ABSTRACT

Title of dissertation: UNCOVERING PATTERNS IN COMPLEX DATA WITH RESERVOIR COMPUTING AND NETWORK ANALYTICS: A DYNAMICAL SYSTEMS APPROACH

Sanjukta Krishnagopal
Doctor of Philosophy, 2019

Dissertation directed by: Professor Michelle Girvan
Department of Physics

In this thesis, we explore methods of uncovering underlying patterns in complex data, and making predictions, through machine learning and network science.

With the availability of more data, machine learning for data analysis has advanced rapidly. However, there is a general lack of approaches that might allow us to 'open the black box'. In the machine learning part of this thesis, we primarily use an architecture called Reservoir Computing for time-series prediction and image classification, while exploring how information is encoded in the reservoir dynamics.

First, we investigate the ways in which a Reservoir Computer (RC) learns concepts such as 'similar' and 'different', and relationships such as 'blurring', 'rotation' etc. between image pairs, and *generalizes* these concepts to different classes *unseen* during training. We observe that the high dimensional reservoir dynamics display different patterns for different relationships. This clustering allows RCs to perform significantly better in generalization with limited training compared with state-of-the-art pair-based convolutional/deep Siamese Neural Networks.

Second, we demonstrate the utility of an RC in the separation of superimposed chaotic signals. We assume no knowledge of the dynamical equations that produce the signals, and require only that the training data consist of finite time samples of the component signals. We find that our method significantly outperforms the optimal linear solution to the separation problem, the Wiener filter.

To understand how representations of signals are encoded in an RC during learning, we study its dynamical properties when trained to predict chaotic Lorenz signals. We do so by using a novel, mathematical fixed-point-finding technique called directional fibers. We find that, after training, the high dimensional RC dynamics includes fixed points that map to the known Lorenz fixed points, but the RC also has spurious fixed points, which are relevant to how its predictions break down.

While machine learning is a useful data processing tool, its success often relies on a useful representation of the system's information. In contrast, systems with a large numbers of interacting components may be better analyzed by modeling them as networks. While numerous advances in network science have helped us analyze such systems, tools that identify properties on networks modeling multi-variate time-evolving data (such as disease data) are limited. We close this gap by introducing a novel data-driven, network-based Trajectory Profile Clustering (TPC) algorithm for 1) identification of disease subtypes and 2) early prediction of subtype/disease progression patterns. TPC identifies subtypes by clustering patients with similar disease trajectory profiles derived from bipartite patient-variable networks. Applying TPC to a Parkinsons dataset, we identify 3 distinct subtypes. Additionally, we show that TPC predicts disease subtype 4 years in advance with 74% accuracy.

UNCOVERING PATTERNS IN COMPLEX DATA WITH RESERVOIR COMPUTING AND NETWORK ANALYSIS: A DYNAMICAL SYSTEMS APPROACH

by

Sanjukta Krishnagopal

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Patterns of the Wild

The echoes of the morning wind
Roar through the meadows,
Big whorls blend into smaller whorls,
As vortices devour each other hungrily.
Tall green grass flapping flippantly,
Bathed in the first rays of dawn.

The mountains groan in unison
As they wipe the snow off their brow.
Avalanches cascading down the slope,
Rolling into each other seemingly randomly.
Slopes intersect; valleys catching the last snow.
Stability at last.

Hiding amongst the trees,
The frenzied singing of cicadas;
Fireflies synchronize their wings
As they light up the night sky.
Order in the pandemonium;
The forest is alive.

The sheet of time
Wrapped snugly around the planets
As they push and pull,
And often flail and die,
Only to be reborn again.
The fractals of the night sky.

The lion prowls in stealth.
The wolf sneers in the dark,
As she follows the trail
Of the white rabbits' tail.
Ants scurry about with purpose.
A tangled web of predator and prey.

Time is ephemeral,
And yet it is eternal.
All things that come must go.
And yet, as the children of the stars
Dissolve into the earth,
Amidst the chaos,
Patterns unravel.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

It goes without saying that this would not have been possible without my advisor, Prof. Michelle Girvan. Michelle has been instrumental in the trajectory of my research interests; it is from her that I have learned much of what I know about network science. She has supported me through my graduate career, indulged my diverse interests and ideas, and taught me through example, not only about research, but also about all the important yet often overlooked things that come with it, and most of all about how to be a compassionate person though it all. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Prof. Brian Hunt, Prof. Edward Ott, Prof. Jim Reggia, Asst. Prof. Garrett Katz, and Prof. Yiannis Aloimonos, Prof. Larry Davis, Dr. Lisa Shulman, and Dr. Rainer Von Coelln, all of whom have contributed to my growth in graduate school. Without their extraordinary ideas, useful feedback, and willingness to invest in me, this thesis would have not have been possible. I'm particularly grateful to Brian for his advisory role towards the end of my PhD. Special thanks are due to Prof. Brian Hunt, Prof. Jim Reggia, Prof. Edward Ott and Prof. Rajarshi Roy for agreeing to serve on my thesis committee and for sparing valuable time in reviewing this thesis.

My colleagues, Sarthak Chandra and Jaideep Pathak, have been a constant wall to bounce ideas off, I value their friendship just as much as I value our scientific discussions. Special thanks to my office-mates Sarthak Chandra and Sarah Burnett for always keeping the door open for me. I would also like to acknowledge the support of Michelle, Daniel and everyone involved in the COMBINE program at UMD.

I owe my deepest thanks to my family - my mother Usha Iyengar and father Srinivas Krishnagopal who have always stood by me and guided me through my career, read my papers in the middle of the night, and have pulled me through against seemingly impossible odds at times. Words cannot express the gratitude I owe them and the love I have for them. Not to be left behind are my grandparents with their unfaltering love.

I'd like to thank my housemates and friends Dina Genkina, Nicholas Mattise, Karthik Menon, and Jacob Bedrossian, who have shown me kindness and love through good times and bad. I would also like to thank Twin Oaks community where I wrote much of my thesis.

It is impossible to list everyone that I am grateful to. Graduate school has been a wonderful and crazy and beautiful and stressful experience all at once, and anyone who I've had the privilege of associating with in this period has touched my life in intangible ways. To all of them, thank you!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RC	Reservoir Computer
ESN	Echo State Network
LSM	Liquid State Machines
ML	Machine Learning
NN	Neural Network
ANN	Artificial Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Units
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
SNN	Siamese Neural Network
IIR	Infinite Impulse Response
DF	Directional Fiber
RMS	Root Mean Square
PD	Parkinson's Disease
PPMI	Parkinsons Progression Markers Initiative
TC	Trajectory Clustering
TPC	Trajectory Profile Clustering
SNP	Single Nucleotide Polymorphism
UMD	University of Maryland

Chapter 1: Introduction

Patterns are ubiquitous in nature, sometimes obvious to an observer, but often encoded in dynamic phenomena and revealing themselves only upon long-term multi-variable observation. Unraveling such patterns requires tools for analysis of complex data, often with time-varying interactions. Such tools have tremendous applications in fields including biological systems, weather prediction, social interactions etc. With the availability of more and more data, there's significant work being done in applying network science and machine learning to understand patterns in them, and in using them to make predictions about the future.

The field of machine learning has been gaining popularity as an agnostic method of uncovering patterns and 'learning' systems. Learning captures the adaptivity of an intelligent agent to its environment and investigates how artificial systems can improve their performance with experience. Neural networks process inputs in high dimensional spaces, and 'learn' about the underlying system through training. While machine learning is becoming increasingly popular, there is a general lack of a dynamical systems approach to neural networks that could explain 'learning'. Additionally, conventional machine learning methods often adopt a black-box approach that require large datasets and high computational resources. Consequently, there is a need to investigate methods that can be explained from a dynamical systems perspective and that can generalize learning with small datasets.

The machine learning approaches for identifying patterns in data in this thesis is mostly concerned with a particular type of machine learning architecture called reservoir computing that falls under the Recurrent Neural Network (RNN) class. A Reservoir Computer (RC) is a brain-inspired dynamical system that consists of recurrent connections between nodes that are driven by an input. Only the output weights of the RC are trained, making it computationally efficient and hardware implementable. Supervised training of big RNN models is often difficult - it converges slowly and usually gets stuck in a local minimum that depends on parameter initialization. Reservoir computing resolves this problem through less learning. In particular, a much more powerful computational model can be used if not all of its parameters (weights) are supervisedly trained, as in the case of a reservoir, leading to superior results. In classical reservoir computing most of the parameters remain randomly generated according to certain rules.

In this thesis we explore network science and non-linear dynamics approaches to extract meaningful information from data through the following experiments:

1. Generalized Similarity Learning in Images

2. Reservoir computing for separating chaotic signals

3. Mathematical analysis of the dynamical states of a reservoir computer

4. Trajectory clustering algorithm for subtype detection in bipartite networks

Much of the research presented in thesis has already been either published or submitted for publication. Project 1 on similarity learning was published in Complexity in 2018. Project 2 on separating chaotic signals is currently submitted to Chaos. Project 3 on dynamical analysis of the reservoir computer was published as conference proceedings of the International Joint Conference on Neural Networks in 2019. Project 4 on subtype detection via bipartite networks was submitted to PLoS One in 2019 and is currently under review.

