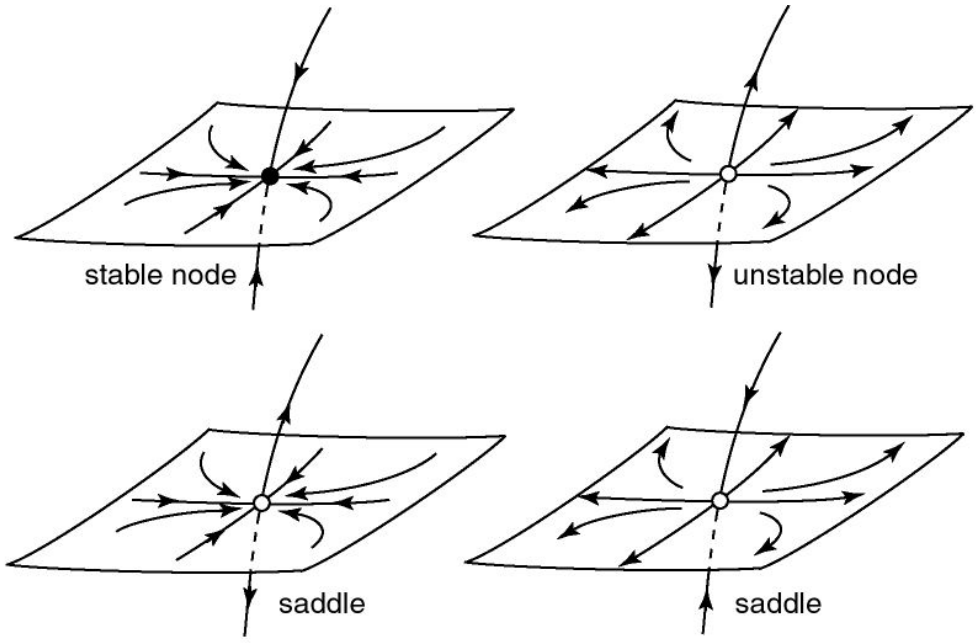


Figure 2.13: The different types of local dynamics exhibited by stationary points in $\mathbb{R}^3$ with real eigenvalues. [Figure from [Izh07a]].

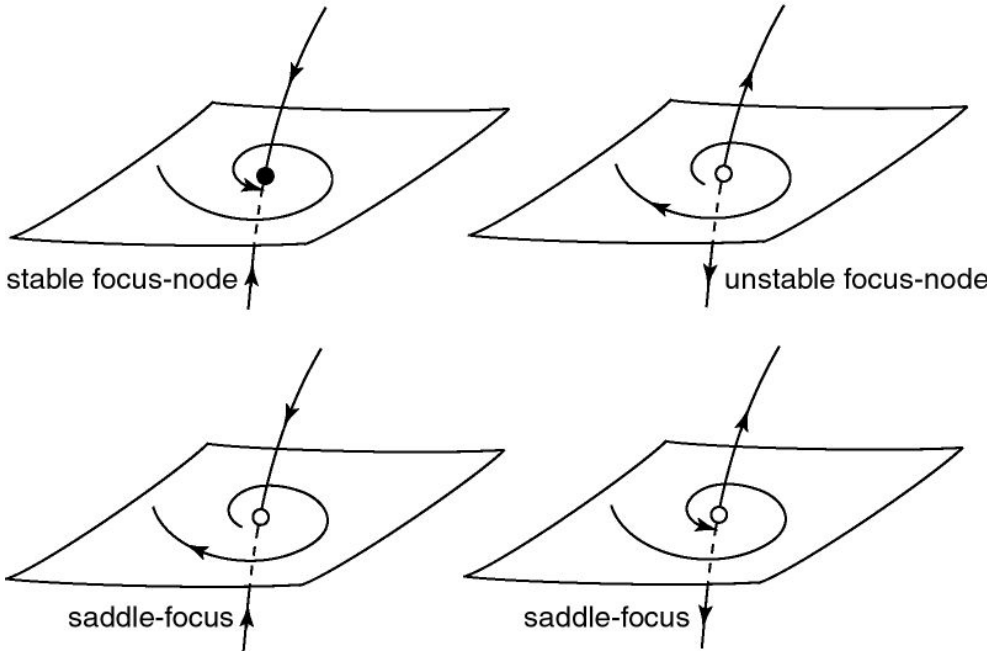


Figure 2.14: The different types of local dynamics exhibited by stationary points in $\mathbb{R}^3$ with a combination of real and complex eigenvalues. [Figure from [Izh07a]].

Chapter 3

Dimensionality of dynamics in driven continuous-time recurrent neural networks

3.1 Precis

Continuous-time recurrent neural networks (CTRNNs) are widely used as models of cortical circuits [SCKS15, MDS16], because they share similar dynamic properties, and also have applications in biologically inspired machine learning [ASS08, HS12, Sus14, BNS+16]. Nonetheless, their dynamics are not yet well understood, especially when they are driven by external stimuli. This chapter presents a study of the response of stable and unstable CTRNNs to simple periodic stimuli of different frequencies. Specifically, the dimensionality of the network's attractor is used as an indicator of the complexity of this response, and using a common dimensionality reduction technique we can visualise the network's trajectory as it exhibits a sequence of phase transitions, manifesting as qualitatively distinct changes in the shape of the low dimensional attractor. Thus, we can understand how complex activity in the network emerges depending on the relative timescales of the network and the periodic input[1].

[1]The basis and part of the work presented in this chapter was published in the paper "An Investigation of the Dynamical Transitions in Harmonically Driven Random Networks of Firing-Rate Neurons" [NMS17b], which was published with Open Access, under the Creative Commons License (http://creativecommons.org/licenses/by/4.0/).

3.2 Introduction

Both human and non-human primates are able to produce a rich repertoire of motor behaviour that is supported by the brain's ability to flexibly switch between various patterns of neuronal activity. For instance, motor-related neural activation patterns in the motor cortex vary significantly to generate the required responses according to task and context, but a comprehensive mechanism for explaining this behaviour is still missing. There are two main hypotheses regarding what is represented by the activity patterns in the motor cortex.

The first one derives from what is known about representation in the visual cortex, i.e. that activity of neurons represent the detection of visual patterns [HW68, SKR13], and hypothesises that neural activation patterns represent kinematic parameters related to movement [Fet92]. This view has led to a yet unresolved controversy, regarding whether neural activation patterns encode lower-level kinetic features, such as the torque applied by the joint or higher-level kinematic features such as position, velocity, and direction of the limb [Hat05].

The second hypothesis, which is more relevant in this work, views the motor cortex as a dynamical system that through the collective activity of the neurons functions as the generator and controller for movement, and postulates that movement-related neural activity can be viewed as low dimensional trajectories of the high dimensional dynamical system. Over the past few years, this hypothesis has overtaken the representational hypothesis and recent work seems to provide evidence in its favour [CCK$^+$10, CCK$^+$12, HVG14, CdLR$^+$15].

One of the first studies was by Churchland *et al.* [CCK$^+$12], who used 469 single-neuron recordings from the motor and pre-motor cortex of four monkeys to analyse their neural patterns during reaching tasks. They found that rotations of the neural population state are a predominant feature of the cortical response during reaching and that these rotations could explain the complex and unpredictable response of individual neurons during movement. Moreover, using a variant of principal component analysis (PCA), which they termed jPCA, they found that these rotations lie at a much lower dimensionality than the actual dimensionality of the system. More precisely, they found that rotations along three orthogonal planes could capture approximately

50-70% of the total variance of their recorded data. These findings support the hypothesis that the representation of movement in the motor and pre-motor cortex is realised at the population level rather than by individual neurons, and hence motivate the view of the cortex as a dynamical system whose function is to generate movement-related activity.

In a follow-up study [KCRS14], it was shown that the neural activity in the monkey motor cortex preceding movement cancels out at the level of the readout to the muscle, thus allowing complex, movement-related computations to take place without initiating any movement. This property can be understood by considering the prior finding, that the actual neural trajectories, both in the preparatory period and during movement, have a much lower dimensionality than the actual dimensionality of the system, and that these trajectories lie in different subspaces of the full space. The subspace in which the preparatory activity evolves is in the null-space of the linear readout to the muscles, meaning that the output of the readout will always be near zero while the activity of the motor cortex lies in this null-space during the preparatory period. Once movement is to be initiated, the activity moves out of the null-space of the readout to the muscles and then the motor cortex can communicate with the muscles to send motor-related signals.

The study of the motor cortices from the stand point of dynamical systems theory is a suitable area for the use of CTRNNs as computational models [Sus14, Bar17]. For instance, in recent work trying to understand population dynamics of motor cortex neurons during movement [SCKS15, MDS16], randomly connected CTRNNs of the form of Eq. 2.3 were used to show that the empirically measured limb movement could be decoded from the neural network population activity, hence supporting the hypothesis that the population is the appropriate level to encode the necessary computation for movement. Furthermore, this hypothesis and associated computational models have also been applied to related directions of research, such as trying to understand interval timing generation [LB13, WNHJ18] and context-dependent computation in the prefrontal cortex [MSSN13, CSFW17]. The feature that makes CTRNNs unique for this purpose is their inherent temporal nature, defined through their connectivity, that can be trained to generate brain-like activity. The development of efficient learning algorithms [SA09] that take advantage of their rich dynamic behaviour has proven important for the wider use of

this class of models.

Even though traditionally, the focus on understanding the dynamics of such networks has been on structural parameters of the network, such us its connectivity, and how the process of training can adjust its dynamic behaviour, the external input has an equally important, but commonly overlooked, effect on the dynamics of the network. This external input represents signals received from other neuronal populations, ranging from the lower sensory layers to other cortical areas, functioning as a medium for communication, and is therefore a common feature of neural circuits. Rajan *et al.* [RAS10] showed that periodic external inputs delivered with a different phase to each neuron can suppress the chaotic activity of intrinsically chaotic networks, but more work is needed to better understand the mechanism through which the input affects dynamics. Inspired by this recent work related to the response of cortical populations to stimuli, the current chapter present a systematic study of how CTRNNs respond to external input signals. The goal is to investigate, both visually and numerically, the attractor dimensionality of externally driven CTRNNs, using two different methods. One is standard PCA and the other one is adapted from Tajima *et al.* [TYFT15], who used this method to show that the attractor complexity of brain dynamics is higher in downstream (cognitive) areas, compared to the upstream (sensory) areas from which they receive direct input.

What this work aims to achieve through this investigation, is to identify the complexity of CTRNNs' response to an external periodic signal, at the timescales of both the input and the network. Its ambition aims to provide additional computational evidence supporting the suitability of CTRNNs for modelling the cortex as a dynamical system, as well as useful insights when considering the design of reservoirs for machine learning applications.

3.3 Methods

3.3.1 Network simulations

Neuron model

The neuron model [WC72, SCS88] used in these experiments was of the form of Eq. 2.3, without an output or any feedback connections and a single time varying input signal $s(t)$, as shown in Fig. 3.1. This simplification leads to the following system of equations:

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} \tanh(x_j) + w_i^{In} s(t) \qquad i = 1, 2, ..., N. \quad (3.1)$$

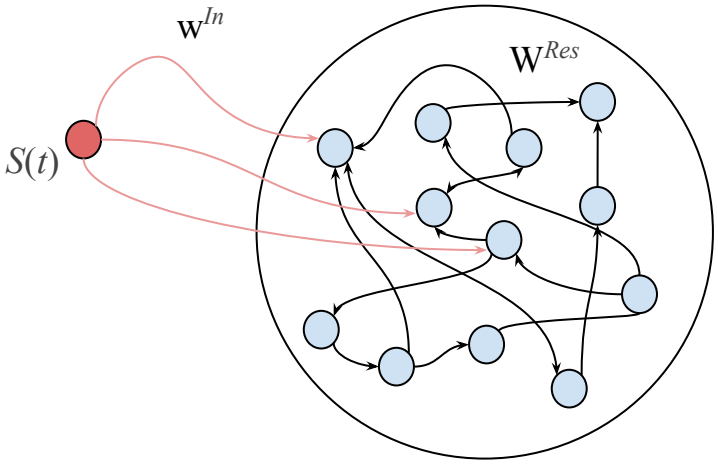


Figure 3.1: The network connectivity used for experiments in this chapter, resembling Eq. 3.1. In the experiments conducted, networks of N neurons were driven by an external periodic oscillator $s(t) = \sin(\alpha t)$, to approximate the predominant type of activity present in the human brain, in the and their response for varying frequencies α was studied. All neurons received input from the external oscillator with a weight $w_i^{In} \sim \mathcal{N}(0, 1)$. The recurrent connections in $\mathbf{W}^{Res}$ were given by a random Erdös-Renyi graph with connection probability $p_{res} = 0.1$, no self-connections and weights $W_{ij}^{Res} \sim \mathcal{N}(0, g_G^2)$, with $g_G = g/\sqrt{p_{res}N}$ acting as a global scaling factor.

g is usually referred to as the *gain* of the connections, an important parameter that strongly affects the dynamic behaviour of the system. It has long been established that autonomous networks with $g < 1$ exhibit attracting dynamics towards a globally stable stationary point, whereas networks with $g \geqslant 1$ can exhibit complex periodic or even chaotic behaviour [SCS88]. On the other hand, the activity of input driven networks is heavily dependent on the input signal. The input signal can play such a pivotal role in the network's activity that it can even suppress chaotic activity in networks with g values greater than 1 [RAS10]. Finally, the variable r_i is the observed activation, or firing rate, of neuron i and lies in the interval $[-1, 1]$.

Once the connectivity matrix $\mathbf{W}^{Res}$ has been specified, the two parameters determining the network's response and complexity are the temporal parameters of the network, namely τ and α. Specifically, the relative timescale of the network and the stimulus can fully determine the network's response. Therefore, the driven system in Eq. (3.1) can be reparametrised using $\hat{t} = \tau^{-1}t$ as

$$\frac{dx_i}{d\hat{t}} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} \tanh(x_j) + w_i^{In} \sin\left(\rho\hat{t}\right) \qquad i = 1, 2, ..., N \ , \quad (3.2)$$

where the only parameter of interest is now the dimensionless quantity ρ, the ratio between the neurons' timescale τ and the period of the driving signal τ_S, defined as

$$\rho = \alpha\tau = 2\pi\frac{\tau}{\tau_S} \ . \quad (3.3)$$

It is important to note that Eq. 3.2 is used here only for aiding the reader to better understand where ρ originates, and what is the relation between the time scale of the network and the input. The network that was actually simulated, as in other work using CTRNNs, was the one defined by Eq. 3.1, where τ and α could be varied independently from each other.

Due to its direct influence on the network's dynamics, it is instructive to think of $s(t)$ as bearing some resemblance to an external driving force acting on the network. However, it is not a force in the strict sense because it is applied to the first-order time derivative of $\mathbf{x}$ rather than the second-order one; as would happen in the case of driven oscillators obeying Newtonian

mechanics. Nonetheless, it is informative to use the analogy of a driving force to gain a more intuitive appreciation of the effect of the driving signal on the network's dynamics.

Numerical simulations

Two different types of simulations were carried out; one was focused on analysing the dimensionality of dynamics at the timescale of the input signal and one at the timescale of the network. This distinction is made because the resolution of the input signal and the network's trajectory vary significantly over the range of ρ values used. This difference is not a mere limitation of our numerical simulations, but also a practical consideration on which of the two timescales is relevant for the purpose of the simulation. For example, in machine learning applications, one is often interested in various types of computation on the input signal. The output of these computations is usually at the same timescales as the input. We are therefore interested in dynamics of the network at the timescale of the input signal, which can be orders of magnitude different than that of the network. On the other hand, the focus might be on the longer term dynamics of the network, irrespective of the timescale of the input. To account for both these perspectives, the analysis was carried out with respect to both timescales separately.

For investigating the dimensionality of the dynamics at the timescale of the input, the frequency of the input was kept constant at $\alpha = 10$ and τ was varied instead. Since these simulations were ran in the range $1 \leq \rho \leq 10^6$, the effective Euler step size was increased to 1 in the worst case, which corresponds to the Euler step used in previous work [LB13]. For this type of experiments, every time step was recorded, thus allowing us to analyse the networks' dynamics at the timescale of the input.

For investigating the dimensionality of the dynamics at the timescale of the network, in order to change the value of ρ between different simulations where $\rho \geq 10$, the frequency of the sinusoidal input $s(t)$, was fixed at $\alpha = 1$, and τ was varied instead. This way, the discretisation of the sinusoidal input was always smooth, even for large values of ρ in all simulations, and hence this method was preferred to avoid any numerical problems that might arise when decreasing the discretisation of $s(t)$ for larger ρ values. In simulations where $\rho < 10$, the input frequency

α was varied instead, by increasing its discretisation and using $\tau = 10$ to avoid numerical instabilities in the network's numerical integration. It is useful to note at this point that τ is inversely proportional to the rate at which the network state evolves (Eq. (3.1)), implying that for networks with large τ, the network state evolves much slower compared to networks with small τ. This slower evolution of the network's trajectory results in a reduction of the curvature of the trajectory, which means that the dynamics appear approximately linear if the timescale is not adjusted. To account for this, the total number of time steps for each simulation was scaled according to the value of τ. For example, a simulation with $\tau = 100$ was run for 10 times the number of total time steps than the simulation with $\tau = 10$, but the number of samples stored for analysis was kept the same by recording the trajectory of the network 10 times less frequently. For the case of $\rho < 10$, since τ was kept constant, the state of the network was recorded at every time step. Simulations were ran over the range $10^{-2} \leq \rho \leq 10^4$ for this type of investigation.

For both types of simulations, the procedure was the following: Simulations were performed using the Euler integration method with step size 0.01 for each time step and run until a total of 3500 points were recorded in each type of simulation. At the end of every time step of the simulation, it was decided whether to save the point or not, depending on the value of τ and the type of simulation. The networks were initialised randomly and received a short, strong pulse of amplitude 5 through $\mathbf{w}^{In}$ after 200 (recorded) time steps, for 50 (recorded) time steps. The purpose of this pulse was to guide the network to a bounded region of its state space with a small hyper-volume, such that all five repetitions of the same network had close, but not identical initial states. The oscillatory input was then applied for the remaining of the simulation. In addition, even though the initial pulse is not strictly necessary for the analysis performed here, it was nonetheless added to the network in order to ensure that the network experiences similar inputs for the analysis presented in this chapter, to those that it will be experiencing in the experiments presented in Chapter 5. Specifically, this input pulse acts as an external "stimulus" that, even though considerably "stresses" the network dynamics, acts as a *cue* that allows the network to associate particular *memories* to particular trajectories in its state space. Hence, to make the analysis presented here as relevant as possible to the

experiments presented in Chapter 5, this initial pulse was also applied here.

The first 1500 recorded points of the simulations were discarded as transients, and the network activity for the remaining simulation was analysed. Discarding the transient behaviour of the network right after the initial pulse, enables us to focus on only the "steady-state" dynamics of the driven network, which makes our analysis simpler. Importantly though, the transients that were discarded might actually incorporate network dynamics that are the most relevant for computation in biological systems, and hence future work should aim to focus on analysing this transient activity. Finally, for each simulation, new sets of input and network connectivity weights, $\mathbf{w}^{In}$ and $\mathbf{W}^{Res}$ respectively, were generated and each one was run for five repetitions. The final values reported are the average of those five repetitions, along with error bars denoting the standard error.

3.3.2 Dimensionality of dynamics

The quantity of interest in this chapter will be the dimensionality of the network's resulting dynamics under the influence of an external periodic signal of different frequencies. The reason we are interested in this quantity is because it is common for individual components of a coupled system, neurons of the network in this case, to have highly correlated activity that results in the system only visiting a subset of its possible configurations.

This usually results in the network having a finite *attractor dimensionality* D, such that the dynamics of the network effectively lie on a lower dimensional attractor, a manifold of dimension d, which is significantly smaller than the number of neurons N in the network. A higher attractor dimensionality indicates more complex dynamics, and usually the presence of higher order correlations between the neurons, which are a feature of particular importance in the framework of reservoir computing, as the network can engage in more sophisticated computation. In this chapter we will explore and compare the attractor dimension of driven stable and unstable networks using two different techniques, one based on a linear estimator and one based on a non-parametric estimator.

Linear estimator — The linear estimator, which will be denoted by D_{PCA}, is based on applying principal component analysis (PCA) to the network's trajectory, after centering the data to zero and scaling the variance to one. The attractor dimensionality is then simply defined as the minimum number of principal components (PC) required to explain 95% of the variance in the network's trajectory, given that the principal components are ordered by percentage of total variance they explain. Even though, ideally, we would want 100% of the variance to be captured by the dynamics on the attractor, we will allow for 5% unexplained variance to account for other directions of motion that lie outside the subspace of the attractor. The fraction of the total variance explained in these later components is significantly smaller than the variance explained by the initial components and hence we will be assuming that the true subspace, wherein the attractor lies, can be approximated by a linear combination of the first D_{PCA}, on average, principal components needed to explain at least 95% of the variance. This of course is an approximation as the PCA method finds the eigenvectors based only on the variance explained; it does not consider higher order correlations. Typical behaviours of the linear estimator are presented in Figs. 3.4a & 3.16a below for the case of stable and unstable networks respectively.

Non-linear, non-parametric estimator — The calculation of the non-linear, non-parametric estimator D_{kNN} is slightly more involved and is taken from recent work by Tajima *et al.* [TYFT15], who used this estimator to show that the attractor dimensionality of brain dynamics is higher in downstream (cognitive) regions compared to the upstream (sensory) regions, from which they receive direct input[2] The algorithm to calculate D_{kNN} is based on Taken's embedding theorem of delayed-time embeddings, which, in simple words, states that it is possible to reconstruct the topology of the manifold of a dynamical system, up to a diffeomorphism, by simply taking delayed-time embeddings of only one of its variables. Since the reconstructed dynamics and the original dynamics are equivalent up to a diffeomorphism – a differentiable and invertible mapping – it follows that the reconstructed attractor is a suitable proxy to study the whole system's dynamics. In particular, standard k-nearest neighbour regression [Bis06] can

[2]Credit for implementing the estimator in MATLAB, based on the description of Tajima *et al.* [TYFT15] goes to Pedro Martinez Mediano.

be used to estimate the reconstructed attractor dimension, using the following steps:

1. Run the simulation five times with different initial conditions. Centre and standardise the data to mean zero and unit variance.

2. Choose 150 pairs of neurons at random; each pair from the same run.

3. Construct the delayed-time embedding of the time series for one of the two neurons, using a delay of τ_d chosen at random from the set $\{4, 5, ..., 50\}$ unless stated otherwise, for the maximum number of dimensions ($d^{max} = 20$).

4. Multiply the embedding from the left with a square random matrix $\mathbf{R}$, which is equivalent to convolving the time series with pre-set random filters. Tajima *et al.* [TYFT15] claims that this random projection makes the estimation less sensitive to the variety of timescales present in the data and hence less sensitive to the choice of τ_d. Furthermore it can destroy autocorrelations and temporal oversampling when choosing nearest neighbours [KBA92, KA02, KG92].

5. Iteratively choose an increasing number of dimensions and for each iteration first find the $k = 4$ nearest neighbours using Euclidean distance and by constructing a search tree (default parameters of MATLAB's `knnsearch` function [MAT5a]), followed by a weighted prediction $\hat{y}_t$ for each time point t. The prediction can be made using the true activation values of the other neuron, y_{t_i}, with corresponding time indices to those of the k nearest neighbours and $i = 1, ..., k$, as follows:

$$\hat{y}_t = \sum_{i=1}^{k} w_i y_{t_i} \ , \tag{3.4}$$

where

$$w_i = \frac{e^{-\left\|\mathbf{x}_t - \mathbf{x}_t^i\right\|^2}}{\sum_{i=1}^{k} e^{-\left\|\mathbf{x}_t - \mathbf{x}_t^i\right\|^2}} \ , \tag{3.5}$$

and $\mathbf{x}_t$ refers to point in the reconstructed space at the current time step t, $\|\cdot\|$ denotes the Euclidean distance operation and $\mathbf{x}_t^i$ refers to the i-th nearest neighbour of the point at the current time step.

6. Once the predictions have been made for all time steps in each iteration, the correlation coefficient $\rho_{\hat{\mathbf{y}},\mathbf{y}}$ between the true activations and the predictions can be calculated.

7. Henceforth, and as per Tajima *et al.* [TYFT15], for the case of prediction performance plateauing with increasing embedding dimension, D_{kNN} is defined as the minimum number of embeddings required to capture more than 95% percent of the variance of the optimal predictor, i.e. the one with the highest $\rho_{\hat{\mathbf{y}},\mathbf{y}}$. For some cases, it might be possible to get a peak, followed by a decrease in performance as the number of embedding dimensions increases. For these cases D_{kNN} is defined as the number embeddings yielding the highest value of $\rho_{\hat{\mathbf{y}},\mathbf{y}}$.

Typical behaviours of the non-linear estimator are presented in Figs. 3.4b & 3.16b below for the case of stable and unstable networks respectively.

3.4 Results

The results section is differentiated into four main subsections. The first two are concerned with networks that we will refer to as *stable* and have a g value of 0.9. With this g value, the resulting connectivity matrix $\mathbf{W}^{Res}$ has a spectral radius of less than unity, meaning that all the eigenvalues of the connectivity matrix lie within a unit circle on the imaginary plane, centred at the origin. This results in autonomous dynamics that are locally contractive and hence the network activity converges to a fixed point of dimensionality 0 if it evolves autonomously. On the other hand, the two last sections are concerned with networks that are *unstable* and have a g value of 1.5, a connectivity matrix with spectral radius higher than unity, and locally expansive

dynamics that give rise to complex, non-periodic autonomous activity with $D_{\text{PCA}} = 18.25 \pm 2.06$ and $D_{\text{kNN}} = 3.50 \pm 0.75$.

As a note to the interested reader, the relationship between the gain of the network g and the spectral radius of the connectivity matrix is, closely linked to results from Random Matrix Theory [TV08], which say that for random matrices with weights drawn independently from the standard normal distribution, their eigenvalues lie uniformly within the unit disk in the complex plane, centred at the origin. The proofs are complicated and deviate significantly from the scope of this work, so they are not reproduced here. Nonetheless, a simple way to think about the connection between the gain g and the spectral radius of $\mathbf{W}^{Res}$, is to see that $W_{ij}^{Res} \sim \mathcal{N}(0, g^2/p_{res}N)$. Here, the variance term, $g^2/p_{res}N$, ensures that the value of g is equal to the radius of the disc – in the complex plain and centred at the origin – that the eigenvalues of $\mathbf{W}^{Res}$ are guaranteed to uniformly lie in.

3.4.1 Dimensionality of dynamics in stable networks at the input timescale

Single network

For the first set of experiments, a stable network ($g = 0.9$) and number of neurons $N = 200$ was created. All weights of the network were randomly generated before the first simulation, as per Methods Section 3.3, and were used throughout this first set of experiments. Then, 200 ρ values were selected at random, by uniformly sampling in the interval $[0, 6]$ and exponentiating the base 10 by each of these samples. A simulation was ran for each of these ρ values, the trajectory recorded at every time step and the dimensionality of the attractor was measured by both D_{PCA} and D_{kNN}. Contrary to the general experimental protocol described in Section 3.3, and only for this set of experiments, a single simulation was ran for each ρ value. The results are shown in Fig. 3.2 below.

The clear trends observed in Fig. 3.2 for the two different estimators show that the network dynamics undergo an increase in dimensionality for intermediate values in the chosen range of

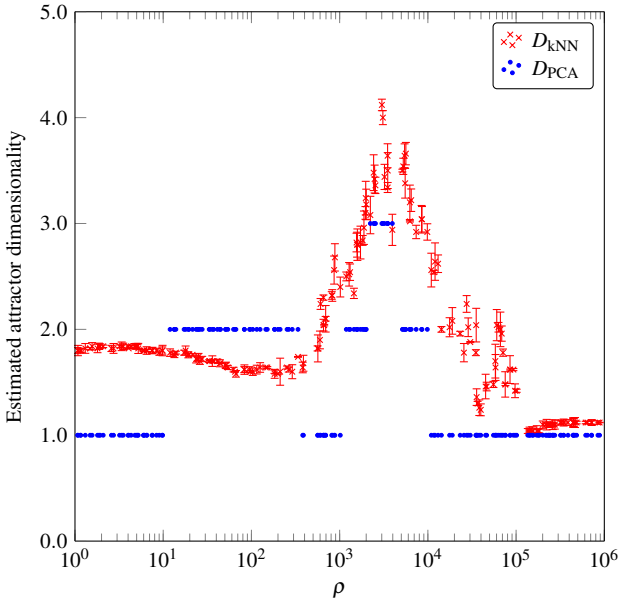


Figure 3.2: Underlying attractor dimensionality, at the input timescale, of a stable network with $N = 200$ neurons and $g = 0.9$ for different ρ values, with the two proposed estimators. The same network weights were used in all simulations. For each ρ value, a single run of the network was simulated. 50 pairs of neurons were used to estimate the attractor dimensionality at each data point for the non-linear estimator D_{kNN}.

ρ. For the non-linear estimator, D_{kNN}, the dimensionality of the network dynamics remains at a value of slightly less than 2 up until about $\rho \approx 10^3$, when it starts increasing until it reaches a peak value of about 4 at $\rho \approx 2 \times 10^3$. from there on, it decreases until it plateaus to a dimensionality of 1 for ρ values between $10^5 - 10^6$. The linear estimator D_{PCA} behaves in similar, but not identical fashion. For very low or very high values of ρ it estimates a low dimensionality of 1, whereas for intermediate values, a peak dimensionality of 3. Contrary to the non-linear estimator, D_{PCA} estimates an increased dimensionality of 2 for values of ρ between $10^1 - 3 \times 10^2$, followed by a drop in dimensionality to 1 at about $3 \times 10^2 - 10^3$, before increasing to its peak value.

The mismatch in the behaviour of the linear estimator for values of ρ between $10^1 - 3 \times 10^2$ indicates that for this range, the total variance of the trajectory of the network spreads from the first to the second principal component (PC). For ρ between $3 \times 10^2 - 10^3$, the total variance gathers back to the first PC and for ρ values where the peak in dimensionality is observed, it spreads until it reaches the third principal component. In general, for a stable network, the two estimators produce somewhat similar trends for the change in dimensionality with increasing ρ.

Randomly generated networks

Having studied the changes in the dimensionality of dynamics for a single network, we will now see whether a similar response holds for randomly generated networks. For each of the about 160 data points, a network with sparse connectivity was generated, as per Methods Section 3.3, and simulated with a ρ value drawn uniformly in exponent space. This aim of this set of experiments was to determine whether randomly generated networks that have connectivity matrices with the same statical properties, exhibit equivalent changes in the dimensionality of their dynamics. Fig. 3.3 below shows the results using the two estimators.

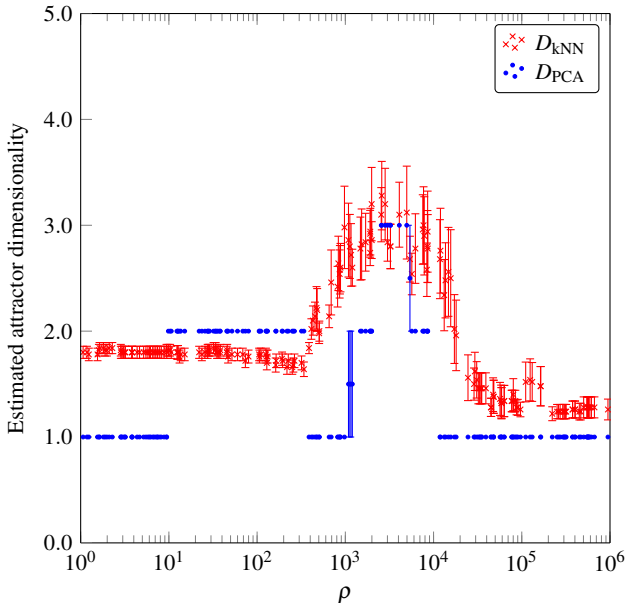
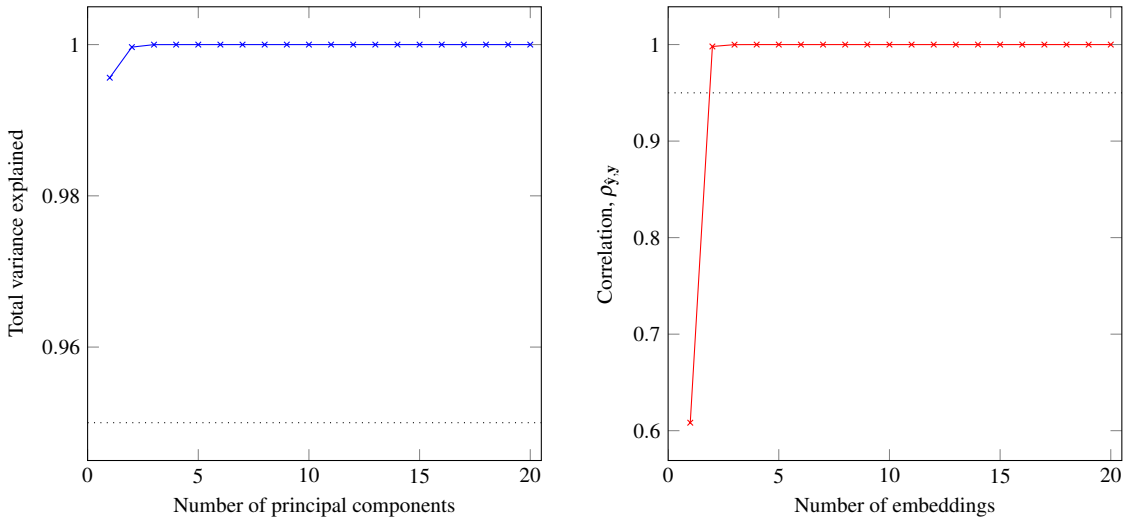


Figure 3.3: Underlying attractor dimensionality, at the input timescale, of stable networks with $N = 200$ neurons, $\tau_d = 4$ and $g = 0.9$ for different ρ values, with the two proposed estimators. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the same network. Figure appeared in [NMS17b].

Fig. 3.3 shows a similar trend for the change in dimensionality as ρ is varied in randomly generated networks to the trend exhibited in the experiments of a single network. Moreover, the variance of both estimators remains relatively low across trials of a given frequency, particularly for very low and very high ρ. Therefore, we can conclude that these results hold for the whole family of networks of 200 neurons generated through this method.

In Fig. 3.4 we can see the typical behaviour of the two estimators used in this study, for the case of stable networks. In Fig. 3.4a we can see how the cumulative total variance explained increases



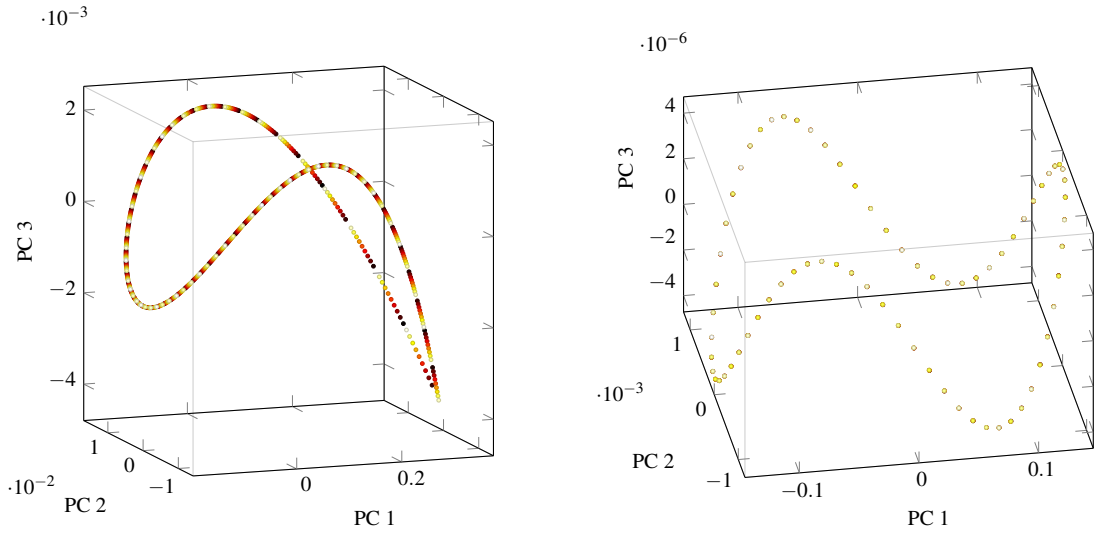
(a) D_{PCA} (b) D_{kNN}

Figure 3.4: Typical plots of how the measures behave for a network of $N = 200$ neurons, $g = 0.9$, $\tau_d = 4$ and $\rho = 10$. For this simulation $D_{\text{PCA}} = 1.00$ and $D_{\text{kNN}} = 1.90 \pm 0.01$. (a) Cumulative total variance explained by principal components. The dotted line corresponds to 95% of the total variance explained. D_{PCA} was estimated as the minimum number of principal components to exceed this value. (b) Total correlation $\rho_{\hat{\mathbf{y}},\mathbf{y}}$ between k-nn prediction $\hat{\mathbf{y}}$ and true activation $\mathbf{y}$ as number of embedding dimensions increases.The dotted line corresponds to 95% of the total variance explained. D_{kNN} was estimated as the the minimum embeddings needed to exceeded this value.

with an increasing number of principal components. Since for the simulation shown, ($\rho = 10$), more than 99% of the variance is already accounted for by the first component, the resulting dimensionality estimated by D_{PCA} for this run is 1. On the other hand, Fig. 3.4b shows that we need at least two embeddings to get a correlation ($\rho_{\hat{\mathbf{y}},\mathbf{y}}$), between the k-NN prediction ($\hat{\mathbf{y}}$) and true activation ($\mathbf{y}$), that is more than 95% of the correlation achieved by the optimal number of embeddings, and hence the estimated dimensionality with D_{kNN} is 2. In both plots, the dotted black lines denote the thresholds used to choose the dimensionality estimates.

In order to aid our intuition, a single run from each of two simulations, one with $\rho = 10$ and the other $\rho = 1000$ were used to visualise the trajectory of the full system, by projecting it on the three principal components extracted with PCA. Fig. 3.5a shows the projection of the periodic attractor exhibited by the system. Its geometric shape can be described as a single full period of a sinusoidal-like wave, whereas the attractor shown in Fig. 3.5b with $\rho = 1000$ appears as having a similar behaviour but with double the frequency of the sinusoidal-like wave per period of the system. This shows that for the case of stable networks, their trajectories become *entrenched*

by the sinusoidal input and exhibit sinusoidal-like oscillations with increasing frequency as the value of ρ increases.



(a) $\rho = 10$ (b) $\rho = 1000$

Figure 3.5: Plots of the activity, at the input timescale, for networks of 200 neurons and $g = 0.9$ driven by different ρ values, projected on their three principal components and one of the five runs. (a) $\rho = 10$, $D_{\text{PCA}} = 1.00 \pm 0.00$, $D_{\text{kNN}} = 1.90 \pm 0.01$. (b) $\rho = 1000$, $D_{\text{PCA}} = 1.50 \pm 0.50$, $D_{\text{kNN}} = 2.75 \pm 0.45$. Notice the difference in the scales of the three components and how they affect the linear estimator D_{PCA}.

Another interesting observation is that the scales of the three principal components appear to be different from one another and their relative difference also appears to change depending on ρ. For instance, for $\rho = 10$ there is only one order of magnitude increase between successive components, whereas it decreases by 2 orders of magnitude between PC1 and PC2, and 3 order of magnitude between PC2 and PC3. Nonetheless, the scale of PC halves from $\rho = 10$ to $\rho = 1000$, indicating that the variance of the trajectory spreads between additional components. In general, this supports our previous conclusion about the lower reliability of the linear estimator D_{PCA}, which is highly sensitive to the relative scale of variance between the components.

Following these results, the next set of experiments will focus on whether the size of the network has any effect on the trend of dimensionality as ρ is varied. Further experiments were conducted with networks of 800, 1400 and 2000 neurons with the same sparsity and sparsity. For these experiments, the results of the non-linear estimator D_{kNN} are presented in Fig. 3.6.

From Fig. 3.6, it is clear that networks larger than 200 neurons also show very similar behaviour,

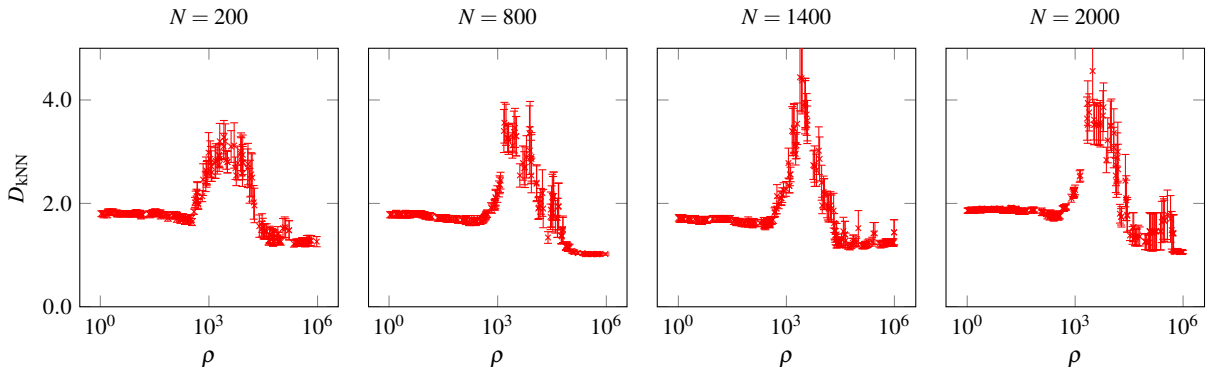


Figure 3.6: Estimated underlying dimensionality D_{kNN}, at the input timescale, for stable ($g = 0.9$) networks of different size and different ρ values. A different network with random connectivity was created for each data point. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the networks. Figure appeared in [NMS17b].

with a peak in underlying attractor dimensionality at intermediate values of ρ. Small differences can be observed between networks of different sizes, specifically during the transition from lower to higher attractor dimensionality. This distinction is more clearly observed for networks with $N = 800$ and $N = 2000$, which indicates some sensitivity to the choice of the variance threshold and the scaling of the exponential kernel in Eq. 3.5. Another minor difference is that the maximum average estimated dimensionality of larger networks is slightly higher for the two larger networks with $N = 1400$ and $N = 2000$, possibly arising from the larger number of interactions in the network, which result in more complex dynamics for individual neurons.