1.1 Generalized Similarity Learning in Images

In chapter 2 of this thesis we explore how transformations between images are encoded in a RC, and explain, through the internal state of the RC, why it is a good model for generalizable learning, i.e., learning transformations on classes of data that it hasn't been trained on. This ability to generalize can be understood through input-feature encoding patterns (or attractors) in the reservoir dynamics. Networks such as reservoir computing are interesting because they are computationally efficient, suitable for temporal data, hardware implementable, and can be tuned to operate at the edge of chaos. Reservoir parameters can be tuned to make the reservoir operate at the edge of chaos, i.e., they have wide range of available reproducible dynamical states. This large space can encode input representations and their semantic meaning through patterns that are unique to the input. For instance, the dog attractor, i.e., the dynamical pattern of the network when a dogs image is input, is likely closer in the high dimensional space to the cat attractor than the car attractor. These attractors allow the system to generalize with small training data. We compare our method to Siamese Neural Networks (SNN) a standard pair-based feedforward neural network and demonstrate that the RC outperforms the SNN for generalization tasks. Thus, reservoirs dynamically encode for image features, providing an efficient method for generalizable pair-based image learning without requiring large datasets, in contrast with conventional machine learning approaches. This provides some insight through dynamical systems into explainable machine learning, a field that is gaining traction rapidly.

1.2 Reservoir computing for separation of chaotic signals

The reservoir computer, because of its dynamical properties, is particularly well suited for predicting temporal signals. In chapter 3 of this thesis we explore the ability of an RC to separate chaotic signals. Signal denoising or signal separation is a popular application in the field of signal processing. In particular, several real life signals such as the weather or human speech are deterministic yet chaotic in nature, making them difficult to predict using conventional machine learning. Several state of the art signal separation methods such as the Wiener filter [1] are spectra-based. In this work, we use reservoir computing for separation of chaotic signals generated from the Lorenz system. The input to the reservoir was a mixed signal $u(t) = \sqrt{\alpha} s_1(t) + \sqrt{1-\alpha} s_2(t)$. where s_1 and s_2 both from chaotic systems and α is the mixing fraction. We extended this work to generalized signal separation (where is unknown) using two RCs in series - one to estimate α, and the other for signal separation for a known α. Our technique can also be applied to parameter estimation using machine learning and outperforms the Wiener filter, commonly used in signal separation and mathematically shown to be the optimal linear filter under infinite data. Thus the RC is able to separate chaotic signal without any information about the system that the signals are generated from, even in cases where the spectra of the signals to be separated are similar, and even when their amplitude ratio is unknown.

1.3 Mathematical analysis of the dynamical states of a reservoir computer

While machine learning as a black box approach has been immensely successful, understanding the dynamics within a machine learning system is crucial is an important and relatively unexplored question. In chapter 4, we understand the mechanisms of learning using ideas from chaos theory and nonlinear dynamics. We train an RC to predict the chaotic Lorenz system. We used a novel mathematical tool called directional fiber [2] to identify attractors in the high dimensional reservoir. Directional fibers are mathematical objects that can be used for systematic root-finding in smooth, non-linear, multivariate equations. Let $f(x) = \mathbf{0}$ be any such equation, where $f : \mathbb{R}^M \rightarrow \mathbb{R}^N$ is a smooth function and $\mathbf{0}$ is an appropriately sized vector containing all zeros. For a constant direction vector $c \in \mathbb{R}^N - \{\mathbf{0}\}$, the directional fiber of c under f is defined as:

$$\gamma^{(c)} = \{x \in \mathbb{R}^M : f(x) \text{ is a scalar multiple of } c\}. \tag{1.1}$$

The directional fiber identifies attractors is the reservoir computer, a network that has been shown to be successful in the prediction of chaotic signals [3]. The three dimensional Lorenz system is known to have three fixed points. We find that the RC, on being trained on data from the chaotic Lorenz system, encodes (in higher dimensional reservoir space) the attractors of the Lorenz system. Thus, the reservoir learns dynamical properties (such as fixed points etc.) of the Lorenz system, without ever explicitly seeing the Lorenz system. Additionally, the reservoir also contains inherent spurious fixed points. The distribution of these spurious points define the basin of attraction of the encoded attractor in reservoir space and contribute to eventual breakdown of prediction. This is a big step in opening the black box and exploring explainable machine learning from a dynamical systems perspective, a field with tremendous tangible impact.

1.4 Trajectory clustering algorithm for subtype detection in bipartite networks

The field of network science is particularly well suited to unraveling patterns in data with evolving interactions. Complex interactions between different types of agents in a system are often better captured in through graphical representations than through a machine learning model. Networks have been used with a lot of success to model neuronal interactions, social dynamics, disease networks etc. Network algorithms such as community detection are useful in extracting properties of such network models that further enable prediction. In chapter 5 of this thesis, we present a 'trajectory clustering' algorithm for the analysis of network clusters based on similarity in trajectory, i.e., the evolution pattern of variables and their interactions over time. In our algorithm, subtypes/communities are identified based on similarities in trajectories of individuals as they evolve through the layers, and not just their variable profile at different times. There has been little previous work on trajectory-based network algorithms in the field of network medicine. To demonstrate the effectiveness of our method, we apply it to data from Parkinson's disease, the second largest neuro-degenerative disease in the world, with different types of variables associated with it such as genetic, clinical, and demographic. However our approach is easily extended to other data with complex multi-variate and time-varying data in other diseases as well as transportation, biological or social networks. Our algorithm demonstrated a success rate of 74% in Parkinsons subtype prediction in test-set patients 4 years in advance. This is an important step in early subtype prediction and personalized medicine, and provides a new useful way of community detection in graphs that can be extended to other disease datasets.

Chapter 2: Similarity Learning and Generalization with Limited Data: A Reservoir Computing Approach

This chapter appears in the following publication: Krishnagopal, Sanjukta, Yiannis Aloimonos, and Michelle Girvan. "Similarity Learning and Generalization with Limited Data: A Reservoir Computing Approach." Complexity (2018).

2.1 Introduction

Different types of Artificial Neural Networks (ANNs) have been used in the areas of feature recognition and image classification. *Feedforward* machine learning architectures such as convolutional neural networks (CNNs)[4], deep neural networks [5], stacked auto encoders[6] etc. and *recurrent* architectures such as Recurrent Neural Networks (RNNs) [7], Long Short-Term Memorys (LSTMs)[8] etc. have been immensely successful for several tasks from speech recognition [9] to playing the game GO [5].

There have also been a number of rapid advances in other *recurrent* machine learning architectures such as Echo State Networks (ESNs) (originally proposed in the field of machine learning) [10] and Liquid State Machines (LSMs) (originally proposed in the field of computational neuroscience) [11] , commonly falling under the term Reservoir Computing [12]. Compared with deep neural networks, Reservoir Computers (RCs) are a brain-inspired machine learning framework, and their inherent dynamics when trained on cognitive tasks have been show to be useful in modeling local cortical dynamics in higher cognitive function [13].

The goal of this work is to demonstrate the unreasonable efficiency of Reservoir Computers (RCs) in learning the relationships between images with very little training data and consequently generalizing the learned relationships to classes of images not seen before. We recognize that other machine learning techniques such as deep learning [14] and CNNs have been proven to be extremely successful at image classification and have also been used for tasks involving learning concepts of similarity [15, 16, 17], however, they generally require large training datasets and high computational resources. To our knowledge, similarity-based tasks have not been systematically investigated using RC architectures. However, RCs, because of their dynamical properties and simple training needs, may inherently be better suited for learning from a small training set and generalization of this learning [18]. While other recurrent architectures, like LSTMs and Gated Recurrent Units (GRUs), may also offer dynamical properties enabling generalization, due to their complex structure and training, they often require comparatively much larger datasets for training and hence are more computationally intensive.

RCs are dynamical systems that non-linearly transform input data in a reproducible way in order to serve as a resource for information processing. They are appealing because of their dynamical properties as well as easy scalability, since only the output weights are trained, while the recurrent connections within the reservoir are fixed randomly. Applications of RCs include processing and prediction of many real world phenomena such as weather patterns, stock market fluctuations, self-driving cars, language interpretation, robotic control, etc., several of which are inherently non-linear phenomenon. RCs are also appealing because of their biologically-inspired underpinnings. Biological systems such as the visual cortex are known to have primarily ($\sim 70\%$) recurrent connections with less than 1 % of the connections being feedforward [19]. RCs (or closely related architectures) provide insights into how biological brains can carry out accurate computations with an 'inaccurate" and noisy physical substrate [20], for example, accurate timing of the way in which visual spatiotemporal information is super-imposed and processed in primary visual cortex [21]. Additionally, models of spontaneously active cortical circuits typically exhibit chaotic dynamics, as in RCs [22, 23].

In biological systems, a recurring method of learning is through analogies, using only a handful of examples [24]. For example, in [25], bees were trained to fly towards the image in an image pair that looked very similar to a previously displayed base image. On training bees to fly towards the visually similar image, the bees were presented with two scents, one very similar to and one different from a base scent. As a consequence of the visual training that induced preference to the very similar category, the bees flew towards the very similar scent. Recent work has also been done on the phenomenon of 'peak shift', where animals not only respond to entrained stimuli, but respond even more strongly to similar ones that are farther away from non-rewarding stimuli [26]. In this way, biological systems have been found to translate learning of concepts of similarity across sensory inputs, suggesting that the brain has a common and fundamental mechanism that comprehends through analogies or through concepts of 'similarity,' allowing for generalization of the relationships to unseen classes of data. Compared with machine learning, humans learn much richer information using very few training examples. Moreover, humans learn more than how to do pattern or object recognition: they learn a concept, i.e., a model of the class that allows their acquired knowledge to be flexibly applied in new and unseen situations [27]. While many machine learning approaches can effectively classify images with human-like accuracy, these approaches often require large training datasets, and consequently increasingly powerful GPUs.

Despite the fact that research in learning from very few images, e.g., one shot learning [28], etc., has gained momentum recently, integrating it with generalization of learning is a relatively unexplored area. One shot learning, which learns a class (e.g. sleeping cats) from one example, is distinctly different from the task of generalization to an entirely new class (e.g., recognizing sleeping dogs after having only been trained to recognize sleeping cats). In our framework, the RC not only requires very few training examples compared to techniques such as deep learning, but can also effectively use analogies to learn relationships, leading to easy generalization.

RCs are built on several prior successful approaches that emphasize the use of a dynamical system, e.g., with temporal reinforcement, for successful, neuro-inspired learning. In the ground-breaking work of Hopfield in [29], the success of Recurrent Neural Networks (RNNs) depend on the existence of attractors. In training, the dynamical system of the RNN is left running until it ends up in one of its several attractors. Similarly, in [30], a unique conceptor is found for each input pattern in a driven RNN. However, training of RNNs is difficult due to training problems like exploding or vanishing gradient. RCs overcome this problem by training only the output weights. RCs offer a convenient solution to some the problems with RNNs, while offering many of the same advantages.

In this work, we explore two RC architectures that broadly resemble neural architecture (sec. 2.2.1). We train the RCs on both the MNIST handwritten digit database (to demonstrate proof of concept) as well as depth maps of visual scenes from a moving camera, to study generalization of the learned relationships between pairs of images. The data and methods are outlined in section 2.2. The methods include training the RC to identify relationships between image pairs drawn from a subset of handwritten digits $(0-5)$ from the MNIST database and generalizing the learned relationships to images of handwritten digits $(6-9)$ unseen during training. Additionally, using a database of depth maps of images taken from a moving camera, we train RCs to learn relationships such as 'similar' (e.g., same scene, different camera perspectives) and 'different' (different scenes) and investigate the system's ability to generalize its learning to visual scenes that are very different from those used in training. In section 2.3.1, we present the performance of our RC architectures in generalization to unseen classes, showing successful generalization for both handwritten digits and depth maps.

We also compare, in section 2.3.2, the RC performance for our generalization task to two pair-based, feed-forward approaches: a deep siamese neural network (SNN) and a convolutional siamese neural network (CSNN). Several recent studies have been very successful in using siamese (pair-based) feed-forward networks for similarity-based tasks such as sketch-based image retrieval [31], gait recognition in humans [32], signature verification [33], verification and one/few shot learning on the omniglot dataset [34, 35] etc. For our generalization task, we show that the reservoir performs significantly better than commonly used deep and convolutional siamese neural networks, both for simpler MNIST images as well as for depth maps, highlighting the utility of the RC approach for generalization to unseen data classes using limited training data. We also show, in section 2.3.3, that the reservoir is able to recognize not only the individual relationships it has been trained on but also combinations of them.

In order to explain the success of the reservoir in generalization, we look for recurring dynamical patterns the reservoir system state trajectories in section 2.3.4. We find that the reservoir state trajectories in response to different types of input pairs effectively cluster, with different clusters corresponding to different relationships between the pair of input images. The reservoir can then be thought of as a nonlinear filter whose goal is to map the input into a high-enough dimensional space that the important features become nearly linearly separable. In addition, the dynamical properties of the reservoir allow for temporally encoded 'memory'. We speculate that this combination of effective nonlinear filtering and temporally encoded memory allows for generalization of the learned relationships to classes of image pairs seen and unseen by the reservoir using a small number of training image pairs.

2.2 Data and Methods

We use two datasets for our study: (1) the handwritten digit MNIST database that consists of 70000 images, each 28×28 pixels in size, of handwritten digits 0-9; and (2) depth maps from a moving camera from 6 different visual scenes recorded indoors in an office setting (refer data availability for access to dataset). Each visual scene has depth maps from at least 300 images, each compressed to 100×100 pixels in size, recorded as the camera is moved within a small distance (∼30cm) and rotated within a small angle (∼30°). A sample of three RBG images from one of the 6 classes is shown in Fig. 2.1.

In our framework, images are always considered in pairs (image 1 and image 2). We study five relationships - noise, rotated, zoomed, blurred, and different. We are interested in exploring relationships between images through concepts of 'similarity' and 'difference'. The relationships we consider are a natural extension of these concepts. Examples of the image pair relationships applied to the MNIST dataset is shown in Fig. 2.2. We create the image pairs as follows:

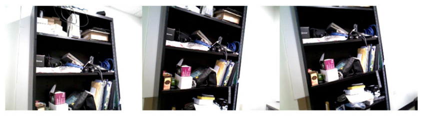


Figure 2.1: Examples of images taken from a moving camera from the same class. A pair of these would be classified under the category 'similar'

Two different images from the same class, i.e. of the same digit, are taken directly from the MNIST database for cases $1-4$. There may be significant variation between these images in spite of them being from the same class.

1. Noise: One of the images in the pair (image 1) remains untransformed, whereas the other (image 2) is transformed by superimposing random noise with peak value given by 20 % of the peak value of image 1 (Ex. Fig 2.2(a)).

2. Rotated: Image 2 is 90° rotated (Ex. Fig 2.2(b))

3. Zoomed: Image 2 is zoomed with a magnification of 2 (Ex. Fig 2.2(c)).

4. Blurred: Image 2 is blurred (Ex. Fig 2.2(d)) by convolving every pixel of the image by a 6×6 convolution matrix with all values 1/36:

5. Different: Two different images from different classes (Ex. Fig 2.2(e)).

All pairs are characterized by the relationship between the image-pair. For instance, we call a pair rotated if we start from two different handwritten images of the same digit and rotate the second image 90° with respect to the first. Since two different handwritten images of the same digit are used, the relationship between the image pair involves an initial non-linear transformation in addition to the applied transformation.

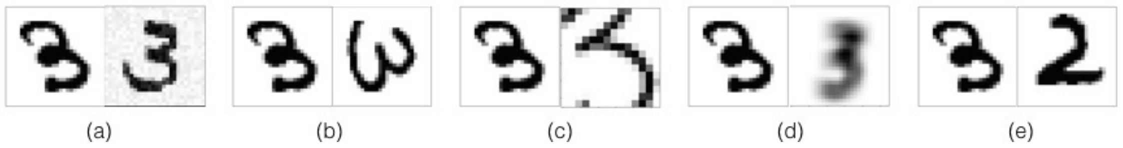


Figure 2.2: Pairs of images that are representative of the transformations classified into five labels: (a) very similar, (b) rotated by 90°, (c) zoomed, (d) blurred and (e) different.

2.2.1 Network Architecture

In this work we use the Echo State Network (ESN) class of RCs for training and classification. Our RCs are neural networks with two layers: a hidden layer of recurrently interconnected non-linear nodes, driven both by inputs as well as through feedback from other nodes in the reservoir layer and an output or readout layer. Only the output weights of the reservoir are trained. RCs have been found to replicate attractors in dynamical systems [36, 37]. It works particularly well for analyzing time-series input data due to its short term memory [18] and high-dimensional encoding of the input [3, 38]. The input images are hence converted into a 'time-series' by feeding the reservoir a column of the input image at each time point (as in [39]). The method of 'temporalization' of the input (row-wise, column-wise etc.) simply changes the input representation and doesn't affect the analysis. While there is limited understanding of the actual processes through which the brain processes analogies, we explore two models that are inspired by cortical processing of relationships between inputs. There has been some evidence [40] of integrated processing, particularly in the visual cortex. To mimic an integrated processing system more closely, we study the Single Reservoir architecture (Fig. 2.3(a)). However, there is also some evidence that analogy processing involves two steps: (1) the brain generated individual mental representations of the different inputs and (2) brain mapping based on structural similarity, or relationship, between them [41]. We create the Dual Reservoir architecture (Fig. 2.4) in an attempt to mimic this parallel processing of signals followed by mapping based on the differences between the processed signal in the cortex. Since, there isn't a consensus in the neuroscience community about the details of cortical processing, we present both the single and dual reservoir architecture here.

2.2.1.1 Single Reservoir Architecture

Input Layer   As discussed above, in order to exploit the memory properties of RCs, the input is converted to a time series. We vertically concatenate the image pair to form the combined image. We then input the combined image, through the input weights matrix W^{in}, column by column (shown in Fig. 2.3(b) for the MNIST database) into the reservoir, i.e., with the time axis to run across the columns of the combined image. While this 'temporalization' may seem artificial, there's a unique reproducible reservoir state trajectory (the sequence of reservoir states) corresponding to each image, causing the results to be independent of order of temporalization, as long as all images are temporalized the same way. W^{in} is randomly chosen and scaled such that the inputs to the reservoir are between 0 and 1.