In summary, we can see that the input signal increases the dimensionality of dynamics at its timescale, in all stable networks considered with a peak around $\rho \approx 10^3$.

3.4.2 Dimensionality of dynamics in stable networks at the network timescale

Next, shift our focus to the dimensionality of dynamics at the timescale of the network. This will allow us to assess whether there is any difference between the dynamics at the two timescales and to identify whether the attractors in Fig. 3.5 are stable attracting sets or long transients.

Results in figure Fig. 3.7 show that if we look at the dynamics at the timescale of the network, the dimensionality of dynamics follows a different pattern than that estimated at the timescale of the input. The main difference is that the peak in dimensionality has shifted to lower ρ values, by about three orders of magnitude. At ρ values where the dynamics exhibited higher dimensional response at the timescale of the input, our estimators now predict responses of much lower dimensionality, and vice versa. The new range is now $10^{-1} \leq \rho \leq 10^{2}$, showing that at the network scale, the dynamics are most complex when the timescales of the input and the network are of similar magnitudes.

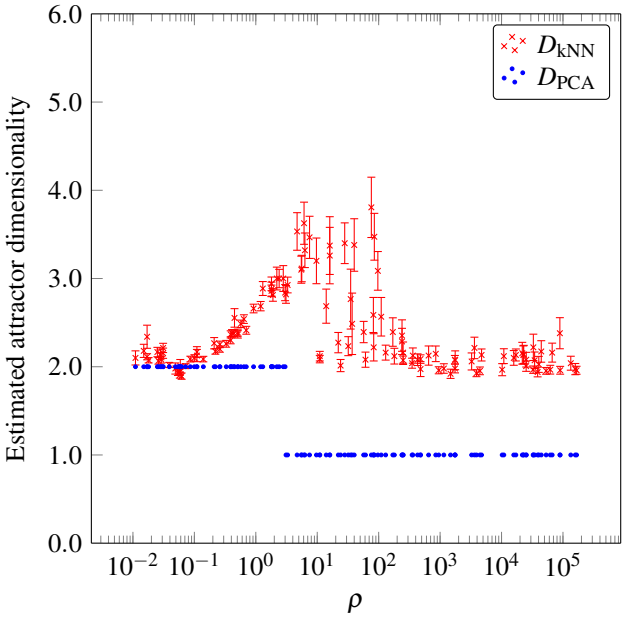
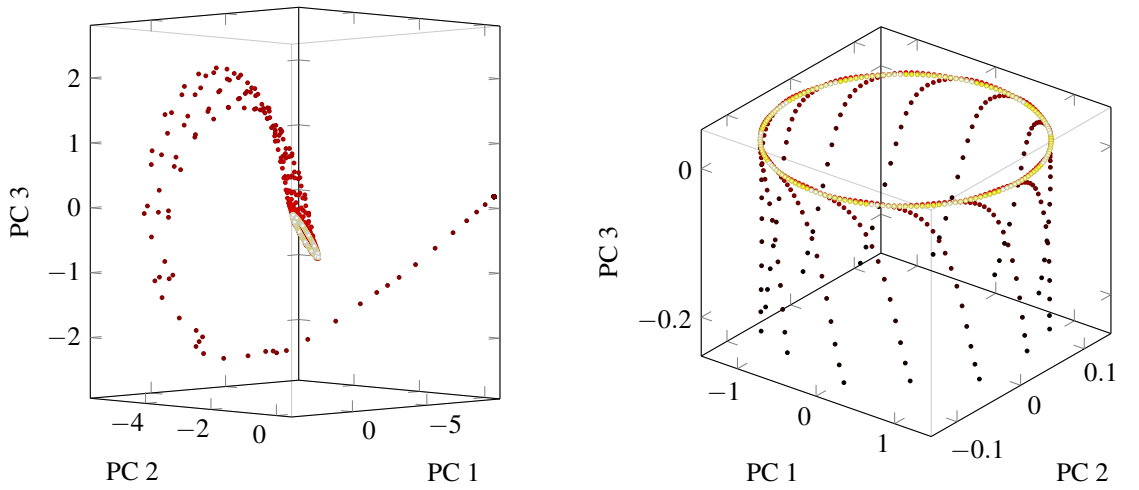


Figure 3.7: Underlying attractor dimensionality, at the network timescale, of stable networks with $N = 200$ neurons and $g = 0.9$ for different ρ values, with the two proposed estimators. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the same network.

When we are considering the network timescale, because only points every $\frac{\tau}{10} - th$ time step were recorded, it is equivalent to considering longer term dynamics of the network than when examining them at the input timescale, where points at every time step were recorded. If the attractors detected at the input timescale were stable and invariant, then the same attractors would have been observed at the timescale of the network. Hence, we would have observed similar trends in their dimensionality. Since the complexity of the response of the network at the longer timescale is different, the attractors observed at the shorter timescale must be transient attractors that evolve very slowly. One example is the attractor observed at $\rho \approx 10^{3}$ and shown

in Fig. 3.5b. Even though it looks stable at the input timescale, when the network activity is plotted at the network timescale is looks completely different, as can be seen in Fig. 3.8 and it's estimated dimensionality changes to $D_{\text{PCA}} = 1.00 \pm 0.00$ and $D_{\text{kNN}} = 2.10 \pm 0.75$.

From Fig. 3.8 we can see that what was identified as a stable attractor at the input timescale is a very slow transient, that approaches something that resembles a periodic attractor, shown in Fig. 3.8b. In these plots, we can see that the network is evolving towards an attracting state, but it takes progressively longer to get there. In Fig. 3.9 we can see the Euclidean distance between the final point at time step 3500 of the simulation and points previous recorded time steps. Points before recorded time step 2000 had a much higher Euclidean distance. This means that the network trajectory is slowly approaching a periodic attractor, illustrated by the yellow points in Fig. 3.8b. This does not mean that the final attractor lies on the yellow points exactly, but it is closer to them than other states the network visited earlier in the simulation.



(a) Time steps $1250 - 3500$ (b) Time steps $3000 - 3500$

Figure 3.8: Plots of the activity, at the network timescale, for a network of 200 neurons and $g = 0.9$ at $\rho \approx 10^3$, projected on its three principal components and one of the five runs. Estimated dimensionality was $D_{\text{PCA}} = 1.00 \pm 0.00$ and $D_{\text{kNN}} = 2.10 \pm 0.75$. (a) PCA on time steps $1250 - 3500$. (b) PCA on time steps $3000 - 3500$. This corresponds to recording the network activity at the network timescale, as opposed to the same ρ value and recording at the input timescale, shown in Fig. Fig. 3.5b.

We can further extend this analysis and for each ρ find the time step of the simulation that the network was within a certain Euclidean distance $d_{Eucl} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ from the final point. This is illustrated in Fig. 3.10a, whereas in Fig. 3.10b we can see the average number of time steps until it is within that distance again. We can see a decreasing trend in the period as

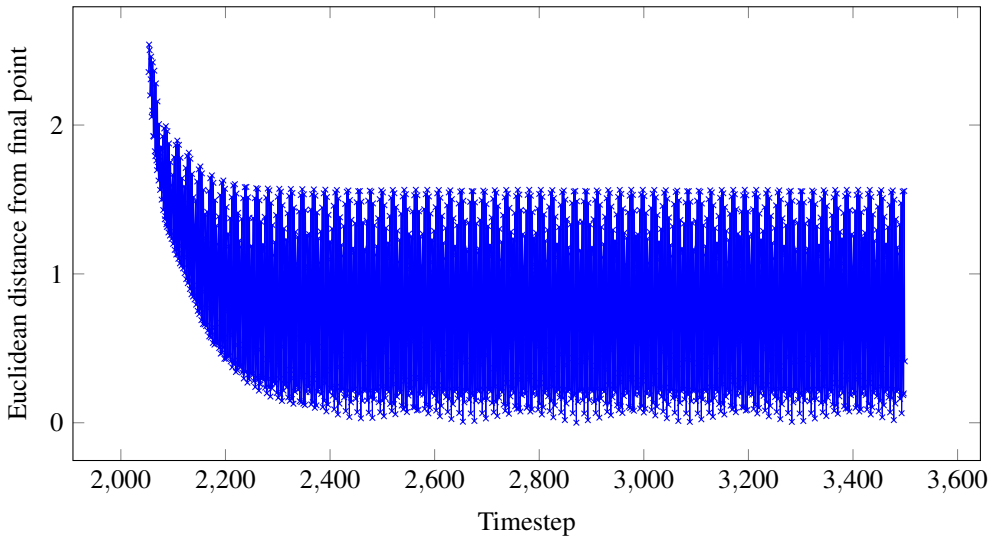
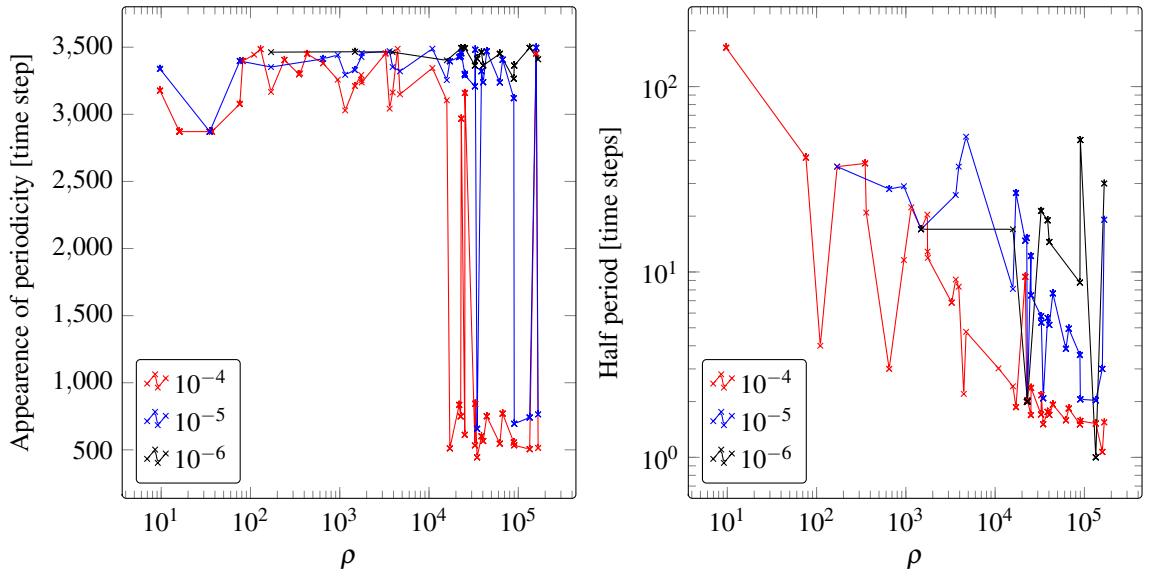


Figure 3.9: Euclidean distance between the final recorded point of the simulation, at the network timescale, and other recorded points. Simulations correspond to the network in Fig. 3.8. The points closer to the final point, are at a distance of about 10^{-4}, meaning that on average, each dimension is less than 10^{-5}.

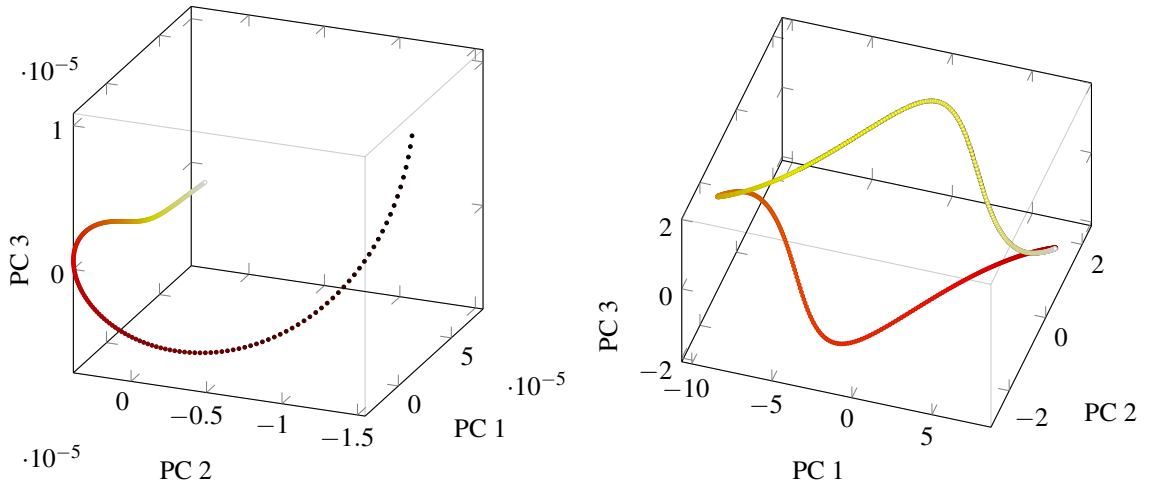
ρ increases, illustrating that the network is more often found at a distance d_{Eucl}. In addition, for very high ρ values, the network is within the distance d_{Eucl} from about the recorded time step 500 of the simulation onwards. Since we know that the network is evolving very slowly (as shown in Fig. 3.11a), we can conclude that for these very high ρ values, a very high frequency input significantly limits the region in the state space that the network visits, effectively slowing down its evolution.

The slowing down of the network dynamics at very large values of ρ can explain the decrease in the estimated dimensionality in stable networks. Fig. 3.12 shows the average Euclidean distance between network states at successive recorded time steps, with a clear peak the range of ρ values that the peak in estimated dimensionality appears in Fig. 3.7. This effect is observed both at the timescale of the input and the network. For the case of $\rho \leq 1$, the lower dimensionality comes from the very long periods of the input signal, about 100 longer than the timescale of the network. This very slow input entrenches the network state as it evolves (Fig. 3.11b), forcing it towards linearity as the frequency of the input increases further. Hence, the higher complexity in the response of the network can be attributed to the range of ρ values over which the input is neither too slow, compared to the network, such that it entrenches the network's state, nor too fast, such that it causes the network to evolve over a very confined subset of its state space.



(a) Time step to periodicity (b) Half period of periodicity

Figure 3.10: (a) Time step at which the network state is first found within a fixed distance $d_{Eucl} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ from the state at the final time step of the simulation. Only simulations where such a point could be found are shown on the plot. (b) The number of time steps, on average, until the network is found within this distance again. Only simulations where the network was within the distance more than once are shown.



(a) $\rho \approx 10^5$ (b) $\rho \approx 10^{-1}$

Figure 3.11: Plots of the activity, at the network timescale, for networks of 200 neurons and $g = 0.9$ projected on its three principal components and one of the five runs. (a) $\rho \approx 10^5$, $D_{\text{PCA}} = 1.00 \pm 0.00$ and $D_{\text{kNN}} = 2.10 \pm 0.55$. (b) $\rho \approx 10^{-1}$, $D_{\text{PCA}} = 2.00 \pm 0.00$ and $D_{\text{kNN}} = 2.00 \pm 0.20$.

At this point, we should also consider whether the different ways in which the simulations were carried out can account for the shift in the peak dimensionality between the input and network timescales. As mentioned in Section 3.3.1 earlier, when considering the timescale of the input (Fig.3.3), the base case of $\tau = 10$ corresponds to $\rho = 100$, rather than $\rho = 10$ when considering

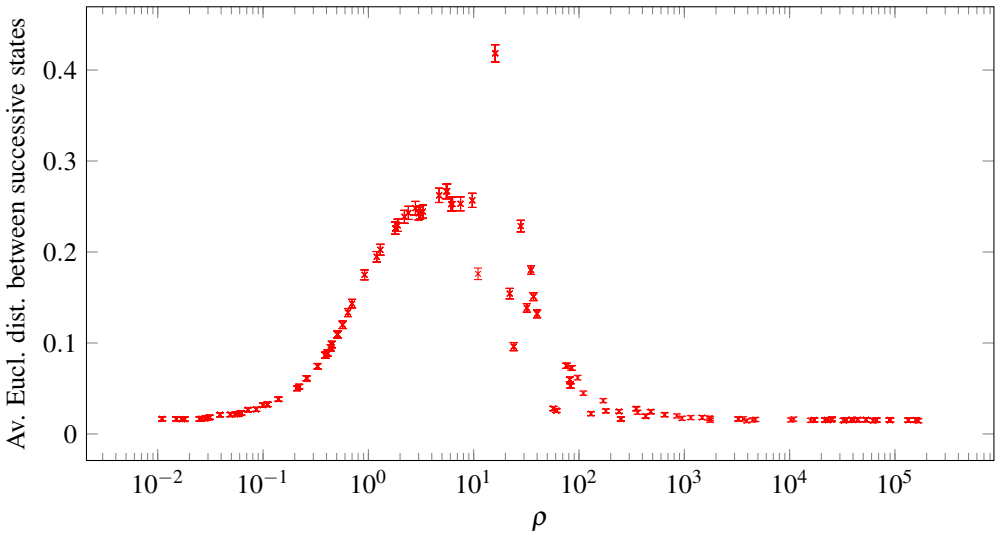


Figure 3.12: Euclidean distance between successive recorded states of the simulation, at the network timescale, for networks of 200 neurons and $g = 0.9$. Error bars denote standard error.

the timescale of the network (Fig.3.7). This can only account for a shift by about an order of magnitude, but since the observed shift was by three orders of magnitude, it means that the complexity of the response of the network to a periodic input is indeed different if we consider it at different timescales.

3.4.3 Dimensionality of dynamics in unstable networks at the input timescale

The results presented in the previous two Sections 3.4.1 & 3.4.2 correspond to stable networks, which exhibit intrinsically stable dynamics, due to the presence of sink attractors. Despite their intrinsic stability, these networks can nonetheless produce interesting behaviour if driven by a time-varying external signal. On the other hand, networks with a higher gain, $g > 1$ that we will be considering in this and the following section, are regarded as intrinsically unstable in the sense that their activity remains complex without converging to a stable state [SCS88]. Finally, for the case of externally driven networks with $g > 1$, previous work [RAS10] has shown that a strong oscillatory input can *entrench* the network dynamics and induce periodic, non-chaotic network activity.

In this section we will be considering networks with $g = 1.5$ and $N = 800$. Beyond investigating

their dimensionality with the two estimators at the input timescale, we will be considering visualisations of the trajectories of some of the networks to aid our intuitive understanding of what the attractors look like. As a benchmark, the activity of an autonomous network (no input signal applied) is shown in Fig. 3.13, illustrating that unstable networks have a highly irregular trajectories that require a large number of linearly independent dimensions to explain 95% of the variance (≈ 18). Moreover, the non-linear estimator estimates a dimensionality of about ≈ 3.5, which is as high as the highest reached in stable networks driven by a periodic input.

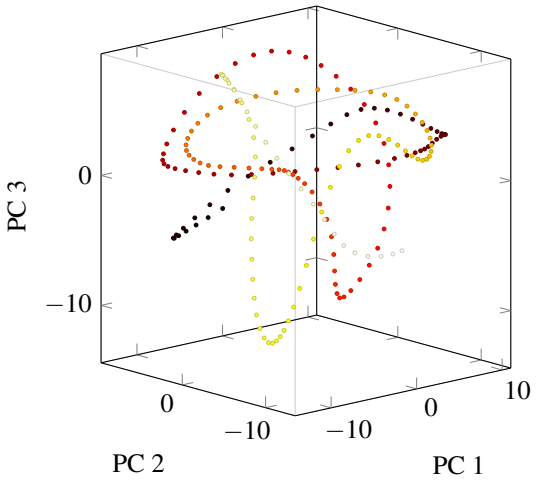


Figure 3.13: Plot of the activity for a network of 800 neurons, $\tau = 10$, and $g = 1.5$ without any input, projected on its three principal components. The estimated dimensionality of this activity was $D_{\text{PCA}} = 18.25 \pm 2.06$ and $D_{\text{kNN}} = 3.50 \pm 0.75$.

Figs. 3.14a and 3.15a show that for low values of $\rho < 60$, the network activity is no longer irregular and of dimension ≈ 3.5, as is the case for autonomous dynamics, but rather lays on a two-dimensional periodic attractor, caused by the slow sinusoidal signal with the same dimensionality. This result is in close agreement with theoretical results by Rajan *et al.* [RAS10], who showed that external stimuli can suppress chaotic activity in an otherwise chaotic network and impose a periodic behaviour on its dynamics.

As the value of ρ increases, the dimensionality of the underlying dynamics, as measured by the estimators, also increases. It is evident that the linear estimator significantly overestimates the dimensionality of dynamics compared to the non-linear estimator, similar to but stronger than the overestimation observed in stable networks, and the two estimates diverge significantly in the region of ρ values near the peak. This effect can be attributed to a rapidly developing and

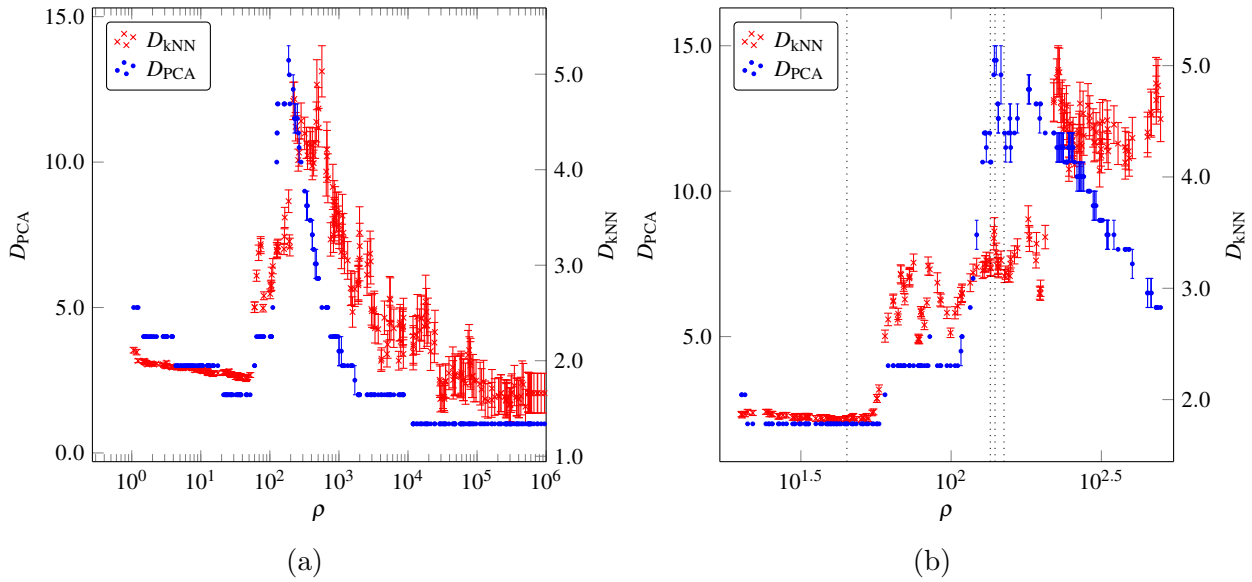
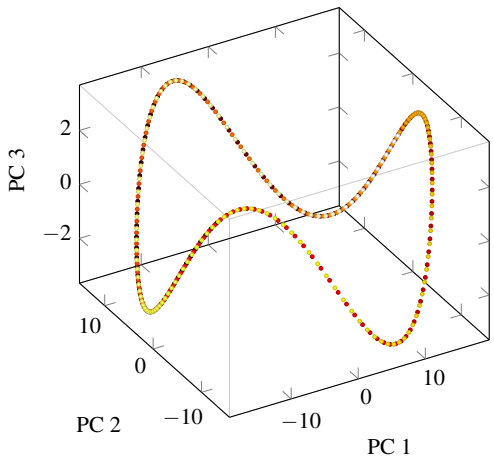


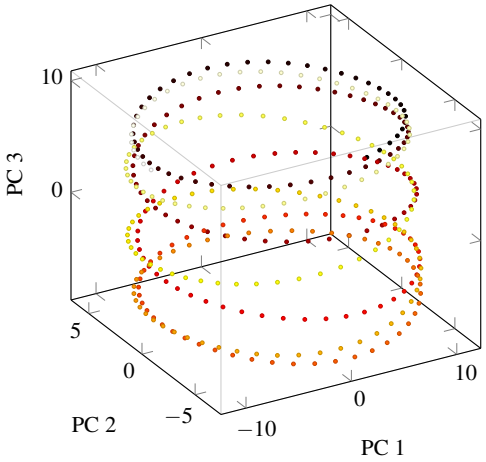
Figure 3.14: Dimensionality of dynamics, at the input timescale, estimated by D_{PCA} and D_{kNN}. Estimates are presented (a) across the full range of ρ values and (b) for ρ values close to the transitions presented in Fig. 3.15. The dotted lines in (b) correspond to ρ values of 45, 135, 140 and 150. Simulations were carried out using an unstable network of 800 neurons and $g = 1.5$. Figure appeared in [NMS17b].

highly non-linear attractor in this region. Although the numerical values of the dimensionality differ significantly between the two estimators, the general trend for the change in dimensionality is very similar for both. Moreover, the increase in dimensionality for unstable networks results in a maximum dimensionality of 5 at ρ values that were an order of magnitude lower than the ones for stable networks, also at the input timescale, which can reach a maximum dimension of 4.

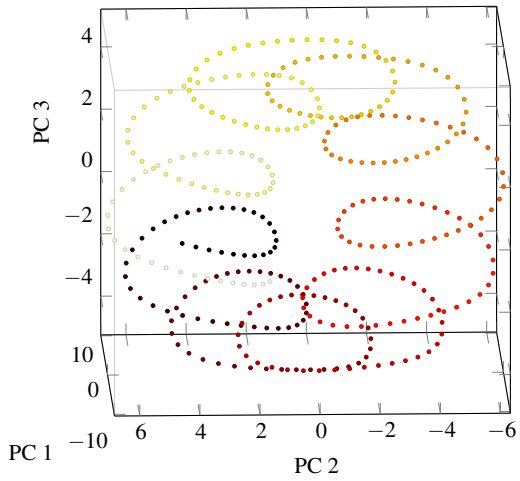
Figs. 3.15b, 3.15c & 3.15d capture the projections in reduced PC-space of transient attractors with a dimensionality of 3, as measured by the non-linear estimator. We refer to these as transient because, as we will see in Section 3.4.4, when considering the timescale of the network, these attractors no longer exist in this form. It is interesting to note that the increase in dimensionality from a value of approximately 2 to approximately 3 for $\rho = 45$ to $\rho = 135$ can be visually observed in Figs. 3.15a & 3.15b. In this transition, the network activity extends from a 2-d period transient attractor to oscillations around a 3-d manifold, whose geometric form resembles that of a cylinder. Furthermore, in the region of ρ values between 130 to 150, all transient attractors have an estimated dimensionality of ≈ 3, but exhibit different geometric forms. Three distinct forms were found in this region and are shown in Figs. 3.15b, 3.15c &



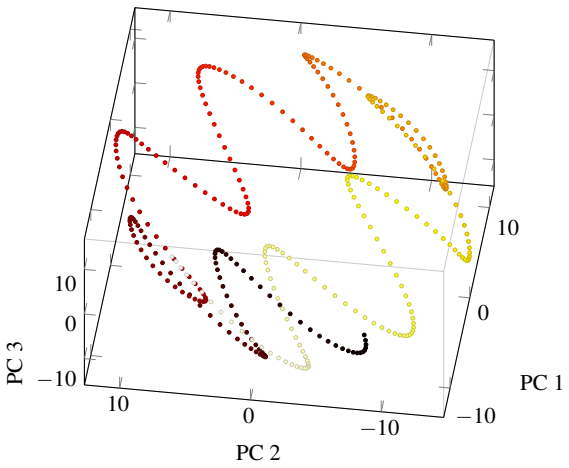
(a) $\rho = 45$, $D_{\mathrm{kNN}} = 1.90 \pm 0.01$



(b) $\rho = 135$, $D_{\mathrm{kNN}} = 3.2 \pm 0.08$



(c) $\rho = 140$, $D_{\mathrm{kNN}} = 3.4 \pm 0.16$

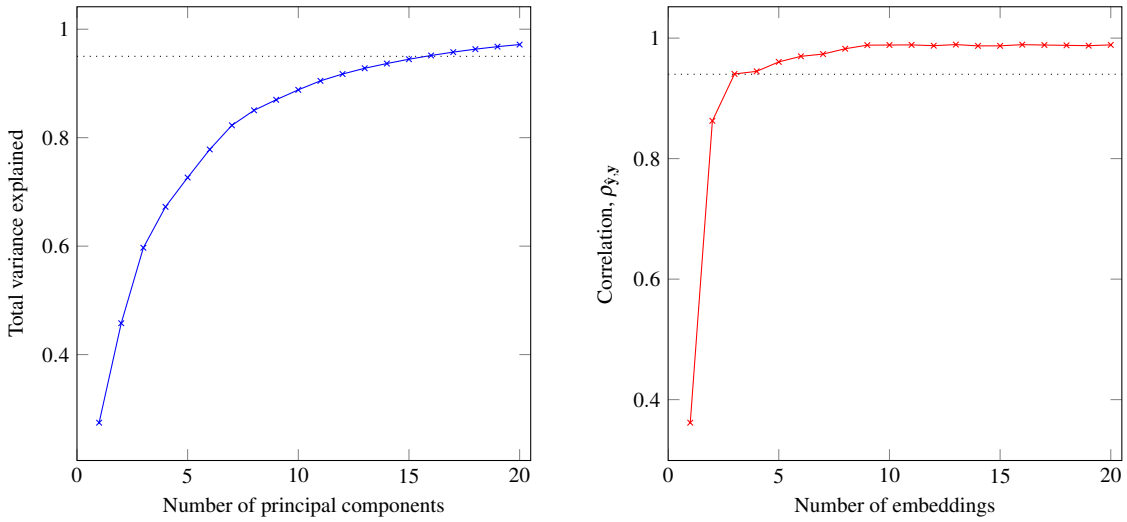


(d) $\rho = 150$, $D_{\mathrm{kNN}} = 3.2 \pm 0.08$

Figure 3.15: Plots of the activity for networks, at the input timescale, of 800 neurons and $g = 1.5$ driven by various ρ values, projected on their three principal components. The increase in transient attractor dimensionality that results from increasing ρ from 45 to 135 is associated with a transition from dynamics resembling (a) a figure-of-eight to (b) oscillations along the surface of a cylinder in the reduced space. Figs. (b), (c) and (d) show the various geometric forms of the system's transient attractors with $D_{\mathrm{kNN}} \approx 3$.Figure appeared in [NMS17b].

3.15d. Unlike trajectories of stable networks that simply follow similar periodic orbits with increased frequencies, unstable networks appear to exhibit much more complex behaviour in their transient attractors. That transient attractors of different geometric forms exist for values

of ρ that differ only by a small amount is evidence for the highly non-linear and flexible dynamics that can be exhibited by unstable networks.



(a) D_{PCA} (b) D_{kNN}

Figure 3.16: Typical plots of how the measures behave for a network of N=800 neurons, $g = 1.5$ and $\rho = 150$. For this network $D_{\text{PCA}} = 17.25 \pm 0.4$ and $D_{\text{kNN}} = 3.62 \pm 0.2$. (a) Cumulative total variance explained by principal components. The dotted line corresponds to 95% of the total variance explained. D_{PCA} was estimated as the minimum number of principal components to exceed this value. (b) Total correlation $\rho_{\hat{\mathbf{y}},\mathbf{y}}$ between k-nn prediction $\hat{\mathbf{y}}$ and true activation $\mathbf{y}$ as number of embedding dimensions increases.The dotted line corresponds to 95% of the total variance explained. D_{kNN} was estimated as the the minimum embeddings needed to exceeded this value.

It is instructive to note here that what can actually be visualised are not the complete high-dimensional transient attractors themselves, but their projections in the reduced, three-dimensional, PC-space. For this reason, care should be taken when forming interpretations from such visualisations and especially for ρ values that result in a dimensionality higher than 3. Nevertheless, all transient attractors presented in Fig. 3.15 had a maximum estimated dimensionality of three, as calculated with the more reliable non-parametric estimator and shown in Fig. 3.14. Moreover, even though the estimated dimensionality with the linear estimator was higher than 10 for Figs. 3.15b, 3.15c & 3.15d, the total variance explained by the first three principal components using the PCA estimator was about 75-80% for these plots and each subsequent dimension added a progressively smaller percentage to the total variance explained. This means that network oscillations along the plotted dimensions capture the majority of the variance of the system and any single additional dimension should not provide any additional

information about the geometric form of the transient attractor. Hence, we conclude that enough information was retained after dimensionality reduction with PCA, in the visual sense, so as for our visual exploration to remain relevant and informative of the effect of the driving signal frequency on the dimensionality and geometric form of these long transients. Finally, and for completeness, the typical behaviour of the two estimators is shown in Fig. 3.16.

3.4.4 Dimensionality of dynamics in unstable networks at the network timescale

In this final results section we will be looking at the dynamics of unstable networks at the timescale of the network. As we can see from Figs. 3.17, 3.18 & 3.19, similar to the case of stable networks, unstable networks also show a peak in the dimensionality of dynamics at much lower values of ρ when considering the network timescale rather than the input one. Furthermore, this pattern is consistent across different network sizes (Figs.3.17 – 3.19), with the linear estimator estimating a higher dimensionality of larger networks. The main differences between unstable and stable networks is that the peak is more sharp for unstable networks and that for high ρ values the linear estimator produces estimates that remain very high after the initial peak, with high variance.

To understand this behaviour, the Euclidean distance between states of the network at successive recorded time steps was estimated for the different simulations and shown in Fig. 3.20. As we can see, the pattern for the case of unstable networks is similar but not identical to that of stable networks. Even though the trend is very similar for $\rho \leq 1$, the behaviour for $\rho \geq 10$ follows a less strongly decreasing trend compared to stable networks, with high variance. A further interesting point, is that the range of ρ values where the variance of the average Euclidean distance is high ($\rho > 1$) coincides with the range of ρ values where the variance of dimensionality, using the linear estimator, is high. Even though we cannot specificy identify

Another interesting property of unstable networks compared to stable ones is that they do not exhibit a strong slow down of their dynamics for very high ρ, as the decreasing distance trend,

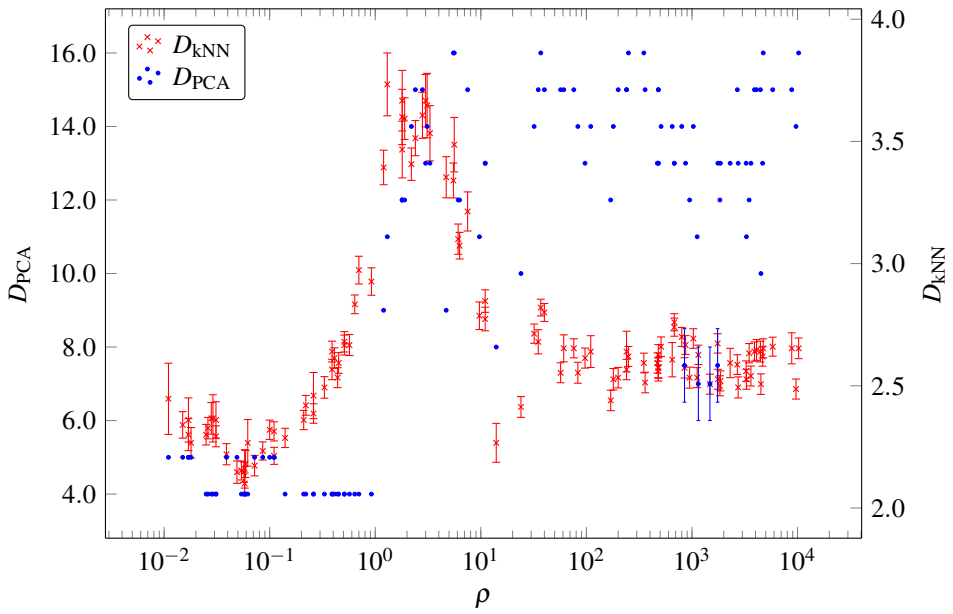


Figure 3.17: Dimensionality of dynamics, at the network timescale, of unstable networks with N=800 neurons and $g = 1.5$ for different ρ values, with the two proposed estimators. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the same network.

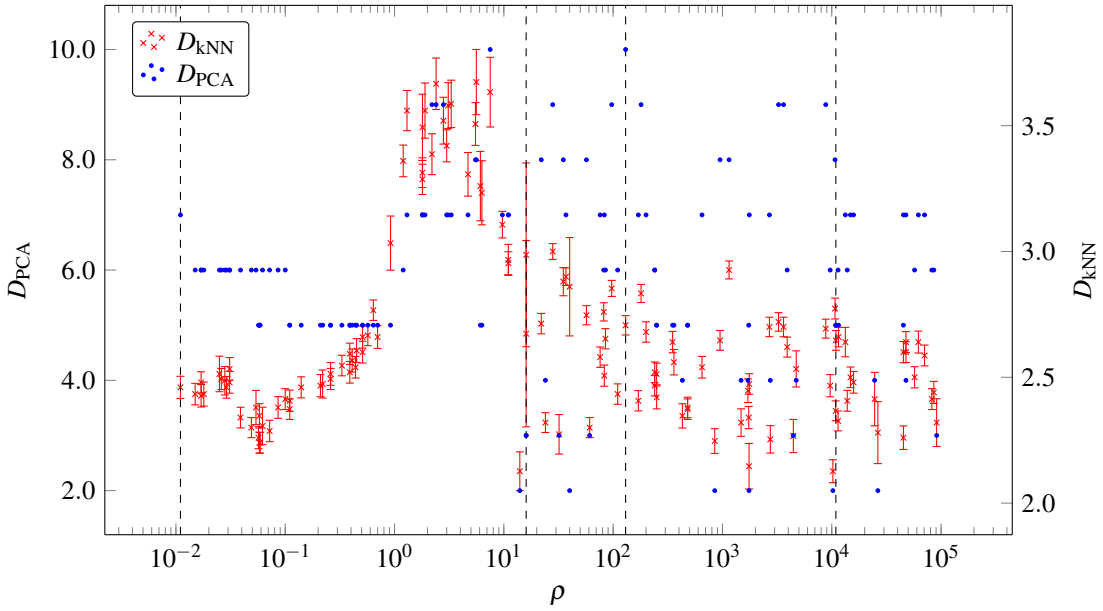


Figure 3.18: Dimensionality of dynamics, at the network timescale, of unstable networks with $N = 200$, $g = 1.5$ for different ρ values, with the two proposed estimators. The dashed lines correspond to ρ values of 0.11, 16, 130 and 1.09×10^4, the network activities of which are plotted in Fig. 3.21. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the same network.

possibly towards a stable attractor, is much smaller compared to stable ones. To further examine this, the projected activity of some of the networks is plotted in Fig.3.21. In the last Fig.3.21 of

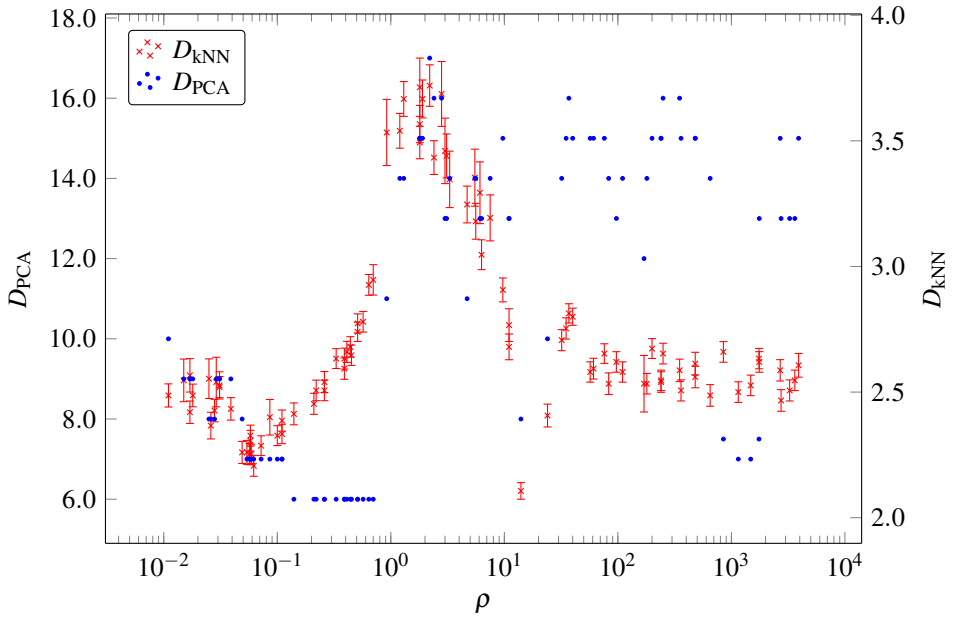


Figure 3.19: Dimensionality of dynamics, at the network timescale, of unstable networks with $N = 2000$, $g = 1.5$ for different ρ values, with the two proposed estimators. Error bars denote the standard error of the estimated dimensionality from different neurons and runs of the same network.

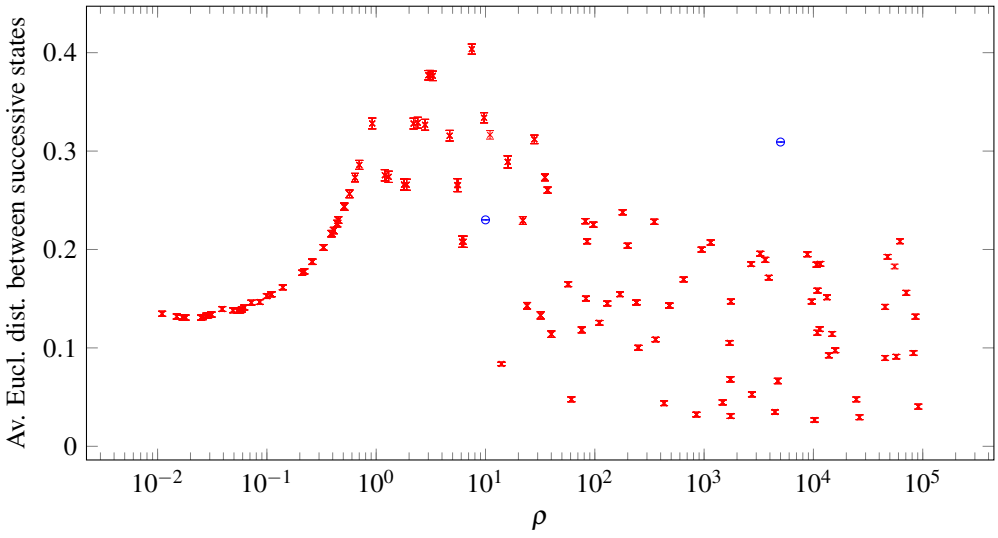


Figure 3.20: Euclidean distance between successive recorded states of the simulation, at the network timescale, for networks of 200 neurons and $g = 1.5$. The blue circles denote the case of networks with no input signal, simulated with $\tau = 10$ ($\rho = 10$) and $\tau = 5 \times 10^3$ ($\rho = 5 \times 10^3$). Error bars denote standard error.

this subsection, the activities of four unstable networks of 200 neurons, at the network timescale, for different ρ values, are shown. Unlike the case of stable networks, unstable networks exhibit a more complex response, even at the timescale of the network. The transition between, possibly transient, attractors is exhibited for networks driven at ρ values of 0.11 and 1.09×10^4, where

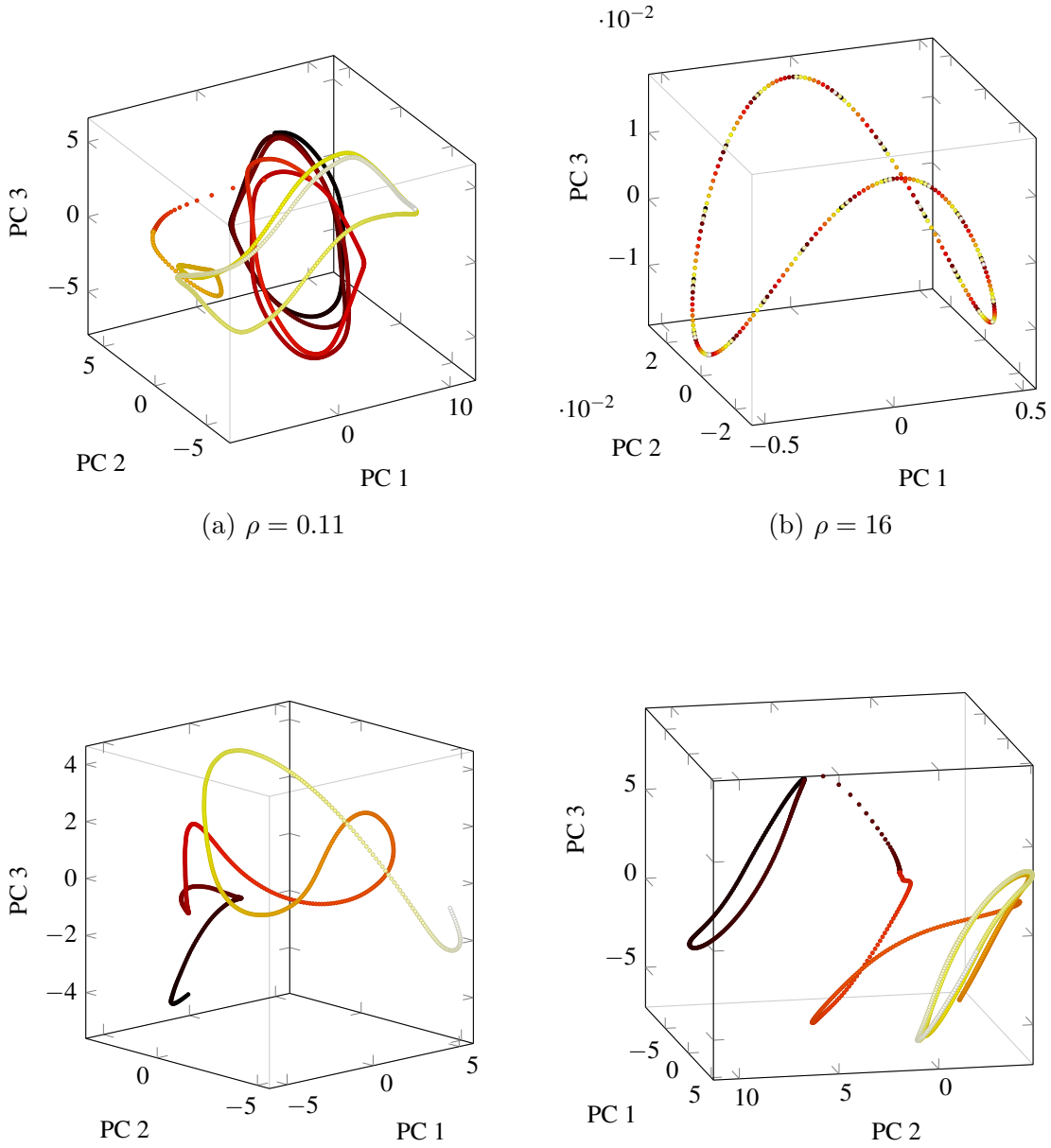


Figure 3.21: Plots of the activity for networks, at the network timescale, of 200 neurons and $g = 1.5$ driven by various ρ values, projected on their three principal components. (a)$\rho = 0.11$, $D_{\text{kNN}} = 2.41 \pm 0.04$ and $D_{\text{PCA}} = 7.00 \pm 0.00$. (b)$\rho = 16$, $D_{\text{kNN}} = 2.67 \pm 0.37$ and $D_{\text{PCA}} = 3.00 \pm 0.00$. (c)$\rho = 130$, $D_{\text{kNN}} = 2.71 \pm 0.04$ and $D_{\text{PCA}} = 10.00 \pm 0.00$. (d)$\rho = 1.09 \times 10^4$, $D_{\text{kNN}} = 2.65 \pm 0.04$ and $D_{\text{PCA}} = 5.00 \pm 0.00$.

the initial part of the trajectory is oscillating in a clearly different pattern in a distinct region of the state space than later on. Furthermore, when $\rho \approx 10$, the input and network timescales are comparable, illustrating that the attractor identified around this ρ value is likely to be a stable

one. What remains unclear is why the dynamics of the network lead to something that looks like an attractor for $\rho \approx 10$ and $\rho \approx 10^4$, but not $\rho \approx 10^2$. What we can safely say with these results though, is that the low dimensional visualisation in Figs. 3.21a & 3.21d, showing the presence simultaneously two periodic attractors, are consistent with the lower dimensionality estimates of $D_{\text{kNN}} \approx 2.5$. What we will aim to uncover in the next chapter is the mechanism through which phenomenon arises.

Finally, to verify that the decreasing trend in the average euclidean distance is indeed because of the input and not the higher τ values used and not an artefact of using a very large τ, the two blue circles in Fig.3.21 correspond to the case of autonomous networks, simulated with the same method. The fact that the average euclidean distance remains high even for large $\tau = 5 \times 10^3$, is evidence in favour of our initial assumption that we can increase ρ by increasing τ and recording states every $\frac{\tau}{\rho}$ time steps, without changing the dynamic behaviour of the network or introducing artefacts.

3.5 Discussion

In the experiments presented in this chapter, the value of ρ was varied to examine how stable and unstable networks respond to input signals that operate at timescales different than those of the network. Networks that respond with higher dimensional dynamics, when considered at the timescale of the input, are more responsive to changes in the input, and hence more useful for input-dependent computation, than networks that respond with lower dimensional dynamics and are hence not affected much by the changing input. From the presented results we can see that both types of networks, irrespective of their size, show a considerable increase in the dimensionality of their dynamics, at the input timescale, for intermediate values of $10^3 < \rho < 10^4$, meaning they respond in a richer and can theoretically have higher computational capacity. When considering the timescale of the network on the other hand, networks have more complex responds at much lower rations ρ; somewhere between two and three orders of magnitude lower. Hence, the relative timescales of the input and network, as well as that over which the dynamics

are considered need to be carefully chosen. This insight is relevant to the field of both machine learning and computational neuroscience.

3.5.1 Reservoir computing applications

Within the paradigm of reservoir computing, the recurrent network serves as a high-dimensional spatio-temporal kernel acting on an input time series [HS12]. Its function is to perform a complex non-linear convolution on the input signal such that, by training linear readout units, the output can be used to approximate the result of a desired computation on the input. This was shown to have applications, among other areas, in robotic manipulation [JM04], localisation [ASS08] and navigation [AS15]. In this latter context, the reservoir acts on an incoming stream of sensory information to either detect events or determine motor actions to complete a task, e.g. navigating between rooms. For such applications the dynamical properties of the reservoir play a crucial role: on the one hand, stability is desirable, in the sense that a similar sensory input should result in similar reservoir activity to guarantee a reliable operation; on the other hand, rich dynamics are also advantageous to ensure that the network activity is expressive enough for the readouts to be appropriately trained.

This investigation can provide insight into the dynamics of the reservoir driven by the appropriate input, such that the recurrent and external connections can be tuned, and to avoid time-consuming trial and error during training. Using the insights gain here, one can initialise the reservoir with a good estimate of g and τ, depending on the type and timescale of the input signal, as well as the task. One can then either use PCA or non-linear embedding to visualise the dynamics and measure their dimensionality from the resulting activity of the network. The effect of the two mentioned parameters on the dynamics can be quickly explored to identify different dynamical regimes and hence guide the parameter search during training.

3.5.2 Biological significance

Beyond the possible applications of these methods for reservoir computing, the work presented here can also be linked to features of neural dynamics to help advance both modelling and interpretation of brain behaviour [EPQD16, Bar17]. As an example, a recent study by Shenoy and colleagues showed that the motor cortex is strongly activated long before the onset of any muscle movement [KCRS14]. Furthermore, they showed that this pre-movement activity lies in the nullspace of the output connections projecting to the muscles. This means that even though the motor cortex is engaged in complex internal dynamics [CCK$^+$12] it has no downstream effect and hence does not initiate any motion. This behaviour allows the motor cortex to integrate information coming from upstream regions of the brain without causing any undesirable, premature movement [ASB$^+$11, Gra11].

The subspace where the network activity lies during preparation without affecting downstream regions could be considered a transient attractor of specific dimensionality determined by the input from upstream regions. Signals in a brain network such as the motor cortex can be thought of as having two important functions: to transfer information from other brain areas and to affect the dimensionality of the region's dynamics so that this information can be reliably processed and result in a useful motor output. Importantly, if the input signal can force the network into a low-dimensional subspace, then it means that this subspace can become a nullspace by only training the linear output weights to the muscles. This further means that the preparatory activity can be performed naturally, without the requirement for a separate gating or threshold mechanism, but rather through the presence of a reliable and reproducible attractor [Cis06, BSK10]. As this work shows for the case of CTRNNs, such attractor can naturally emerge in a network through the action of a driving periodic input signal that can operate over a range of frequencies. Even though the orders of magnitude considered for the frequency of the input signal used in this work were much larger than the range of $0 - 50$ Hz that researchers typically consider when studying the brain, the fact that the complexity of dynamics is strongly affected by the input signal frequency in CTRNNs, suggests that the brain might be using dynamics for computation at a much bigger extent than previously thought.

Interestingly, the non-linear attractor dimensionality estimator was used by Tajima *et al.* [TYFT15] to show that the complexity of brain dynamics is higher in downstream (cognitive) areas, compared to the upstream (sensory) areas from which they receive direct input. Further studies also showed that the attractor complexity in downstream areas of awake monkeys is significantly higher than that of anaesthetised monkeys – which lead to the conclusion that the conscious state and regions associated with higher cognitive processing are characterised by an increased attractor complexity [MBR$^+$11]. From the viewpoint of this work, we can also hypothesise that the increased dynamic complexity of these cognitively relevant areas could be partially explained by a change in the input signals coming from upstream sensory areas. Furthermore, changes in global power spectrum in the anaesthetised brain [MBR$^+$11] could be responsible for maintaining the necessary brain activity without inducing any actions, by restricting dynamics to a particular subspace of brain states.

3.5.3 Closing remarks

In this chapter we have looked at how the dimensionality of the dynamics of driven networks is affected by two parameters, specifically ρ and g, as well as the timescale over which the dynamics are considered. To do this we solely considered the activity of the simulated network to produce estimates of the dimensionality of the dynamics. In a sense, the analysis only consisted of measuring the *emergent phenomenon*, i.e. the resulting trajectory, from which we concluded *how* it is affected by the two parameters and the chosen timescale. In the next chapter the focus will shift to the *local interactions* around the stationary points of the system, in order to understand *why* this collective phenomenon emerges.

Chapter 4

Stationary points in continuous-time recurrent neural networks

4.1 Precis

In this chapter we will explore how the dynamical *skeleton* of the underlying vector field of continuous-time recurrent neural networks changes as the value of the input changes. Specifically, we will investigate the trajectory of stationary points and changes in the locally linear dynamics of these networks, and relate these to the phenomena reported in the previous chapter, in order to identify the origin of input-dependent dynamical transitions. What this work aims to to achieve is to build an intuitive understanding of the mechanics of how the relative timescales of the periodic input and the network *interact* with the underlying vector field to produce the rich behaviour exhibited by these networks[1].

[1]The basis and part of the work presented in this chapter was published in the paper "An Investigation of the Dynamical Transitions in Harmonically Driven Random Networks of Firing-Rate Neurons" [NMS17b], which was published with Open Access, under the Creative Commons License (http://creativecommons.org/licenses/by/4.0/).

4.2 Introduction

In the previous chapter we saw that the response of CTRNNs, when driven by external periodic stimuli with different ρ, exhibit a variety in their complexity that is also dependent on the timescale at which they are considered. These responses were examined by considering the network trajectory directly, rather than analysing the system of equations describing the systems. Hence, we can consider the results of the previous chapter as an investigation of a phenomenon, i.e. the global, resulting behaviour of the system captured by its trajectory, whereas in this chapter we will be considering the system of equations that determines the interactions between the neurons of the network. By differentiating this investigation into considering the system from two distinct perspectives, i.e. one focused on the trajectory of the system and the other on the local behaviour around stationary points, we are able to gain a more intuitive understanding of how the two perspectives are related, and can hence predict how the different parameters affect the observed behaviour. Furthermore, the highly non-linear behaviour of these networks and the large number of components make it hard to reason analytically about the presence and operation of the mechanisms that underlie them. The aim in this chapter is to show that by thinking of these networks from two different but related perspectives, we are able to gain more insight into the operations of mechanisms present in large scale non-linear systems.

As already mentioned, this chapter will consider the underlying dynamical skeleton of the system, defined as the stationary points and locally linear dynamics in the vector field. Our attempt is more challenging compared to the simpler systems described in Section 2.3, mainly because of two differences. The first one is that CTRNNs have a large number of neurons, making them very difficult to solve analytically, and the second one is that they are driven by an external signal, making them non-autonomous. In order to overcome the first challenge, a numerical approach will be used instead of an analytical one, according to Sussillo and Barak [SB13], and for the second one we will be considering a sequence of autonomous networks for each instantaneous value of the input signal. This will enable us to use simple tools from dynamical systems theory to relate how the instantaneous values of the input signal change the underlying vector field as the input changes. We will then attempt to interpret the changes in

the complexity of the trajectory as a result of the different timescales over which the vector field and the network state change.

4.3 Related work

The study of CTRNNs from the standpoint of dynamical systems theory has a long history. In the late 1980s Sompolinsky *et al.* [SCS88] used dynamical mean-field theory to predict that for the case of networks with infinitely many neurons, a sharp transition from a stable to a chaotic regime appears as the gain of the network increases from $g < 1$ to $g > 1$. They have also reported the appearance of limit cycles in numerical simulations for $1 < g < 1.5$ and networks with $100 - 1000$ neurons, indicating that for finite networks, the transition to a chaotic regime is much smoother, with a rich intermediate dynamic behaviour. In a related paper [RAS10], published two decades later, Rajan et at. showed that periodic input, delivered with an independent phase to each neuron, can suppress chaotic activity and enforce periodicity. In addition, they showed that for an intermediate range of input frequencies, probably analogous to the natural frequency of the network, chaotic activity is completely eliminated. In more recent work [SSA14], Stern, Sompolinsky and Abbott showed that CTRNNs with self connections can exhibit bistable dynamics. The relative strengths of self-connections and connections with other neurons can be used as tunable parameters to adjust the dynamic behaviour of such network; from stable to chaotic and bistable and transient chaos with many stationary points; the latter two were also reported in the results of the previous chapter in this work. Interestingly, they also used numerical simulations to estimate the lifetime of the long transient activity before reaching a stable point for networks of different sizes and found that the lifetime increases proportionally to the network size. Even though the networks they used were autonomous and cannot directly be compared to the periodically driven networks used in this work, the self-interaction term can be considered as a type of time-dependent additional input to each neuron which evolves in a complicated way and is tightly coupled to each neuron.

In another other paper, Laje and Buonomano [LB13] showed that by modifying a subset of the

recurrent connections of CTRNNs, it is possible to enforce locally stable trajectories as a response to a particular input pattern. These trajectories were shown to be robust to small perturbations (post-training), as shown by the decrease in the estimated largest Lyapunov exponent (LLE) around the stable trajectory, from about 7 to a positive value close to 0 (0.05). Other trajectories of the system maintained an estimated LLE of about 3, indicating diverging local dynamics and hence the presence of regions with chaotic dynamics in the network. Structurally, this was reflected (post-training) in an increased variance of the recurrent weight distribution and an increase in both the non-cyclic and cyclic clustering coefficients, reflecting global changes in the dynamics of these networks after modifying a subset of the recurrent connections. The exhibited dynamic behaviour, they claim, is a new form of phenomenon observed in complex dynamical systems, that differs from other previously discovered dynamic behaviours, the most relevant ones being *stable chaos* and *chaotic transients*. Stable chaos has been shown to manifest in randomly connected networks of firing rate neurons [ZBH09] and its main property is that a stable solution is reached after a very long, usually irregular transient with negative LLE, whereas chaotic transients are related to the presence of non-chaotic attractors that are reached via chaotic transients with a positive LLE [GOY83].

In a different line of research, bifurcations in time delayed CTRNNs have also been studied analytically for the case of networks with less than 10 neurons [Bee06, XL16, XZW16], but a similar analytical approach for larger networks comprising hundreds of neurons would be extremely challenging. For this reason, another approach based on finding stationary points in these networks using numerical approaches has emerged. This was proposed by Sussillo and Barak [SB13], who used numerical optimisation to find stationary and *slow* points in the state space of these networks. By linearising the dynamics around stationary points, they were able to reveal the underlying mechanism of how a trained network uses its dynamics to implement a 3-bit flip-flop task. In particular, they found that saddle points play a crucial role in transitioning between the different stable stationary points used to encode the various memory states. Furthermore, they explored how a network can learn to compute the moving average of the last two input values and found that the network creates a two-dimensional manifold of "slow" points, which they have termed an approximate plane attractor.

Using the same approach, Mante *et al.* [MSSN13] trained a CTRNN to reproduce the prefrontal activity of macaque monkeys during a selection and integration task with two noisy sensory inputs, followed by a stationary point analysis of the trained model. What they found was that the process of stimulus selection and evidence integration could be fully described through the dynamics of the trained network. In particular, they identified the presence of a line attractor along which evidence for the relevant stimulus is accumulated, while evidence for the irrelevant one is ignored. This is achieved by using two orthogonal sensory selection vectors. Evidence that is relevant to the task lies along the relevant selection vector and is accumulated along the stable line attractor, whereas irrelevant evidence lies along an orthogonal vector and results in no change of the network state along the line attractor. Furthermore, they concluded that the dynamics of these networks can accommodate the coexistence of two line attractors (one for each task), each with its two corresponding selection vectors. This led them to speculate that one of the functions of the prefrontal cortex might be to generate these types of separable representations.

Beer and Barak [BB18] used this method of stationary point analysis to elucidate the changes that take place in the dynamics of CTRNNs as a consequence of training with different learning algorithms. By assessing how close to a stationary point the states visited by a network are, before and after training, they were able to explain the better performance exhibited by FORCE learning in terms of the stability of the network dynamics during training. Furthermore, they showed that the network stability is very sensitive to the order with which the trials are presented during the training phase and related this to similar findings in animal training [Ski58] and machine learning [BLCW09]. Moreover, a simpler version of these networks, without a non-linear activation of the neurons, has been studied by Bondanelli and Ostojic [BO18], who were interested in understanding how the transient trajectories of neural responses to stimuli, as opposed to the resulting steady-state, can give rise to neural population coding. To this end, they identified the maximum eigenvalue of *the symmetric part* of the connectivity matrix – as opposed to the connectivity matrix itself – as the determining factor of whether input perturbations will first be amplified before dying out. However, it is much more difficult to perform a similar analysis for networks with a non-linear activation function, and hence analytic

conditions about important dynamical properties are still lacking.

In this chapter we will see how the stationary points and the locally linear dynamics around them can can be used to understand the mechanism underlying the rich behaviour exhibited by CTRNNs driven by periodic inputs reported in the previous chapter. Even though Rajan *et al.* [RAS10] tackled a closely related question, i.e. what is the effect of the frequency of an external stimulus on the dynamics of CTRNNs, (a) they used networks with an external stimulus that arrives out of phase at each neuron and (b) they used mean-field theory to predict whether the dynamics will be chaotic or not, without providing a deeper understanding on the mechanism underlying this. Moving forward from their study, we will combine the results presented in this chapter, along with those of the previous chapter, to gain a more intuitive understanding of the link between the changes in the vector field and the dynamics observed at the level of the trajectory.

4.4 Methods

4.4.1 Identifying stationary points

The focus of this chapter is the identification and analysis of the stationary points arising in continuous-time recurrent neural networks of the form of Eq. 3.1. As we have already seen in Section 2.3.4, a stationary point is a point where all components of the system's gradient are zero and can be classified as a source, sink, limit cycle or saddle point for low-dimensional systems, but its classification can get more complicated as the dimensionality of the state-space increases. Moreover, stationary points are key to predicting the long-term behaviour of the system and its response to perturbations. In this section we will build on the work of Sussillo and Barak on stationary point analysis in recurrent neural networks [SB13], to understand how the instantaneous value of the input changes the underlying vector field of the system.

For finding the stationary points of the driven reservoir, the system of equations describing the evolution of the network can be rewritten in matrix form for a constant stimulus, as

$$F(\mathbf{x}, s) \triangleq \dot{\mathbf{x}} = \frac{1}{\tau}\left(-\mathbf{x} + \mathbf{W}^{Res}\tanh(\mathbf{x}) + \mathbf{w}^{In}s\right) \ . \tag{4.1}$$

Since we cannot easily derive an analytical solution to the system, because of the non-linearities involved in the equations, we need to resort to numerical methods. Formulating the equations of the system as in Eq. 4.1, we see that finding the set of stationary points $\phi(s) := \{\mathbf{x^*}_1, \mathbf{x^*}_2, ..., \mathbf{x^*}_k\}$ of the system for an instantaneous input value s is equivalent to finding all solutions of the system of equations $F(\mathbf{x}, s) = \mathbf{0}$. In other words, finding unique state vectors $\mathbf{x}^*$ that minimise $|F(\mathbf{x}, s)|^2$, subject to different initial conditions. This leads to the definition

$$\mathbf{x}^* = \arg_{\mathbf{x}} \min |F(\mathbf{x}, s)|^2 \ . \tag{4.2}$$

For the case where there exists only one stationary point for a given value of s, we have $\phi(s) := \{\mathbf{x}^*\}$ and we will use the convention $\phi(s) \equiv \{\mathbf{x}^*\}$. The most common problem in finding stationary points with this method is the practical feasibility of this optimisation – in the standard reservoir computing scenario the number of neurons in the network ranges from several hundred to a few thousand, making the optimisation computationally expensive. The method proposed here, which is most effective for stable networks with $g < 1$, leverages the smoothness of the networks' dynamics to find all the stationary points for the full range of values s takes, by formulating the full optimisation as a set of sequential optimisation problems. The method followed for stable networks is following:

1. We know that for stable networks with no external input the stationary point is the origin. We can therefore start from the trivial solution $\phi(0) = \mathbf{0}$ by setting $s = 0$ and initial value of the optimisation also at $\mathbf{0}$.

2. Increment the value of s by small steps $\Delta = 0.01$, and minimise Eq. 4.2 again, using the result of the previous step $\phi(s)$ as the starting point for the current next optimisation $\phi(s + \Delta)$. The optimiser used was the MATLAB function `fsolve` with parameters as shown in Table 4.1 [MAT5a]. The stopping criterion was that every element of the gradient

vector had an absolute value below 10^{-15}.

3. Repeat this until the full positive range of $s(t)$ is considered. Then begin the second part by starting at $s = 0$ and covering the negative part of the range in a similar manner, by subtracting $\Delta = 0.01$ from s at every optimisation step.

Table 4.1: Parameters of `fsolve` used for finding stationary points.

Parameter Name	Value
`MaxFunEval`	10^6
`TolFun`	10^{-16}
`TolX`	10^{-16}
`DerivativeCheck`	`on`
`Jacobian`	`on`
`Algorithm`	`trust-region-reflective`

To ensure that the proposed optimisation scheme did not get trapped in local optima for the case of stable networks, additional optimisations were run with 50 random initialisations for each value of s, all of which converged to the single stationary point determined by the proposed scheme.

For the case of unstable networks, because of the larger number of stationary points present, the optimisation was only carried out using random initialisations that allowed for better exploration of the state space when searching for the stationary points.

In the experiments conducted, all the stationary points identified for each value of s were compared pair-wise and those that had a Euclidean distance of less than 10^{-9} were considered duplicates. Furthermore, it was found that incrementing s in steps of $\Delta = 0.01$ and 50 random initialisations offered a good balance between convergence probability, speed and identification of multiple unique stationary points. Finally, the trajectory of the stationary points is obtained after $\phi(s)$ is successfully computed for all relevant values of s (in this case, a grid in the $[-1, 1]$ interval).

In summary, this method allows us to capture"snapshots" of the location of stationary points in the vector field of the network, for each instantaneous value of the input signal. In other words, if the input remained constant at any of its possible values, the network trajectory would

have evolved over the corresponding vector field, since the only change from the autonomous case is the addition of a constant to each of the equations. Importantly, for the case of stable networks, because the vector field deforms smoothly for very small changes in the instantaneous value of the input signal (shown in the Results section later on), if the frequency of the signal is very low compared to the speed with which the network evolves (very low values of ρ), then the stationary point trajectory obtained with the proposed method will be a good approximation of the actual trajectory of the system. As the frequency of the input increases though (higher ρ), this approximation becomes less relevant, and the trajectory of the system is no longer guaranteed to closely follow the stationary point trajectory. This effect is clearly shown in Fig. 4.2, where simulations with lower values of ρ are shown to follow the stationary point trajectory more closely compared to simulations with higher ρ.

Locally linear dynamics — In order to characterise the local dynamics of the network near the stationary points, a similar approach described in Section 2.3.4, and also used by Sussillo and Barak [SB13] and Mante [MSSN13] in previous work, was taken to approximate the non-linear dynamics of the network shown on Eq. (4.1) as a linear system using their 1^{st} order Taylor expansion.

Thus, the dynamics around a very small neighbourhood $\delta\mathbf{x}$ around a stationary point $\mathbf{x}^*$ can be approximated as

$$\frac{d}{dt}(\mathbf{x}^* + \delta\mathbf{x}) = F(\mathbf{x}^* + \delta\mathbf{x}) = F(\mathbf{x}^*) + F'(\mathbf{x}^*)\delta\mathbf{x} + \frac{1}{2}F''(\mathbf{x}^*)\delta\mathbf{x}^2 + \frac{1}{6}F'''(\mathbf{x}^*)\delta\mathbf{x}^3 + \dots \tag{4.3}$$

Given that $\delta\mathbf{x}$ is sufficiently small, terms containing $\delta\mathbf{x}^2$ or above can be ignored. Further, the term $F(\mathbf{x}^*) = \mathbf{0}$ since $\mathbf{x}^*$ is a stationary point and hence we can obtain the locally linear form of the dynamics of the original system as

$$\frac{d}{dt}(\mathbf{x}^* + \delta\mathbf{x}) = F(\mathbf{x}^* + \delta\mathbf{x}) \approx F'(\mathbf{x}^*)\delta\mathbf{x}. \tag{4.4}$$

The matrix $F'(\mathbf{x}^*)$ is the Jacobian of the linearised system, $\mathbf{J}(\mathbf{x}^*)$, where each element $J_{ij}(\mathbf{x}^*)$, after dropping the dependency of τ, is the partial derivative

$$J_{ij}(\mathbf{x}^*) = F'_{ij}(\mathbf{x}^*) = \frac{\partial F_i}{\partial x_j}\Big|_{\mathbf{x}^*} = \frac{\partial}{\partial x_j}\Big|_{\mathbf{x}^*} \left(-x_i + \sum_{j=1}^{N} W_{ij}^{Res} \tanh(x_j) + w_i^{In} s \right), \tag{4.5}$$

where $\mathbf{x}^* \in \phi(s)$ and results in

$$J_{ij}(\mathbf{x}^*) = -\delta_{ij} + W_{ij}^{Res} \left(1 - \tanh(x_j^*)^2\right), \tag{4.6a}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases} \tag{4.6b}$$

To aid computations, the Jacobian of the linearised system can then be expressed in the compact matrix notation

$$\mathbf{J}(\mathbf{x}^*) = \mathbf{W}^{Res} \odot \mathbf{B} - \mathbf{I}\ , \tag{4.7}$$

where $\odot$ denotes an element wise multiplication between the two matrices, $\mathbf{B}$ is an $N \times N$ matrix with identical rows $B_{ij} = (1 - \tanh(x_j)^2)$ and $\mathbf{I}$ is the identity matrix.

The Jacobian can then be calculated at each stationary point, $\mathbf{x}^* \in \phi(s)$, followed by diagonalising to extract the eigenvectors corresponding to the eigenvalues of the linearised system. This was done using the `eig` function of MATLAB. In order to determine the nature of the stationary point all the eigenvalues of the linearised system were taken into account and using results from dynamical systems theory described in Section 2.3 the nature of the fixed point was determined. Points with all eigenvalues having strictly non-positive real parts were considered stable. If any of the eigenvalues of a stationary point had a positive real part, the point was classified as a saddle.

The challenge for visualisation comes if the eigenvectors have a non-zero imaginary part, as

is often the case. Dynamically, this indicates the presence of rotational dynamics near the stationary points. To visualise this effect, the diagonal form of the Jacobian at each stationary point can be transformed into its real Jordan canonical form, where pairs of complex conjugate eigenvectors are transformed into a pair of real vectors that define the basis of the hyperplane that the rotation lies on. This pair of real eigenvectors directly determines the orientation of the rotation plane of the locally linear dynamics.

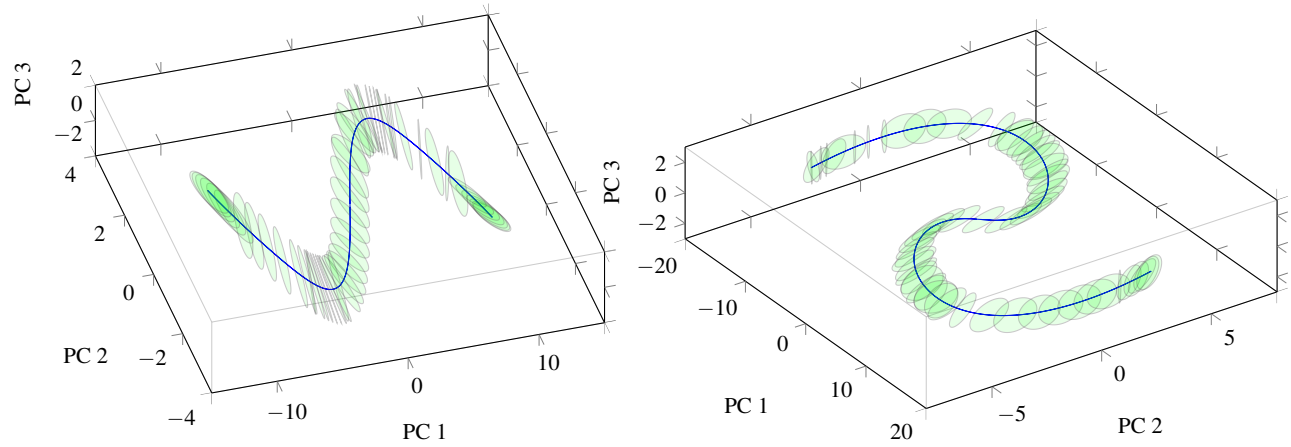
4.5 Results and discussion

4.5.1 The trajectory of stationary points in stable networks

We will begin by examining the trajectory of stationary points, using the technique described in Section 4.4, for two networks of sizes 200 and 800 neurons, and $g = 0.9$. By performing PCA on the stationary point trajectories and plotting their projections in the three principal components we obtain Fig. 4.1.

The first result we observe is that for all stationary points and both networks, the eigenvalues with the two largest absolute value were complex conjugate pairs with negative real parts. This indicates that the local dynamics exhibit an attracting rotational effect towards the stationary points – i.e. if the input is constant, the network state spirals towards a stable point along the green circles in Fig. 4.1. This is consistent with numerical simulations of CTRNNs, where no matter the instantaneous value of the input, the network will always condense to a single fixed value if $g < 1$.

A second point of interest in Fig. 4.1 is the regularity with which the stationary point shifts as the input changes, resulting in a smooth trajectory in the reduced PC-space. This validates the rationale behind the sequential optimisation approach which allowed the optimisation to be completed much faster than by using a random initial point. A further point of interest is that the planes spanned by the two dominant eigenvectors also change smoothly along the stationary point trajectory. The smoothness with which the plane changes orientation is more obvious for



(a) 200 neurons, $g = 0.9$ (b) 800 neurons, $g = 0.9$

Figure 4.1: Trajectories of the stationary points as the input varies shown are in blue, projected onto their three PCs. The origin represents the stationary point for the case of no input, while the two extrema of the curves represent the extrema of the input values (-1 and 1). Attracting planes spanned by the pair of complex conjugate eigenvectors corresponding to the pair of eigenvalues with the highest absolute value at each stationary point are shown in light green. Figure appeared in [NMS17b].

the case of the smaller network than the larger one, because of the larger number of dimensions projected away in the case of the latter.

From this first set of results we can conclude that even in the presence of a varying input, stable networks remain in a stable dynamical regime that is characterised by the presence of dominant stable spirals. A time-varying input (in the range $[-1, 1]$) results in a smooth change in the position of the stable point, along with a smooth deformation of the vector field that preserves the locally linear dynamics around this stable point. Hence, we can interpret the behaviour of a stable network as follows: for each value of the input signal s, the network follows a vector field $F(\mathbf{x}, s)$ which, if given enough time, will bring the network to $\phi(s)$. However, since the input value changes over time, the network is not guaranteed to reach this stable fixed point, and is instead under the influence of a different vector field at every time step. Linking this back to the results of the previous chapter, the parameter ρ, being the ratio between the timescales of the input and the network, regulates how fast the stationary point moves along its trajectory compared to how fast the network evolves in its state-space.

We can visualise the interplay between a changing vector field and the network state by plotting the activity of the same network driven by signals of different ρ values, projected and overlaid in

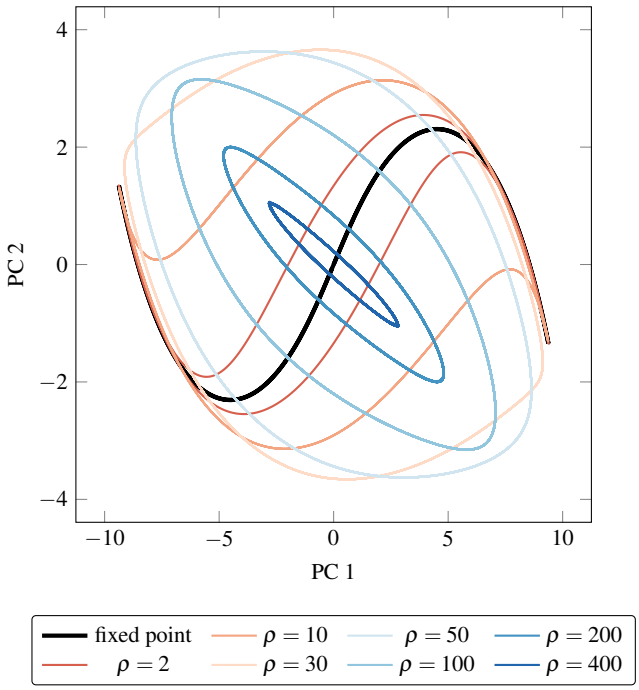


Figure 4.2: Set of stationary points (centre black curve) and trajectories of an $N = 200$ stable network for different ρ values at the input timescale. All trajectories have been projected onto the two largest principal components of the set of points visited by the stationary point. Projected network trajectories for lower ρ values lie closer to the projected stationary points, indicating that the fixed points move at a *speed* that is comparable to that of the network's evolution and can hence *pull* the network activity close to them. As ρ increases to about 30, the projected network activity initially *opens up* and then, for $\rho \gtrsim 50$, falls into an oscillation mode that is orthogonal to the oscillation mode of the stationary points in reduced PC space, while condensing around the origin. This illustrates that as τ becomes much larger than τ_S, the network activity progressively moves into a subspace that is orthogonal to the two principal components of the stationary point trajectory, indicating a *disentanglement* of the two trajectories. Figure appeared in [NMS17b].

the PC-space of the set of stationary points (Fig. 4.2). For each trajectory we can also calculate the radius R of the smallest enclosing sphere and its maximum instantaneous distance d_ϕ from the stationary point in the reduced three-dimensional PC-space (Fig. 4.3).

These results match our intuitions – for lower values of ρ, the stationary point moves slowly along its trajectory and at a speed through the state-space which is comparable to the speed at which the network activity evolves. Due to the fact that all stationary points remain sinks, the state of the network is being constantly pulled by the moving stationary point and remains close to it, yielding a small d_ϕ. As ρ increases, the stationary point moves faster and can no longer be followed by the network state, such that the trajectories of the network become wider and progressively less similar to that of the stationary point, as reflected by the rapidly

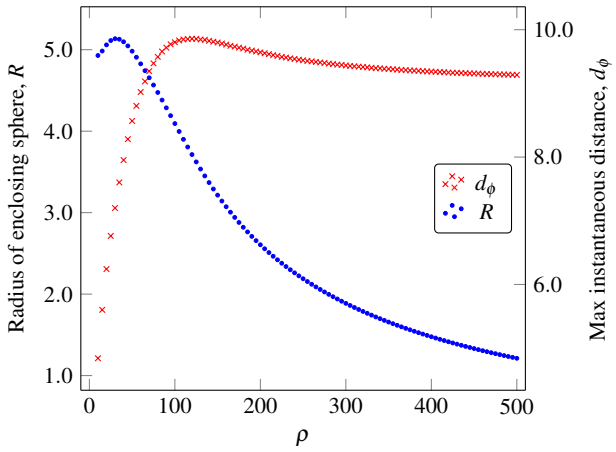


Figure 4.3: Stable network with 200 neurons and $g = 0.9$. In blue, R is the radius of the smallest enclosing sphere of the network trajectory at the timescale of the input, projected onto the reduced 3-dimensional PC-space of the stationary point trajectory. In red, d_ϕ is the maximum instantaneous distance in same space between the network state and the stationary point trajectory. Figure appeared in [NMS17b].

increasing d_ϕ. Then, as ρ gets much larger, the vector field changes much faster than the network can accommodate and trajectories start condensing around the origin in the reduced PC-space. Eventually R vanishes into a subspace orthogonal to the reduced PC-space and d_ϕ asymptotically approaches the maximum distance of the set of stationary points to the origin. This behaviour is exhibited because the network activity is no longer able to follow the moving stationary point closely, due to the fact that the stationary point and network state move with very different speeds[2].

Linking Figs. 4.2, 4.3 & 3.3, we observe that the complexity of the response appears to start increasing at around the same value of $\rho \approx 500$ that the radius of the enclosing sphere in the trajectory PC-space shrinks to about 1. For ρ values lower than that, the network state appears to be driven by the moving stationary point and follow a periodic trajectory surrounding the trajectory of stationary points, which is consistent with the estimated dimensionality of 2. As the state of the network can no longer keep up with the moving stationary point, is starts oscillating in an orthogonal sub-space, which means that it is no longer as closely coupled to the

[2]This behaviour is loosely reminiscent of the resonance behaviour observed in conventional driven periodic oscillators. If the oscillation of the driving force is slow compared to the intrinsic timescale of the oscillator, the system follows the trajectory imposed by the force. With higher frequencies of the driving force the amplitude of the resulting oscillation increases, as resonance effects start playing a role. Finally, for even faster frequencies, the force changes too fast for the system to assimilate, and the amplitude of the oscillation shrinks around the origin. As already mentioned, the input is not strictly speaking a force, as it is applied on the first time derivative rather than the second.

2-dimensional stationary point trajectory and can hence exhibit more independent and complex behaviour, as reflected by the increase in the estimated dimensionality. An example of this behaviour can be seen in Figs. 3.5b & 3.8, showing geometrically richer trajectories for networks driven with $\rho = 1000$, compared to trajectories driven with lower ρ values, at the timescale of the input and network respectively.