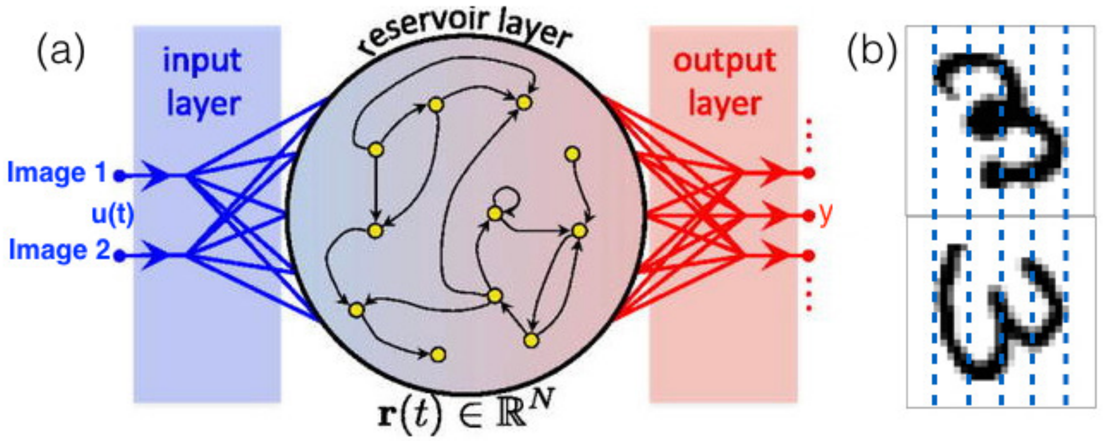


Figure 2.3: (a) Reservoir architecture with input state of the two images at time t denoted by $\overrightarrow{u}(t)$, reservoir state vector at a single time by $\overrightarrow{r}(t)$ and output state by $\overrightarrow{y}$. (b) shows one image pair from the rotated 90° category of the MNIST dataset split vertically and fed into the reservoir in columns of 1 pixel width, shown to be larger here for ease of visualization.

Reservoir Layer The reservoir can be thought of as a dynamical system characterized by a reservoir state vector $\vec{r}(t)$ which describes the state of the reservoir nodes as a function of time t. $\vec{r}(t)$ is given by:

$$\overrightarrow{r}(t+1) = \tanh\left(W^{\text{in}} \cdot \overrightarrow{u}(t) + W^{\text{res}} \cdot \overrightarrow{r}(t) + b\right) \tag{2.1}$$

The input weights matrix $W^{\text{in}} \in \mathbb{R}^{N_R \times N_u}$, where N_R is number of nodes in the reservoir and N_u is the dimension of the input vector $\overrightarrow{u}(t)$; here N_u is the number of rows of the concatenated image. The activity of the reservoir at time t is given by $\overrightarrow{r}(t)$, of size N_R. The recurrent connection weights $W^{\text{res}} \in \mathbb{R}^{N_R \times N_R}$ are set randomly between -1 and 1. b is a scalar bias. We use hyperbolic tangent as the non-linear activation function. We rescale W^{res} to have a spectral radius γ (maximal absolute eigenvalue) of 0.5, but we observe no conclusive correlation or robust pattern between performance and this choice as seen in Fig. 2.10. Our choice of spectral radius is in part influenced by the analysis of the effect of spectral radius on performance presented in [42]. The reservoir is a dynamical system that transforms the low dimensional input into a much higher dimensional reservoir space and isn't affected by W^{in} and W^{res} being sparse, making it computationally faster. Matrix sparsity is 0.9 (90% of the entries are randomly chosen to be zero) unless otherwise stated.

Output Layer In the single reservoir architecture, for one combined input image, the reservoir state trajectory, X, is formed by concatenating the $N_R \times 1$ reservoir state vectors (the state of all reservoir nodes) at every timestep $\overrightarrow{r}(t)$ as follows:

$$X = [\overrightarrow{r}(0)\ \overrightarrow{r}(t=1)\ \dots\ \overrightarrow{r}(t=T)], \tag{2.2}$$

where X is an augmented matrix of size $N_R \times c$ and c is the number of columns in the image (number of time steps, T, through which the entire image is input). For the single reservoir case, X is the same as the reservoir *system* state trajectory, denoted by $\tilde{X}$ for both single and dual reservoir architectures). $\tilde{X}$ is the matrix obtained by processing the input through the reservoir architecture that is then used to generate the output weights.

The output/readout layer representation (Y_i) for a very similar pair is $(1, 0, 0, 0, 0)$, rotated pair is $(0, 1, 0, 0, 0)$, zoomed pair is $(0, 0, 1, 0, 0)$, blurred pair is $(0, 0, 0, 1, 0)$ and different pair is $(0, 0, 0, 0, 1)$. The output weights convert the reservoir system state trajectories $\tilde{X}_k$ into the reservoir output y_i (whose values are reservoir predicted probabilities of each category). Ridge regression (see Appendix 2.5) is then used to train the output weights of the reservoir. While testing, a fractional probability is allotted to each output label, and the image pair is classified into the label with the highest probability.

2.2.1.2 Dual Reservoir Architecture

Input Layer In order to exploit the memory properties of RCs, for the dual reservoir architecture, the input is again converted to a time series. However unlike for the single reservoir architecture, we input each image (image 1 and image 2) column by column into two identical reservoirs, allowing the time axis to run across the columns of the image.

Reservoir Layer The reservoir state vectors for the two reservoirs (corresponding to image 1 and image 2), $\overrightarrow{r_1}(t)$ and $\overrightarrow{r_2}(t)$, are given by :

$$\begin{aligned}\overrightarrow{r_1}(t+1) &= \tanh\left(W^{\text{in}} \cdot \overrightarrow{u}(t) + W^{\text{res}} \cdot \overrightarrow{r_1}(t) + b\right) \\ \overrightarrow{r_2}(t+1) &= \tanh\left(W^{\text{in}} \cdot \overrightarrow{v}(t) + W^{\text{res}} \cdot \overrightarrow{r_2}(t) + b\right)\end{aligned} \tag{2.3}$$

where $\overrightarrow{u}(t)$, $\overrightarrow{v}(t)$ are inputs from image 1 and 2 respectively. The properties of the internal dynamics of the reservoir are identical and the same as the single reservoir. W^{res} for both reservoirs are identical and randomly chosen as outlined in the single reservoir case.

Output Layer Contrary to the single reservoir case, here we have two distinct reservoirs. The reservoir state trajectory X_k of one individual reservoir for one image k is then formed by concatenating the reservoir state vector as in 2.2. However, for the dual reservoir, we obtain two individual reservoir state trajectories, whose difference forms the reservoir system state trajectory $\tilde{X}_k$, that is used in the determination of the output weights.

The k^{th} reservoir system state trajectory is given by $\tilde{X}_k = |X_{k1} - X_{k2}|$, where X_{k1}, X_{k2} are the reservoir state trajectories corresponding to the images 1 and 2 respectively, for the kth input image pair. The readout layer representations for different relationships are the same as that in the single reservoir case. Ridge regression

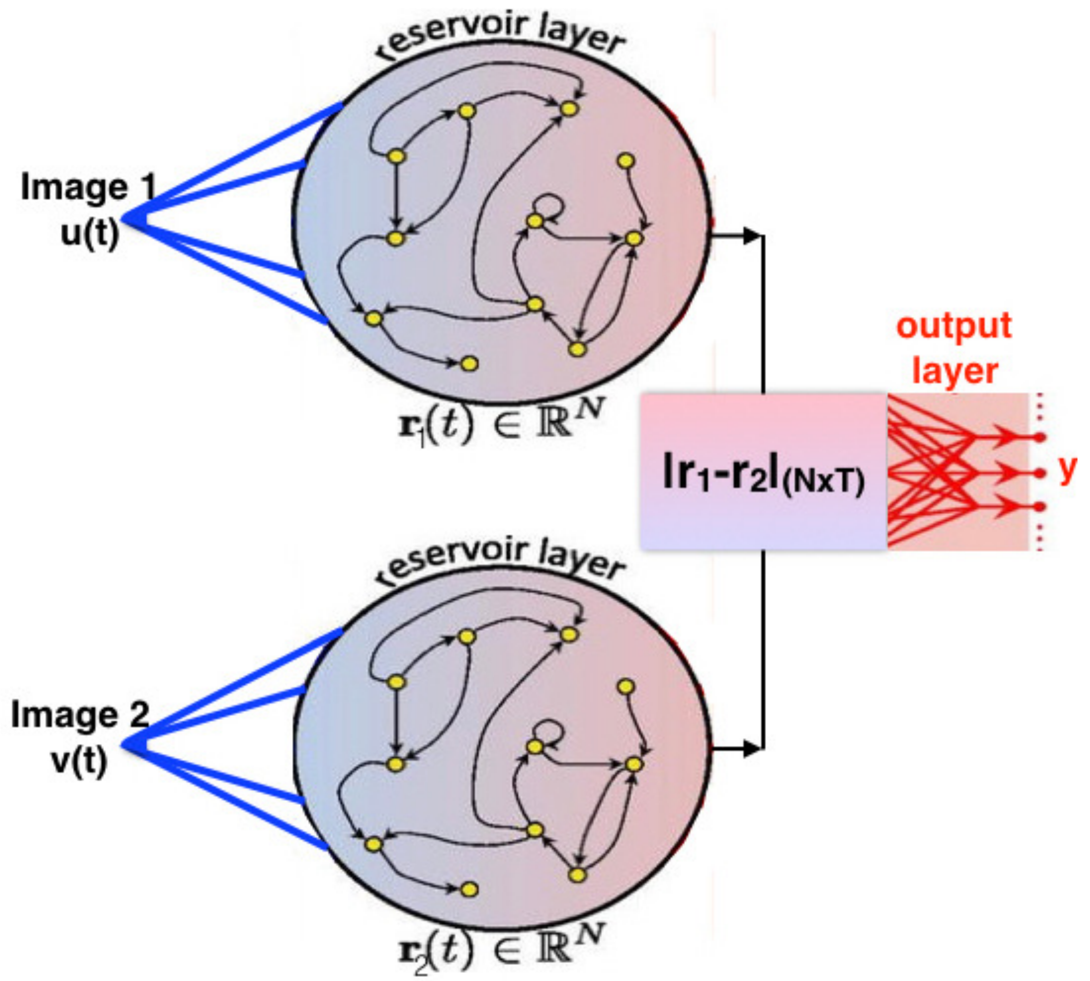

reservoir layer
Image 1
u(t)
$\mathbf{r}_1(t) \in \mathbb{R}^N$
output layer
$|r_1 - r_2|_{(N \times T)}$
y
reservoir layer
Image 2
v(t)
$\mathbf{r}_2(t) \in \mathbb{R}^N$

Figure 2.4: (a) Dual reservoir architecture with input state of the two images at time t denoted by $\overrightarrow{u}(t)$ and $\overrightarrow{v}(t)$, reservoir state vectors by $\overrightarrow{r_{1,2}}(t)$ and output state by $\overrightarrow{y}$.

(refer Appendix 2.5) is then used to train the output weights of the reservoir.

2.3 Results

2.3.1 Generalization to Untrained Image Classes

In this section we discuss the performance of the single and dual reservoir in the task of generalization of learned relationships. We present the results obtained on the MNIST dataset as proof of concept. The systems were trained on the five relationships – noise added, 90° rotation, blur, zoom, different (i.e. no relationship), on image-pairs of handwritten digits 0-5. Then they were tested on identifying the same relationships (in equal measure) between image pairs of handwritten digits 6-9 (digits they have never seen before). We use fraction correct (1- error rate) as a metric of performance.

In Fig. 2.5(a&c), we see that the reservoir performance increases rapidly with training set size and plateaus at around 200 training pairs. A training set size of ∼250 image pairs gives a reasonable trade-off between performance and computational efficiency. This is significantly lower than the training set sizes typically used in machine learning. Hence, our system achieves an important goal for many biomimetic architectures, i.e., the ability to train with relatively few training examples.. Fig. 2.5(b&d) shows that for a constant training data size (250 pairs) the performances increase as expected with reservoir size up to ∼750 nodes after which it saturates. The overall performance of the single reservoir appears to be better than that of the dual reservoir for a given reservoir size. Further, we examine the

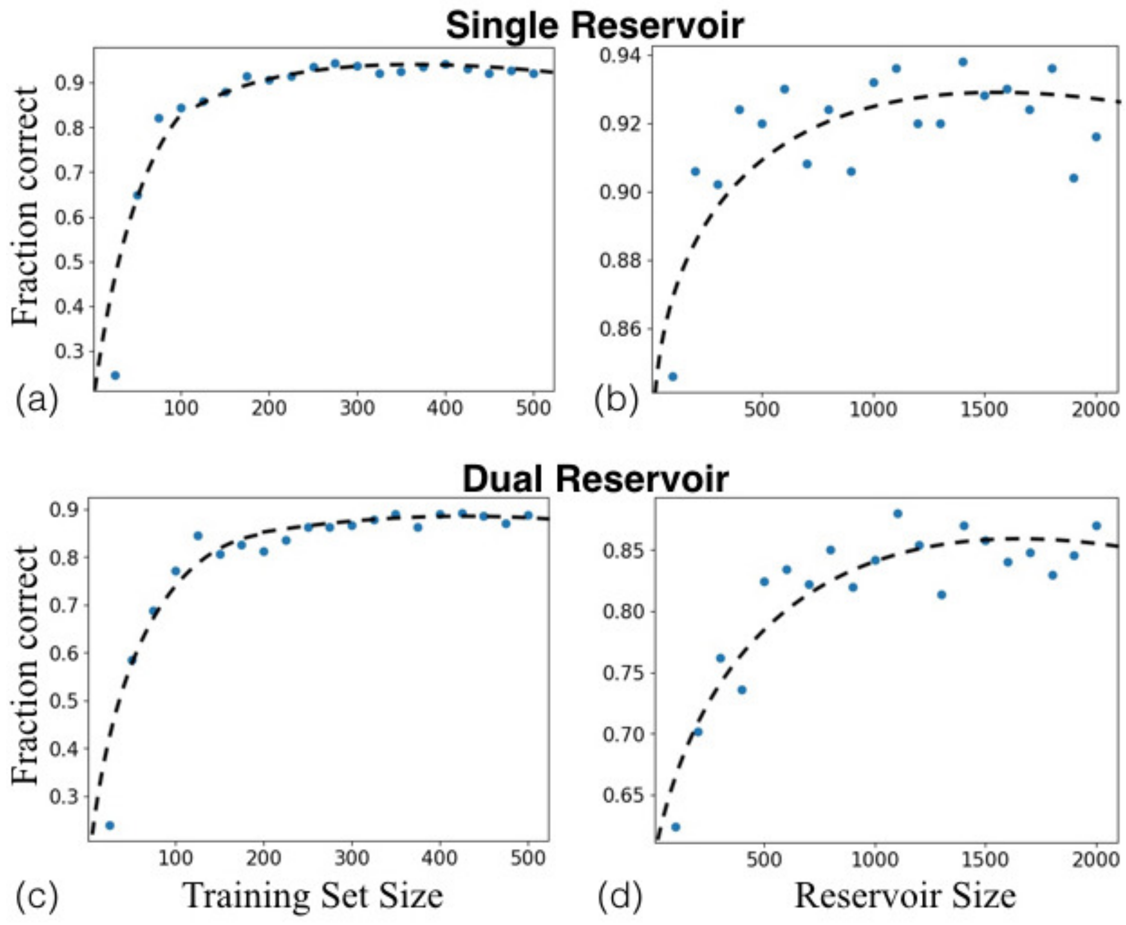


Figure 2.5: Fraction of image-pairs correctly classified versus training set size (a&c) and reservoir size (b&d). Single reservoir results show in (a&b). Dual reservoir results shown in (c&d). Dashed curves denote fourth order polynomial best fit. Reservoir size=1000 nodes for (a&c); training size=250 pairs for (b&d). Spectral radius $\gamma = 0.5$, sparsity $= 0.9$, testing size$=$ 500 pairs.

reservoir performance as a function of the spectral radius γ in Fig. 2.10; we observe a significant spread in performance values across γ, however we see no definitive pattern or conclusive correlation between the spectral radius and performance for the range investigated. While we notice a better performance for $\gamma = 0.1$ in the single reservoir, this is neither consistent across the single and dual reservoir architectures, nor is the boost in performance robust across all small γ values. For reference, reservoir activity, single node activity and output weights are shown in the Appendix 2.6.

2.3.2 Comparison with Siamese Neural Networks

The topic of generalized learning has, to the the best of our knowledge, not been satisfyingly addressed using a dynamical-systems-based machine learning approach that renders itself to easy analysis. To assess the effectiveness of our approach, we compare the performance of RCs with variants of a Siamese Neural Network (SNN), a successful pair-based machine learning technique. Specifically, we compare the single and dual reservoir model to three other architectures: a base SNN multilayer perceptron with 4 fully connected layers of 128 neurons each, a deep SNN multilayer perceptron with 8 fully connected layers of 128 neurons each, and a convolutional SNN (convolutional layer with 32 filters, 3×3 kernel and a rectified linear non-linearity, followed by 4 fully connected layers with 128, 64,32 and 2 neurons each). We compared performance for two binary classification tasks (Fig. 2.7(c)): (1) Learning the $90°$ rotation operator on MNIST image pairs (2) Learning to detect depth maps that come from the same visual scene class for the dataset of depth maps from a moving camera.

All SNN architectures were trained using contrastive loss (following [43]) and the optimizer Adadelta with a self adjusting learning rate. The objective of our SNN is not classification but differentiation. Hence the contrastive loss function that pulls neighbors together and pushes non-neighbors away is a natural choice compared to classification loss functions such as cross entropy. The single and dual