The last result from the previous chapter we will try to understand is the trend in the average distance moved by the network at its own timescale as ρ varies, shown in Fig. 3.12. Building on the intuition gained in this chapter, we understand that for very low values of ρ the network moves much faster than the stationary point. This means that it has enough time to reach the slow-moving stationary point, which is the limiting factor that determines the average distance traversed by the network per recorded time step. As ρ increases, the speed of the input increases and the limiting factor is no longer the speed of the stationary point, which causes the average speed to reach a maximum. As ρ increases, the network state moves out of its PC-space and does not "experience" a strong "pull" of the stationary point directly. Rather, it "experiences" a weaker, indirect "pull" through the lower principal components of the stationary point trajectory. This forces the network to oscillate in a much smaller region of its state space that does not overlap with the sub-space where the stationary point oscillates, and results in less displacement between successive recorded network states. This understanding is consistent with the results shown in Figs. 3.8 & 3.10, which show the state of stable networks progressively shrinking to a small part of their state space as ρ increases.

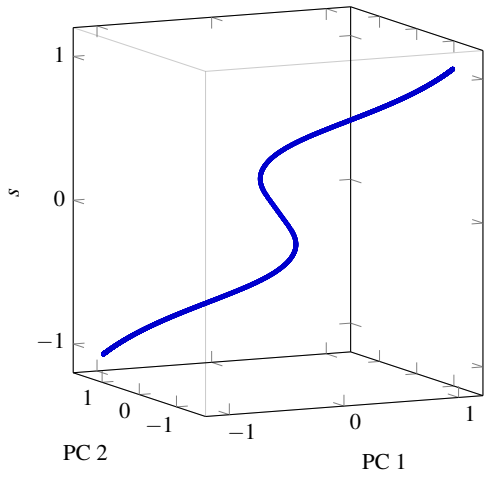
Having gained a more intuitive understanding of the effect of the relative speeds of the network and the stable stationary points on the resulting dynamics of the network, for the case of stable networks with $g < 1$, our focus will now shift to unstable networks with $g \geq 1$.The aim is to use our new intuitions to identify the different properties that unstable networks have and link these to their resulting dynamic behaviour.

4.5.2 Beyond stable networks

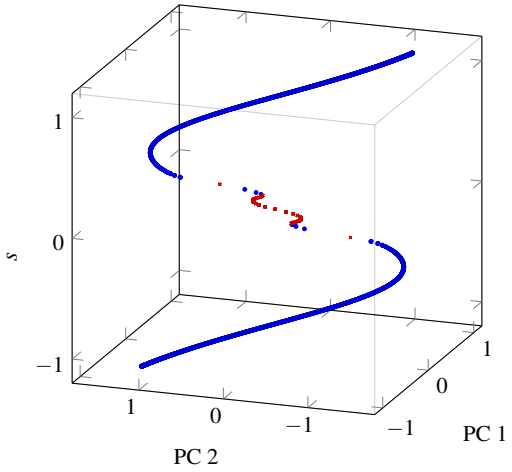
Following a similar approach for identifying the position and type of stationary points, we can examine how the gain of the network g affects the dynamical skeleton of CTRNNs. Using the optimisation scheme described in Section 4.4, for each instantaneous value of s in $[-1, 1]$ 50 random initialisations were performed, in order to find as many stationary points as possible that satisfy our criteria. Again, the eigenvalues of the linearised system at each stationary point were examined to classify it. The plots in Fig. 4.4 show how the position and type of the stationary points change with g and s for networks of 200 neurons. For each subplot in Fig. 4.4, the number of complex, positive-real-part and strictly real eigenvalues obtained for each identified stationary point at the different values of s are shown in Figs. 4.5, 4.6 & 4.7 respectively. No stationary point was found to have repeated or purely imaginary eigenvalues.

The plots in Fig. 4.4 convey the interesting point that both the network gain g and the instantaneous value of the input s can be considered to be bifurcation parameters, in the sense that the position and nature of the stationary points depend on their precise values. Even though for networks with $g < 1$ we know that there exist only stable points, Fig. 4.4 shows that the value of s plays an equally important role. This can be seen by the fact that even networks with $g \geq 1$ appear to have stable stationary points, if the absolute value of s is high. More specifically, Fig. 4.4a for $g = 1$ shows that all stationary points remain stable points for all values of s, similar to the case of networks with $g < 1$. This can also be seen by the absence of eigenvalues with a positive real part for any value of s, shown by Fig. 4.6a. The majority of eigenvalues of all stationary points, for any value of s, are pairs of complex conjugate eigenvalues, all with negative real parts (Fig. 4.5a), while the remaining are negative and real (Fig. 4.7a). These results are consistent with those of Sompolinsky *et al.* [SCS88] who also observed that finite networks show a smooth transition to chaos.

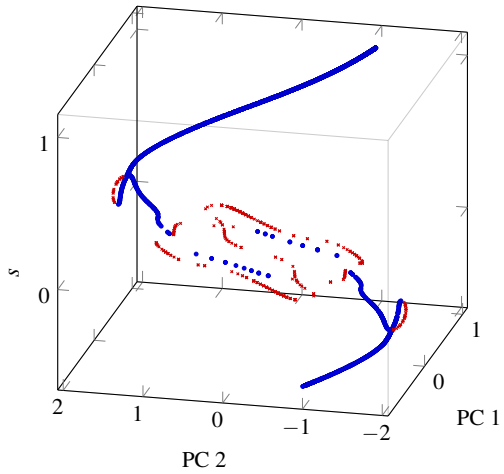
As g increases by a small amount, the range of s values in the vicinity of the origin generate multiple saddle points that prevent the network from reaching a stable state if run with very small or no input. This is in agreement with the well-known fact that autonomous networks are unstable for $g > 1$ and also reflected by the appearance of eigenvalues with positive real parts



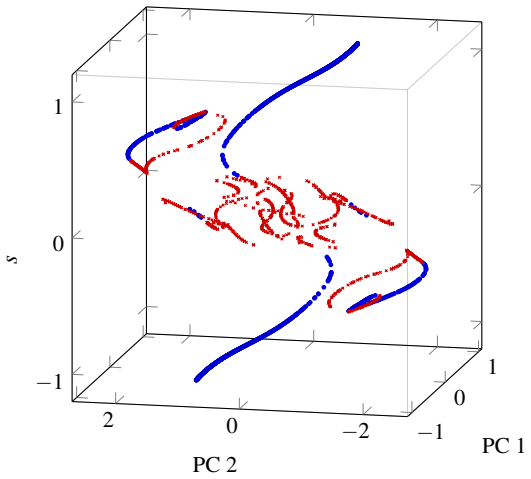
(a) $g = 1$, 315 unique



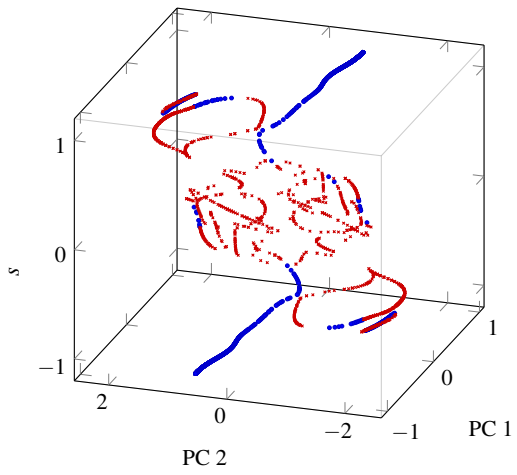
(b) $g = 1.2$, 345 unique



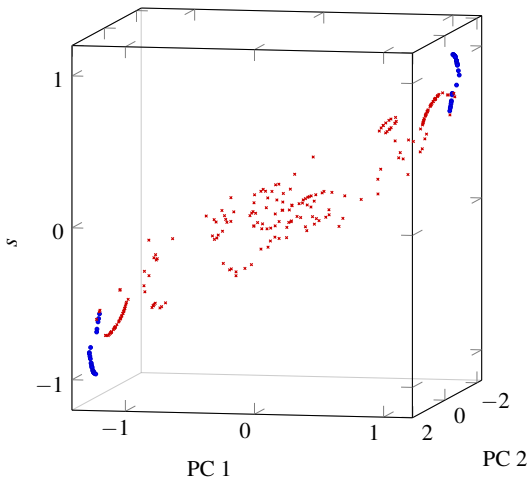
(c) $g = 1.3$, 423 unique



(d) $g = 1.5$, 597 unique

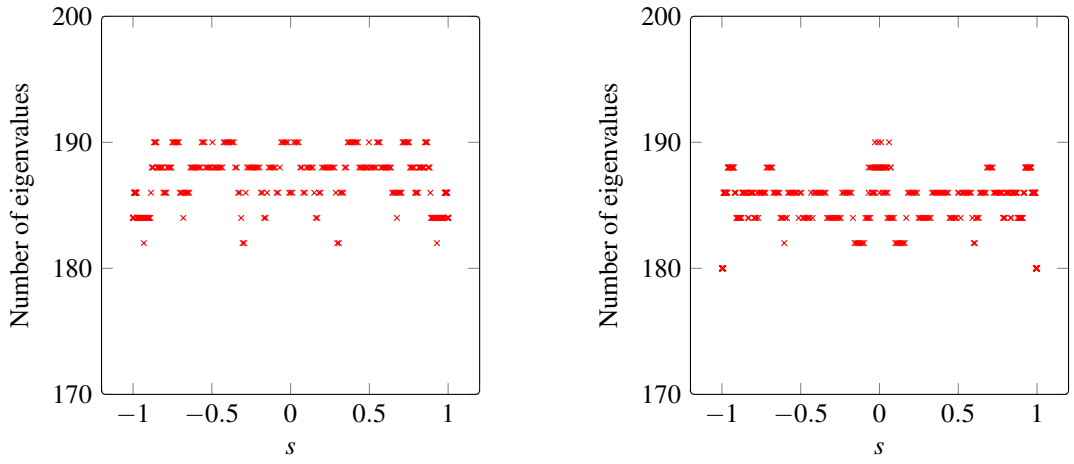


(e) $g = 1.6$, 654 unique



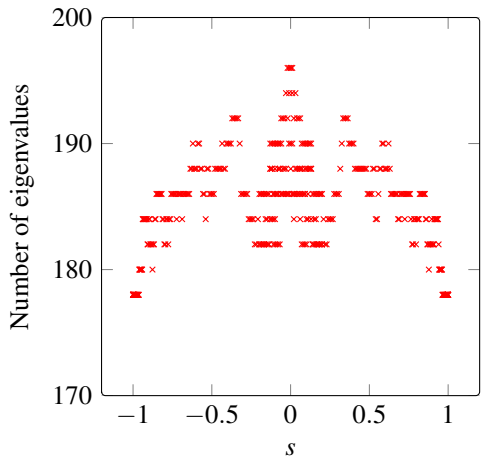
(f) $g = 2$, 126 unique

Figure 4.4: Stationary point visualisations for different values of g in networks of 200 neurons. Stationary points were discovered with 50 random initialisations of the optimisation procedure (similar to Sussillo and Barak [SB13]) for each value of the input signal s. Linearisation of the dynamics, each stationary point was classified as a sink (blue dots) or a saddle (red crosses), depending on the principal eigenvalues. The number of unique points identified for each sub-figure is denoted in its legend. Figure appeared in [NMS17b].

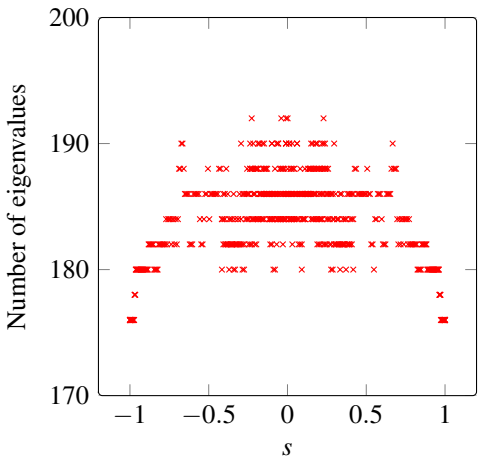


(a) $g = 1$

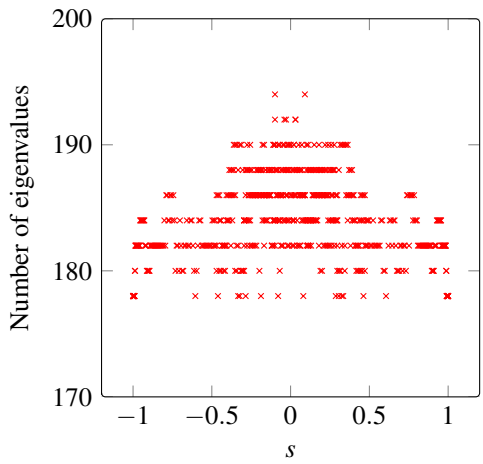
(b) $g = 1.2$



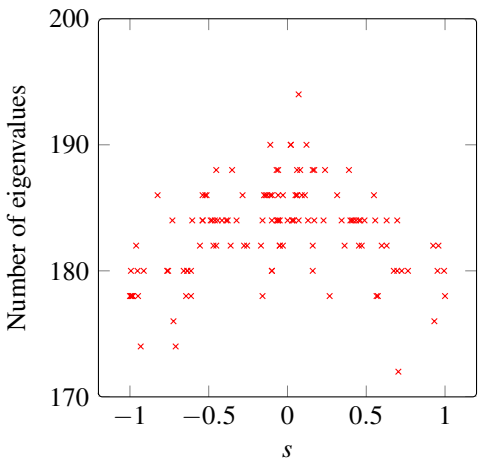
(c) $g = 1.3$



(d) $g = 1.5$

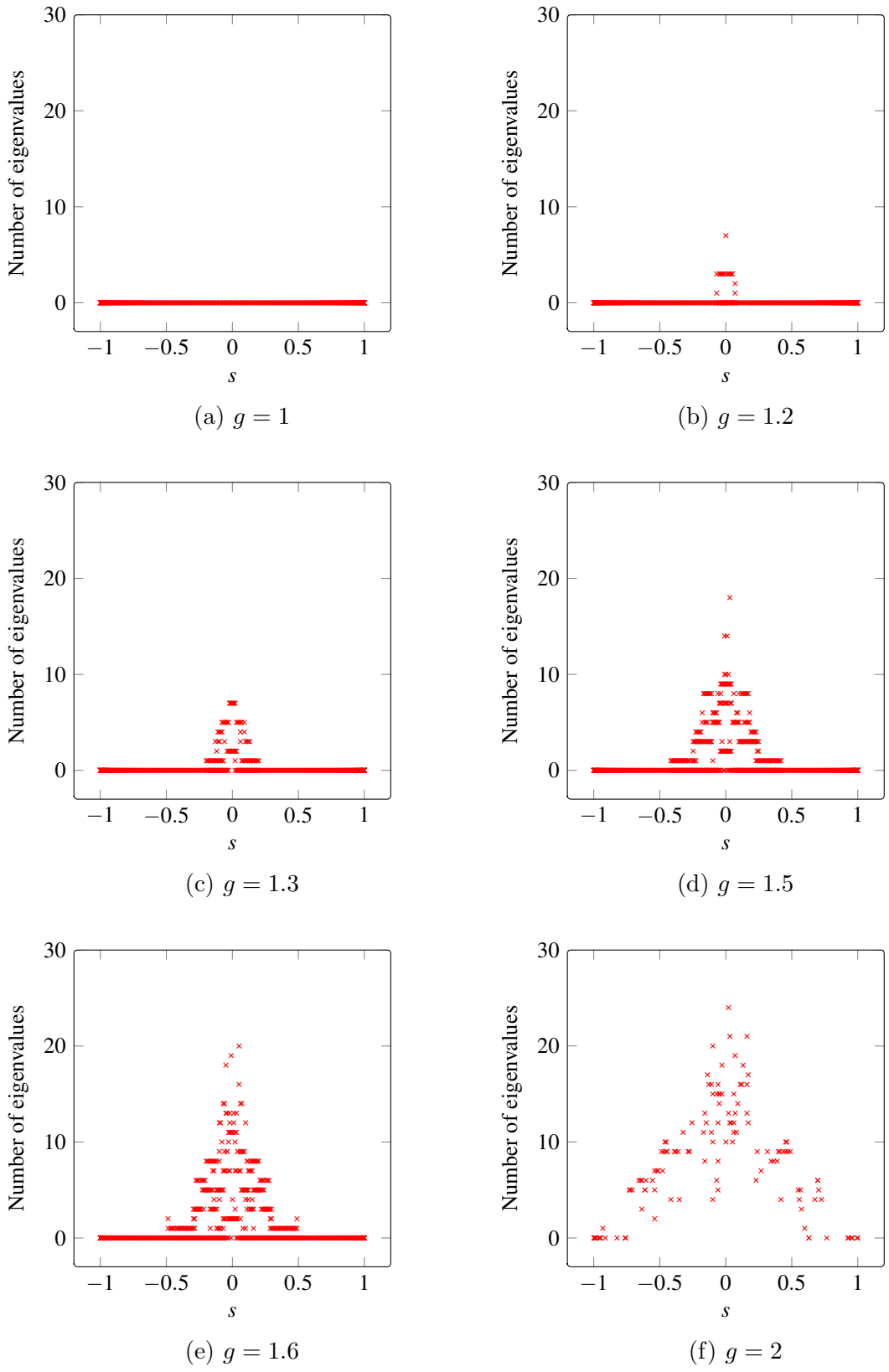


(e) $g = 1.6$



(f) $g = 2$

Figure 4.5: The number of complex eigenvalues obtained at fixed points of instantaneous values of the input signal, for each of the sub-figures in Fig. 4.4. These eigenvalues relate to rotational dynamics in the network. The total number of eigenvalues obtained after linearisation is 200.

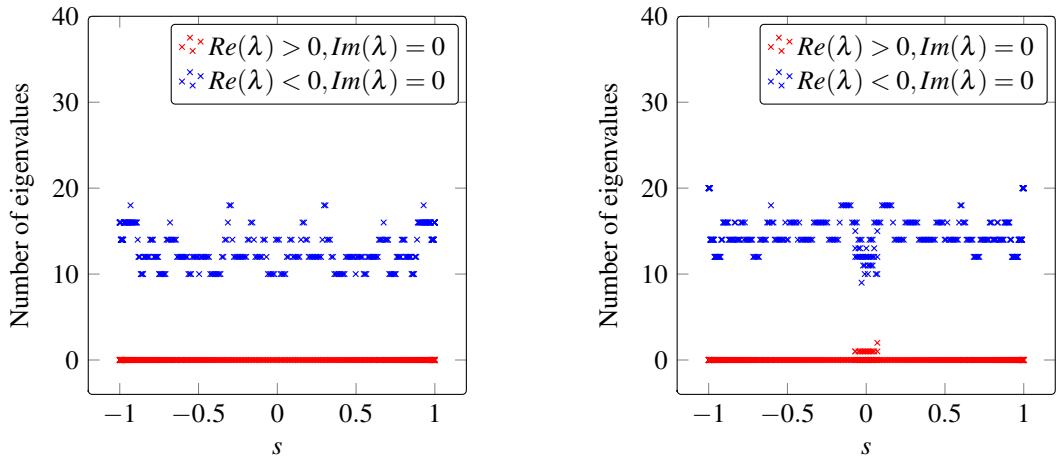


(a) $g = 1$ (b) $g = 1.2$

(c) $g = 1.3$ (d) $g = 1.5$

(e) $g = 1.6$ (f) $g = 2$

Figure 4.6: The number of eigenvalues with a positive real part obtained at fixed points of instantaneous values of the input signal, for each of the sub-figures in Fig. 4.4. There is no differentiation between purely real or complex eigenvalues here. Positive real parts of eigenvalues are associated with instabilities around the stationary points. The total number of eigenvalues obtained after linearisation is 200.



(a) $g = 1$ (b) $g = 1.2$

(c) $g = 1.3$ (d) $g = 1.5$

(e) $g = 1.6$ (f) $g = 2$

Figure 4.7: The number of positive and negative real eigenvalues obtained at fixed points of instantaneous values of the input signal, for each of the sub-figures in Fig. 4.4. The total number of eigenvalues obtained after linearisation is 200.

(Figs. 4.6a & 4.7a). As the value of g increases to 1.3, 1.5, 1.6 and finally 2, three trends can be observed: first, the number of saddle points increases and they spread out in the projected PC-space (Fig. 4.4); second, the range of s values where saddle points appear increases towards higher absolute values, thus increasing the fraction of the range of s that is unstable; third, the number of positive eigenvalues in the saddle points increases both with g and with a diminishing signal amplitude (Fig. 4.6). The first point was also proven from a theoretical perspective [WT13]; that is, the topological complexity of the state space of CTRNNs increases at a similar rate as the dynamical complexity of the network. This means that as g increases above 1, the number of stationary points in the state space increases, along with the estimated largest Lyapunov exponent, commonly associated with diverging trajectories and chaotic dynamics.

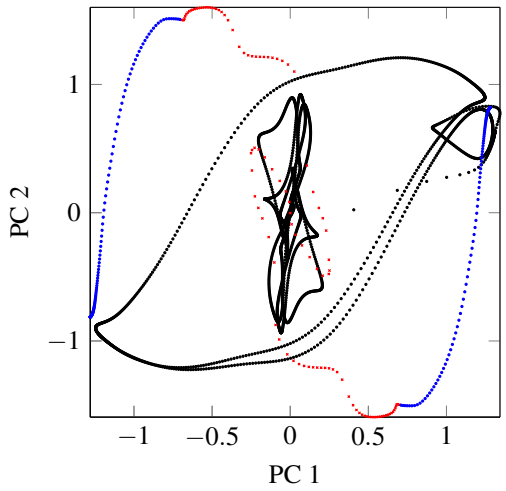
In light of this investigation and in agreement with previous work [NP08], the varying input signal can be considered as an *on-line* bifurcation parameter that can regulate the stability of the system. As the input changes from strong and positive (and thus from an operating regime with stable points) to values close to zero, a series of bifurcations occur that nudge the dynamical landscape into an unstable operating regime. This rhythmic switching between stable and unstable operating regimes results in a higher maximum attractor dimensionality for networks with $g > 1$ as opposed to networks with $g < 1$, that always experience stable nodes. We also note that the edge case $g = 1$, even though usually considered at the edge of stability, consistently behaved similar to the stable cases (Fig. 4.4a).

We will now try to understand various observations about the dynamic behaviour of unstable networks in the previous chapter by linking it to findings in the current one. The first result is the decreasing trend, but with higher variance compared to stable networks, of the average Euclidean distance between successive recorded states as ρ increases. A possible reason for the presence of the decreasing trend relates to the reason for the presence of a similar decreasing trend in stable networks, i.e. because of a resonance-like phenomenon arising from the vastly different speeds of the stationary points and the network state. The higher variance observed for the case of unstable networks though, might arise because of the presence of many saddle points, which prevent this phenomenon from manifesting as strongly. Evidence that supports this speculation can be found when comparing the behaviour of the linear estimator in Figs. 3.7

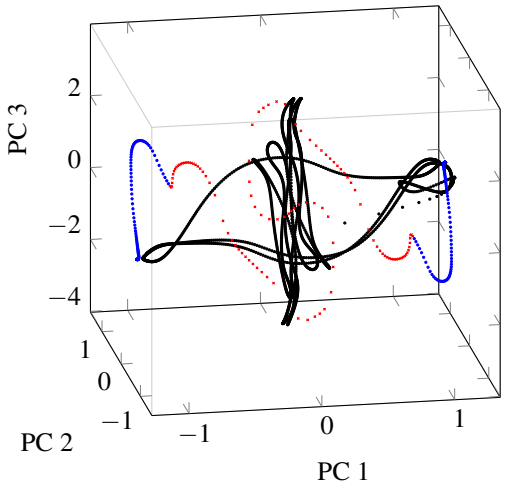
& 3.18. In particular, for stable networks of 200 neurons, a drop in dimensionality to 1 is estimated at high ρ values, indicating the concentration of the variance of the trajectory of the network in a single dimension, whereas for the case of unstable networks of 200 neurons, a high estimate of the dimensionality is maintained even for high ρ values, indicating that the variance of the trajectory is spread across multiple dimensions. Linking this back to the presence of saddle points in unstable networks, I speculate that due to the presence of multiple unstable stationary points, the network state is not strongly attracted by a single stable point, but has more flexibility to move in its state space. In other world, there is no longer a single globally attracting point that strongly affects the network state, as in the case of stable networks, but rather a collection of multiple unstable points that *nudge* the network state around in complicated ways.

The speculation made in the previous paragraph can be supported by evidence shown in Fig. 4.8, depicting the trajectories of two networks driven by two different ρ values, and from two different viewing angles each, projected onto the first three principal components of their corresponding stationary point trajectory. For the network with $\rho = 0.11$, corresponding to a slow input compared to the network evolution speed, the state was initially "trapped" into an irregular oscillation inside a region near the origin that was rich in saddle points. After some time there, the network state "escaped" this region and evolved in a new mode that can be described as a periodic orbit surrounding the region of saddle points. This new orbit appears to have been more strongly affected by the stable points rather than the saddle points. This behaviour observed for small ρ values is similar to that observed in stable networks, whereby the network state was strongly affected by the slow changes in the vector field and position of stationary points, which forced the network state to closely follow.

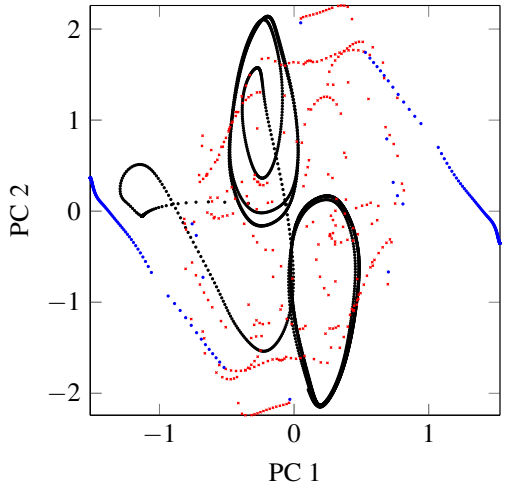
The difference between stable and unstable networks can be observed in the case of a very high ρ value, whereby the network remained in a region near the origin, surrounded by saddle points. As can be seen by the plots, the network activity did not vanish outside the principal dimensions of the trajectory of stationary points, as was the case in stable networks. Though this difference is intriguing, more investigation is required before concluding whether this is because of the dynamics remaining near the trajectory of stationary points or a side-effect of applying PCA



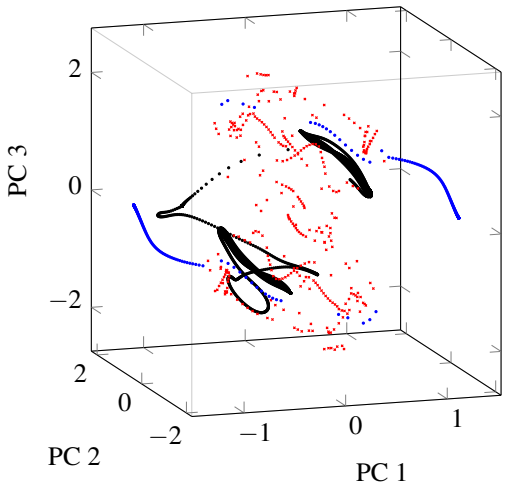
(a) $\rho = 0.11$



(b) $\rho = 0.11$



(c) $\rho = 1.09 \times 10^4$



(d) $\rho = 1.09 \times 10^4$

Figure 4.8: Plots of the activity for networks, at the network timescale, of 200 neurons and $g = 1.5$ driven by various $\rho = 0.11$ ((a) & (b)) and $\rho = 1.09 \times 10^4$ ((c) & (d)). The network trajectories in black are projected on the three principal components of the stationary point trajectory (red crosses for saddle points and blue dots for stable points. (a) & (b) correspond to two different view angles of the trajectory in Fig. 3.21a and (c) & (d) to that in Fig. 3.21d.

on saddle points of unstable networks that have "*spread out*" more in state space compared to stable points in stable networks that are concentrated on a line. The evolution of the network state for this high ρ value can be described by a transition from one periodic oscillation in

one sub-region, to a similar oscillation in a different sub-region, possibly indicating metastable behaviour. This behaviour is much richer compared to that observed in stable networks for very high ρ values, whereby the network approaches a periodic behaviour that occupies a decreasing volume in the state space.

The final point of interest concerns the distribution of stationary points in the reduced PC-space for networks with $g = 1.5$. Even though the connectivity matrices of the two networks were drawn from the same distribution, as described in Section 3.3.1, Fig. 4.8 indicates that the identified stationary points for two such networks appear to be "spread" differently in their PC-space. Even though the optimisation was carried out in exactly the same manner, it resulted in about 350 unique points for the top two plots and about 600 for the bottom two. Even when the optimisation was repeated, it resulted in similar numbers for the two cases. This is an interesting observation, because it illustrates the highly non-linear relationship between the connectivity matrix values and the vector field of CTRNNs. A better understanding of how the two are related could uncover interesting relations between structure and function of high dimensional dynamical systems that could help us design better models and machine learning systems.

4.6 Conclusion

In these last two chapters we set out to investigate the changes in the dynamic behaviour of stable and unstable networks with random sparse connectivity when driven by external periodic stimuli. It was found that both types of networks exhibited a peak in the dimensionality of their response over a specific range of ρ, the ratio of the timescales of the network and the input signal. Moreover, by using dynamical systems theory, the underlying dynamical skeleton responsible

for this behaviour was uncovered, which helped explain the origin of the dynamical transitions observed. The main findings were that small changes in ρ result in the vector field changing with a slightly different speed, as a result of the change in the input frequency, while the speed with which the network evolves remains constant. Due to the appearance and disappearance of stable and saddle points as the vector field changes, the local vector field near the instantaneous state of unstable networks can be significantly different for slightly different input frequencies. This can result in the networks orbiting around attractors that attain complicated geometric forms and that are either periodic or transient, depending on the value of ρ.

These transient attractors have been studied in the context of evidence accumulation and decision making in primates [RBW$^+$13]. The work in these two chapters illustrates that random networks of firing rate neurons can intrinsically accommodate a vast repertoire of attractors, which emerge as a response to inputs with different frequency. In addition, the uncovering of the dynamical skeleton underpinning the dynamics of such networks shows that untrained random networks have a rich dynamical structure that can support the generation of complex neural activation patterns that are considered important for decision making [RBDRWF10, RBW$^+$13, CSFW17] and generation of motor patterns [LB13]. This means that since the necessary degree of complexity is already present, computationally cheap online methods [SA09] can be used to adjust the readout weights, by taking advantage of the rich internal dynamics, without the need to learn explicitly how to represent the input signals. This provides computational evidence to support the hypothesis that the brain uses the dynamics of randomly connected networks as one of the predominant resources for computation [Buo09, Sus14].

In a machine learning context, there are settings in which the network is connected to more than one external source, but only one of them is active at any given time. This is the case, for example, in Sussillo's 3-bit flip-flop neural gate, in which three sources can inject the same signal into the network through different sets of input weights [SB13]. We can interpret the operation of such trained network using the insights gained from this work. When a pulse of ± 1 is injected through any of three input weights, the network state is pushed towards a region in the state space that is dominated by the effect of a stable point. Through training, the network learned to output a constant value of ± 1 with the same sign as the input pulse,

through the readout weights. This output is then continuously fed back to the network through the feedback weights, such that the vector field around the current state of the network is not changing, meaning that the stable dynamics are sustained through the feedback activity. Once a different input pulse arrives, the network is pushed to another region that is also locally stable and different stable dynamics are similarly produced. This example, though simple, illustrates that by intelligently harnessing the dynamical properties of these networks we can create simple and yet robust computational neural circuits.

Moving beyond analysing simple neural circuits, this work also shows that sensory signals that vary much faster than the network processing them can lead to two different timescales over which computation could take place. Our results show that the complexity of dynamics in the network can also be different depending on the timescale over which they are considered. For example, around values of $\rho \approx 1000$, the estimated complexity of dynamics at the input timescale is high, implying high computational capabilities. If we consider the same network and ρ range though, at a timescale that is 100 times longer, the estimated dimensionality is lower. This can be interpreted as the long-term behaviour of the network being *insensitive* to variations taking place in shorter timescales. Even though more work is required to verify that this can be exploited and develop suitable training techniques to do so, FORCE learning is an existing method [SA09] used in training these networks in the paradigm of reservoir computing, that can accommodate weight updates over two – largely different – timescales. The reason that FORCE learning is relevant is because the most important quantity used in training is a running estimate of the inverse correlation matrix, which can be iteratively updated over any desired timescale. Using one of these inverse correlation matrices for each timescale that is relevant to the task, can in theory capture all the correlations necessary for learning over these two timescales. If this speculation is verified, reservoir computing, which currently suffers from catastrophic forgetting [MC89, Fre99] when sequentially learning multiple tasks, can prove a computationally cheaper alternative for training on such tasks than conventional RNNs and LSTMs that are trained using the more expensive backpropagation through time scheme [Wer90]. Even though the goal is promising, there still lies a long path ahead.

Chapter 5

Reservoir computing for episodic-like memory using continuous-time recurrent neural networks

5.1 Precis

Much has been claimed about the computational capabilities of the dynamics of CTRNNs. In this chapter we will focus on a simple computational task; that of memorising a sequence of high-dimensional data and recalling it. The datasets consist of sequences of synthetic and natural images, representing different episodes, while the the training procedure follows the paradigm of reservoir computing, using FORCE learning. The aim here is to assess the ability of CTRNNs in this task and to explore how performance on the task depends on a few important parameters[1].

[1]The basis and part of the work presented in this chapter was published in the paper "A reservoir computing model of episodic memory" [BNS+16] with permission from the publisher ©2011 IEEE. The dataset of used for experiments of "natural images" were kindly provided by Dr. Eren Erdal Aksoy from KIT in Karlsruhe, Germany. This collaboration was part of the EU FET grant (GA:641100) TIMESTORM - Mind and Time: Investigation of the Temporal Traits of Human-Machine Convergence.

5.2 Introduction and background

In the paradigm of reservoir computing, the reservoir acts as a high-dimensional spatio-temporal kernel that can be used to project a usually lower-dimensional signal into the high-dimensional space of the reservoir and to convolve it in time. In this high-dimensional space, even complicated and seemingly similar signals become easier to separate [HS12] and hence linear mappings to a lower-dimensional output space can be easily learned. The main benefit of reservoir computing over other paradigms for training recurrent neural networks is that the connectivity of the reservoir remains fixed, while only simple linear readouts are trained to produce the desired output signals. Traditionally, recurrently connected networks of various types of neuron have been used as reservoirs [Jae01, MNM02, SA09], but in more recent years, time-delayed optoelectronic circuits have enjoyed great attention in reservoir computing implementations [PDS$^+$12, VMVV$^+$14, KHS$^+$18].

The temporal convolution that the reservoir performs on the input signal is closely related to the *echo state property* [YJK12], which in simple terms is the tendency of the reservoir state to maintain a "fading memory" of its input, which given enough time is lost. Even though this property was defined and proven for neural networks implemented with discrete maps, rather than ordinary differential equations, this terminology can also be applied to CTRNNs. Relating this back to the work of the previous two chapters, the echo state property holds if the dynamics of the network are sensitive enough to the input signal, such that a change in the signal causes a change in the reservoir state, but also stable in the long run, such that the effect of this change vanishes after some time, and the reservoir returns back to a stable trajectory. This means that when using CTRNNs in reservoir computing, one of the most important properties that the network must have is stable and reproducible dynamics that are *also* sensitive to the input.

Moreover, the echo state property is closely related to the *separation* and *approximation* properties defined in the context of the Liquid State Machine, developed by Maas, Natschläger and Markram [MNM02]. The name Liquid State Machine was chosen to differentiate it from the Turing Machine, which operates on static and discrete inputs through the operation of a head which can read and write to memory using an instruction table. The Liquid State Machine

on the other hand operates on analogue input signals, in continuous time, and does not need to have a rigid, pre-defined program in order to process the input. Rather, it is defined as a system that adapts over time, based on some form of learning, to perform useful computations based on perturbations of an input signal. This was shown to be a valid model for computation if two necessary properties hold: (a) its state can differentiate between similar but different inputs, termed the *separation* property, and (b) a set of readouts can map the state onto the desired output, termed the *approximation* property. Even though it has proven difficult until now to use these definitions in a practical way so as to develop powerful computational machines consisting of dynamical systems, these properties are nonetheless closely related to how researchers think that the human brain is able to process huge amounts of information in an efficient and intelligent way. Interestingly, in the human brain, the dentate gyrus (DG) and CA3 regions of the hippocampus are thought to respectively support the separation of similar input patterns, analogous to the separate property, and pattern-completion, analogous to the approximation property [OM94, KLH06, KHM16].

Inspired by this correspondence, this chapter will focus on the well defined computational problem of memorising and recalling sequences of temporally ordered data, and in particular high-dimensional images. This task is closely related to episodic memory in humans, for which the aforementioned regions in the hippocampus play a central role. Moreover, this type of task is also of interest to the wider Artificial Intelligence research community, as it was shown that artificial agents trying to maximise the total reward received from their environments through reinforcement learning, can considerably increase their performance if they are equipped with a type of external memory that is analogous to episodic memory in humans [BUP$^+$16, PUS$^+$17, HKSB17, WHA$^+$18], thus suggesting that episodic memory is necessary for exhibiting intelligent behaviour.

As we have briefly discussed here and in Section 1.2.2, episodic memory in humans involves large brain structures that have specific connectivity patterns and multiple neuron types [KHM16], and are orchestrated through complicated interactions to store and retrieve memories. In this chapter our aim is not to provide a physiologically plausible model that explains how episodic memory is supported by brain dynamics, but rather to use such a crucial, but unsolved problem,

as a source of inspiration to guide us in the choice of a simple, yet relevant, computational task. The common theme that underlies both episodic memory in humans and our reservoir computing model is that of recalling high-dimensional, temporally extended data. Hence, we will see how CTRNN dynamics, in the paradigm of reservoir computing, can be used to memorise this kind of sequentially-arriving data, how the information is represented in the state space of the networks, and what determines the reservoirs' ability to successfully complete such task. A convention used throughout this chapter, is that the word *reservoir* will be used interchangeably with the word *network*, and in particular, it will be preferred when referring to a network trained for a task.

5.3 Methods

The learning algorithm chosen for training the output of CTRNNs to memorise sequences of high-dimensional images in this chapter was FORCE learning. Even though a detailed review of FORCE learning has already been given in Section 2.2.2, the update equations used during learning are repeated below for convenience. The running estimate of the inverse of the correlation matrix at time step t, $\mathbf{P}(t)$, which was introduced in Eq. 2.17, was initialised as the $N \times N$ identity matrix at the beginning of each training loop, where N was the number of neurons in the reservoir, and updated through

$$\mathbf{P}(t) = \mathbf{P}(t - \Delta t) - \frac{\mathbf{P}(t - \Delta t)\mathbf{r}(t) \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{P}(t - \Delta t)}{1 + \left(\mathbf{r}(t)\right)^{\mathrm{T}} \mathbf{P}(t - \Delta t)\mathbf{r}(t)}, \tag{5.1}$$

where Δt is the update step size, and $\mathbf{r}(t) = \Big[\tanh(x_1(t)),\ \tanh(x_2(t)),\ ...,\ \tanh(x_N(t))\Big]^{\mathrm{T}}$ is the vector of activations of the reservoir neurons. The update to the weights of the readout $D_{Tar} \times N$ matrix, where D_{Tar} is the dimensionality of the high-dimensional target, was performed using the error vector $\mathbf{e}\_(t)$ between the reservoir output and the target, $\mathbf{y}(t)$, at the current

timestep, using the expressions

$$\mathbf{W}^{Out}(t) = \mathbf{W}^{Out}(t - \Delta t) - \mathbf{e}_{-}(t) \left(\mathbf{P}(t)\mathbf{r}(t)\right)^{\mathrm{T}} \tag{5.2a}$$

$$\mathbf{e}_{-}(t) = \mathbf{W}^{Out}(t - \Delta t)\mathbf{r}(t) - \mathbf{y}(t). \tag{5.2b}$$

The experiments presented here are separated in two set of experiments, one with grey scale synthetic images and one with frames from coloured video of natural scenes. For all experiments, the quality of the reconstructed episode was measured by the peak signal-to-noise ratio (PSNR) (Eq. 5.3a), which makes use of the mean squared error (MSE) and the range of possible values that the channels of each pixel can attain. The reconstructed image quality is considered to be better for higher PSNR values. For the case of the images used here, each 8-bit channel value can range from 0 to 255, and typical values for good image reconstruction are at about $20 - 30$dB. The PSNR was calculated for each frame independently and averaged over all reconstructed frames. The subscript l in Eq. 5.3b denotes the index of all pixel channels of in a frame, and L the total number of pixel channels in a frame. A reconstructed channel of a pixel is thus denoted by z_l, while of the target frame is denoted by y_l. Before training, all pixel values were normalised by the maximum possible value of 255. During reconstruction, the reservoir output was converted back to the $0 - 255$ range for the MSE calculation.

$$\mathrm{PSNR} = \frac{10}{N_{frames}} log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right), \tag{5.3a}$$

$$\mathrm{MSE} = \frac{1}{L}\sum_{l=1}^{L}\left(z_l - y_l\right)^2. \tag{5.3b}$$

For the natural images, an additional metric assessing the structural similarity between the true and reconstructed image was used, namely the mean structural similarity (MSSIM) index, which can attain values in the range $[-1, 1]$. The closer to 1 the value of MSSIM, the more

structurally similar the image is. This measure has been suggested as an alternative to the PSNR, which is based on the MSE, and has been criticised as not being a very good metric to assess image reconstruction quality [WBSS04, WB09, HTG12]. The main problem with MSE based methods is that is can give the same value for two reconstructed images, where one is classified by humans as having a good quality, while the other not. The MSSIM on the other hard, takes a more local approach to the image quality assessment, by using the statistics of localised patches in the two images and then averaging out the similarity across all patches. To calculate the MSSIM for the case of the natural images dataset, the RGB frames, as well as the reconstructed frames, were first converted to the luma format, which uses a single 8-bit channel for each pixel to encode its brightness. Then, following Wang *et al.* [WBSS04], the structural similarity (SSIM) index between two corresponding small patches from the frames, was calculated by

$$\mathrm{SSIM}(\mathbf{z}_m, \mathbf{y}_m) = \frac{\left(2\mu_z\mu_y + (K_1+255)^2\right)\left(2\sigma_{zy} + (K_2+255)^2\right)}{\left(\mu_z^2 + \mu_y^2 + (K_1+255)^2\right)\left(\sigma_z^2 + \sigma_y^2 + (K_2+255)^2\right)} \quad m = 1, 2, ..., N_{pat}. \quad (5.4)$$

In the equation above, $\mathbf{y}_m$ corresponds to a small patch of the image, while $\mathbf{z}_m$ corresponds to the network pixel-wise reconstruction of the same patch. The default values $K_1 = 0.01$ and $K_2 = 0.03$ for the regularising parameters were used as in Wang *et al.* [WBSS04]. In addition, the means of the reconstruction and the original image are respectively μ_z and μ_y, the variances are σ_z^2 and σ_y^2, and the covariance σ_{zy}. The variance of the Gaussian filter used to calculate the means of the patches in the frames had a standard deviation of 1.5, while the patch size used was 11×11. Finally, to obtain the MSSIM, the SSIM values were averages over all N_{pat} patches of a frame and then over all N_{frames} frames of all D_{ep} episodes under consideration.