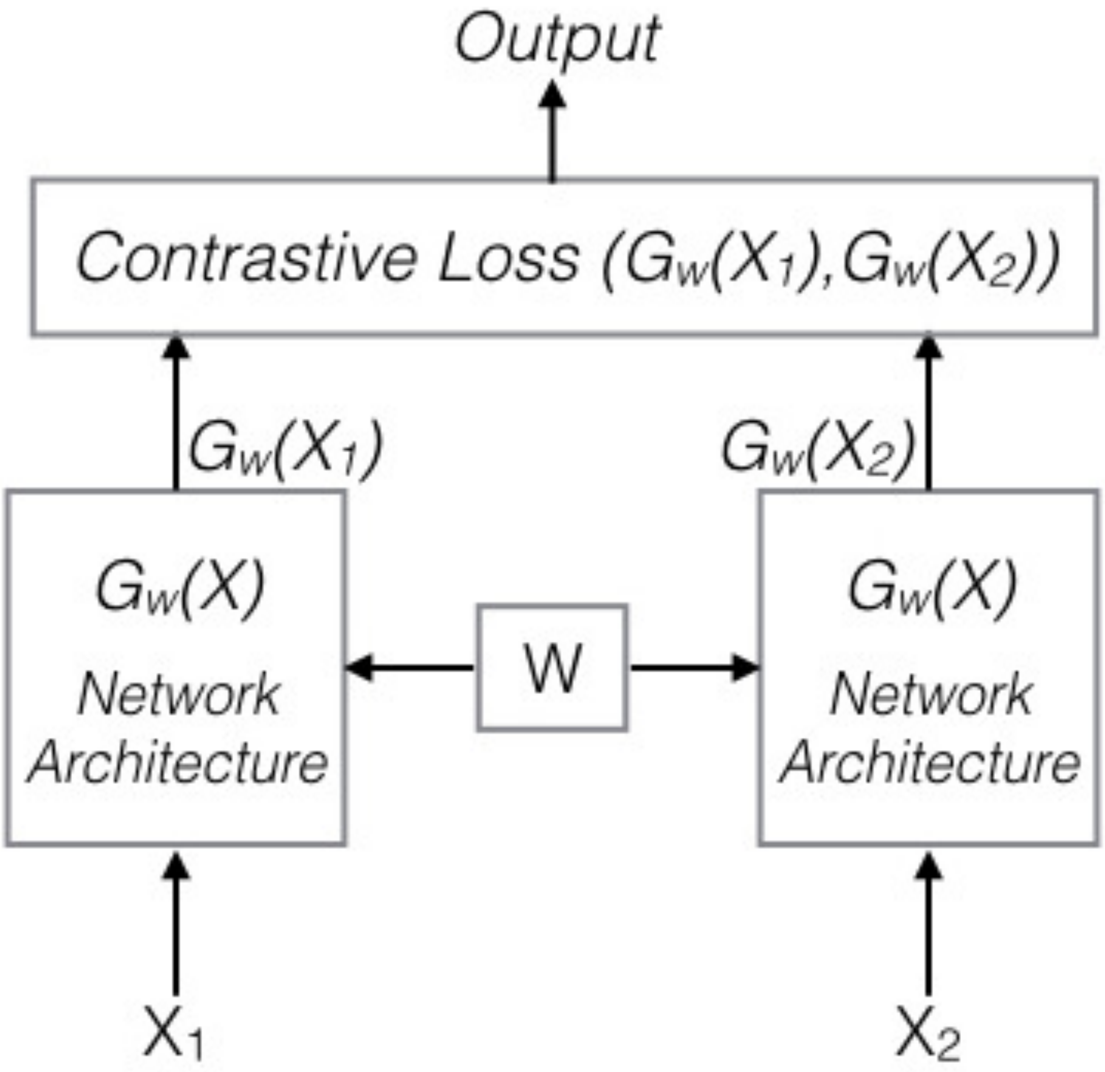


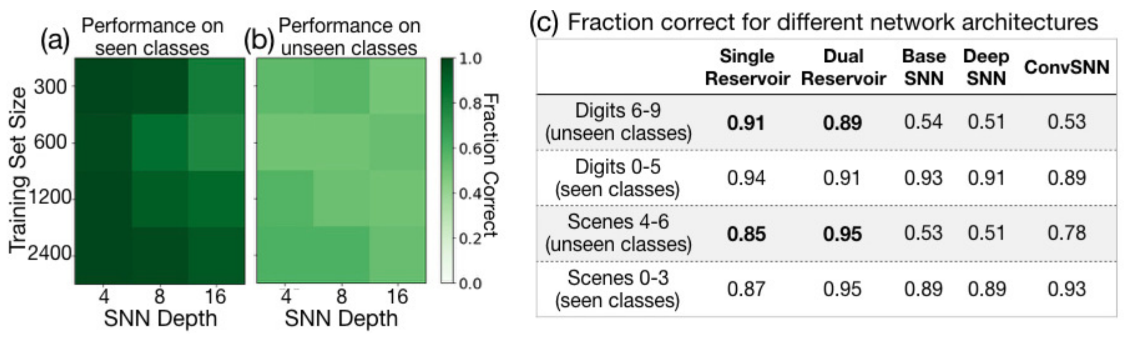
Figure 2.6: Siamese Network Architecture. Two inputs $X1$ and $X2$ are fed into two identical networks. $G_w(X)$ is the network transformation of the input X. W is the shared weights between the two heads of the siamese architecture.

reservoirs have 1000 nodes with $\gamma = 0.5$ and sparsity 0.9. Training is done for a 100 (40) epochs on the base and deep SNN multilayer perceptrons, 40 (20) epochs on the convSNN for MNIST (visual scenes) data respectively and once on the reservoirs on 500 image pairs.

While we present a select few SNN architectures here (and selected choices of parameters), we tried several other SNN architectures including VGG16-SNN and deep convSNN and found their performance to be comparable to the representative SNN performances we have shown. We also show SNN multilayer perceptron performance on varying depth (number of layers) and varying training data size (varied in the lower range compared to traditional deep network training sizes for comparison with the RCs and to motivate the question of biological plausibility) while testing on seen (trained) classes and unseen (test) classes (Fig. 2.7(a&b) respectively) and find that while the network performs fairly well on the trained classes, it performs consistently poorly on the unseen classes. The loss and accuracy plots for the SNN architectures for both tasks are in Appendix 2.7.

2.3.2.1 Generalized Learning of the Rotation Operator on the MNIST dataset

We train the reservoir on a simple binary classification task, i.e., classify image pairs from the MNIST dataset as having the relationship 'rotated' or not. Our training set consists of rotated and not rotated images of digits 0-5. Fig. 2.7(c))shows the fraction of correct classification by the RCs and the SNNs on the training classes



(c) Fraction correct for different network architectures

	Single Reservoir	Dual Reservoir	Base SNN	Deep SNN	ConvSNN
Digits 6-9 (unseen classes)	**0.91**	**0.89**	0.54	0.51	0.53
Digits 0-5 (seen classes)	0.94	0.91	0.93	0.91	0.89
Scenes 4-6 (unseen classes)	**0.85**	**0.95**	0.53	0.51	0.78
Scenes 0-3 (seen classes)	0.87	0.95	0.89	0.89	0.93

Figure 2.7: SNN perceptron performance on trained (seen) classes (a) and test (unseen) classes (b) of MNIST data as a function of training dataset size and SNN perceptron depth. (b) Classification accuracy (fraction correct) of the single and dual reservoir, base SNN, deep SNN and convSNN on seen (trained) classes and unseen (test) classes, on (1) identifying rotation transformation in MNIST images, and (2) identifying similar visual scenes from a moving camera. Training size: 500 images, testing size: 500 images.

(seen, digits 0-5) and testing classes (unseen, digits 6-9), as rotated or not rotated. We observe that, while the performance of all the networks is comparable on training set digits (digits 0-5), all the SNN architectures seems to have a near-random percent correct for untrained digits (6-9). Performance didn't improve on increasing the depth of the base SNN (Fig. 2.7(a&b)). The reservoir performance remains equally good over trained digits (0-5) and untrained digits (6-9), indicative of learning of the underlying relationship in the pairs and not the individual digits themselves. From observations in 2.3.4, we speculate that the generalization ability of the reservoir may be attributed to the convergence of parts of the reservoir system state trajectories for all rotated image-pairs. The dynamical properties of the reservoir create temporal patterns that enable memory. These properties may make learning on small datasets easier by requiring the RC to learn only some features of the dynamical patterns instead of the whole reservoir space. By contrast, the feedforward SNN isn't a dynamical system that enables temporally encoded memory, and training occurs explicitly on the images as opposed to the classes of relationships, which may be a possible cause for poorer performance while generalizing. For comparison, we present performance of a fully connected SNN upon varying depth, training data size in Fig 2.7(a&b).

2.3.2.2 Generalizing Similarities in Depth Perception from a Moving Camera

Identifying similarities in scenes, properties of scenes such as depth, style etc. from a moving camera is an important problem in the field of computer vision [44, 45]. We are interested in studying how the reservoir could learn and generalize relationships between images of visual scenes from a moving camera, frames of which may be non-linearly transformed with respect to each other. To demonstrate the practicality of our method, we implement it on depth maps from 6 different visual scenes recorded indoors in an office setting. Each visual scene has depth maps from 300 images, recorded as the camera is moved within a small distance ($\sim$30cm) and rotated within a small angle ($\sim 30^\circ$). We then train the networks to identify pairs of depth-maps as very similar (same visual scene) or different (different visual scenes), learning to capture small spatial and rotational invariance. Training is done on 500 images each from the first three visual scenes. We study whether the systems are able to generalize, i.e., identify relationships between depth maps from the other three visual scenes. Fig. 2.7(c) shows the reservoir performs significantly better on untrained scenes than the SNN, which classifies randomly. Both systems have a comparable and very high performance on the trained scenes. Thus, the reservoir is able to identify frames with similar depth maps from scenes it hasn't seen before. This has potential applications in scene or object recognition using a moving camera.

2.3.3 Combining Relationships

In this section we train the reservoir independently on the five relationships as in previous sections. However our test images have a linear combination of multiple relationships applied on them simultaneously (e.g., rotated as well as blurred). We then study the ability of the reservoir to recognize all the separate relationships applied to the test input pair.

Several tests on subsets/combinations of relationships were performed, however we only present a few demonstrative cases here.Training is done on the five individual relationships (noise, rotated, blurred, zoomed and different) for digits 0-5. Here we present testing on a combination of 3 relationships (90° rotation, zoom and blur), combination of 2 relationships (90° rotation and blur) as well as solo 90° rotation for digits 6-9. For testing image-pairs with n relationships applied simultaneously, we consider the reservoir to have classified correctly if the n highest label probabilities predicted by the reservoir during testing correspond to the n applied relationships. In Fig. 2.8 we observe that both the single and dual reservoirs perform very well (in terms of percent correct) at identifying combined relationships in images that they have never seen before. The single reservoir, on average, performs

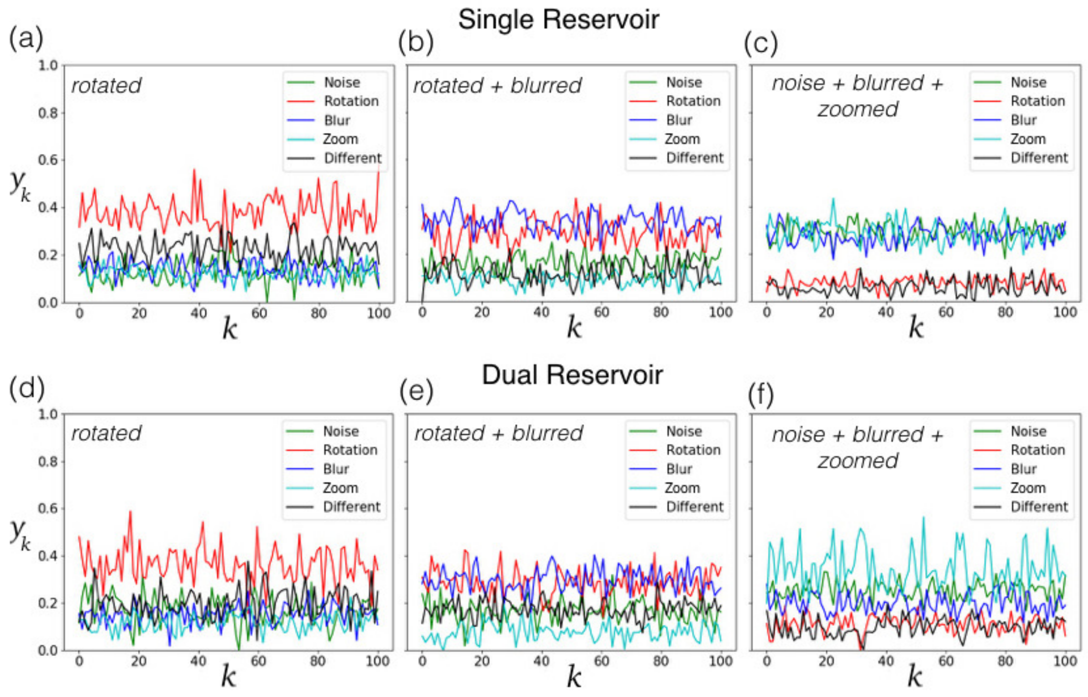


Figure 2.8: Graphing probabilities of the reservoir output (y_k) versus the image pair number k, for image pairs that are rotated (a,d) 2 combination: rotated and blurred (b,e), 3 combination: noise, blurred and zoomed (c,f), for single and dual reservoir respectively. The fraction correct, where classification is considered to be correct if the n predicted maximum probability labels are the n transformations applied to the test image-pair (shown on top left of each panel), are $0.97, 0.97, 1.0, 0.93, 0.84, 0.93$ for (a,b,c,d,e,f) respectively. γ=0.5, reservoir size=1000. Training digits: 0-5, testing digits: 6-9. Training size: 250 pairs.

slightly better than the dual reservoir. While there may be some inherent biases (ex. in Fig. 2.8(f), the dual reservoir shows a bias towards the zoomed category), inspite of the biases, the reservoirs are able to not only identify and separate linear combinations of these relationships, but also generlize this knowledge to previously unseen classes. We speculate that this ability to generalize combinations of multiple relationships is a result of overlap of reservoir system state trajectory clusters that correspond to the separate relationships. The cases shown in Fig. 2.8 are representative of the higher end of the range of accuracies obtained with other combinations (not presented here) as well.

2.3.4 Clustering Reservoir Space

Here we present a study of the features reservoir system state trajectories that may be important for generalization. In order to generalize, for a given relationship between the input image pair, there must be a corresponding relationship between the reservoir activity, dependent only on the relationship between the input images and not on the specific features of the input images themselves. As discussed earlier, the reservoir serves as a non-linear filter, whose goal for classification problems is to map the input into a high-dimensional space where the different relationships become linearly separable. In addition, the dynamical properties of the reservoir allow it to encode memory (because the reservoir state at time t depends on its state at time $t-1$). In this way, the reservoir's dynamical activity pattern in response to the input can highlight important features/relationships within the temporalized input.

In this section we illustrate that reservoir system state trajectories corresponding to a relationship do indeed cluster/become separable in reservoir space, allowing for generalization.

In Fig. 2.9, we plot a representation of 500 reservoir system state trajectories for each relationship (using different input digits; equally sampled) for (a) the single reservoir and (b) the dual reservoir. We show here the five standard relationships for MNIST - noise, rotate, blur, zoom, different, as well as one combined relationship - blur+rotate. A single reservoir system state trajectory has a very high dimensionality ($N_R \times T$). We are interesting in viewing this high dimensional data in a reduced dimensional space. Hence, we use the following dimensionality reduction techniques - first, we use Principal Component Analysis (PCA) to extract the 100 largest principal components (PCs) of each reservoir system state trajectory. We then use the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [46] on the extracted PCs for further dimensionality reduction. t-SNE, being particularly well suited for the visualization of high-dimensional datasets, has been used very successfully in recent years along with PCA.

We visualize the reservoir system state trajectories in a two-dimensional space and find that relationships between images cluster. We observe from Fig. 2.9, that the separation of relationships is more prominent for the dual reservoir (Fig. 2.9(b)) compared to the single reservoir (Fig. 2.9(a)). This may be attributed to the architecture of the dual reservoir, which takes the difference between the individual image trajectories, thus more directly encoding the classification features, i.e., the features of the *differences* between the image pair (blur, scale, rotation feature etc.), unlike

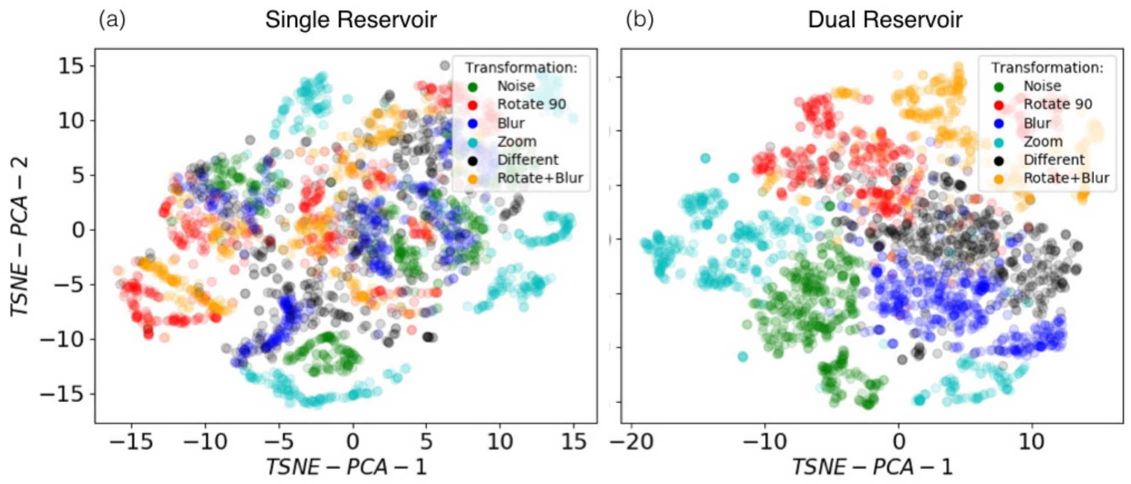


Figure 2.9: 500 reservoir system state trajectories for each relationship in the reduced dimensional space spanned by the two largest components obtained using t-SNE on the 100 largest principal components of the reservoir system state trajectory for (a) single reservoir and (b) dual reservoir. Input images: digits 0-9 of MNIST dataset. N_R: 1000. t-SNE iterations: 300, perplexity: 40.

the single reservoir. However, we note that despite the fact that we see better separation of of clusters for the dual reservoir, the single reservoir slightly outperforms it on the MNIST data (see Fig 2.7 (b)). One possible reason for this is that Fig. 2.9 only shows a two-dimensional representation of the clusters and perhaps the single reservoir shows a better separation than the dual reservoir in higher dimensions. Another possible reason is that the reservoir system state trajectories do not take into account the training, which, in addition to the clustering of reservoir trajectories, is a key component of the reservoir's performance. There may be some features of the reservoir system state trajectories from the single reservoir architecture that are not captured in Fig. 2.9 yet allow for more effective training.

We speculate that the separation of the system trajectories in reservoir space is important for generalizing with small datasets when using a linear training procedure like ours. Here, we have demonstrated that the reservoir does indeed function as an effective non-linear filter that acts upon the image pairs and separates them in high-dimensional reservoir space into clusters characterized by the relationships between the two input images.