5.3.1 Synthetic images

Dataset – For the first set of experiments, the Columbia University Image Library of 20 objects[2] [NNM+96] was used. Each of the 20 objects in the dataset consists of 64 unique, 128×128 grey scale frames, showing a full rotation of the object. For this first set of experiments, episodes were constructed by concatenating sequences of frames to create two full rotations of one or more objects, i.e. 128 frames for each object used in an episode. Example images for the first 10 objects are shown in Fig. 5.1

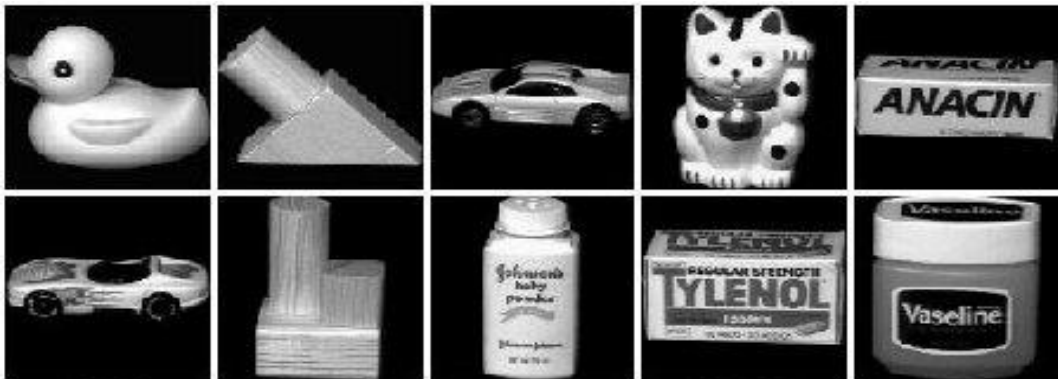


Figure 5.1: The first 10 objects from the Columbia University Image Library of 20 objects [NNM+96].

Multilayer Perceptron (MLP) for image classification – A simple multilayer perceptron was trained to classify the images of the dataset into their respective classes. The MLP consisted of 3 hidden layers, with `tanh` activation functions, details of which are shown in Table 5.1[3]. The purpose of the MLP is to provide an abstract signal, using the sequence of activations of the third hidden layer of 40 neurons extracted per frame presented, that can be fed into the reservoir to serve as a stabilising signal, $\mathbf{s}^{Fb}$. The reason that this signal is required is that, as we have seen, in the absence of an input signal, the reservoir dynamics are unstable, meaning that initially small deviations in the state of the reservoir will be amplified after a few time steps, resulting in a different trajectory. The MLP was chosen because the activations of the hidden layers represent a higher-level abstraction of the original input image.

[2]Dataset available from: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php (date last accessed: 5 Oct 2018)

[3]Credit for implementing and training the MLP in MATLAB goes to Dr. David Bhowmik and Radu Cristian Cioata

Table 5.1: Parameters of the multilayer perceptron used to classify the Columbia University Image Library dataset[NNM+96].

Layer	Number of units	Activation function	Input weights
Input	16384	–	–
Layer 1	190	`tanh`	linear with bias
Layer 2	80	`tanh`	linear with bias
Layer 3	40	`tanh`	linear with bias
Output	20	`linear`	linear with bias

Reservoir – A CTRNN was used as the reservoir, according to Eq. 5.5a during training and Eq. 5.5b during recall. The reason for the two different, but very similar set of equations, is that for this set of experiments only, the network was trained with a slightly different procedure [Jae01] compared to the natural images. This procedure is illustrated in Fig. 5.2. Following this procedure, two sets of input weights were used. $\mathbf{W}^{In}$ was an $N \times D_{Ep}$ weight matrix, where D_{Ep} denotes the number of episodes to be learned, that was used to provide an input pulse, denoted as the instantaneous input $\mathbf{s}^{In}$, of amplitude 2, which lasted for 20 time steps, in the beginning of each episode during training and to signal the recall of the episode after training. This pulse was sent through the weights corresponding to the episode under consideration and its purpose was to push the reservoir state into a unique, small region in its state space, acting as an online reset of the reservoir initial state at the beginning of an episode. For the duration of the pulse, all other input weights received no input. Moreover, throughout the rest of the episode, no input was received through $\mathbf{W}^{In}$.

The second set of input weights is $\mathbf{W}^{Fb}$, a $N \times 40$ matrix, which during training fed the signal $\mathbf{s}^{In}$ from the 40 neurons of the third hidden layer of the MLP to the reservoir for stabilisation, as shown in Eq. 5.5a. During recall, the reservoir instead received the output, $\mathbf{z}^{Fb}$, generated through the readouts feedback weights, $\mathbf{W}^{FOut}$, as shown in Eq. 5.5b. This was a $40 \times N$ matrix that was trained using the Echo State Network approach [Jae01], as shown in Fig. 5.2, at the end of the training period. To find the right set of weights, the complete time series of MLP activations $\mathbf{S}_t^{Fb}$ and reservoir states $\mathbf{R}_t$ were used along with the equation $\mathbf{W}^{FOut} = \left(\mathbf{R}_t^{-1}\mathbf{S}_t^{Fb}\right)^{\mathrm{T}}$. We should note here that the $\mathbf{R}_t$ is usually singular, but the MATLAB function `mldivide` can find a good approximation to this problem, through QR factorisation [MAT5a]. Also, the

input to the reservoir from the MLP, $\mathbf{s}^{Fb}$, as well as the reconstruction, $\mathbf{z}^{Fb}$, were rounded to 3 decimal places, to avoid destabilisation of the dynamics that might result from small deviations in different runs. Finally, the readout matrix $\mathbf{W}^{Out}$, of size $16384 \times N$, was trained with FORCE learning such that the reservoir output, $\mathbf{z} = \mathbf{W}^{Out}\mathbf{r}$, was able to reproduce the high-dimensional target frame. The matrix $\mathbf{P}$ and readout weights $\mathbf{W}^{Out}$ were updated once for each frame. The Euler integration step was 0.02. The connectivity diagram during the training and testing phase is shown in Fig. 5.2.

$$\tau\frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} r_j + \sum_{k=1}^{D_{ep}} W_{ik}^{In} s_k^{In} + \sum_{l=1}^{40} W_{il}^{Fb} s_l^{Fb} \qquad i = 1, 2, ..., N \tag{5.5a}$$

$$\tau\frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} r_j + \sum_{k=1}^{D_{ep}} W_{ik}^{In} s_k^{In} + \sum_{l=1}^{40} W_{il}^{Fb} z_l^{Fb} \qquad i = 1, 2, ..., N \tag{5.5b}$$

$$r_i = \tanh(x_i) \tag{5.5c}$$

$$z_o = \sum_{j=1}^{N} W_{oj}^{Out} r_j \qquad o = 1, 2, ..., N_{Out} \tag{5.5d}$$

$$z_l^{Fb} = \sum_{m=1}^{N} W_{lm}^{FOut} r_m \qquad l = 1, 2, ..., 40 \tag{5.5e}$$

For this set of experiments, two reservoirs with $N = 1000$ and $N = 800$ neurons was used. $\mathbf{W}^{Res}$ had no self-connections, the probability of connection between any two neurons was 0.1, $g = 1.5$, its connection weights were drawn from $\mathcal{N}(0, \frac{g}{0.1N})$ and remained fixed. The readout matrix $\mathbf{W}^{Out}$ was initialised as a fully connected matrix with values drawn from the standard normal $\mathcal{N}(0, 1)$ and scaled by $\frac{1}{\sqrt{N}}$. The input matrix $\mathbf{W}^{In}$ weights were drawn from the standard normal, while $\mathbf{W}^{Fb}$ had weights drawn from the standard normal with a connection probability of $p = 0.1$. Both of these input weight matrices also remained fixed.

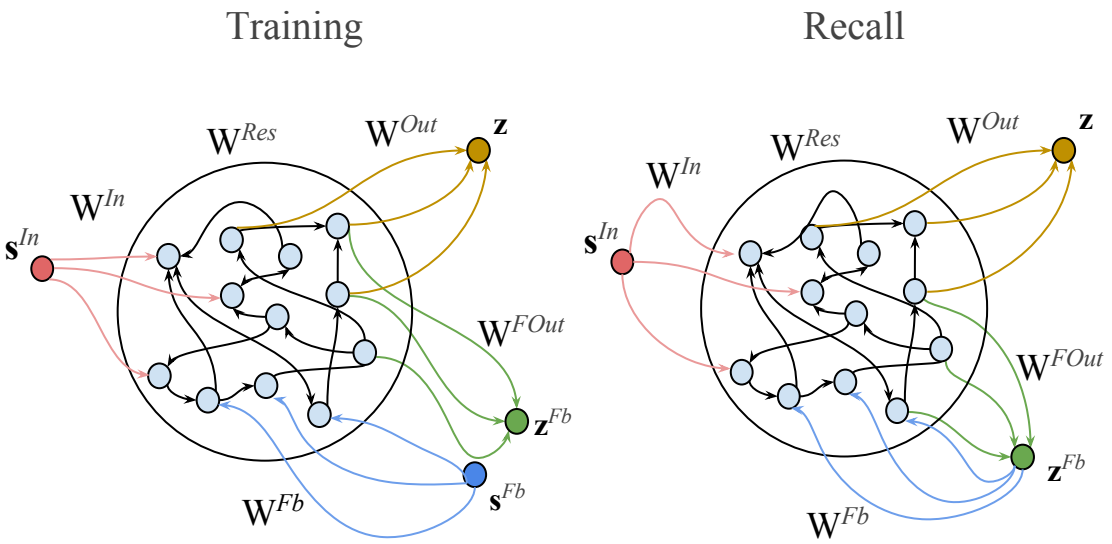


Figure 5.2: Differentiation of the training and recall phase for the first set of experiments, according to the Echo State Network training procedure [Jae01]. During the training phase the reconstruction $\mathbf{z}^{Fb}$ of the stabilising signal $\mathbf{s}^{Fb}$ is disconnected from the feedback and the signal from MLP is injected to the reservoir through the feedback connections $\mathbf{W}^{Fb}$ instead. During memory recall, the reservoir output $\mathbf{z}^{Fb}$ is fed back to the reservoir to guide the reservoir state in a stable way such that it reconstructs the desired target output $\mathbf{z}$ through $\mathbf{W}^{Out}$.

5.3.2 Natural images

In the second set of experiments, images from the natural world were used instead of synthetic ones. The first aim of these experiments was to assess whether reservoirs of CTRNNs, in addition to simpler synthetic images, are able to also memorise and recall sequences of images from the natural world, which tend to have a higher temporal variability and carry more information than synthetic ones.

Dataset – Four videos were used for this set of experiments, kindly provided by Dr. Eren Erdal Aksoy from KIT in Karlsruhe, Germany as part of the TIMESTORM project. The videos were recorded using a still camera capturing the view of a table, where a human enters the scene and uses the ingredients laid out on the table to prepare a salad. Four videos were used in total, each one with small variations in the ingredients and sequence of actions taking place. The videos were preprocessed to extract 30 frames per second of video in frames of

(a) Episode 1

(b) Episode 2

(c) Episode 3

(d) Episode 4

Figure 5.3: An example frame from each of the four episodes of used in the natural images experiments. The videos were kindly provided by Dr. Eren Erdal Aksoy from KIT in Karlsruhe, Germany as part of the TIMESTORM project.

255×255 pixels in RGB. These amount to 195075 channels, and hence outputs to learn, which are more than an order of magnitude more than the number of output that were learned for the case of synthetic images. The sequence of frames extracted from the four videos were used as independent episodes. From Episode 1 to 4, the episodes consisted of 643, 692, 486 and 416 frames. Example frames are shown in Fig. 5.3.

Reservoir – As in the previous set of experiments, a CTRNN was used as the reservoir, but the stabilising feedback signal in this case came from two randomly chosen outputs, with indices l_1 and l_2, instead of the hidden layer activations of a MLP, and was trained using FORCE learning instead of the Echo State Network approach. This is expressed in Eqs. 5.6 and shown in Fig. 5.4. As before, two sets of input weights were used, the $N \times D_{Ep}$ matrix $\mathbf{W}^{In}$ that

was used to send initial pulses to denote the beginning of the episodes, and the $N \times 2$ matrix $\mathbf{W}^{Fb}$ that was used to feed back the stabilising signal generate by the reservoir output. As per the experiments with synthetic images, the input matrix $\mathbf{W}^{In}$ weights were drawn from the standard normal, while $\mathbf{W}^{Fb}$ had weights drawn from the standard normal with a connection probability of p and scaled by the feedback scaling factor α_{fb}. The $N \times N$ connectivity matrix $\mathbf{W}^{Res}$ and the $195075 \times N$ readout matrix $\mathbf{W}^{Out}$ were constructed in a similar way as in the other experiment. Finally, an Euler integration step of 0.01 was used.

The aim of the experiments was to assess how the performance of the reservoirs, measured by the average PSNR between the reconstructed and original images, depends on (1) the gain of the reservoir g, (2) the number of episodes learned, (3) the number of neurons in the reservoir N, (4) the number of training loops, (5) feedback scaling factor α_{fb} and (6) the feedback weights connection probability p. A number of experiments were run where during the training phase, an input pulse of amplitude 2 was send to the reservoir for 100 time steps, followed by the presentation of the episode while applying FORCE learning. For each training loop, the estimate of the inverse correlation matrix $\mathbf{P}$ was reset to its initial value, followed by the presentation of a single occurrence of all episodes to be memorised. Once all the training loops were completed, the testing phase began, by running five different realisations of the experiment, with different initial conditions. The average PSNR for the five runs and its standard error were calculated.

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} r_j + \sum_{k=1}^{4} W_{ik}^{In} s_k^{In} + \sum_{l=1}^{2} W_{il}^{Fb} z_l^{Fb} \qquad i = 1, 2, ..., N \tag{5.6a}$$

$$r_i = \tanh(x_i) \tag{5.6b}$$

$$z_o = \sum_{j=1}^{N} W_{oj}^{Out} r_j \qquad o = 1, 2, ..., N_{Out} \tag{5.6c}$$

$$\mathbf{z}^{Fb} = \begin{bmatrix} z_{l_1} & z_{l_2} \end{bmatrix} \tag{5.6d}$$

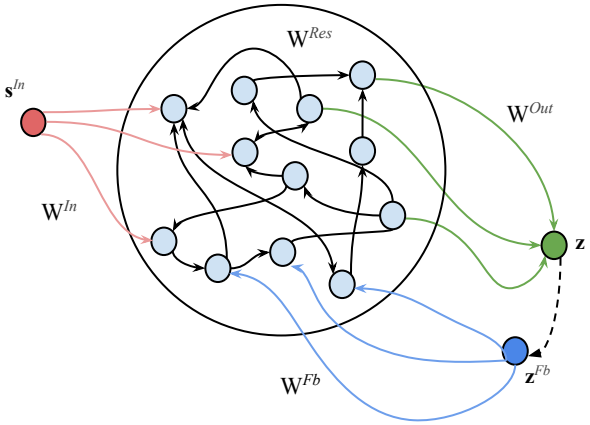


Figure 5.4: The connectivity of the reservoir used for the experiments on natural images. Two random outputs from the reconstructed frame $\mathbf{z}$ were chosen to be used as the feedback stabilising signal $\mathbf{z}^{Fb}$, which was sent back to the reservoir both during training and testing. This is shown as a black dashed line from $\mathbf{z}$ to $\mathbf{z}^{Fb}$.

5.4 Results

5.4.1 Synthetic images

For the first experiment a reservoir with 1000 neurons was used to memorise and recall five independent episodes, where each episode corresponded to two full rotations of a single object from the Columbia University Image Library of 20 objects [NNM+96]. For each episode, an initial pulse of amplitude 2 was sent through a separate set of input weights, corresponding to different columns of $\mathbf{W}^{In}$, for 20 time steps to provide a cue to the reservoir. Both training and recall began at the end of this input pulse. The reconstruction ability of the reservoir assessed after 5 training and testing runs and was found to be 28.40 ± 0.02dB. Example frames from the reconstruction of the five objects during recall are shown in (A-E) Fig. 5.5.

Following the successful independent memorisation and recall of the five episodes, the reservoir state was visualised using PCA, as shown in Fig. 5.6. The reservoir was able to memorise and recall the five episodes by encoding them in five different reservoir trajectories that are able to reproduce the high-dimensional image when read out by the readout weights in $\mathbf{W}^{Out}$. Each episode began from a different initial reservoir state after the initial pulse ended, as a result of the different input weights used for each episode. Because the input weights were different

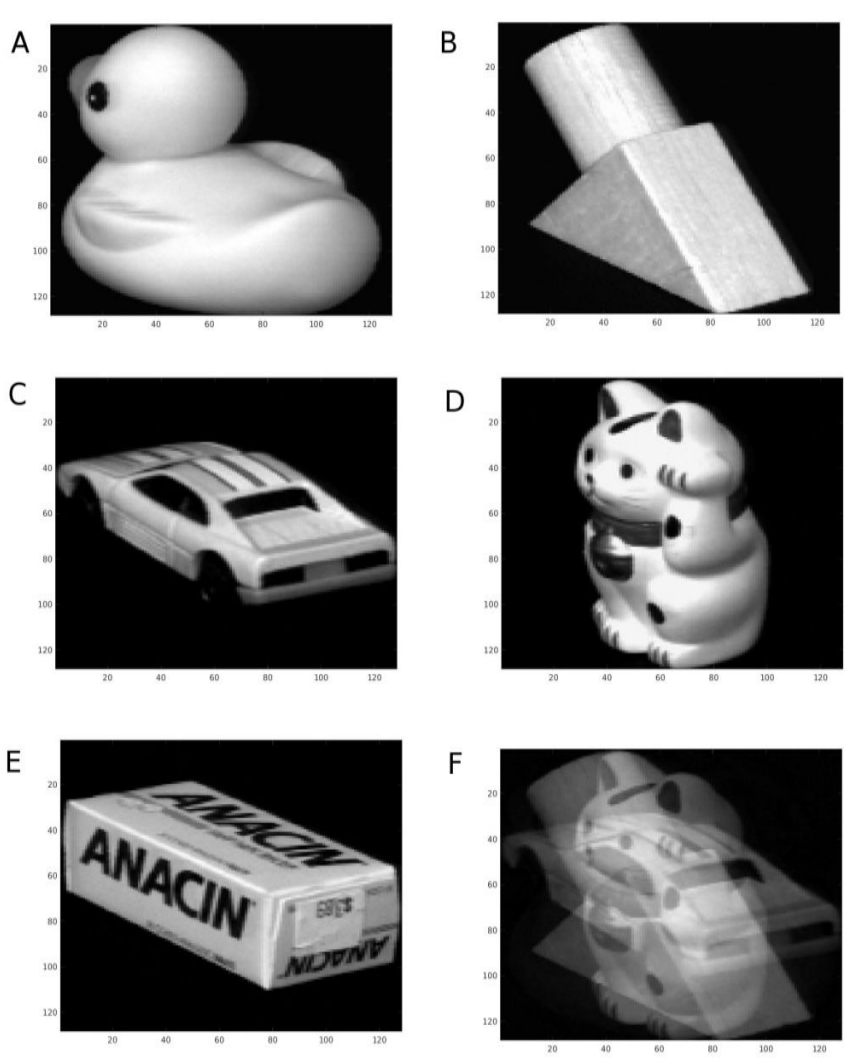


Figure 5.5: Recall of objects from the Columbia University Image Library [NNM+96]. (A-E) The first five objects in the dataset used for the first experiment. (F) Merged images of the objects, as reconstructed by the reservoir output during recall in the second experiment. This output was produced at the branch point, shown in Fig. 5.8, and is a superposition of the three objects that could potentially follow from the branch point. The figure was reproduced from [BNS+16] with permission from the publisher ©2016 IEEE.

for each episode, the reservoir state was attracted towards a different stable point during the time that the pulse was applied. From then on, the different readout weights for the pixels of the reconstructed frame could independently map the high-dimensional state of the reservoir to the many one-dimensional pixel values at each time step. Concurrently, the feedback weights, $\mathbf{W}^{FOut}$ and $\mathbf{W}^{Fb}$, first mapped the reservoir state to the 40-dimensional stabilising signal corresponding to the particular episode, which was then fed back to the reservoir to provide the necessary stability for "guiding" it through the appropriate stable trajectory. This resulted in the reservoir memorising five distinct trajectories that were used as temporal, high-dimensional

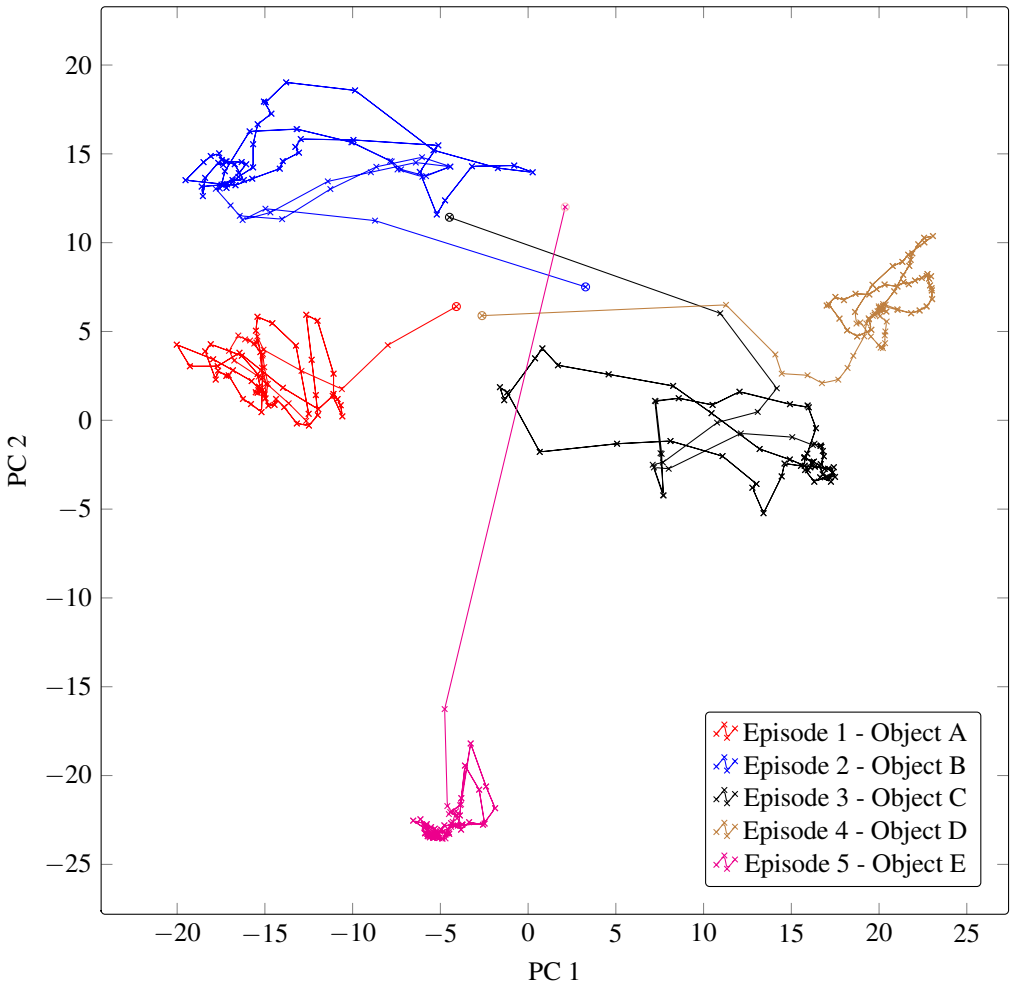


Figure 5.6: PCA on the reservoir trajectories during independently recalling the five episodes. Each object has an associated letter corresponding to a unique ID, which is shown in Fig. 5.5. The beginning of each episode is denoted by a circle mark. The results appeared in [BNS+16].

"reservoir" states for memory storage and retrieval. These trajectories were separated out in the two-dimensional PC-space, as seen in Fig. 5.6.

An interesting point to note is that for all episodes, in the second rotation of the object in the image, the reservoir trajectory returned near the state visited in the first rotation during the corresponding frames. In fact, in all episodes, the reservoir state for corresponding frames was closer in later frames rather than earlier ones. The reason for this is because during the initial frames of the first rotation, the reservoir state was near the stable point defined by the input pulse and hence required more time steps before returning to the stable trajectory imposed by the stabilising signal. Initial frames of the second rotation on the other hand, were already close to this stable trajectory and hence a large Euclidean distance between corresponding frames was

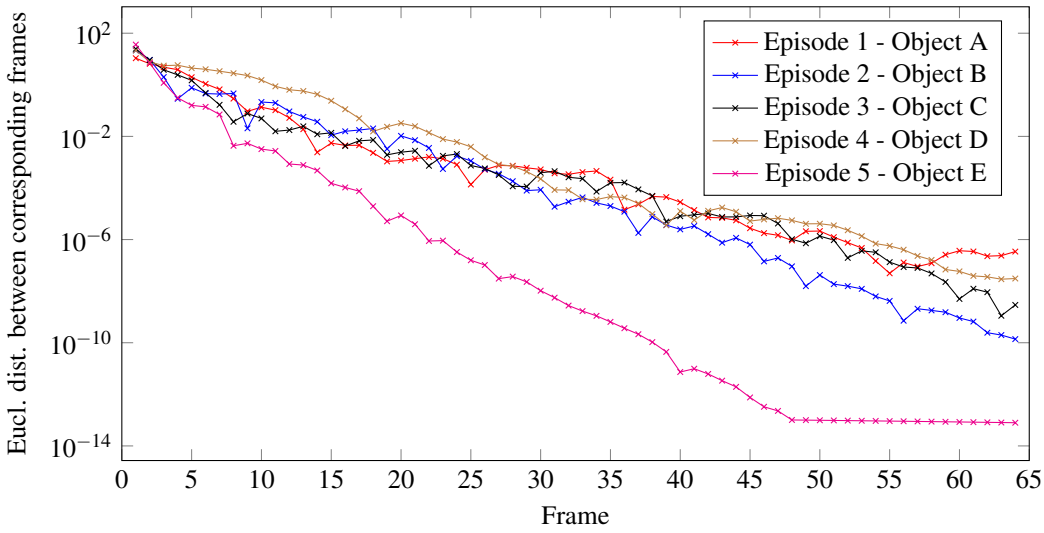


Figure 5.7: Euclidean distance in the reduced two-dimensional PC-space between corresponding frames of the two rotations of each objects for each of the five episodes in Fig. 5.6. Initial frames were further apart because of the initial pulse, but subsequent frames were reproduced from very similar reservoir states due to the formation of attracting dynamics by the stabilising signal.

measured during initial frames of each episodes and shown in Fig. 5.7. These results show that the reservoir dynamics were successfully adjusted by the stabilising signal such that the stable trajectory was effectively reused for the reconstruction of the same high dimensional images during its second occurrence in the same episode.

To explore this property more, a second experiment was carried out, where two episodes were learned and recalled using a reservoir with 800 neurons. The first episode was longer than the episodes in the first experiment (Fig. 5.6), while the second episode had the same the same number of frames. In this second experiment, each object was only shown for a single rotation, but two or more objects were shown in each episode. Specifically, in Episode 1, the sequence of objects A, B, A, C was memorised and recalled, whereas in Episode 2 a simpler sequence of A followed by D was used. The aim of this experiment was to see whether the stable dynamics imposed by the reconstructed stabilising signal could differentiate between different objects succeeding the same object within the same episode, as well as across different episodes. If this is not possible, it means that the dynamics relating to each object could only lead to a single successor, something that can severely limit the possible applications of these reservoirs.

On the contrary, as shown in Fig. 5.8, the reservoir dynamics could both learn an efficient representation of temporal sequences through the feedback mechanism and maintain a memory

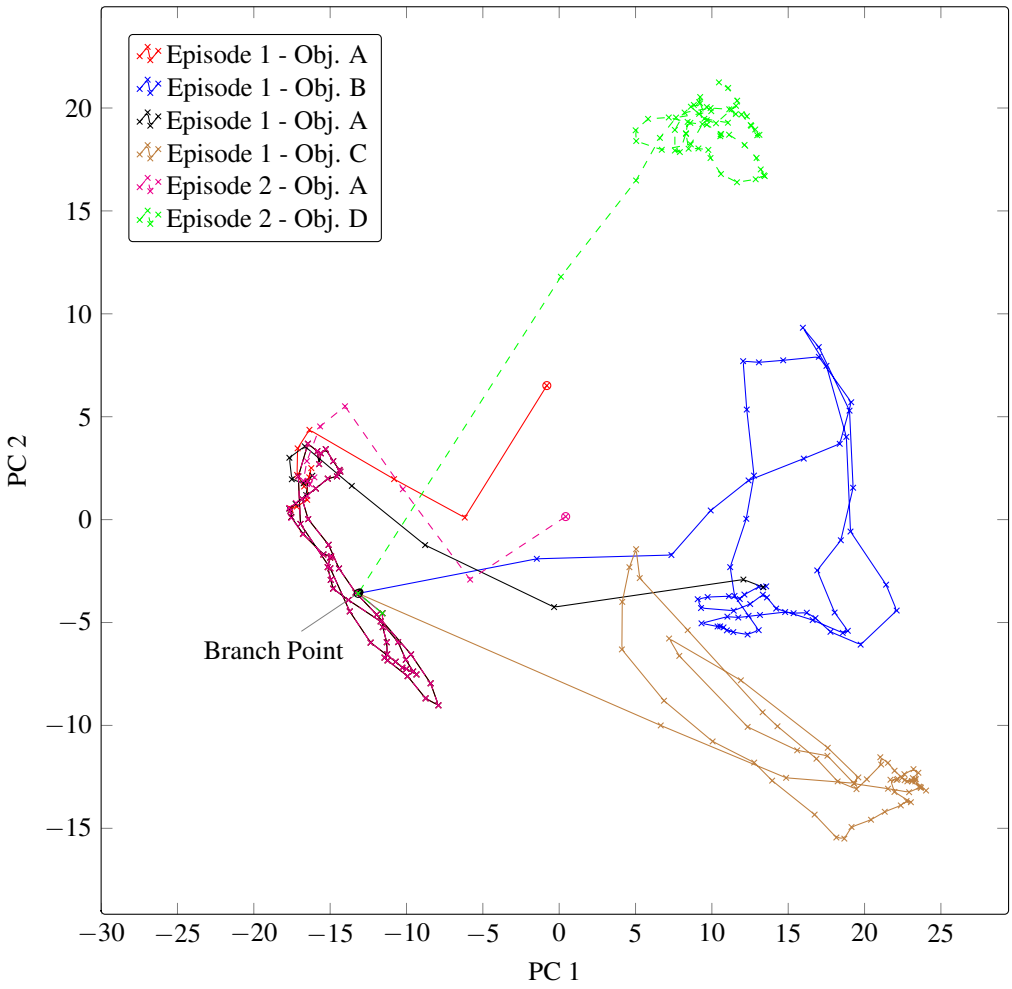


Figure 5.8: PCA on the reservoir trajectories during independently recalling two episodes. In Episode 1, shown by solid lines, the sequence of objects A, B, A, C was memorised. Both occurrences of object A were recalled when the reservoir followed approximately the same trajectory in the PC-space, followed by branching out to B or C via the branch point. The image reconstruction at the branch point was a superposition of all three possible images of objects that could have followed (Fig. 5.5F). Episode 2, shown by dashed lines, was a simpler episode made up of two objects A and D. Even in this case, the reservoir state followed a trajectory that was consistently near the other occurrences of A, before separating out via the branch point. The beginning of each episode is denoted by a circle mark. The figure was adapted from [BNS+16].

of past states, which meant that the repeated object was followed by the appropriate successor every time. In Fig. 5.8, efficient representation translates to trajectories for all occurrences of object A being overlaid in the reduced PC-space, while a memory of the past states translates to the different successors of A being successfully recalled every time. This memory of past states can be attributed to small differences in the reservoir state throughout the stable trajectory corresponding to object A and at the branch point, which are small enough to always reproduce A, while different enough to lead to the reconstruction of a slightly different stabilising signal at

the branch point. This small difference is quickly amplified after the branch point to guide the reservoir to the stable trajectory corresponding to the appropriate successor of A. Moreover, as we have seen, the readout weights overlay the reconstructed images of the three objects (Fig. 5.5F) at the branch point. This interpolation is representative of the linear mapping from reservoir state to individual pixel values that the readout performs. Finally, the PSNR after five repetitions of this experiment was 29.33 ± 0.01dB, which was much better than when learning five independent episodes.

5.4.2 Natural images

Now that we have seen how stable trajectories imposed by the stabilising signal can be used to encode the images, we will begin our investigation of how the various reservoir parameters affect the quality of the reconstructed video, as measured by the average PSNR and MSSIM. The most important reservoir parameter that caused the biggest difference in the quality of the reconstructed episodes was the gain of the reservoir g. This can be seen in Fig. 5.9, where the unstable reservoirs with $g \geq 1.0$ reconstructed the episodes with considerably better quality compared to stable ones ($g = 0.8$). This can be seen through both the PSNR and MSSIM quality measures, which consistently estimated similar trends in the quality of reconstruction. The PSNR value obtained when using unstable reservoirs was in the range of $20 - 30$dB, which is usually considered to correspond to a good quality reconstruction, while the stable reservoirs led to a PSNR value of about 15dB, which is usually considered low. When evaluated with MSSIM, unstable reservoirs scored values mostly in the range $0.8 - 0.95$, while stable ones scored in the range $0.4 - 0.5$.

To visually assess the frame reconstruction quality, a typical unstable reservoir of 1200 neurons, $g = 1.5$, $\alpha_{fb} = 0.5$, $p = 0.1$ and trained for 15 loops to memorise and recall four episodes, was

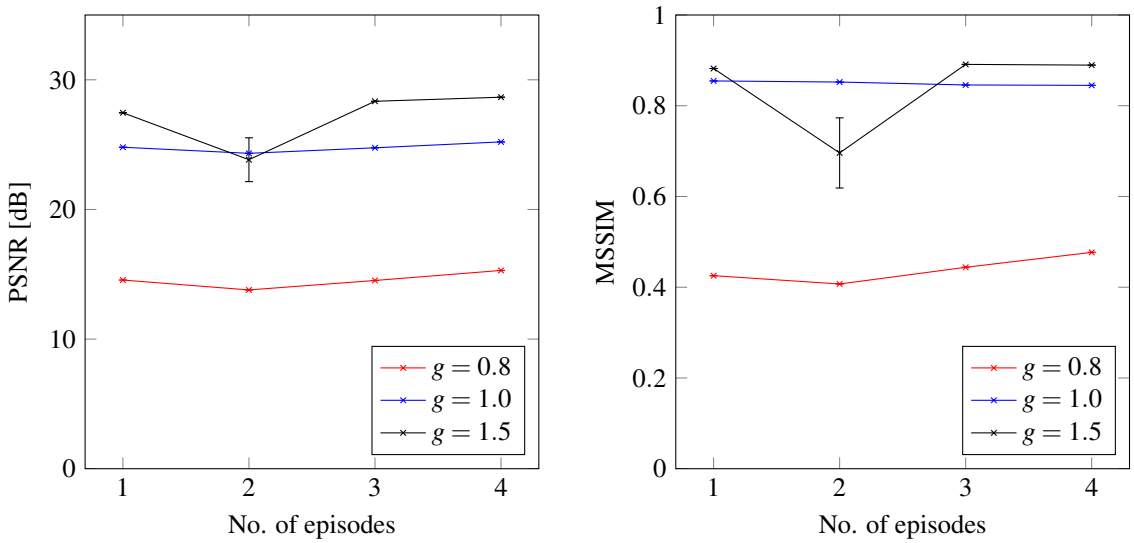


Figure 5.9: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The reservoirs' gain was varied, while the other parameters used were $N = 1200$, $\alpha_{fb} = 0.5$, $p = 0.1$ and the number of training loops used was 15.

selected to memorise and construct all episodes. Four frames from each of the four episodes reconstructed are shown in Fig. 5.10. The reconstructions come from a reservoir that performed very well (PSNR $= 28.65 \pm 10^{-3}$dB & MSSIM $= 0.89 \pm 10^{-3}$) As we can see, the reconstructed frames, even though blurry at times, retained a good structure, which allows a human to identify what kind of action is taking place and what objects are present in the scene. Furthermore, the good quality of reconstruction was retained for, and throughout, every episode.

Also, we can see that the quality of reconstruction did not decrease as the number of episodes (to be memorised and recalled) increased. This means that the reservoirs used in these experiments were equally capable of remembering one, three or four episodes. An exception can be seen for reservoirs, with $g = 1.5$, trying to learn two episodes, where the quality of reconstruction decreases. This is probably an effect of the small number of repetitions used and would disappear if more repetitions of the experiment were carried out.

To further assess how the reconstructed frames were corrupted for experiments with a low performance, the same frame of the first episode – from different experiments – was chosen and shown in Fig. 5.11. Each of them is reconstructed by a reservoir with different parameters. Comparing Fig. 5.11b, reconstructed using an unstable reservoir, to Fig. 5.11a, which is the original frame, we can see that the reconstruction closely matched the original frame. There

Frame 250 Frame 300 Frame 350 Frame 400

Figure 5.10: Example reconstructed frames from the four episodes. Going from top to bottom row, Episodes 1, 2, 3 and 4 using a reservoir of $N = 1200$, $\alpha_{fb} = 0.5$, $g = 1.5$, $p = 0.1$, and trained for 15 loops and four episodes. The calculated PSNR was 28.65dB, while the MSSIM was 0.89. The reconstructed frames are sometimes slightly blurred, yet easily comprehensible to the human eye. The videos were kindly provided by Dr. Eren Erdal Aksoy from KIT in Karlsruhe, Germany as part of the TIMESTORM project.

were minor blurs in the reconstructed frame, especially at regions of the image where movement is taking place. Going from Fig. 5.11c to 5.11d, 5.11e and then 5.11f, the blurring in the reconstructed frame increases, as indicated by the lower quality measured. Also, in each of these

(a) SSIM = 1.0

(b) SSIM = 0.86
PSNR = 29.46dB

(c) SSIM = 0.57
PSNR = 10.63dB

(d) SSIM = 0.33
PSNR = 8.64dB

(e) SSIM = 0.18
PSNR = 7.63dB

(f) SSIM = 0.12
PSNR = 6.24dB

Figure 5.11: Example reconstructed frames of varying quality. The reconstructed frame shown is frame 200 of the second episode. (a) Original image. (b) Reservoir parameters: $N = 1200$, $\alpha_{fb} = 0.5$, $g = 1.5$, $p = 0.1$, and trained for 15 loops and four episodes. PSNR = 28.70dB and MSSIM = 0.89 for this run. (c) Reservoir parameters: $N = 1200$, $\alpha_{fb} = 0.5$, $g = 0.8$, $p = 0.1$, and trained for 25 loops and two episodes. PSNR = 14.14dB and MSSIM = 0.44 for this run. (d) Reservoir parameters: $N = 1200$, $\alpha_{fb} = 0.5$, $g = 0.8$, $p = 0.1$, and trained for one loop and two episodes. PSNR = 13.85dB and MSSIM = 0.41 for this run. (e) Reservoir parameters: $N = 1200$, $\alpha_{fb} = 0.5$, $g = 0.8$, $p = 0.1$, and trained for 15 loops and two episodes. PSNR = 13.44dB and MSSIM = 0.36 for this run. (f) Reservoir parameters: $N = 600$, $\alpha_{fb} = 0.5$, $g = 1.5$, $p = 0.1$, and trained for 15 loops and four episodes. PSNR = 13.97dB and MSSIM = 0.40 for this run. In the label of each reconstructed frame, the PSNR and MSSIM of the frame shown is reported. The decrease in quality can be easily seen from the frames. The videos were kindly provided by Dr. Eren Erdal Aksoy from KIT in Karlsruhe, Germany as part of the TIMESTORM project.

reconstructed frames, which correspond to reconstructions from stable reservoirs, objects from the second episode are overlaid on the frame of first one, which is reflected in their lower quality.

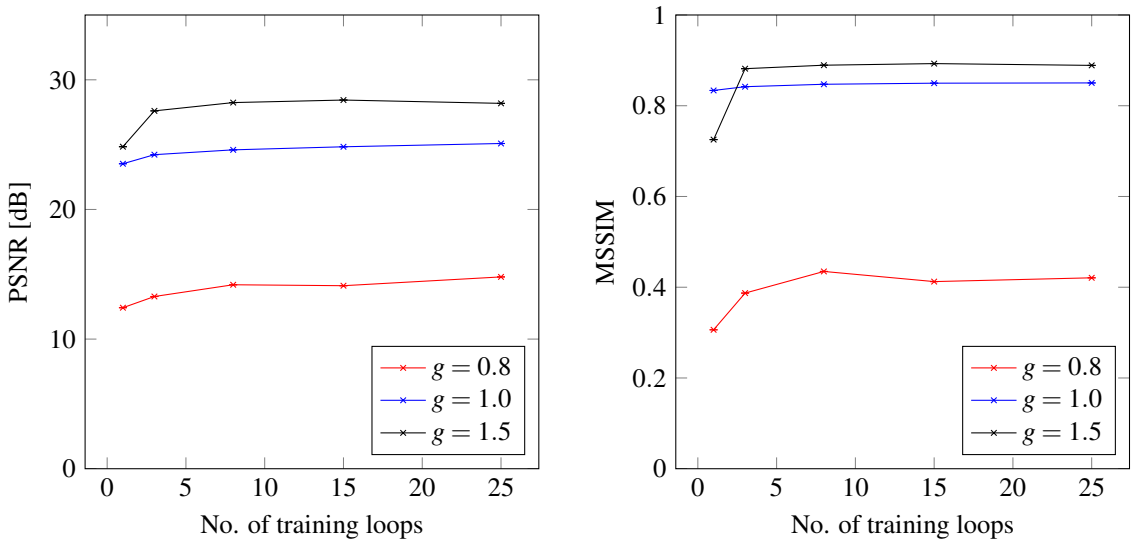


Figure 5.12: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The reservoirs' gain and number training loops were varied, to assess the quality of reconstruction when memorising and recalling three episodes, while $N = 1200$, $p = 0.1$ and $\alpha_{fb} = 0.5$.

Next, the quality of reconstruction subject to changing the number of training loops was evaluated. As can be seen from Figs. 5.12 & 5.13, when the number of training loops was increased to more than one, the quality of reconstruction increased for all reservoirs, but it was more evident for the case of unstable ones. This can be expected, because reservoirs with unstable dynamics require more training to be stabilised than stable ones and hence, a smaller-than-necessary number of training loops leads to a considerable decrease in performance. The improvement in measured quality – as more training loops were used – became smaller, reaching a saturation point, whereby increasing the number of training loops did not improve the quality of the reconstruction any further.

From the preceding experiments, we can see that the gain of reservoirs, g, plays the most important role in their ability to successfully store and recall high-dimensional temporal data. In particular, unstable reservoirs consistently outperformed stable ones in this task. The reason is that unstable reservoirs are able to respond to the feedback signal with richer dynamics compared to stable reservoirs, which allows them to produce different, yet stable trajectories, that are used to store the memory. This property is closely related to the separation property described earlier, whereby, the state evolution of reservoirs with good separation is able to differentiate between similar, but different input signals. This allows them to store each memory

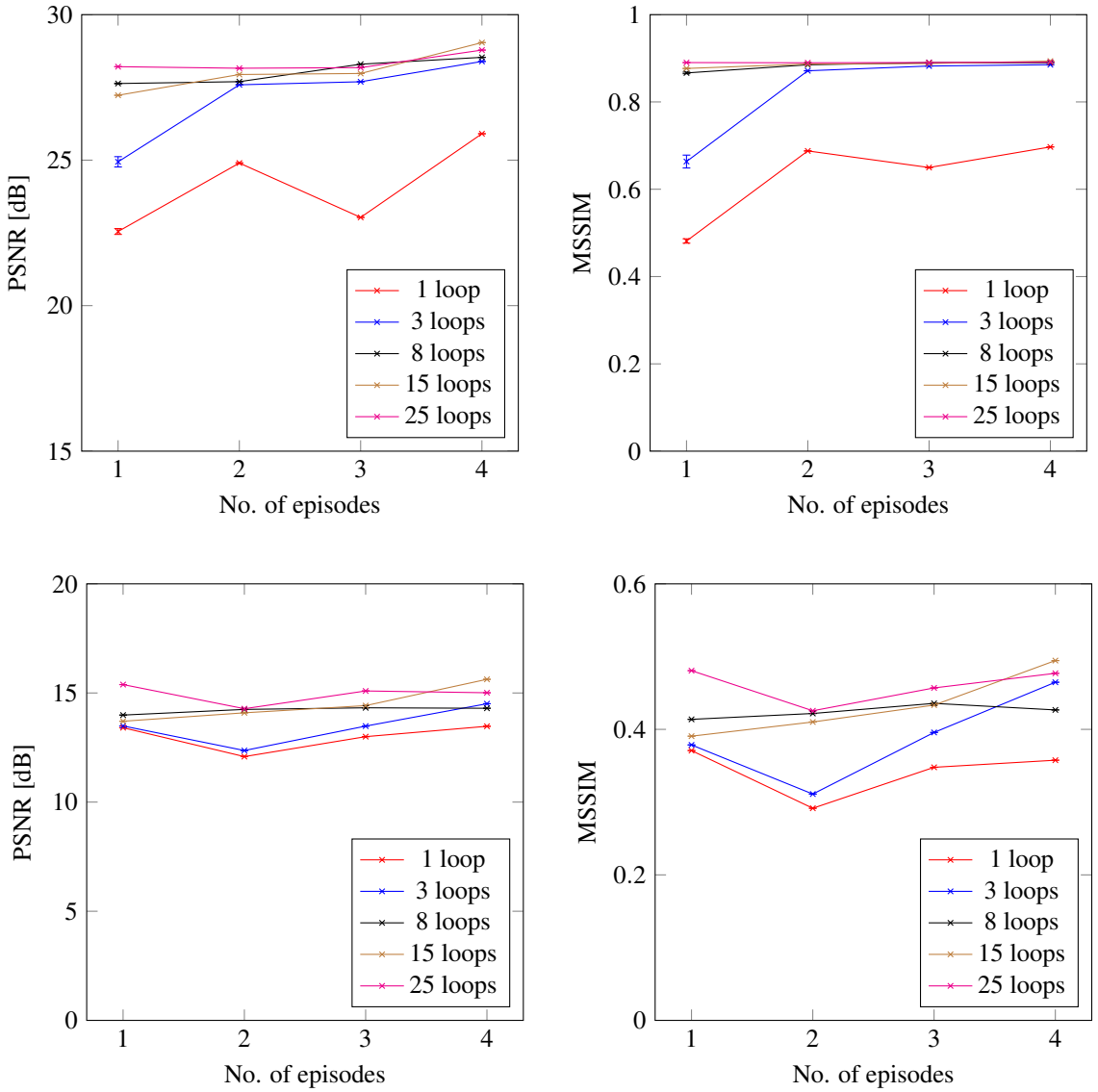


Figure 5.13: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The number of training loops was varied, while the other reservoir parameters used were $N = 1200$, $\alpha_{fb} = 0.5$ and $p = 0.1$. The top row corresponds to unstable reservoirs with $g = 1.5$, while the bottom row to stables ones with $g = 0.8$. Stable reservoirs perform much better than unstable ones, and are more affected by the number of training loops used. The results show that increasing the number of training loops leads to a decreased performance, in most cases, as measured by the two measures.

in a different trajectory, which makes the reconstruction – linked to the approximation property – much easier. The separation taking place in stable reservoirs on the other hand is much less, due to the very stable dynamics, and is not able to effectively differentiate its response to different input signals. This also leads to poorer quality in the reconstructions, whereby different episodes appear to be blurred together (Fig. 5.11f). This effect is characteristic of the network state following trajectories which are very close to each other, which leads to the linear readout

matrix reconstructing a merged image that combines the two memories.

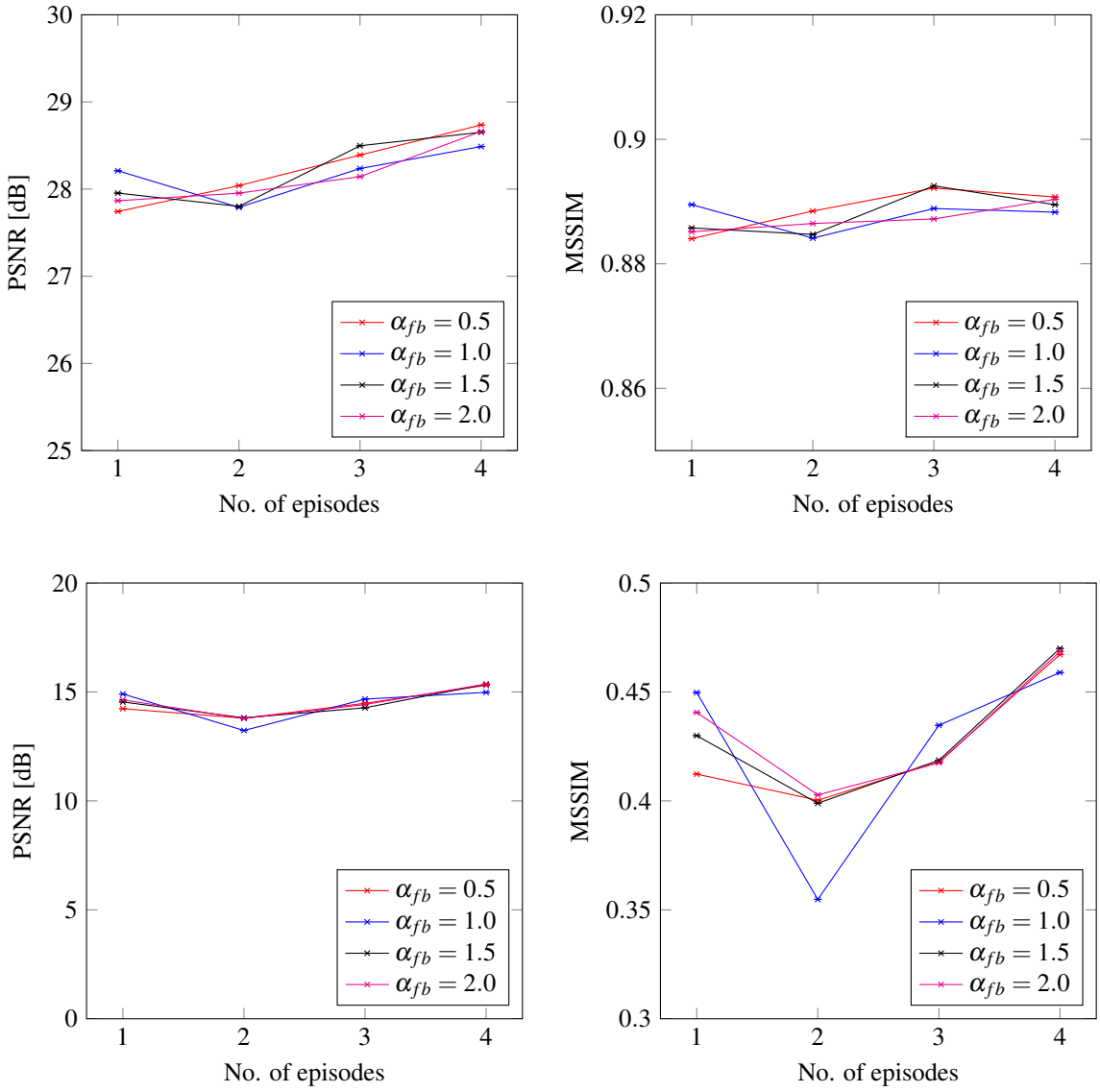


Figure 5.14: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The reservoirs' feedback scaling was varied, while the other parameters used were $N = 1200$, $p = 0.1$, and the number of training loops used was 15. The top row corresponds to unstable reservoirs with $g = 1.5$, while the bottom row to stables ones with $g = 0.8$.

From what we have seen in previous chapters though, the input signal to the CTRNNs is very important for the resulting dynamics. It can drive the state of stable reservoirs, or even induce stable dynamics in unstable ones. To asses whether increasing the feedback scaling α_{fb} leads to improved performance in this task, α_{fb} was varied from 0.5 to 2.0. Moreover, the connection probability, p, of the feedback connectivity matrix, $\mathbf{W}^{Fb}$, was also varied between 0.1 and 0.8. The aim here was to examine whether making the density of feedback connections

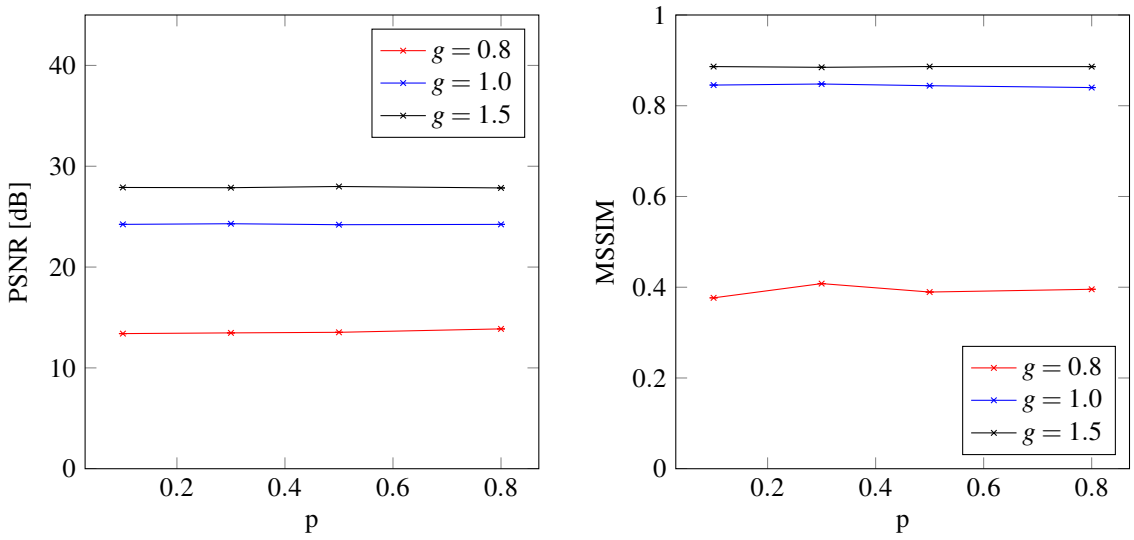


Figure 5.15: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The reservoirs' gain and feedback connection probability were varied, to assess the quality of reconstruction when memorising and recalling two episodes, while $N = 1200$, $\alpha_{fb} = 0.5$ and 8 training loops were used.

higher, or increasing the intensity of the feedback signal to the network, leads to any change in performance.

Experiments with $N = 1200$, $g = 1.5$ and 15 training loops were conducted to assess the effect of the feedback scaling α_{fb}. The results in Fig. 5.14, show that values of the feedback scaling factor in the range $0.5 - 2.0$, led to similar performance for both stable and unstable reservoirs. Hence, higher amplitude signals do not lead to changes in the dynamics of reservoirs that can offer improvements in task performance.

For the second experiment, networks with $N = 1200$ and $\alpha_{fb} = 0.5$ were trained for 8 training loops to memorise and recall two episodes. The gain g and the probability of feedback connection p were varied to assess whether providing more dense input connectivity for the stabilising signal has any effect on the quality of episode reconstruction. The results from this experiment are shown in Fig. 5.15, where we can see that increasing the density of connections does not lead to improved performance. Therefore, combining the results of all of the above experiments, we can conclude that unstable reservoirs are better suited for memorising and recalling sequences of high dimensional data than stable ones. Following this conclusion, it should not come as surprising that the dynamics of biological brains seem to exhibit similarly complex dynamics

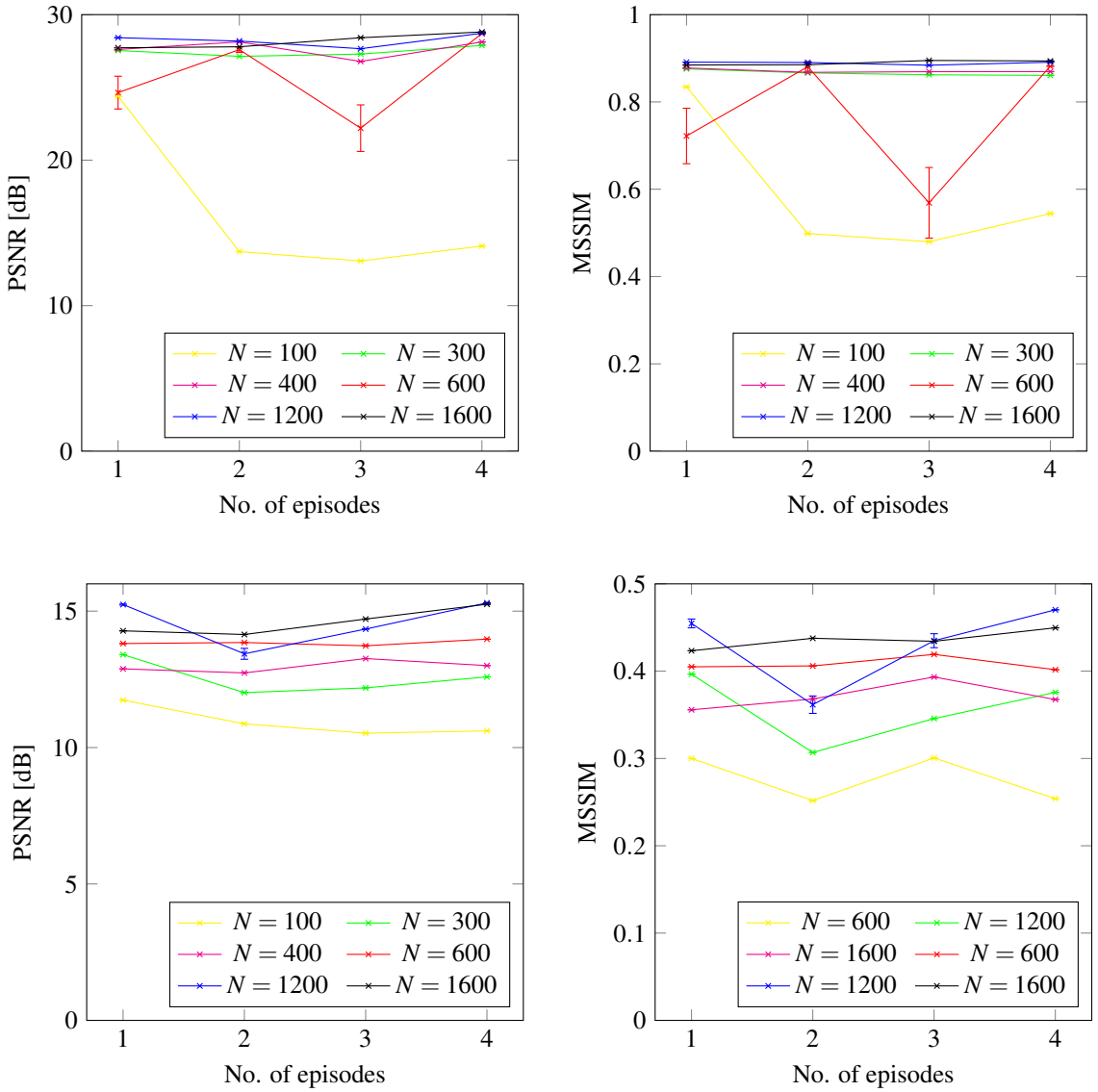


Figure 5.16: Quality of the reconstructed videos, as measured by the average PSNR (left plot) and the MSSIM (right plot). The reservoirs' number of neurons was varied, while the other parameters used were $\alpha_{fb} = 0.5$, $p = 0.1$ and the number of training loops used was 15. The top row corresponds to unstable reservoirs with $g = 1.5$, while the bottom row to stables ones with $g = 0.8$.

[STE00, Chi10], since from the experiments conducted here, they are ideal for memorising and recalling high-dimensional temporal patterns.

The final parameter considered was the number of neurons, N, in the reservoir. Fig. 5.16 shows that both types of reservoirs suffered from a decreased performance when a smaller number of neurons was used. Even though this effect is more intense in stable reservoirs, it becomes more apparent if the number of neurons used in unstable ones is reduced significantly. This can be linked back to the approximation property of reservoirs, which relates to how well the reservoir

state can be used to approximate a target output. Having a higher dimensional reservoir, i.e., more neurons, makes it easier to find a set of linear readout weights that can map the reservoir states to the pixel channel values across the episodes. The reason for this, is because states in higher dimensions are more separable than states in lower dimensions, and also because the linear readout can use more coefficients to satisfy the required mappings from states to outputs.

5.5 Discussion

The primary point of interest in the mechanism for memory storage and recall using reservoirs composed of CTRNNs, is that the memory storage and retrieval of an episode depend on three sets of connections. The first set is the connections inside the reservoir, that, in line with reservoir computing practice [HS12], remain fixed, and are specified by the $\mathbf{W}^{Res}$ matrix. These connections determine the magnitude of interaction between connected neurons and hence the dynamic behaviour of the network in the absence of any input or feedback. From what we have seen, the scaling of these connections, specified by the gain g, plays a vital role for the performance of the reservoir in memorising and recalling temporal memories. In particular, unstable networks with $g \geq 1$ outperformed stable ones, $g < 1$, because their dynamics respond by producing more complex activity, which gives promotes greater separation, a property that is vital for memorising and recalling different episodes with good quality.

The second set is composed of the connections specified by the $\mathbf{W}^{Out}$ matrix that is trained to map the trajectory of reservoir activity to the pixel channel values of the visual image. This set of weights contributes only in the reconstruction of the high-dimensional target, which is closely linked to the approximation property. Moreover, using the reservoir activity to store and reconstruct memories, as opposed to utilizing static attracting states of the system (e.g. as in Hopfield networks [Hop82]), is in agreement with other recent theories of neural computation [MNM02].

The last set of connections includes all the connections that are part of the feedback loop. These include the subset of weights in $\mathbf{W}^{Out}$ that are used to reconstruct the outputs that are fed

back to the reservoir, and all feedback weights in $\mathbf{W}^{Fb}$. Either using a subset of the outputs, or training another set of readout weights to reconstruct a different stabilising signal, is equivalent to training the reservoir to drive its own activity in a way that helps in memorising and recalling episodes. In this way, the reservoir is able to autonomously reproduce appropriate dynamics, relevant to the memory it has stored and is trying to reproduce, by only making use of a distinct initial pulse, which acts as a "recall cue". Since training the feedback loop is sufficient for the reservoir to learn these reproducible dynamics, we can conclude that training this feedback loop is necessary and sufficient for memorising and recalling episodes. By this we mean that the most important part of training is learning rich, yet reproducible dynamics which can be used by the readout to learn the appropriate mappings.

This insight has recently been the focus of a new learning approach, that is an extension of FORCE learning, called full-FORCE [DCR$^+$18]. This approach aims at explicitly enforcing dynamics in the network that are useful for the task at hand. To achieve this, it uses a second, target-generating network, that receives the input signal, the target output and, optionally, additional input "hint" signals about the task. All these signals are relevant for the task, and hence the dynamics of this target-generating network are heavily driven by them. The task-performing network is then trained, using FORCE learning, by modifying the connections, $\mathbf{W}^{Res}$, inside the network, such that the activity of the task-performing network matches the activity of the target-generating network. Because this modifies the connections in the network, it no longer follows the paradigm of reservoir computing, but the resulting task-performing network dynamics are more suitable for successfully training on the task. The readout weights in $\mathbf{W}^{Out}$ can then be trained using vanilla FORCE to learn a suitable mapping from the network state to the target output. Importantly, the feedback connections are no longer necessary if full-FORCE is used, because during the initial training phase, when $\mathbf{W}^{Res}$ is adapted, the necessary stability is implicitly imposed in the network. In this way, the authors of the paper [DCR$^+$18] showed that by modifying the weights $\mathbf{W}^{Res}$, the dynamics can be adapted to become more stable and relevant for the task, and hence they could use smaller networks to successfully perform task, that would otherwise require larger networks.

The conclusions arrived at by the authors of the full-FORCE paper [DCR$^+$18] are compatible

with the results presented in this chapter in mainly two ways. The first way is through the use, in this work, of feedback connections from a subset of the output back to the reservoir, which are equivalent to providing a "hint" signal to the network, to "shape" and stabilise its activity during learning. This is especially relevant since FORCE learning was used, which ensures that the network output is very close to the target from early-on during training, resembling the "hint" signal from [DCR+18]. The second way that the two are compatible, is because a subset of the trainable readout weights, $\mathbf{W}_{Fb}^{Out}$, were part of the feedback loop. This means that in addition to the neuron interactions defined by $\mathbf{W}^{Res}$, neurons also interact through the matrix product of $\mathbf{W}^{Fb}$ and $\mathbf{W}_{Fb}^{Out}$. Hence, by modifying $\mathbf{W}^{Out}$, the total network connectivity matrix, $\hat{\mathbf{W}}^{Res} = \mathbf{W}^{Res} + \mathbf{W}^{Fb}\mathbf{W}_{Fb}^{Out}$, is also modified indirectly. Since permanently modifying recurrent connections can lead to self-generated dynamics that are relevant for the task, the full-FORCE approach naturally extends current approach and proposes an explicit modification of these connections in an offline manner, in order to fully exploit any useful information that can be provided. Nonetheless, the method used in the experiments presented here is more biologically plausible, as it can be implemented in an online manner, and using only the target as a "hint" signal, thus overcoming the need for any additional relevant signals.

5.5.1 Data compression using reservoirs

Interestingly, for most experiments, the quality of reconstruction did not drop as the number of episodes increased from one to four, indicating that the number of frames to be memorised and recalled is well within the capacity of the networks used. We would expect the opposite to be true, as memorising more episodes requires the linear readout to learn a larger number of mappings from states to pixel values. We would also expect that the more the mappings that need to be successfully learned, the higher the probability that later weight updates will cause changes in the linear readout weights, $\mathbf{W}^{Out}$, and the feedback weights, $\mathbf{W}^{Fb}$, that will lead to the forgetting of previously learned mappings. The effect of this would be that the reservoir has "forgotten" earlier memories. From the results obtained here though, we see that the number of episodes learned, as well as the number of frames per episode, fall well within the capacity of

the reservoirs that have been used.

In general, using a large number of parameters to learn a small number of mappings is an easy task, but is commonly avoided, as it can lead to over-fitting, where the trained model is able to learn the training dataset very well, but performs poorly when tested on cases that were not part of the training set. Of course, in the task used in this work, the aim *was* to memorise the training dataset and be able to reproduce it, so the task required the network to over-fit to *all* the episodes in the dataset. An interesting question then arises: what is the greatest number of samples – i.e. frames – that the reservoir can learn successfully? Ideally, the number of samples learned should be as high as possible. This would indicate that the reservoir is utilising its dynamics to efficiently encode the episodes it is learning. In fact, if the reservoir is able to memorise and reproduce a number of frames that is above a certain threshold, the reservoir must be using its dynamics to encode the episodes in a compressed representation. A good way to check whether the reservoir is compressing the information in the episodes in its dynamics is through the Compression Ratio

$$\text{Compression Ratio} = \frac{\text{Memory requirement of episode [bits]}}{\text{Memory requirement of reservoir [bits]}}, \tag{5.7}$$

which gives a ratio greater than one if compression is taking place.

Even though the images used here require only one byte per pixel channel, we will assume that for any general "episode", each "channel" is represented by floats, just like each of the reservoir weights. This allows us to compare directly the number of weights that need to be stored to define the reservoir, and the number of target outputs that need to be learned. Also, to make it simple, we can assume that all target values are part of a single episode and that there exists a single feedback signal, which allows us to rewrite Eq. 5.7 above to

$$\text{Compression Ratio} = \frac{N_{out} \cdot N_f}{N \cdot (1+p) + N^2 \cdot p_{res} + N \cdot N_{out}}, \tag{5.8}$$

where N is the number of neurons in the reservoir, p is the probability of connection of the

feedback to the reservoir, p_{res} is the probability of connection between the neurons of the reservoir, N_{out} is the total number of output channels of the target output and N_f is the total number of "frames" in the episode. Hence, the numerator represents the total memory requirement storing all the information in an episode, while the denominator represents the memory needed to store the weights that define the reservoir tasked with memorising and recalling the episode. The probabilities of connection of the feedback, p, and reservoir connectivity, p_{res} can be included if a memory-efficient, sparse representation of the weights can be used, otherwise they are dropped. Since compression is taking place when the ratio is greater than one, we can rearrange Eq. 5.8, to yield the following expression

$$N_f > \frac{N \cdot p + N^2 \cdot p_{res} + N \cdot N_{out}}{N_{out}}. \tag{5.9}$$

Moreover, if we assume that the number of output "channels", N_{out}, is much greater than N, Eq. 5.9 further can be further simplified to

$$N_f > N. \tag{5.10}$$

Hence, if the reservoir is able to memorise an episode consisting of more "frames" than neurons in the reservoir, then the reservoir must be compressing information.

So, let us now calculate the compression ratio of the reservoirs used here. Let us start by choosing an optimistic scenario of $N = 600$, $p = p_{res} = 0.1$, four episodes, hence the total number of input weights including feedback connections is $(4 + 2p)N$, $N_{out} = 255 \times 255 \times 3 = 195075$ and $N_f = 643 + 692 + 486 + 416 = 2237$. A further assumption is that the pixel channel values are of type 64-bit float in all episodes, and hence the terms are directly comparable. Hence, using Eq. 5.8,

$$\begin{aligned}\text{Compression Ratio} &= \frac{195075 \cdot 2237}{(4 + 2 \cdot 0.1) \cdot 600 + 0.1 \cdot 600^2 + 600 \cdot 195075} \\ &= \frac{436382775}{117143520} = 3.727 \text{ (3 d.p.).}\end{aligned} \tag{5.11}$$

We can see that a reservoir of $N = 600$ has a compression ratio above 1, indicating that the reservoirs are performing some form of compression, under the assumptions used. The corresponding compression ratio of reservoirs with $N = 1600$ becomes 1.397 (3 d.p), but also comes with more robustness and better quality. Therefore, and as expected, increasing the reservoir size leads to an increased robustness and quality of reconstruction, but comes with a higher memory requirement and computational cost.

Compressing data into the reservoir – as well as reconstructing it – is computationally very expensive. Moreover, the shortcomings of using reservoirs for compressing temporal data become more obvious if we compare them to the compression ratios achieved with conventional video encoders, for example the `MPEG` encoder. In particular, the typical compression ratio range for recent `MPEG` encoders is $20 - 50$, without much loss in quality and without the high computational cost that comes with training reservoirs. Nonetheless, we should be aware that reservoir computing was *not designed* to specifically perform compression and hence, this comparison with methods that *were* specifically designed to compress videos should not discourage us from further investigating reservoir computing and trying to integrate it into other applications.

5.5.2 Relation to other work on episodic memory

In this work, it was claimed that the reservoir was used to perform a task that is related to episodic-like memory. The interpretation of episodic-like memory employed was that of storing and recalling temporal high-dimensional information, similar to saving a video in memory for later replay. Intuitively, this is related to the ability of humans to remember "informationally rich" episodes from their everyday interactions; which can include how the scene looked, what conversations took place, where the cat was at the time of the episode, and so on. All this

information might not have been necessary for achieving a particular goal at the time of the episode, but the recall of the episode can be used retrospectively to access specific information that is necessary for another goal, at a later time, i.e., trying to remember when was the last time I saw the cat, who is nowhere to be found now.

The value of such a cognitive skill, is that it creates a long-lived buffer, where information that might be relevant for achieving a goal in the near future is stored and easily accessed. Moreover, very novel and surprising situations can also be stored in episodic memory to help take better decisions the next time a similar situation is encountered. For example, remembering that this particular cat, unlike other cats, got very upset and aggressive when I touched her ear, will help me avoid an unpleasant interaction the next time I encounter her. To remember this, it is very useful to remember that it was *this particular cat*, and that it was *the touching of her ear* that got me in an unpleasant situation. Episodic memory allows both these necessary pieces of information to be kept in a coherent "narrative" of a previously experienced episode, which can be recalled upon a future encounter with this cat. The work carried out here suggests that, were the brain to follow reservoir computing principles, it could use its huge number of synapses as readouts, to decode – from the brain's current activity – the high-dimensional sensory information it has memorised during a relevant past experience. This suggestion emerges because we saw that CTRNNs can generate stable and reproducible activity, that supports the memorisation and recall of memories in the aforementioned way, while other studies showed that the dynamic activity of CTRNNs is compatible with cortical dynamics in many ways [MSSN13, SCKS15, MDS16, WNHJ18].

The notion of episodic-like memory has already been integrated in agents trained to explore virtual worlds or play computer games deep reinforcement learning [Lin93, MKS$^+$15] and showed considerable improvement. This has been termed *experience replay* [MMO95, Lin93, OPBDC10], and consists of keeping a buffer of past state-action-new state-reward instances, from which samples are randomly chosen and replayed throughout training. The aim is to remove correlations and bias in the new samples, which are present when the agent explores only a small portion of the environment for some time. By replaying "memories" from this buffer, more diverse samples, which were experienced in the past, are introduced during learning to reduce the

bias and speed up the process. Another variant, called *prioritised experience replay* [SQAS15], prioritises the replayed experiences based on how "surprising" they were, as measured by the Temporal Difference (TD) method. This approach is closely linked to theories about the role of the hippocampus in episodic and semantic memory [KLH06, KHM16], whereby episodic memory acts as a buffer where unexpected events are stored, and replayed from, until the relevant knowledge is "distilled" in semantic memory, which is updated more slowly.

Recent work [SMS15] on action classification and future frame prediction tasks in videos, has illustrated that networks of long short-term memory (LSTM) units are also able to perform a similar, but not identical role, to the one explored in this chapter. The authors trained LSTM networks to encode, in a latent space, representations of frame sequences. These representations could be then used by another set of LSTM units to reconstruct the original input sequence and predict future frames. The LSTM network showed qualitatively good performance on reconstructing 32×32 RGB frames of human actions from the UCF-101 dataset [SZS12]. In addition, the learned hidden representations proved useful for classifying the actions shown in the frames, through the use of an additional classification layer on top of the latent space. The main difference in the task and performance of the LSTM networks compared to CTRNNs used in this thesis was that LTSMs were trained only on sequences of 16 input and 13 output frames, using $4,096$ LSTM units and $9,500$ training videos. Qualitative results presented in the paper show that the predicted frames became blurry very soon, thus indicating a better memorisation and reconstruction performance in CTRNNs, which were able to memorising sequences of hundreds of frames. Nonetheless, the focus of [SMS15], was on learning useful latent representations for action recognition, something that the work presented in this thesis does not address. On the contrary, CTRNNs are much better in memorising few, long episodes for later reconstruction, as they are more data efficient but suffer from lower generalisation capabilities compared to LSTMs.

In a different line of research, more biologically realistic models, inspired by the brain regions involved in episodic memory, have been proposed [EH12, EH14, BBU$^+$18], that have a connectivity structure inspired by the brain. The aim of these models is to use similar structure and interactions to those found in regions of the hippocampus and prefrontal cortex, and in particular

grid cells, for reaching rewarding states in an environment. Grid cells are believed to utilise their connectivity to create hexagonal, grid-like representations of locations of the environment, which can be used for planning towards reaching a goal. Usually, the input to such models is a low dimensional signal, which provides pre-processed information about the first-person view of the environment and the velocity of the agent, in order to update its representation about the agent's head direction and position in the environment, as well as keep track of where the reward is. Simple agents using these models of grid cells, were shown to successfully learn to navigate towards goals in their environment. Moreover, in [BBU$^+$18], it was shown that grid-like representations emerge naturally (i.e. without explicitly designed into the model) if agents controlled by artificial neural networks are trained to path-integrate within a square area in order to navigate to a goal.

In sum, the field of episodic-like memory for artificial agents has been gaining much attention in recent years, as it can provide agents with a "buffer" of relevant, informative, and specific experiences. Even though there has been significant progress in incorporating such types of memory in current state-of-the-art models of reinforcement learning, and a huge effort by the neuroscientific community is currently being undertaken to elucidate the intricate neural pathways supporting it, there is still room for many more developments in this direction.

Chapter 6

Identifying information broadcast in complex networks with continuous-time firing-rate neurons

6.1 Precis

This last chapter suggests a simple, yet theoretically well-founded methodology, that can be used for investigating how complex dynamical systems with modular, small-world connectivity, are able to use their dynamic activity to broadcast module-specific information to other modules through a connective core. This methodology is demonstrated using CTRNNs with modular, small-world connectivity, where an external stimulus is injected to one of the modules. To measure how information about this stimulus is shared among the other modules of the network, via "information sharing channels", two measures from information theory are used: mutual information and transfer entropy. Specifically, the time-resolved version of the measures is used to measure information broadcast through the network's connective core, to provide additional computational evidence for the plausibility of the connective core hypothesis[1].

[1]The work presented in this chapter was developed with the help of Pedro Mediano and presented in the conference abstract "Identifying information broadcast in complex networks" [NMS17a]. Pedro Mediano developed the estimators in the JIDT (v1.4) package [Liz14] used in this work.

6.2 Introduction

The Connective Core Hypothesis [Sha12] attempts to explain cognitive integration in the human brain – a highly parallelised information processing system. This hypothesis, which is closely related to the Global Workspace Theory (GWT) [Baa93], rests on a view of how the information-bearing dynamics of different brain regions can interact to produce a single unified behaviour via competition for access to, and control of the *connective core*: a common set of neural hubs that are well-connected to each other and other brain areas. By gaining control of the connective core, modules can form coalitions, influence other modules, and transmit their information to decision-making brain regions.

Empirical evidence supporting this hypothesis comes from recent work that shows that cortical networks exhibit: (i) modular organisation, (ii) multiple parallel information processing paths, and (iii) the presence of highly connected hubs [BB06, ZLZK11, Spo13, DCKM14], which make the brain a modular, small-world network. More formally, small-world networks [WS98], are networks with a high mean clustering coefficient and low mean path length, relative to a randomly connected network with the same number of nodes and edges. The clustering coefficient of a node denotes the fraction of all possible edges between the neighbours of that node that are actually realised, while the path length between any two nodes is the minimum number of edges that needs to be traversed to go from one node to the other. A different way of characterising networks is through their modularity. A modular network is a network whose nodes can be partitioned into subsets (or modules), such that the connectivity within each subset is high, while the connectivity between the different subsets is low. It can also be the case that modular small-world networks have *connector hubs*, sets of nodes that are well connected with other nodes, both from the same module and other modules, and provide a short path between any pair of nodes in the network [vdHS13].

Interestingly, networks with small-world connectivity, present in systems like the human brain [You93] or man-made communication systems [LM01], are highly "efficient", in the sense that there is high global and local connectivity in the network, which promotes efficient information sharing and fault tolerance within the network [LM01]. It is also interesting to note that Echo

State Networks with small-world connectivity have recently been shown to have a better memory capacity compared to randomly connected ones [KPA19]. The authors of the work claim that such connectivity promotes information propagation in the network and enhances its echo state property.

Whether by using the activity of the biological brain [DC11], or through computer simulations [Sha10b, WS12], current research methods try to assess the extent to which the above hypotheses are plausible, by examining the co-activation, synchronisation, and correlation between the activity of cortical regions in the primate brain, or modules in a simulated network. The Temporal Generalisation Method by King and Dehaene [KD14] on the other hand, attempts to assess whether patterns of neuronal activation in the brain, at different moments in time, can be successfully used to decode the presence of a stimulus, or other information relevant to the task. Examples of information that the authors demonstrated to be decodable from the neural activity were the presence (or absence) and position of the stimulus, the subjects' response, and their assessment of whether they responded correctly or erroneously to the stimulus. To achieve this, they trained a classifier on pairs of neural activation patterns and target labels, in a supervised manner. The trained classified was then used to assess whether new instances of neural activity samples, that were not in the training set, could successfully decode the presence or absence of the task-relevant information. The decoding performance of the classifiers, trained and tested at different points in time, was then used to identify the moments during the experiment that information about the stimulus was present in the neural activity.

These established methods were well received by the research community, as they could provide evidence for useful insight into information-related brain activity. Their biggest shortcomings is that they either incorporate strong model assumptions, for example a linear form for correlation or a specific measure for synchronisation, or they make use of a classifier, which is not very informative about the underlying correlations that are present in the activity of neurons. In this chapter, a simple computational experiment is presented, using CTRNNs, that aims to provide an alternative approach for shining some more light on how information broadcast in complex networks can be quantified and visualised. This experiment does not attempt to answer all questions about information integration in the brain; rather, it is a simple study

that aims to illustrate how CTRNNs can be set-up with a modular, small-world connectivity, to produce interesting dynamics that can be analysed within the framework of information theory, and produce intuitive visualisations of emergent information broadcast channels in the network. The basic assumption employed here, is that information is encoded at the level of the neural activities and that interactions between the modules constitute the means through which modules can broadcast stimulus-related information to other regions in the network.

To this end, the work in this chapter uses the time-resolved [GHWR$^+$15] version of mutual information to assess how information encoded in the neural dynamics of the module receiving the stimulus is broadcast through the CTRNN's connective core. In addition, the time-resolved version of transfer entropy is used to identify the directionality of the interaction responsible for the information broadcast. Both of these measures are temporal in nature, which allows us to create a temporal analysis of the spread of information and the interactions in the network. Finally, their main strengths over other widely used methods for calculating correlation and causal interaction is that they impose fewer assumptions about the statistics of neural activity and can hence identify all non-linear relationships between the activities of the network's neurons. Their main limitation, which has hindered their widespread application so far, is that they require a large number of data points to be estimated. Fortunately, such large amounts of data can be generated through a computer simulation, and hence we will now see how these measures can be applied on CTRNNs.

6.3 Methods

6.3.1 Network

Continuous-time recurrent neural networks, of the form shown in Eq. 6.1 below, were used to examine how information is broadcast in modular, small-world networks.

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} W_{ij}^{Res} \tanh(x_j) + \sum_{k=1}^{5} W_{ik}^{In} s_k \qquad i = 1, 2, ..., N. \quad (6.1)$$

As before, N is the number of neurons in the network, x_i represents the hidden state of neuron i, $\mathbf{s}$ in an external stimulus, $\mathbf{W}^{Res}$ is the network connectivity matrix and $\mathbf{W}^{In}$ the matrix defining all input weights. Unlike the networks in the previous chapters, all of which had a random connectivity, the network used here had a modular, small-world connectivity. Specifically, a single network consisting of 5 modules, each with 200 neurons, was constructed. Intra-modular connectivity was random and set at 10%, with weights drawn independently from $\mathcal{N}(0, \frac{1.6^2}{20})$ and no self-connections. Importantly, there were no inter-modular connections, other than through the connective core of 10 neurons. Two neurons from each module were then selected at random to be part of the fully-connected core. External input connections to the modules were defined in the $N \times 5$ matrix $\mathbf{W}^{In}$. Each column had non-zero entries, independently drawn from $\mathcal{N}(0, 20^2)$, only at the rows for its corresponding module's neurons. This high variance of the input weights was chosen such that the effect of the brief input stimulus pulses on the network dynamics was amplified. None of the connective core neurons received external input. This network connectivity (Fig. 6.1) resulted in a Small-World Index of 2.81. This was obtained after converting $\mathbf{W}^{Res}$ to a binary adjacency matrix, followed by the use of the relevant Brain Connectivity ToolBox [RS10] functions for binary directed graphs, implemented in `MATLAB`.

The Small-World Index, σ_G, of a graph G, is defined as the ratio [HG08]

$$\sigma_G = \frac{\gamma_G / \gamma_{rand}}{\lambda_G / \lambda_{rand}}, \quad (6.2a)$$

where γ_G and λ_G are respectively the average clustering coefficient and average path length between the nodes of the graph, while γ_{rand} and λ_{rand} are the equivalent quantities for a randomly connected graph with the same number of nodes and edges as G. Interestingly, for the case of

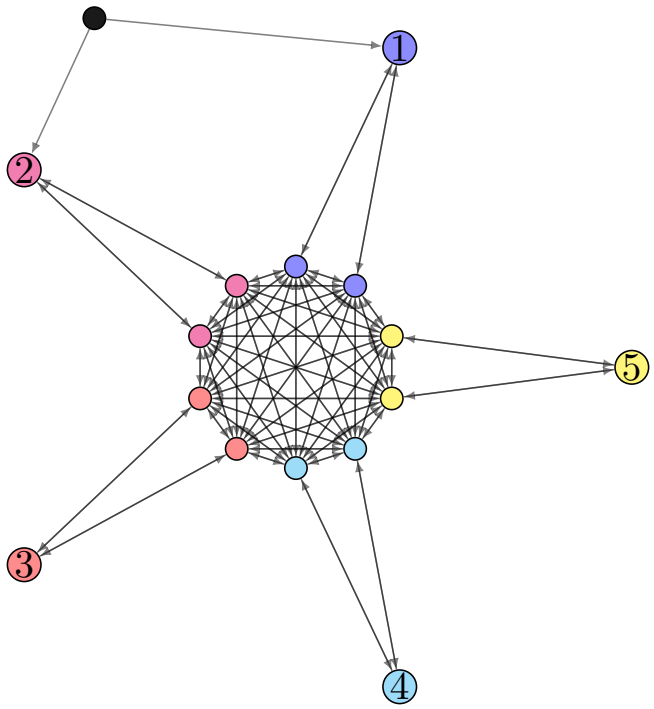


Figure 6.1: Structural connectivity of the network. Outer nodes represent the modules of 200 neurons, while inner nodes represent the connective core neurons. Nodes with the same colour belong to the same module. The black, top-left node represents the stimulus. Lines with arrows between nodes represent direct connections between the nodes, with the arrows showing the directionality of the connection.

random graphs with N nodes and an average degree κ, which is the ratio between the total number of edges and total number of nodes in the graph, $\gamma_{rand} \approx {}^{\kappa}/_{N}$ and $\lambda_{rand} \approx {}^{log(N)}/_{log(\kappa)}$ [RS10].

Each stimulus was constructed by randomly placing 50 brief pulses within a window of $3,000$ time steps. Each pulse was modelled by a Gaussian, with width 20 time steps and maximum amplitude of 2. The same network was randomly initialised 400 times, each run for $4,000$ time steps with no input, then for $3,000$ time steps with 50 pulses to module 1 and finally for $3,000$ time steps with 50 pulses to module 2. The first $1,000$ time steps were discarded as transients and the analysis was performed on the remaining $9,000$ time steps.

6.3.2 Measuring information broadcast

The aim in this chapter is to estimate non-linear correlations between pairs of non-stationary time series of neuronal activity in the network. Information theory was chosen as a suitable,

mathematically well-founded framework to measure these correlations. In particular, two well-known and widely used measures from information theory were chosen: (a) mutual information (MI) in order to quantify the instantaneous shared information between the different parts of the network, and (b) transfer entropy (TE) [Sch00], which allows the quantification of directed information flow. Mutual information and transfer entropy between two discrete, random variables X_1 and X_2, are respectively given by the equations [CT06]

$$\mathrm{MI}(X_1, X_2) = I(X_1; X_2) = \sum_{x_1, x_2} p(x_1, x_2) \log \left(\frac{p(x_1, x_2)}{p(x_1) p(x_2)} \right), \tag{6.3a}$$

$$\mathrm{TE}(X_1 \leftarrow X_2) = T_{X_1 \leftarrow X_2} = \sum_{x_1, x_1', x_2} p(x_1, x_1', x_2) \log \left(\frac{p(x_1' | x_1, x_2)}{p(x_1' | x_1)} \right), \tag{6.3b}$$

where x_i denotes the realisation of the random variable X_i and x_i' denotes the future of X_i that is not present in x_i. Moreover, $p(x_1, x_2)$, $p(x_1|x_2)$ and $p(x_1)$ are respectively the joint, conditional, and marginal probability densities. Note that the summation can be replaced by the integral for the case of continuous random variables, and that the measure is symmetric, i.e. $I(X_1; X_2) = I(X_2; X_1)$, and only equal to zero if the two variables are statistically independent, i.e. $p(x_1, x_2) = p(x_1)p(x_2)$.

Unlike other quantities that can be used to characterise correlations between two random variables, for example linear correlation, or in fact any quantity that makes assumptions about the form of the correlation between the two variables, non-parametric mutual information makes no underlying assumptions about the form of the correlation, and hence quantifies all, both linear and non-linear, correlations between pairs of random variables. Hence, this is a more "holistic" correlation measure, especially when estimated using the non-parametric Kraskov-Stögbauer-Grassberger (KSG) method [KSG04], which is based on k-nearest neighbour distances. According to the authors of the paper proposing this method, it can overcome problems that are common when estimating probability densities with bin-based methods, such as bias in the estimation, lack of the possibility for adaptive discretisation, and data inefficiency.

Beyond mutual information, which measures the correlation between two variables, transfer entropy [Sch00] was also used to measure how one set of neurons affected the future evolution of another. Transfer entropy is a suitable measure for this, as it is a directed and time-asymmetric measure that quantifies the amount of information gained about the future of a variable X_1, by knowing the past state of another variable X_2, over and above the information provided by the past of X_1. Therefore, the use of transfer entropy makes intuitive sense if we are interested in characterising the effect of one set of neurons on another, as it allows us to predict the influence exerted.

Another measure, that is closely related to transfer entropy is Granger Causality [Gra69], which uses the same notion – i.e. increase in the predictability of the future state of one variable, when information about another variable is included – as transfer entropy. Interestingly, Granger Causality has been shown to be equivalent to transfer entropy if the variables are assumed to follow a normal distribution [BBS09]. Moreover, adaptations of Granger Causality have been proposed that consider causal relations in the spectral – rather than the time – domain [KDTB01]. Granger Causality has been preferred when studying causal relations in the interactions between different brain regions, [KDTB01, DRD08, DSA$^+$09] because of its higher data efficiency compared to transfer entropy, due to the normality assumptions it entails. Nonetheless, the normality assumption is not always ideal for dynamics of nonlinear systems, and hence, the use of transfer entropy based on the non-parametric k-nearest neighbour based method of Kraskov, Stögbauer and Grassberger [KSG04] was preferred for this work. Moreover, an alternative formulation of transfer entropy was used, based on the conditional mutual information, which makes its estimation easier:

$$T_{X_1 \leftarrow X_2} = I(X_1'; X_2 | X_1) \ . \tag{6.4a}$$

Since the aim here is to characterise the temporal dynamics of information flow in the network, the conventional method of measuring mutual information and transfer entropy, where the

measures are averaged over the whole simulation, is not very useful. This method would provide a single estimate for the whole simulation, without revealing how the information is propagated through the network over time. Instead, to generate a time series of information broadcast, the time-resolved form of these measures was adopted, following Gómez *et al.* [GHWR$^+$15]. The way that a large enough number of data points was obtained for the estimation was not by providing a longer time series for each of the simulations, but by repeating the simulations multiple times, subject to random initialisations.

In this set of experiments, the following method was used: First, a sliding window of width $\Delta_w = 5$ was defined, where the samples inside it were used to estimate the relevant probability densities inside the logarithms of Eq. 6.3, and to evaluate the measures at each point in time. Then this procedure was repeated by sliding the window by a single time step, hence yielding a time series of mutual information and transfer entropy values. The specific window size was chosen such that (a) the distribution of data inside the window was approximately stationary, and (b) the window size was less than the overall duration of a single external pulse. Moreover, and as already mentioned, the key point of this method is that log-probabilities were not averaged across time but across trials, which makes the measure localised in time and allows us to obtain a temporal resolution of the information broadcast in the network, through the connective core.

The evaluation of the measures was carried out using the JIDT (v1.4) package [Liz14], which implements these measures using the KSG method. Specifically, the classes `measures.continuous.kraskov.MutualInfoCalculatorMultiVariateKraskov1` and `measures.continuous.kraskov.TransferEntropyCalculatorKraskov` were used from the package, with each trial added using the `addObservations()` method, followed by averaging the local estimates for each trial obtained using the `computeLocalOfPreviousObservations()` method. From the value obtained, the average of 5 surrogates was subtracted as a baseline, in order to remove any random correlations present in the data.

Finally, the measures were calculated for the following cases: (a) within each module, by averaging the measure for 20 randomly selected pairs of the module's neurons, (b) between each

module (average of 20 randomly chosen neurons from each module) and every connective core neuron, (c) all pairs of connective core neurons, (d) the stimulus and all connective core neurons and (e) the stimulus and each module, by averaging over 20 randomly chosen neurons from each one. For all figures shown in the results section, the temporal mutual information, and transfer entropy, between the external stimulus and the rest of the network was only used in the calculations for the plots corresponding to the stimulus. Calculations for the other plots included only the shared information and influence within the network.

6.4 Results

Using the time-resolved measures made it possible to quantitatively assess the information sharing and broadcast in this modular small-world network, when the external stimulus perturbed one of the modules. Fig. 6.2 shows changes in the average instantaneous mutual information and transfer entropy between neurons of each of the modules. Connective core neurons were excluded when calculating the average.

In Fig. 6.2, we can see the effect of the stimulus on the average information shared and transferred between neurons of the same module. The first point of interest is that during the first period (time steps $3,000 - 4,000$), when there was no external stimulation, all modules showed a steady amount of both mutual information and transfer entropy amongst their neurons. This is in contrast to the other two periods, where the module receiving the external stimulation showed a considerable inter-modular variation in both measures. Moreover, the large increases in mutual information took place during the time steps that the stimulus was applied, and dropped over the time steps that it was removed. Similar patterns can be observed for the transfer entropy, but the magnitude of the increase, as well as the baseline value, were smaller for transfer entropy compared to those for mutual information. From these results, we can see that neurons within each module (a) share more information than the information they broadcast to each other, and (b) this shared information increases when the external stimulus is applied to that module, relative to when applied to other modules.

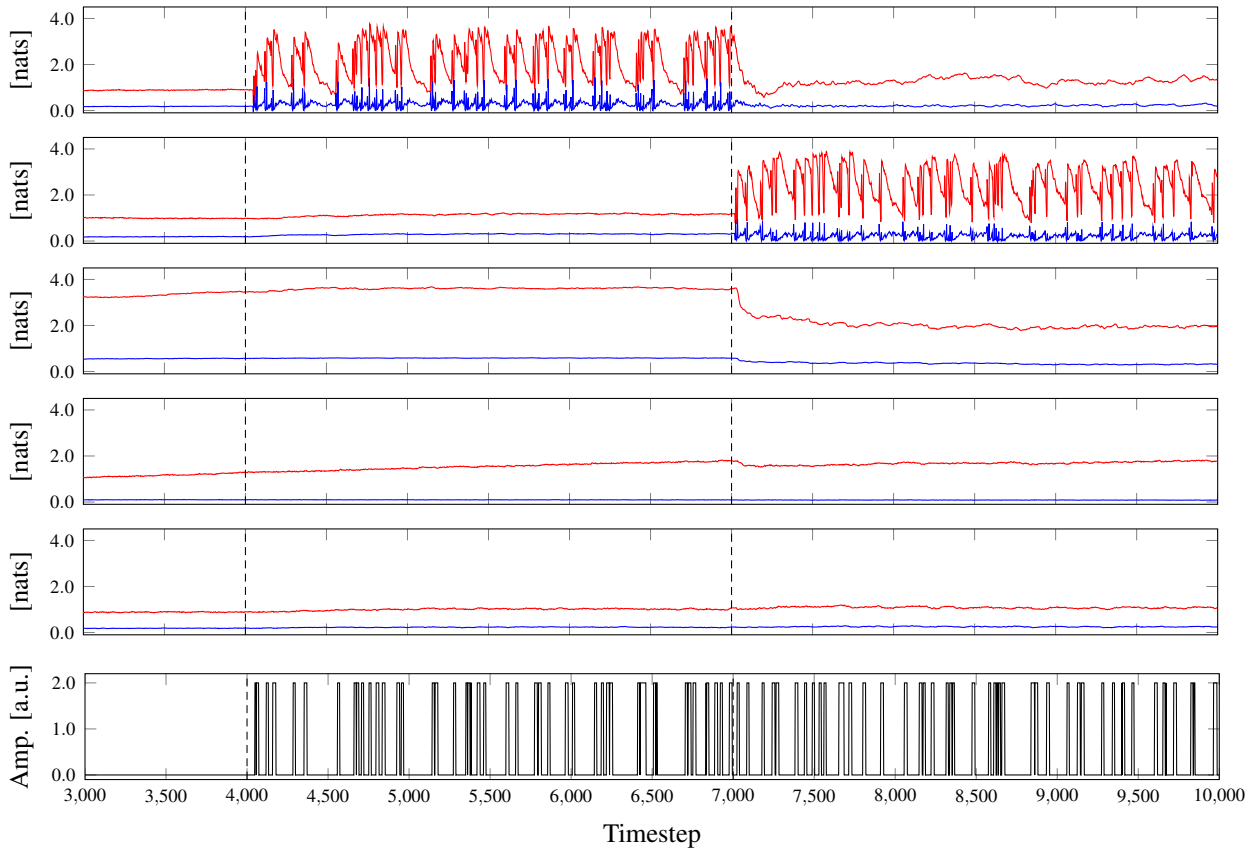


Figure 6.2: Temporal mutual information (red) and temporal transfer entropy (blue) within each of the 5 modules, calculated by averaging between 20 randomly chosen pairs of neurons from each module. From top to bottom, plots correspond to modules 1 to 5 (as labelled in Fig. 6.1), while the bottom plot corresponds to the amplitude of the stimulus. The first section (time steps $3,000 - 4,000$) corresponds to the last $1,000$ time steps of the stimulus-free period. The second section (time steps $4,000 - 7,000$) to the period when the stimulus was injected into module 1, and the third sections (time steps $7,000 - 10,000$) to the period when the external stimulus was injected into module 2. The differentiation of the three periods is denoted by vertical dashed lines.

Another interesting observation is that inter-modular mutual information for module 3 did not change during the period when the stimulus was applied in module 1, but dropped significantly, and remained low, when module 2 received external stimulation. This means that the external perturbation in module 2, was exerting such an effect on the activity of module 3, that resulted in a consistent decrease in the mutual information between module 3 neurons, throughout the duration of that period. Moreover, no significant change was observed for the two measures in any period for the cases of modules 4 and 5. This shows that this specific realisation of the network connectivity can support arbitrary influences in the network, for example the stimulation of one module leading to a decrease in the correlation between neuronal activations

in a single module, but not in others.

We can further assess how much information was shared between the different nodes in the network shown in Fig. 6.1, and how much influence each node exerted on the others, through temporal transfer entropy. Fig. 6.3 shows the average temporal mutual information between each node and all others in the network, whereas Fig. 6.4 shows the average temporal transfer entropy *from* each node *to* the rest, and Fig. 6.5 the average temporal transfer entropy *to* each node *from* all other.

Let us now look at how information about the stimulus emerged and disappeared in the network. The second plot from the bottom in Fig. 6.3, shows that the stimulus shared a non-zero amount of mutual information during the time that it was applied on the network, and dropped back to very close to zero during the intervals when its amplitude was zero. This makes intuitive sense, as it means that during the time that the stimulus was applied, the neurons' activations were causally affected by the changes it produced in the dynamics of the network. Moreover, and in agreement with our intuitive understanding, the corresponding plots in Figs. 6.4 & 6.5 show that during each pulse of the stimulus, the transfer entropy *from* the stimulus *to* the rest of the network considerably increased, while transfer entropy *from* the network *to* the stimulus remained very close to zero.

Let us now consider the neurons in the module receiving the stimulus. Their average mutual information with all other modules and connective core neurons fluctuated strongly during the period when they received external stimulation (first and third plots in Fig. 6.3). Notably, a drop in the mutual information between them and the rest of the network was observed at the time of the external pulses, followed by an increase during the inter-pulse intervals. This indicates that during the duration over which the stimulus is applied, the pairwise correlation of the neurons' activities drops, because neurons in the stimulus-driven network respond in a more independent way, whereas neural activations are more correlated in the autonomous network. This can be thought of as an external pulse "forcing" the activities of some of the neurons to "decouple" from the rest of the network, and when the pulse is gone, the influence from the neurons of rest of the network strengthens again.

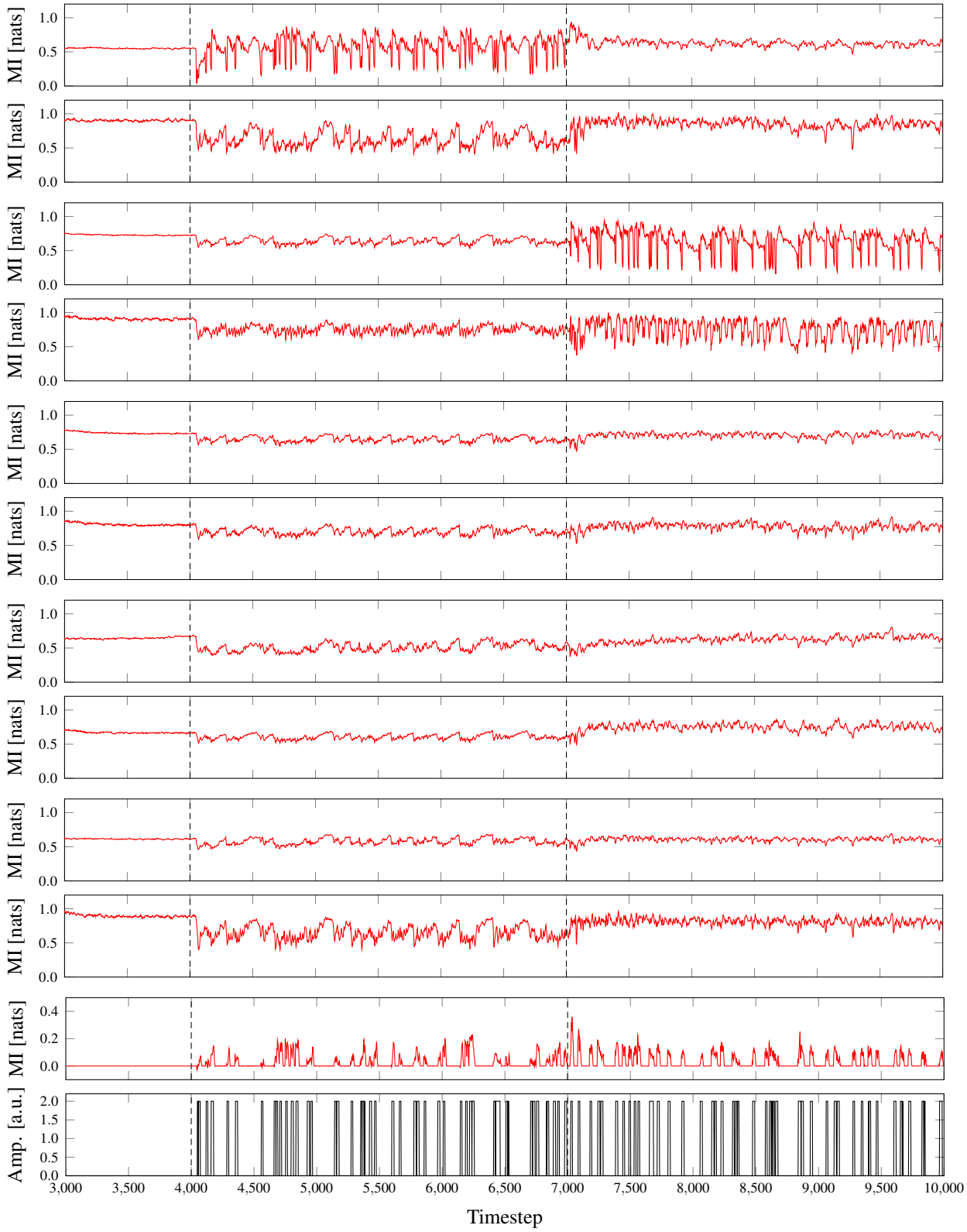


Figure 6.3: Average temporal mutual information between each node in the network (Fig. 6.1) and all other nodes. The top plot is for the neurons in module 1, followed by the plot for the average of the two connective core neurons of module 1. The plots for all other modules follow below in the same pattern of pairs. The last two plots are the average temporal mutual information between the stimulus and all other nodes shown in Fig. 6.1 (one but last), and the time series of the amplitude of the external stimulus (last one).

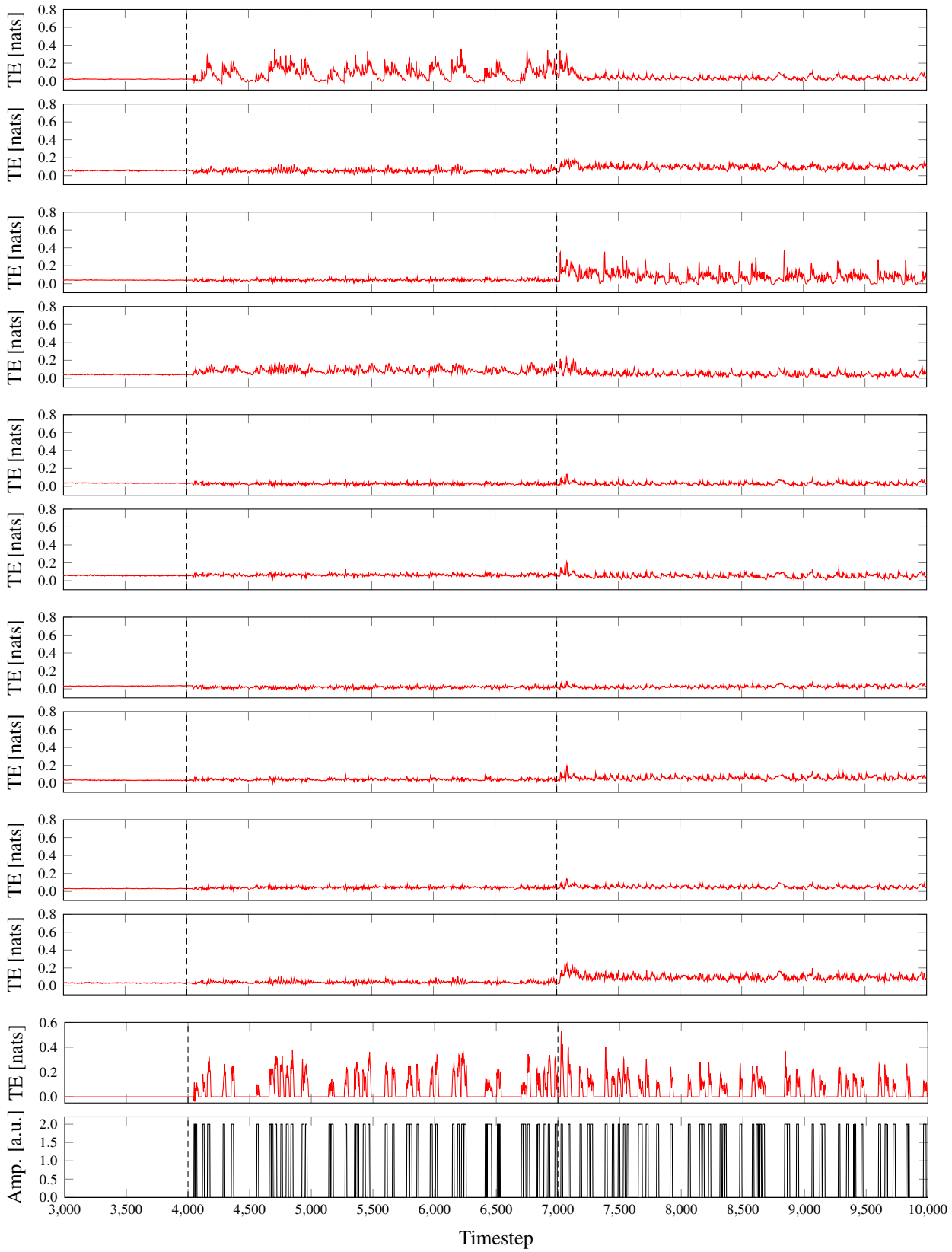


Figure 6.4: Average temporal transfer entropy *from* each node in the network (Fig. 6.1) *to* all other nodes. The top plot is for the neurons in module 1, followed by the plot for the average of the two connective core neurons of module 1. The plots for all other modules follow below in the same pattern of pairs. The last two plots are (one but last) the average temporal transfer entropy *from* the stimulus *to* all other nodes in the network (Fig. 6.1), and the time series of the amplitude of the external stimulus (last one).

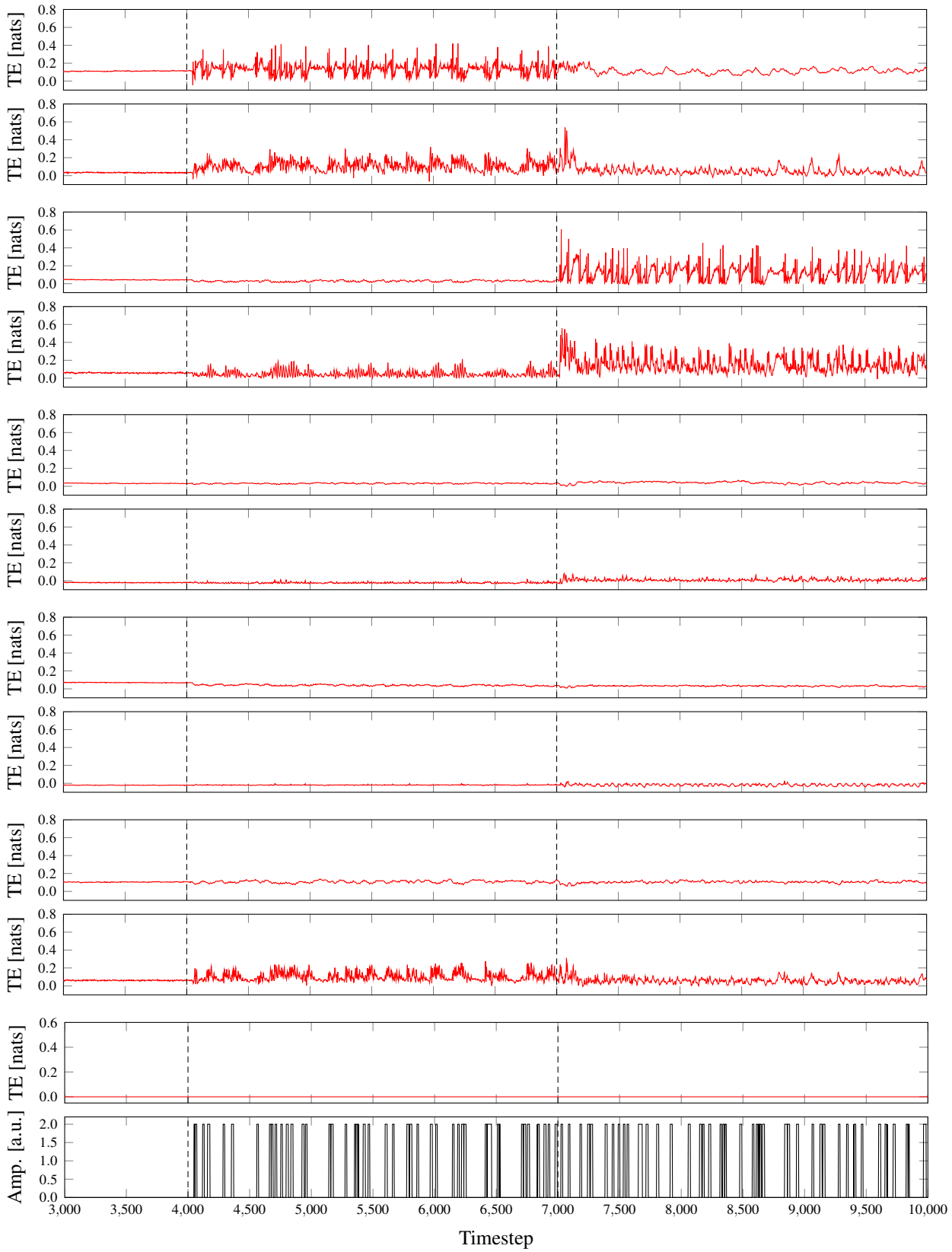
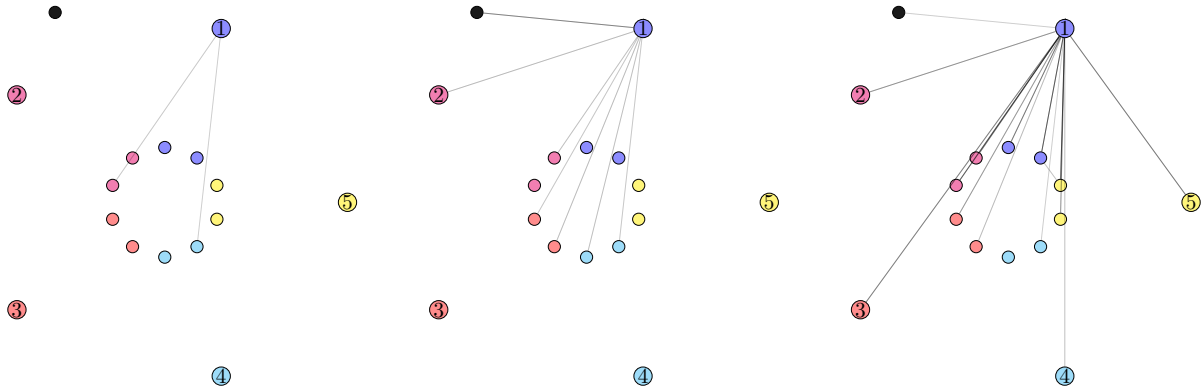


Figure 6.5: Average temporal transfer entropy *to* each node in the network (Fig. 6.1) *from* all other nodes. The top plot is for the neurons in module 1, followed by the plot for the average of the two connective core neurons of module 1. The plots for all other modules follow below in the same pattern of pairs. The last two plots are (one but last) the average temporal transfer entropy *to* the stimulus *from* all other nodes in the network (Fig. 6.1), and the time series of the amplitude of the external stimulus (last one).

Similarly strong fluctuations in mutual information can also be seen in the plots of their corresponding connective core neurons (second and fourth plots in Fig. 6.3). Moreover, milder fluctuations were present in the plots of the average mutual information of the other modules and their respective connective core neurons. These fluctuations were even smaller during the injection of the stimulus to module 2, compared to when it was injected to module 1. This effect is especially noticeable for the connective core neurons of module 5 (third plot from bottom in Fig. 6.3), indicating a possible stronger coupling between the connective core neurons of module 1 and 5.

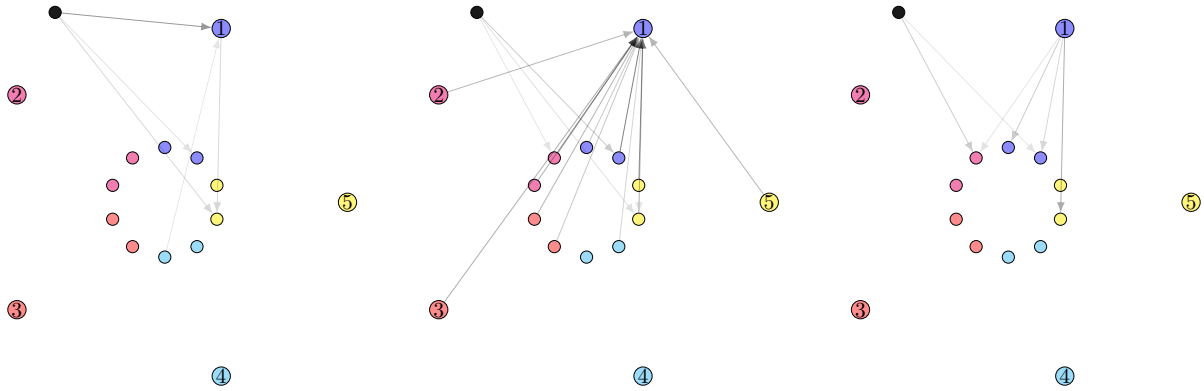
The directionality of these interactions can be seen from the temporal transfer entropy plots in Figs. 6.4 & 6.5. In Fig. 6.4, we can see that module 1 neurons show a considerable influence on the rest of the network during the period that they received the external stimulus. Interestingly, this increase completely coincided with the presence of the external pulse, unlike the case of temporal mutual information. Moreover, during the same period, most of the connective core neurons did not show a considerable change in the influence they exerted on the rest of the network, with the exception of connective core neurons of module 2 (instead of those of module 1), that compared to the rest of the network, demonstrated slightly higher fluctuations in their outwards temporal transfer entropy (fourth plot in Fig. 6.4). A similar phenomenon was also observed for the period during which module 2 received the external stimulation. During this period, a small, but consistent increase in the average outwards influence of connective core neurons of modules 1 and 5 can be seen (second & tenth plots in Fig. 6.4).

In Fig. 6.5, the influence of the rest of the network onto each of the nodes is shown. Surprisingly, the modules, and their respective connective core neurons, that were receiving external stimulation, showed an increased influence from the rest of the network during that period. This can, as before, be explained through the timing of the appearance of the peaks in transfer entropy and the external pulses. Since the peaks in the *inwards* transfer entropy appeared towards the final time steps of each of the external pulses, it means that the influence of the rest of the network increases, as the module's neurons are no longer coupled as strongly to the stimulus, but rather, the coupling with the rest of the network dominates.



(a) At perturbation time (b) Immediately after (c) Shortly after

Figure 6.6: Increases in mutual information between all pairs of nodes in the network (as shown in Fig. 6.1). The three plots represent different moments in time around the time when the stimulus was injected to module 1. Figure appeared in [NMS17a].



(a) At perturbation time (b) Immediately after (c) Shortly after

Figure 6.7: Increases in directional transfer entropy between all pairs of nodes in the network (as shown in Fig. 6.1). The directionality of the measure is denoted by the presence of arrow heads in each of the connections. The three plots represent different moments in time around the time when the stimulus was injected to module 1.

Full videos of the resulting analysis were produced and are available online at `https://goo.gl/SXJpjn`. In these videos, the instantaneous thickness of the edges between the nodes of the network indicates the value of each of the measures for each pair of nodes. The reader is encouraged to watch these videos, as they provide an intuitive and informative visualisation of how exactly information and influence propagate and reverberate in the network, as the external stimulation varies[2].

Example "snapshots" from the video version showing all influences in the network are presented in Figs. 6.6 & 6.7. Both figures show how the two measures increase at the onset, immediately after, and shortly after the stimulus. The difference between consecutive time steps was taken, and only the positive ones were kept. Bolder lines between each pair of nodes denotes a higher instantaneous increase of the measure.

6.5 Link with network dynamics

This section begins by looking at how the trajectory of the network activity and the stationary points changed during the experiment, depending on which module was stimulated. The procedure for finding the stationary points and the linearisation performed to extract the eigenvalues of each one was the same as the one described in Section 4.4.1. Also, the dimensionality reduction was performed on the network trajectory, followed by projecting the positions of the stationary points onto this reduced PC-space.

As can be seen from Fig. 6.8, the network activity during the three different periods is well separated in the reduced PC-space. During the initial period (shown in red), when no stimulation was provided, the network activity was concentrated in a region that is distinct from the other two periods. Moreover, injecting the stimulus to either module 1 (shown in blue) or module 2 (shown in magenta), also causes the network to shift in a distinct "activity subspace". This can be explained by the positions of the stationary points influencing the network during

[2]Four videos are present in the playlist. The two of them show the calculated value of each of the measures at each time step, while the other two show the positive increases in each of the measures between successive time steps. Decreases were set to zero, so as to emphasise the instantaneous increases in the shared information and the interplay of the influence between the nodes in the network.

the two periods when a stimulus was present. Stationary points in each period appear to be concentrated more closely around the network trajectory of the corresponding period, illustrating that depending on which module is stimulated, the network is operating under the influence of different stationary points, situated in different regions of the state space.

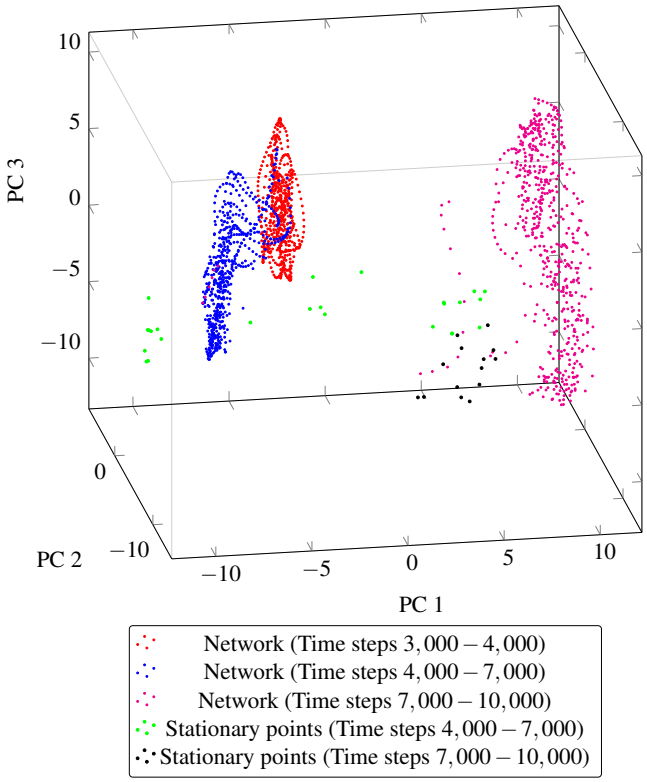


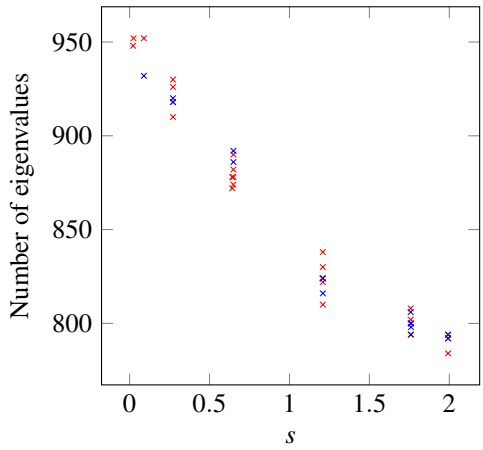
Figure 6.8: Activity of the network and stationary points, projected on the PC-space of the network trajectory. The network trajectory is shown in different colours for each of the three periods. The same holds for the stationary points of the network appearing during the two periods when the stimulus was injected in module 1 (time steps $4,000 - 7,000$) and module 2 (time steps $7,000 - 10,000$). All stationary points had principal eigenvalues with a negative real part, and were hence classified as stable stationary points.

This result is in agreement with the arguments made in Section 3.5.2, regarding the diversity in dynamic behaviour gained when different stimuli are injected to different subsets of the network. As we have seen here, this results in guiding the network activity in different sub-spaces of the state space, depending on which stimulus is "processed". Moreover, the modular small-world networks explored in this section gave rise to *different communication channels altogether*

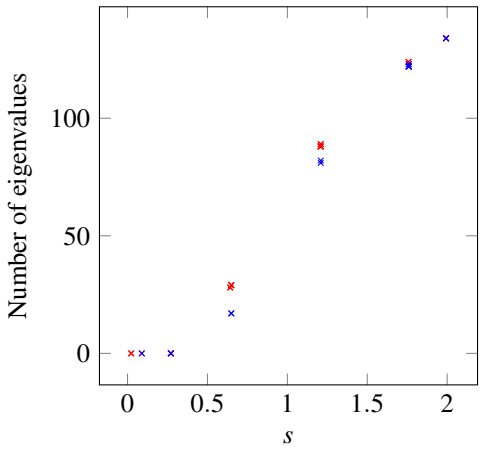
between the different nodes, depending on which module was externally stimulated, as opposed to *simply* a different amount of shared information through *pre-determined channels*.

Examining the eigenvalues associated with the stationary points identified, we see that most of them were complex with a negative real part (as shown in Fig. 6.9), meaning that they exerted an "attracting force" that was "pulling" the network activity to their vicinity. Additionally, and as can be seen in Fig. 6.9, an increase in the stimulus amplitude corresponded to an increase in (a) the real eigenvalues, (b) the eigenvalues with a negative real part, and interestingly, (c) the number of repeated eigenvalues.

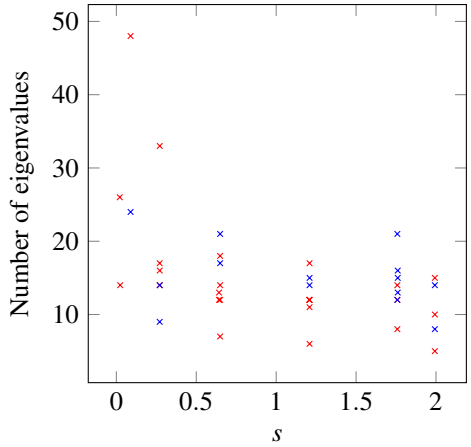
The first two observations are in agreement with the results shown for randomly connected networks in Figs. 4.5, 4.6 & 4.7, but an interesting deviation from randomly connected networks of $N = 200$ receiving input to all neurons, is that modular small-world networks of $1,000$ neurons receiving input through only one of the modules, appear to have stationary points with repeated eigenvalues. Furthermore, the number of repeated eigenvalues increased as the instantaneous value of the input injected to one of the modules was increased. This points to an interesting direction for future work, whereby additional experiments can be carried out to disambiguate the cause of appearance of these repeated eigenvalues. Currently, three different causes seem plausible: (a) the bigger number of neurons in the network, (b) the modular, small-world connectivity, and (c) the stimulation of only a subset of the network's neurons.



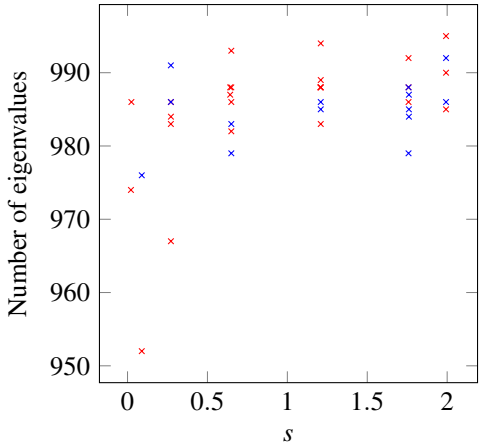
(a) Complex eigenvalues



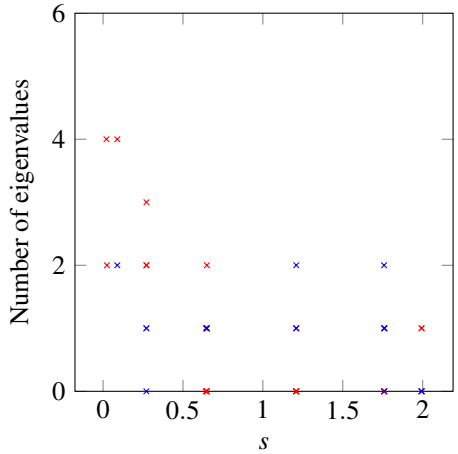
(b) Repeated eigenvalues



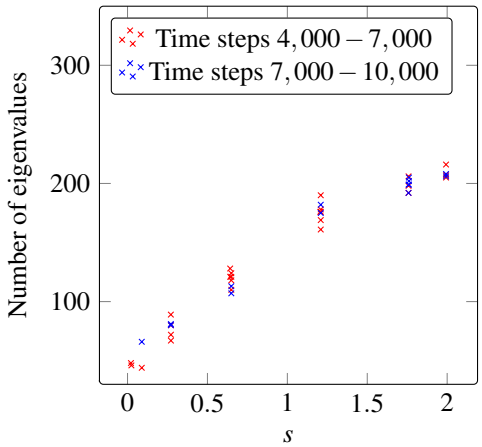
(c) Positive eigenvalues



(d) Negative eigenvalues



(e) Real and positive eigenvalues



(f) Real and negative eigenvalues

Figure 6.9: The number of the different types of eigenvalues of the network, for different stimulus values. During the stationary points identification process, 200 random initialisation of the optimisation where performed for each of the 40 possible instantaneous values of the input signal (20 values for the stimulus in each period). A total of 39 unique stationary points were identified during the optimisation process, each of which had $1,000$ eigenvalues after linearisation. The stimulus was injected in module 1 during time steps $4,000 - 7,000$, and module 2 during time steps $7,000 - 10,000$.

6.6 Discussion

From the results shown in this chapter, we can see that time-resolved measures of mutual information and transfer entropy can be used to visualise the emergence and growth of communication channels in simulated network with a modular, small-world connectivity when one of the modules is externally stimulated. These results provide additional computational evidence for the plausibility of the mechanisms believed to underlie conscious processing, such as the process of "ignition" taking place in the global neuronal space [DCKM14] and the formation of information-sharing coalitions in the connective core [Sha12].

Nonetheless, this simple computational experiment has its limitations. The first one is that it only assessed mutual information and transfer entropy between pairs of nodes, without quantifying any correlations between multiple nodes of the network. A possible extension of this work would be consider the multivariate version of these measures [LHH$^+$11], instead of the univariate version used here. This would mean that instead of calculating the temporal mutual information and transfer entropy between pairs of univariate time series, pairs of multivariate time series can be used, whereby each additional variable is the activity of another neuron from the same module, or another connective core neuron. This can give us more insight into how much information a group of neurons shares with another. Of course, this procedure produces a large number of possible combinations of groups that would need to be considered, and most importantly, it would require a much larger amount of data to be accurately estimated.

In a similar spirit, recent attempts to formulate a theoretical framework that allows us to move beyond pairwise mutual information, have resulted in the formulation of Partial Information Decomposition (PID) by Williams and Beer [WB10]. Within this framework, it is possible to decompose the information that a set of variables has about a target variable. In particular, the information that two variables have about a third one can be decomposed into the *redundant information*, which is the information about the target shared by both variables, the *unique information* that each of the two variable have individually and the *synergistic information* that is the information about the target, which can only be provided if both variables are considered simultaneously.

This framework should be able to provide additional useful insight into the particular types of information that the network's nodes share, compared to the current method, as it is able to differentiate between the possible ways in which the variables of the system interact to provide information about the state of another. Unfortunately though, PID scales prohibitively with the number of variables in the system. Williams and Beer [WB10] noted that for a system with just 9 variables, there are more than 5×10^{22} terms to be considered when decomposing the information of the system, which prohibits the use of the current version of PID for extending this work in the immediate future.

A more plausible set of future experiments, which can be conducted until a scalable PID formulation is conceived, can use the time series of the two measures generated in this experiment, and try to quantify different aspects of information broadcast and interaction taking place in the network. For example, a simple way in which these time series can be used, is to identify the duration of "bursts" of increases in the two measures in the network – similar to Dehaene's "ignition" process – and whether they cover the whole network. Moreover, groups of nodes that are commonly, and persistently "ignited", can be identified over the course of the experiment, to further support the claims for the presence of broadcast channels through the connective core. Another possibility is to assess the same measures when more than one modules receive external stimulation over the same period, and try to assess how broadcast channels are affected by the presence of two external sources of stimulation acting on the network through different modules.

Despite the fact that the experiment presented in this chapter has high computational requirements (both in terms of compute time and memory), there is a wide range of possible experimental configurations that can, in the future, extend this work. For example, the density of connections and number of neurons in the connective core can be varied, while keeping the number of neurons in each module constant, in order to assess how the broadcast of information is affected by the size and sparsity of the connective core. Even more, the connective core can be slowly replaced by additional random, and direct connections between the different modules. The number of these random connections between the modules can also be increased, so that the structural connectivity of the network gradually transitions to resemble that of a randomly connected network. This way, it will be possible to assess the relation between the information

broadcast capabilities and the structural connectivity of the network, as measured by the size and sparsity of the connective core, as well as the modularity and small-world index of the networks. In light of current limitations both in time and resources, these suggestion are left for future work.

Chapter 7

Conclusion

7.1 Overview

At the beginning of this thesis, and inspired by the extraordinary capabilities of the human brain, we set out to investigate a particular type of high-dimensional complex system, namely the continuous-time recurrent neural network, that has been shown to share similar aspects of its dynamic behaviour with cortical regions in the brain. Even though these networks have received much less attention lately, compared to their counterparts used in machine learning applications, they have nonetheless been used in recent work as a model of cortical dynamics. This illustrates that research using these models is relevant and that there remains much to be discovered. Motivated by this view, and using simulations of this mathematical model, this work has investigated non-linear and high-dimensional dynamics that are comparable to those observed in cortical regions. Novel experiments were conducted and analysed using well-established methods, to help develop a better understanding of the underlying information processing principles that could pertain the brain.

7.2 What have we learned and what still remains

The investigation focused on three distinct, yet related directions. Let us now briefly revisit each of them to evaluate what have we learned, and identify possible directions for future work.

7.2.1 Modelling brain networks are dynamical systems

In the first couple of experiments (Chapters 3 & 4), the dynamics of periodically driven continuous-time recurrent neural networks were investigated. This investigation used randomly connected sparse networks, driven by an external periodic signal, to reveal how these networks respond to such input, and explain the underlying mechanism responsible for this. To this end, the dimensionality of the resulting network dynamics was measured using a recently proposed, non-parametric method [TYFT15], and the resulting trajectories were visualised to provide an intuitive understanding of how the networks respond to the external periodic input. These results represent novel and non-trivial findings about the effect of the relative timescales of the input and the network on (a) the dimensionality of the resulting dynamics, and (b) the geometric shape of transient and periodic attractors of the network. Moreover, these results show that, depending on the timescale over which the samples of the network activity are recorded, the measured dimensionality of the system peaks under different experimental conditions; specifically, at different values of the ratio between the timescale of the network and the input.

Additionally, tools from dynamical systems theory were used to explore the structure of the underlying vector field, for different values of the network gain and of the input signal. These results showed that the value of the input to the network is an additional bifurcation parameter, that is commonly overlooked, and that leads to qualitative changes in the vector field of the network. Using the new findings obtained through this analysis, it was possible to explain in an intuitive way how simple neural circuits previously discussed in the literature [SB13] can perform computation on their input. In this way, these first two experiments constitute a new approach to analysing high-dimensional complex systems. Their aim was to try and explain the resulting behaviour of the network, using the underlying features of the vector field. This

work is related to other attempts in the field of computational neuroscience [SB13, MSSN13], that try to elucidate how computation is performed by the prefrontal cortex, and attempts to provide a more general framework for analysing these simulated networks.

A common criticism when using these types of networks as models of the cortex is the biological implausibility of the connectivity used, violating Dale's principle [SH99], which posits that each neuron can project either excitatory or inhibitory synapses to other neurons. Hence, the fact that the same neuron can project both excitatory (positive weights) and inhibitory (negative weights) activity to other neurons in these networks, and that a neuron's activity can be both positive and negative throughout a simulation, are the two most important features of the network criticised, when used for modelling cortical dynamics. Nonetheless, future work using CTRNNs can be undertaken whereby the connectivity matrix of the network is appropriately restricted, and the activation function of its neurons is replaced with a more biologically plausible one [HB18]. It would be interesting to see whether it would be possible to reproduce similar results to the ones obtained in the current work, as that would provide additional evidence that CTRNNs can help us to better understand cortical dynamics.

There are a number of possible directions for extending this work. To begin with, the results obtained for the dimensionality of the network dynamics can be complemented with a modal analysis of the networks' response, subject to the external periodic signal. Since we have seen that there is strong periodicity in their response, and that the dimensionality was strongly dependent on ρ, such an analysis could provide valuable insight by allowing us to see, in the frequency domain, how the network transforms the periodic input. Furthermore, more complex node degree distributions can be used that would allow us to compare the dynamics they produce to those produced by networks with a homogeneous degree, like the ones presented here, which have a random Erdös-Renyi connectivity. In this way, we could uncover how the topology of the network affects the networks dynamics, and perhaps discover qualitatively different responses to the ones presented here. When choosing how to alter the topology of CTRNNs, the field of complex networks can be a valuable source of inspiration. Further, and as already mentioned, it would be very instructive to further validate whether these results are still valid when more numerically stable, Runge-Kutta based numerical integration methods are used, so as to assess

how which of the phenomena observed are artefacts of the simpler Euler integration method, and which emerge from the equations of the system.

Another possible direction for future work is to try to conduct the same set of experiments with spiking neurons [HH52, Izh03], which are biologically more realistic. One of the main problems when analysing the collective activity of networks of spiking neurons, that use for example the Izhikevich model [Izh03], is the presence of a discontinuity in the update equations of the neurons. This is introduced as a simplification via the use of threshold, to determine when the neuron will fire, which makes the analysis of the collective dynamics very challenging. An obstacle that prevents us from using the simple tools used here, is that the state of a spiking neural network evolves by visiting a sequence of vertices of an N-dimensional hypercube, which is in contrast to CTRNNs, that evolve by producing a smooth trajectory in $\mathbb{R}^N$. A possible way for making progress in this direction is via the concept of polychronisation, i.e. spiking neural network that "*exhibit reproducible time-locked but not synchronous firing patterns with millisecond precision...*" [Izh06]. It is possible that by thinking of these polychronous groups as analogous to network trajectories, progress can be made by using more advanced methods from dynamical systems theory – or even by developing new ones – that will enable us to analyse the collective activity of spiking neural networks.

7.2.2 Reservoir computing for episodic-like memory

In Chapter 5 we looked at how these networks can be used in combination with FORCE learning to perform a simple episodic-like memory task, i.e. memorising and recalling sequences of high-dimensional data. Interestingly, what we saw was that when a stabilising signal was used, the networks could use reproducible trajectories in their state-space to reconstruct, through a linear readout, the sequence of high-dimensional frames. These "encoding" trajectories were shown to be stable and efficient, in the sense that similar memories can be reconstructed from similar trajectories, while they are able to maintain a memory of the initial state, which allowed *different subsequent* memories to follow the reconstruction of an *identical intermediate* memory, in different episodes.

Moreover, the parameters of the network affecting its performance in the task were identified through a sensitivity analysis, that showed that the gain of the network plays the most important role in the quality of reconstructions. As an exploratory investigation, the compression capability of the network was examined, showing that by trading-off reconstruction quality, the network can indeed compress the information it is memorising. Future work should investigate this property further, and evaluate whether there are cases where this capability of CTRNNs can outperform those of other, commonly used methods.

In addition, recent advancements in the learning algorithm used, i.e. the development of full-FORCE [DCR+18], open up the possibility for extending the range of tasks that these networks can be tested on. This will be done by moving away from the paradigm of reservoir computing, and allowing the direct modification of all the network weights. Since algorithms for modifying the internal weights of the network are now available, task-relevant dynamics can be imposed on the network. By imposing such dynamics, we will be able to closely examine the underlying vector field of the trained networks, using the method for finding and analysing stationary points that was used in the first two chapters of this thesis. This should further help us in understanding which task-relevant information to include when shaping the dynamics of future, fully-trained networks.

7.2.3 Information broadcast in modular small-world networks

In the last Chapter, we looked at CTRNNs from a different perspective, i.e. as information sharing networks, using tools from information theory. In the spirit of a computational exercise, the emergence of information broadcast channels was explored, by generating time series of the pair-wise mutual information and transfer entropy between the main components of a CTRNN with modular, small-world connectivity. The endeavour was inspired by recent hypotheses concerning the formation of information sharing coalitions in the brain, for the plausibility of which, additional computational evidence were offered.

A link was then briefly made with the previous sections, by visualising the position of the network's stationary points and examining their types. Interestingly, a number of repeated

eigenvalues were identified in this experiment, in contrast to the experiments presented in the previous chapters, where no repeated eigenvalue was identified. This points to an interesting direction for the future, whereby the origin of the repeated eigenvalues, and their effect on the network dynamics can be investigated. Finally, additional future experiments were identified, whereby, by varying the network's structural connectivity and external stimulation, it may be possible to investigate how (a) the emergence of information sharing coalitions, and (b) the appearance of repeated eigenvalues of the dynamics around the stationary points are affected.

7.2.4 Concluding remark and outlook

In conclusion, this work presents a collection of novel findings and insights into the dynamic behaviour of continuous-time recurrent neural networks, obtained by borrowing simple, yet powerful tools from information and dynamical systems theory. In an attempt to shine more light on how these networks behave in response to external input, and how they can be used to create episodic-like memory, various new experiments were devised. Throughout this thesis, an effort has been made to make clear the links between these experiments and other relevant work, in the fields of both machine learning and computational neuroscience. Hopefully, this work can offer an intuitive and comprehensive way to think about these networks and their dynamics, as well as inspire further research, and the development of new computational models that use such networks.

Bibliography

[AOR+13] Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, 10(5):413–420, 2013.

[AS15] Eric A Antonelo and Benjamin Schrauwen. On learning navigation behaviors for small mobile robots with reservoir computing architectures. *IEEE transactions on neural networks and learning systems*, 26(4):763–780, 2015.

[ASB+11] Afsheen Afshar, Gopal Santhanam, M Yu Byron, Stephen I Ryu, Maneesh Sahani, and Krishna V Shenoy. Single-trial neural correlates of arm movement preparation. *Neuron*, 71(3):555–564, 2011.

[ASS08] Eric A Antonelo, Benjamin Schrauwen, and Dirk Stroobandt. Event detection and localization for small mobile robots using reservoir computing. *Neural Networks*, 21(6):862–871, 2008.

[Baa93] Bernard J Baars. *A cognitive theory of consciousness*. Cambridge University Press Cambridge University Press, 1993.

[Bar17] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1 – 6, 2017. Computational Neuroscience.

[BB06] Danielle Smith Bassett and ED Bullmore. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.

[BB18] Chen Beer and Omri Barak. Dynamics of dynamics: following the formation of a line attractor. *arXiv preprint arXiv:1805.09603*, 2018.

[BBS09] Lionel Barnett, Adam B Barrett, and Anil K Seth. Granger causality and transfer entropy are equivalent for gaussian variables. *Physical review letters*, 103(23):238701, 2009.

[BBU+18] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429, 2018.

[BCM82] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[BD15] Ralph Bourdoukan and Sophie Deneve. Enforcing balance allows local supervised learning in spiking recurrent networks. In *Advances in Neural Information Processing Systems*, pages 982–990, 2015.

[Bee95] Randall D Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[Bee06] Randall D Beer. Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18(12):3009–3051, 2006.

[Bis06] Christopher M Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[BLB+15] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[BLCW09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[BM09] Dean V Buonomano and Wolfgang Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature reviews. Neuroscience*, 10(2):113, 2009.

[BMF+15] Yoshua Bengio, Thomas Mesnard, Asja Fischer, Saizheng Zhang, and Yuhuai Wu. Stdp as presynaptic activity times rate of change of postsynaptic activity. *arXiv preprint arXiv:1509.05936*, 2015.

[BNS+16] David Bhowmik, Kyriacos Nikiforou, Murray Shanahan, Michail Maniadakis, and Panos Trahanias. A reservoir computing model of episodic memory. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 5202–5209. IEEE, 2016.

[BO18] Giulio Bondanelli and Srdjan Ostojic. Coding with transient trajectories in recurrent neural networks. *arXiv preprint arXiv:1811.07592*, 2018.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[BSK10] Paul R Benjamin, Kevin Staras, and György Kemenes. What roles do tonic inhibition and disinhibition play in the control of motor programs? *Frontiers in behavioral neuroscience*, 4:30, 2010.

[Buo09] Dean V Buonomano. Harnessing chaos in recurrent neural networks. *Neuron*, 63(4):423–425, 2009.

[BUP+16] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

[CCK+10] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Stephen I Ryu, and Krishna V Shenoy. Cortical preparatory activity: Representation of movement or first cog in a dynamical machine? *Neuron*, 68(3):387 – 400, 2010.

[CCK+12] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51, 2012.

[CdLR+15] Federico Carnevale, Victor de Lafuente, Ranulfo Romo, Omri Barak, and Néstor Parga. Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty. *Neuron*, 86(4):1067–1077, 2015.

[Chi10] Dante R Chialvo. Emergent complex neural dynamics. *Nature physics*, 6(10):744, 2010.

[Cis06] Paul Cisek. Integrated neural processes for defining potential actions and deciding between them: a computational model. *Journal of Neuroscience*, 26(38):9761–9770, 2006.

[CL00] Tommy WS Chow and Xiao-Dong Li. Modeling of continuous time dynamical systems with input by recurrent neural networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(4):575–578, 2000.

[Cri89] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

[Csá01] Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[CSFW17] Warasinee Chaisangmongkon, Sruthi K Swaminathan, David J Freedman, and Xiao-Jing Wang. Computing by robust transience: How the fronto-parietal network performs sequential, category-based decisions. *Neuron*, 93(6):1504–1517, 2017.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, Hoboken, NJ, 2006.

[DC05] Stanislas Dehaene and Jean-Pierre Changeux. Ongoing spontaneous activity controls access to consciousness: a neuronal model for inattentional blindness. *PLoS biology*, 3(5):e141, 2005.

[DC11] Stanislas Dehaene and Jean Pierre Changeux. Experimental and theoretical approaches to conscious processing. *Neuron*, 70(2):200–227, 2011.

[DCKM14] Stanislas Dehaene, Lucie Charles, Jean Rémi King, and Sébastien Marti. Toward a computational theory of conscious processing. *Current Opinion in Neurobiology*, 25:76–84, 2014.

[DCR+18] Brian DePasquale, Christopher J Cueva, Kanaka Rajan, Larry F Abbott, et al. full-force: A target-based method for training recurrent networks. *PloS one*, 13(2):e0191527, 2018.

[DGSGR10] Hugo De Garis, Chen Shuo, Ben Goertzel, and Lian Ruiting. A world survey of artificial brain projects, part i: Large-scale brain simulations. *Neurocomputing*, 74(1):3–29, 2010.

[Dom98] Peter F Dominey. A shared system for learning serial and temporal structure of sensori-motor sequences? evidence from simulation and human experiments. *Cognitive Brain Research*, 6(3):163–172, 1998.

[DRD08] Mukeshwar Dhamala, Govindan Rangarajan, and Mingzhou Ding. Analyzing information flow in brain networks with nonparametric granger causality. *Neuroimage*, 41(2):354–362, 2008.

[DSA+09] Oguz Demirci, Michael C Stevens, Nancy C Andreasen, Andrew Michael, Jingyu Liu, Tonya White, Godfrey D Pearlson, Vincent P Clark, and Vince D Calhoun. Investigation of relationships between fmri brain networks in the spectral domain using ica and granger causality reveals distinct differences between schizophrenia patients and healthy controls. *Neuroimage*, 46(2):419–431, 2009.

[EH12] Uğur M Erdem and Michael Hasselmo. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35(6):916–931, 2012.

[EH14] Uğur M Erdem and Michael E Hasselmo. A biologically inspired hierarchical goal directed navigation model. *Journal of Physiology-Paris*, 108(1):28–37, 2014.

[EPQD16] Pierre Enel, Emmanuel Procyk, Ren Quilodran, and Peter F Dominey. Reservoir computing properties of neural dynamics in prefrontal cortex. *PLOS Computational Biology*, 12(6):1–35, 2016.

[ESC+12] Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205, 2012.

[Fet92] Eberhard E Fetz. Are movement parameters recognizably coded in the activity of single neurons? *Behavioral and Brain Sciences*, 15(4):679690, 1992.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[FMR16] Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66 – 74, 2016. Neurobiology of cognitive behavior.

[FN93] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[Fre99] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[Fri11] Karl J Friston. Functional and Effective Connectivity: A Review. *Brain Connectivity*, 1:13–36, 2011.

[GHC+09] Gaolang Gong, Yong He, Luis Concha, Catherine Lebel, Donald W Gross, Alan C Evans, and Christian Beaulieu. Mapping anatomical connectivity patterns of human cerebral cortex using in vivo diffusion tensor imaging tractography. *Cerebral Cortex*, 19(3):524–536, 2009.

[GHWR+15] Germán Gómez-Herrero, Wei Wu, Kalle Rutanen, Miguel C Soriano, Gordon Pipa, and Raul Vicente. Assessing coupling dynamics from an ensemble of time series. *Entropy*, 17(4):1958–1970, 2015.

[Gle88] James Gleick. *Chaos, Making a New Science*, volume 56. Viking Penguin Inc, 1988.

[GLR16] Jordan Guergiuev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *arXiv preprint arXiv:1610.00161*, 2016.

[GOPY84] Celso Grebogi, Edward Ott, Steven Pelikan, and James A Yorke. Strange attractors that are not chaotic. *Physica D: Nonlinear Phenomena*, 13(1-2):261–268, 1984.

[GOY83] Celso Grebogi, Edward Ott, and James A Yorke. Crises, sudden changes in chaotic attractors, and transient chaos. *Physica D: Nonlinear Phenomena*, 7(1-3):181–200, 1983.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Gra69] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[Gra11] Michael SA Graziano. New insights into motor cortex. *Neuron*, 71(3):387–388, 2011.

[GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[Hat05] Nicholas G Hatsopoulos. Encoding in the motor cortex: Was evarts right after all? focus on motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks. *Journal of Neurophysiology*, 94(4):2261–2262, 2005. PMID: 16160087.

[Hay02] Simon Haykin. Adaptive filter theory. *Prentice Hall*, 2:478–481, 2002.

[HB18] NF Hardy and Dean V Buonomano. Encoding time in feedforward trajectories of a recurrent neural network model. *Neural computation*, 30(2):378–396, 2018.

[HBF+01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[HCG+08] Patric Hagmann, Leila Cammoun, Xavier Gigandet, Reto Meuli, Christopher J Honey, Van J Wedeen, and Olaf Sporns. Mapping the structural core of human cerebral cortex. *PLoS biology*, 6(7):e159, 2008.

[Heb49] Donald O Hebb. The organization of behavior: a neuropsychological theory. 1949.

[HG08] Mark D Humphries and Kevin Gurney. Network small-world-ness: a quantitative method for determining canonical network equivalence. *PloS one*, 3(4):e0002051, 2008.

[HH52] Alan Lloyd Hodgkin and Andrew Fielding Huxley. Propagation of electrical signals along giant nerve fibres. *Proceedings of the Royal Society B*, 140(899):177–183, 1952.

[HKSB17] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[HS97a] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS97b] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[HS12] Michiel Hermans and Benjamin Schrauwen. Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, 24(1):104–133, 2012.

[HTG12] Quan Huynh-Thu and Mohammed Ghanbari. The accuracy of psnr in predicting video quality for different video scenes and frame rates. *Telecommunication Systems*, 49(1):35–48, 2012.

[HVG14] Guillaume Hennequin, Tim P Vogels, and Wulfram Gerstner. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82(6):1394–1406, 2014.

[HW68] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[IC92] Nathan Intrator and Leon N Cooper. Objective function formulation of the bcm theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5(1):3–17, 1992.

[Izh03] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[Izh06] Eugene M Izhikevich. Polychronization: computation with spikes. *Neural computation*, 18(2):245–282, 2006.

[Izh07a] Eugene M Izhikevich. Equilibrium. *Scholarpedia*, 2(10):2014, 2007. revision #91238.

[Izh07b] Eugene M Izhikevich. Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452, 2007.

[Jae01] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[Jae07] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.

[JM04] Prashant Joshi and Wolfgang Maass. Movement generation and control with generic neural microcircuits. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 258–273. Springer, 2004.

[KA02] Matthew B Kennel and Henry D I Abarbanel. False neighbors and false strands: A reliable minimum embedding dimension algorithm. *Physical Review E*, 66:026209, Aug 2002.

[Kan07] Takashi Kanamaru. Van der Pol oscillator. *Scholarpedia*, 2(1):2202, 2007. revision #138698.

[KBA92] Matthew B Kennel, Reggie Brown, and Henry D I Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45:3403–3411, Mar 1992.

[KCRS14] Matthew T Kaufman, Mark M Churchland, Stephen I Ryu, and Krishna V Shenoy. Cortical activity in the null space: permitting preparation without movement. *Nature neuroscience*, 17(3):440, 2014.

[KD14] J. R. King and S Dehaene. Characterizing the dynamics of mental representations: The temporal generalization method. *Trends in Cognitive Sciences*, 18(4):203–210, 2014.

[KDTB01] Maciej Kamiński, Mingzhou Ding, Wilson A Truccolo, and Steven L Bressler. Evaluating causal relations in neural systems: Granger causality, directed transfer function and statistical assessment of significance. *Biological cybernetics*, 85(2):145–157, 2001.

[KG92] Daniel T Kaplan and Leon Glass. Direct test for determinism in a time series. *Physical Review Letters*, 68:427–430, Jan 1992.

[KHM16] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[KHS+18] Andrew Katumba, Jelle Heyvaert, Bendix Schneider, Sarah Uvin, Joni Dambre, and Peter Bienstman. Low-loss photonic reservoir computing with multimode photonic integrated circuits. *Scientific reports*, 8(1):2653, 2018.

[Kir91] K Kirby. Context dynamics in neural sequential learning. In *Proceedings of Florida AI Research Symposium*, pages 66–70, 1991.

[KLH06] James J Knierim, Inah Lee, and Eric L Hargreaves. Hippocampal place cells: parallel input streams, subregional processing, and implications for episodic memory. *Hippocampus*, 16(9):755–764, 2006.

[KNST16] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016.

[KPA19] Yuji Kawai, Jihoon Park, and Minoru Asada. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, 2019.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LB13] Rodrigo Laje and Dean V Buonomano. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nature neuroscience*, 16(7):925–933, 2013.

[LBOM12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[LHH+11] Joseph T Lizier, Jakob Heinzle, Annette Horstmann, John-Dylan Haynes, and Mikhail Prokopenko. Multivariate information-theoretic measures reveal directed information structure and task relevant changes in fmri connectivity. *Journal of computational neuroscience*, 30(1):85–107, 2011.

[Lin93] Long-Ji Lin. Reinforcement learning for robots using neural networks. PhD dissertation, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[Liz14] Joseph T Lizier. Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI*, 1:11, 2014.

[LJ09] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[LM01] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

[Lor63] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[LR06] Christophe Letellier and Otto E Rossler. Rossler attractor. *Scholarpedia*, 1(10):1721, 2006. revision #91731.

[Mar06] Henry Markram. The blue brain project. *Nature reviews, Neuroscience*, 7(2):153, 2006.

[Mar10] James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.

[MAT5a] MATLAB. *version 8.5.0.197613 (R2015a)*. The MathWorks Inc., Natick, Massachusetts, 2015a.

[May76] Robert M May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459, 1976.

[MBR+11] Michael Murphy, Marie-Aurélie Bruno, Brady A Riedner, Pierre Boveroux, Quentin Noirhomme, Eric C Landsness, Jean-Francois Brichant, Christophe Phillips, Marcello Massimini, Steven Laureys, et al. Propofol anesthesia and sleep: a high-density eeg study. *Sleep*, 34(3):283–291, 2011.

[MC89] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[MDS16] Jonathan A Michaels, Benjamin Dann, and Hansjrg Scherberger. Neural population dynamics during reaching are better explained by a dynamical system than representational tuning. *PLOS Computational Biology*, 12(11):1–22, 11 2016.

[Mic17] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6:e20899, 2017.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[MLB10] David Meunier, Renaud Lambiotte, and Edward T Bullmore. Modular and hierarchically modular organization of brain networks. *Frontiers in neuroscience*, 4, 2010.

[MMO95] James L McClelland, Bruce L McNaughton, and Randall C O'reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[MNM02] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[Moz89] Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.

[MS11] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040, 2011.

[MS16] Jonathan A Michaels and Hansjörg Scherberger. hebbRNN: A reward-modulated hebbian learning rule for recurrent neural networks. *The Journal of Open Source Software*, 1(5), sep 2016.

[MSGVK97] Mortimer Mishkin, Wendy A Suzuki, David G Gadian, and Faraneh Vargha-Khadem. Hierarchical organization of cognitive memory. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 352(1360):1461–1467, 1997.

[MSSN13] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78, 2013.

[NMS17a] Kyriacos Nikiforou, Pedro Mediano, and Murray Shanahan. Identifying information broadcast in complex networks. Paper presented at: *Cognitive Computational Neuroscience*, 2017.

[NMS17b] Kyriacos Nikiforou, Pedro A. M. Mediano, and Murray Shanahan. An investigation of the dynamical transitions in harmonically driven random networks of firing-rate neurons. *Cognitive Computation*, 9(3):351–363, Jun 2017.

[NNM+96] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

[NO03] Kenneth A Norman and Randall C O'reilly. Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach. *Psychological review*, 110(4):611, 2003.

[Nor10] Kenneth A Norman. How hippocampus and cortex contribute to recognition memory: revisiting the complementary learning systems model. *Hippocampus*, 20(11):1217–1227, 2010.

[NP08] Mario Negrello and Frank Pasemann. Attractor landscapes and active tracking: the neurodynamics of embodied action. *Adaptive Behavior*, 16(2-3):196–216, 2008.

[Oja82] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[OKK16] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[OM94] Randall C O'reilly and James L McClelland. Hippocampal conjunctive encoding, storage, and recall: Avoiding a trade-off. *Hippocampus*, 4(6):661–682, 1994.

[OPBDC10] Joseph ONeill, Barty Pleydell-Bouverie, David Dupret, and Jozsef Csicsvari. Play it again: reactivation of waking experience and memory. *Trends in neurosciences*, 33(5):220–229, 2010.

[Ott02] Edward Ott. *Chaos in dynamical systems*. Cambridge university press, 2002.

[PDS+12] Yvan Paquot, Francois Duport, Antoneo Smerieri, Joni Dambre, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Optoelectronic reservoir computing. *Scientific reports*, 2:287, 2012.

[PE13] Alison R Preston and Howard Eichenbaum. Interplay of hippocampus and prefrontal cortex in memory. *Current Biology*, 23(17):R764–R773, 2013.

[PUS+17] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. *arXiv preprint arXiv:1703.01988*, 2017.

[RAS10] Kanaka Rajan, Larry F Abbott, and Haim Sompolinsky. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, 82:011903, Jul 2010.

[RBDRWF10] Mattia Rigotti, Daniel Ben Dayan Rubin, Xiao-Jing Wang, and Stefano Fusi. Internal representation of task rules by recurrent dynamics: The importance of the diversity of neural responses. *Frontiers in Computational Neuroscience*, 4:24, 2010.

[RBW+13] Mattia Rigotti, Omri Barak, Melissa R Warden, Xiao-Jing Wang, Nathaniel D Daw, Earl K Miller, and Stefano Fusi. The importance of mixed selectivity in complex cognitive tasks. *Nature*, 497(7451):585, 2013.

[RHW86a] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–538, 1986.

[RHW86b] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[RS10] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

[SA09] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[SAC09] Thomas Suddendorf, Donna R Addis, and Michael C Corballis. Mental time travel and the shaping of the human mind. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1521):1317–1324, 2009.

[SB05] Murray Shanahan and Bernard Baars. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, 2005.

[SB13] David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

[SB17] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11, 2017.

[SBM15] Markus Siegel, Timothy J Buschman, and Earl K Miller. Cortical information flow during flexible sensorimotor decisions. *Science*, 348(6241):1352–1355, 2015.

[Sch00] Thomas Schreiber. Measuring information transfer. *Physical Review Letters*, 85(2):461–464, 2000.

[SCKH04] Olaf Sporns, Dante R Chialvo, Marcus Kaiser, and Claus C Hilgetag. Organization, development and function of complex brain networks. *Trends in cognitive sciences*, 8(9):418–425, 2004.

[SCKS15] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025–1033, 2015.

[SCS88] Haim Sompolinsky, Anrea Crisanti, and Hans-Jürgen Sommers. Chaos in random neural networks. *Physical Review Letters*, 61(3):259–262, 1988.

[SH99] Piergiorgio Strata and Robin Harvey. Dales principle. *Brain Research Bulletin*, 50(5):349 – 350, 1999.

[Sha48] Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Sha10a] Murray Shanahan. *Embodiment and the inner life*. Oxford University Press, 2010.

[Sha10b] Murray Shanahan. Metastable chimera states in community-structured oscillator networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1):013108, 2010.

[Sha12] Murray Shanahan. The brain's connective core and its role in animal cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1603):2704–2714, 2012.

[SHK07] Olaf Sporns, Christopher J Honey, and Rolf Kötter. Identification and classification of hubs in brain networks. *PloS one*, 2(10):e1049, 2007.

[Sho07] Harel Z Shouval. Models of synaptic plasticity. *Scholarpedia*, 2(7):1605, 2007. revision #144654.

[Ski58] BF Skinner. Reinforcement today. *American Psychologist*, 13(3):94–99, 1958.

[SKR13] Tatyana O Sharpee, Minjoon Kouh, and John H Reynolds. Trade-off between curvature tuning and position invariance in visual area v4. *Proceedings of the National Academy of Sciences*, 110(28):11618–11623, 2013.

[SMS15] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[Spo10] Olaf Sporns. *Networks of the Brain*. MIT press, 2010.

[Spo13] Olaf Sporns. Network attributes for segregation and integration in the human brain. *Current Opinion in Neurobiology*, 23(2):162–171, 2013.

[SQAS15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Squ86] Larry R Squire. Mechanisms of memory. *Science*, 232(4758):1612–1619, 1986.

[Squ92] Larry R Squire. Memory and the hippocampus: a synthesis from findings with rats, monkeys, and humans. *Psychological review*, 99(2):195, 1992.

[SS03] Jon S Simons and Hugo J Spiers. Prefrontal and medial temporal lobe interactions in long-term memory. *Nature reviews. Neuroscience*, 4(8):637, 2003.

[SSA14] Merav Stern, Haim Sompolinsky, and Larry F Abbott. Dynamics of random neural networks with bistable units. *Physical Review E*, 90(6):062710, 2014.

[SSC13] Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual review of neuroscience*, 36, 2013.

[STE00] Olaf Sporns, Giulio Tononi, and Gerald M Edelman. Connectivity and complexity: the relationship between neuroanatomy and brain dynamics. *Neural networks*, 13(8-9):909–922, 2000.

[Ste04] Jochen J Steil. Backpropagation-decorrelation: online recurrent learning with o (n) complexity. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 843–848. IEEE, 2004.

[STK05] Olaf Sporns, Giulio Tononi, and Rolf Koötter. The human connectome: A structural description of the human brain. *PLOS Computational Biology*, 1(4), 09 2005.

[Str14] Steven H Strogatz. *Nonlinear Dynamics and Chaos*. Press, Westview, second edi edition, 2014.

[Sus14] David Sussillo. Neural circuits as computational dynamical systems. *Current opinion in neurobiology*, 25:156–163, 2014.

[SVVC07] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European Symposium on Artificial Neural Networks. p. 471-482 2007*, pages 471–482, 2007.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[TS09] Endel Tulving and Karl K Szpunar. Episodic memory. *Scholarpedia*, 4(8):3332, 2009. revision #91235.

[TS14] Filipe P P Teixeira and Murray Shanahan. Does plasticity promote criticality? In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2383–2390. IEEE, 2014.

[TS15] Filipe P P Teixeira and Murray Shanahan. Local and global criticality within oscillating networks of spiking neurons. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015.

[Tul85] Endel Tulving. *Elements of Episodic Memory*. Oxford University Press, 1985.

[TV08] Terence Tao and Van Vu. Random matrices: the circular law. *Communications in Contemporary Mathematics*, 10(02):261–307, 2008.

[TYFT15] Satohiro Tajima, Toru Yanagawa, Naotaka Fujii, and Taro Toyoizumi. Untangling brain-wide dynamics in consciousness by cross-embedding. *PLoS computational biology*, 11(11):e1004537, 2015.

[vdHS13] Martijn P van den Heuvel and Olaf Sporns. Network hubs in the human brain. *Trends in cognitive sciences*, 17(12):683–696, 2013.

[VMVV+14] Kristof Vandoorne, Pauline Mechet, Thomas Van Vaerenbergh, Martin Fiers, Geert Morthier, David Verstraeten, Benjamin Schrauwen, Joni Dambre, and Peter Bienstman. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nature communications*, 5:3541, 2014.

[WB09] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[WB10] Paul L Williams and Randall D Beer. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[WC72] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972.

[WCJ14] Xiaogeng Wan, Björn Crüts, and Henrik J Jensen. The causal inference of cortical neural networks during music improvisations. *PloS one*, 9(12):e112776, 2014.

[Wer88] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[Wer90] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[WHA+18] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[WNHJ18] Jing Wang, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible timing by temporal scaling of cortical responses. *Nature neuroscience*, 21(1):102, 2018.

[WS98] Duncan J Watts and Steven H Strogatz. Collective dynamics of "small-world" networks. *nature*, 393(6684):440, 1998.

[WS12] Mark Wildie and Murray Shanahan. Metastability and chimera states in modular delay and pulse-coupled oscillator networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):043131, 2012.

[WT13] Gilles Wainrib and Jonathan Touboul. Topological and dynamical complexity of random neural networks. *Physical review letters*, 110(11):118101, 2013.

[XL16] Changjin Xu and Peiluan Li. Dynamics in four-neuron bidirectional associative memory networks with inertia and multiple delays. *Cognitive Computation*, 8(1):78–104, 2016.

[XZW16] Changjin Xu, Qiming Zhang, and Yusen Wu. Bifurcation analysis of two-neuron networks with discrete and distributed delays. *Cognitive Computation*, 8(6):1103–1118, 2016.

[YJK12] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

[YL10] Wenjie Yang and Yiping Lin. Bifurcation and chaos analysis for a lotka-volterra system with time delay. In *Chaos-Fractals Theories and Applications (IWCFTA), 2010 International Workshop on*, pages 346–349. IEEE, 2010.

[You93] Malcolm P Young. The organization of neural systems in the primate cerebral cortex. *Proc. R. Soc. Lond. B*, 252(1333):13–18, 1993.

[ZBH09] Rüdiger Zillmer, Nicolas Brunel, and David Hansel. Very long transients, irregular firing, and chaotic dynamics in networks of randomly connected inhibitory integrate-and-fire neurons. *Physical Review E*, 79:031909, Mar 2009.

[ZLZK11] Gorka Zamora-López, Changsong Zhou, and Jürgen Kurths. Exploring brain function from anatomical connectivity. *Frontiers in Neuroscience*, 5:1–11, 2011.

Appendix A

A.1 Copyright Permissions

The work presented in Chapter 5 was published in the paper "A reservoir computing model of episodic memory" [BNS+16]. This has been acknowledged by adding the following statement in the beginning of the chapter:

"The basis and part of the work presented in this chapter was published in the paper "A reservoir computing model of episodic memory" [BNS+16] with permission from the publisher ©2011 IEEE."

Furthermore, I am reusing one of the figures from the paper, for which I include the following statement in the caption of the figure: "The figure was reproduced from [BNS+16] with permission from the publisher ©2016 IEEE."

This is an accordance with the guidelines from the publisher's permissions, a screen shot of which is show in Fig. A.1.

Figure A.1: Permission from IEEE publisher. Link to permission can be found through `https://ieeexplore.ieee.org/abstract/document/7727887`.