2.4 Conclusion

In this paper we have used Reservoir Computers (RCs) for image classification problems that involve generalization of relationships learned between image pairs using limited training data. While image classification has been studied extensively before, here we present a biologically-inspired recurrent network approach that not only generalizes learning, but also allows us to build an interpretation of the results. We present our results on the simple handwritten digits database, as well as on a video dataset of depth maps from a moving camera, useful in identification of similar scenes from different camera perspectives. We observe that the reservoir system state trajectories obtained from input image-pairs with the same relationship cluster in reservoir space. This can be interpreted as the reservoir trajectory exhibiting dynamical patterns corresponding to image-pair relationships. Because the reservoir system state trajectories separate in the high dimensional reservoir space according to the input pair relationships, a linear method of training such as ridge regression is effective. The separability of the clusters allows for training to converge relatively quickly and with limited training data. By reducing dimensionality from the reservoir space to the space mapped by the clusters, we obtain a well-generalizing reservoir using only a small training dataset, whereas contemporary methods such as deep learning require much larger datasets. Although we see strong performance with a sparse reservoir and few training images in our proof-of-concept study, we suspect that for more complex input images, a more powerful (and possibly more sophisticated) architechture would be required to match performance.

We find that the RCs perform significantly better than deep/convolutional SNNs for the task of generalization (sec. 2.6). From a computation perspective, the RC has the added advantage of speed since only the output weights are trained and the reservoir is sparsely connected. Our system is biologically-inspired in two ways. First, the learning mimics biological learning through comparisons and analogies. Second, the internal dynamics of the reservoir are known to broadly resemble neural cortex activity. We conclude that although state of the art machine learning techniques such as SNNs (for pairwise input) work exceedingly well for traditional image classification, they do not work as well for generalization of learning, for which RCs significantly outperform them in our study, perhaps due, in part, to their dynamical 'memory' properties that lead to distinctive dynamical patterns in the reservoir state trajectories. While more complex architectures such as LSTMs may also have much greater success in generalization than non-recurrent architectures, they require much larger training data and more computational power. However, implementing the experiment on an LSTM network could be an interesting future direction, especially for more challenging generalization problems.

Data Availability The visual scenes captured from a moving camera (images and depth maps) dataset used to support the findings of this study are available from the corresponding author upon request. The dataset has not been included in the article/ supplementary material due to its large size. Code is available at https://github.com/a-jan-tusk/Reservoir-for-generalization.

Funding Statement This research was supported in part by the University of Maryland's COMBINE (Computation and Mathematics for Biological Networks) program through NSF award number 1632976, as well as a DoD contract under the Laboratory of Telecommunication Sciences Partnership with the University of Maryland.

2.5 Ridge Regression and Training

Only the output weight matrix W^{out} is optimized during training such that it minimizes the mean squared error $E(y, Y)$ between the output of the reservoir y and the target signal Y. The reservoir output is:

$$y = W^{\text{out}} \delta X \tag{2.4}$$

$W^{\text{out}} \in \mathbb{R}^{N_y \times N_R}$ where N_y is the dimensionality of the readout layer.

δX or the concatenated reservoir system state trajectory is the matrix containing all reservoir system state trajectories during training phase, $\delta X = [\tilde{X}_0 \; \tilde{X}_1 \; \ldots \; \tilde{X}_M]$ where M is the total number of training image-pairs, input one after the other, and $Y = [Y_0 \; Y_1 \; \ldots \; Y_M]$ is the matrix containing the corresponding readout layer for all images. W^{out} is computed using Ridge Regression (or Thikonov regularization) [47], which adds an additional small cost to least square error, thus making the system robust to overfitting and noise. Ridge regression calculates W^{out} by minimizing squared error $J(W^{\text{out}})$ while regularizing the norm of the weights as follows:

$$J(W^{\text{out}}) = \eta |W^{\text{out}}|^2 + \sum_i ((W^{\text{out}})^T \delta X_i - Y_i)^2. \tag{2.5}$$

where δX is the concatenated reservoir system state trajectories over all training image pairs, Y contains the corresponding label representations and the summation is over all training image pairs. Upon solving the stationary condition $\frac{\partial J}{\partial W^{\text{out}}} = 0$ is

$$W^{\text{out}} = (\delta X \delta X^T + \eta I)^{-1} \delta X Y. \tag{2.6}$$

where η is a regularization constant and I is the identity matrix.

2.6 Reservoir Dynamics and Performance

We present the performance of the single and dual reservoir as a function of spectral radius γ. γ is varied from 0 to 1 while looking for the optimal performance region where the reservoir has memory or is in the 'echo state' (edge of charos) [48], however we find no indicative pattern (Fig. 2.10).

Performance with Spectral Radius: Fig. 2.10 shows fraction correct as a function of reservoir dynamics for (a) single and (b) dual reservoir.

Reservoir Dynamics:

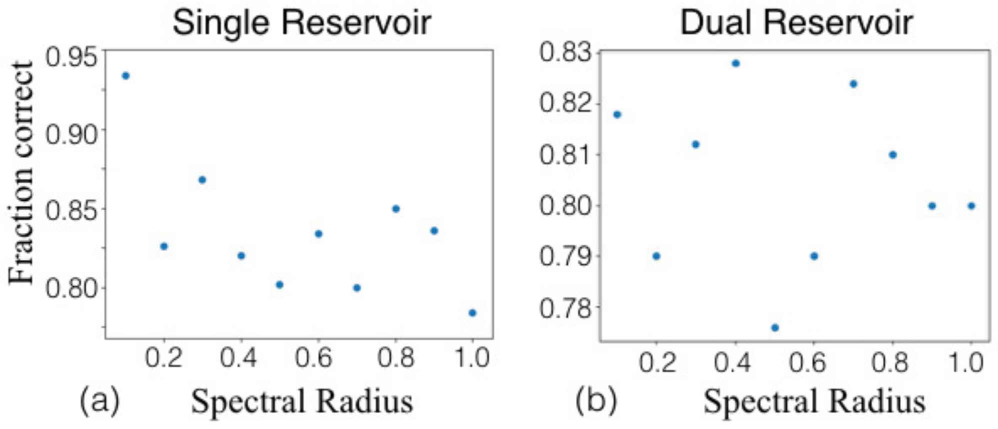


Figure 2.10: Fraction correct as a function of spectral radius for (a) single reservoir (b) dual reservoir. N_R=1000, training size=250 pairs, testing size=500 pairs $\gamma = 0.5$, sparsity = 0.9.

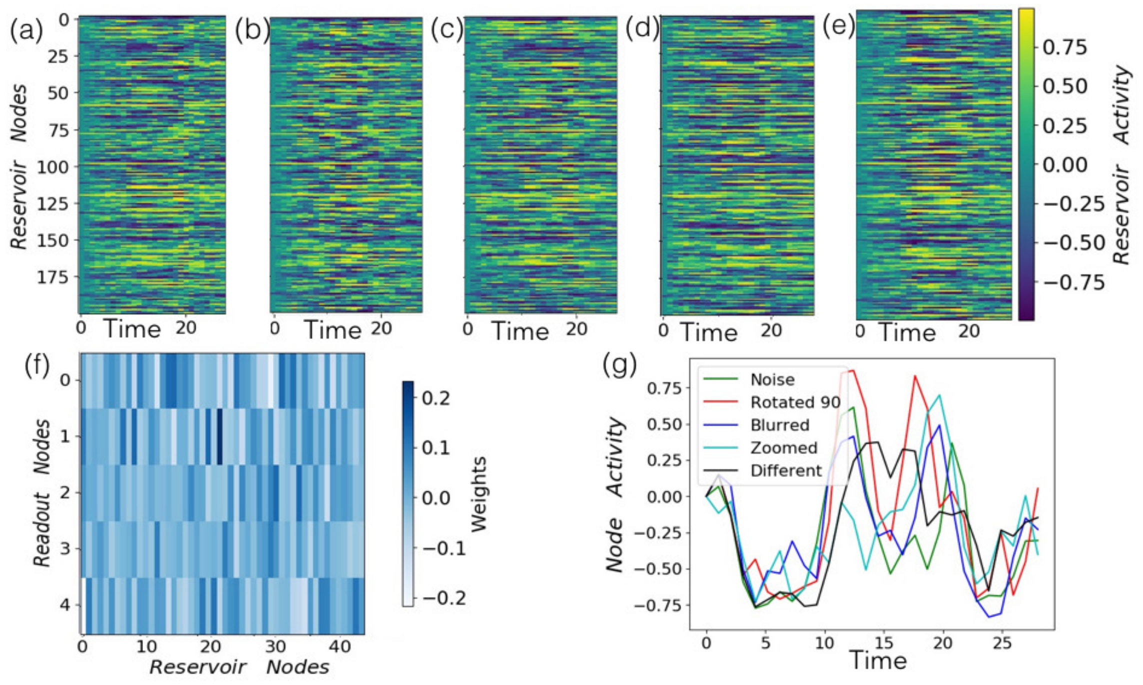


Figure 2.11: Reservoir activity for the single reservoir architecture. (a), (b), (c), (d), (e) show the differential reservoir activity of 200 nodes over 28 timesteps for input relationships noise, rotated, zoomed, blurred and different respectively. (f) shows the output weight matrix(W^{out}) for 50 reservoir nodes. (g) shows activity of a random node for all output labels over 28 timesteps. N_R: 1000, $\gamma = 0.5$, sparsity= 0.9.

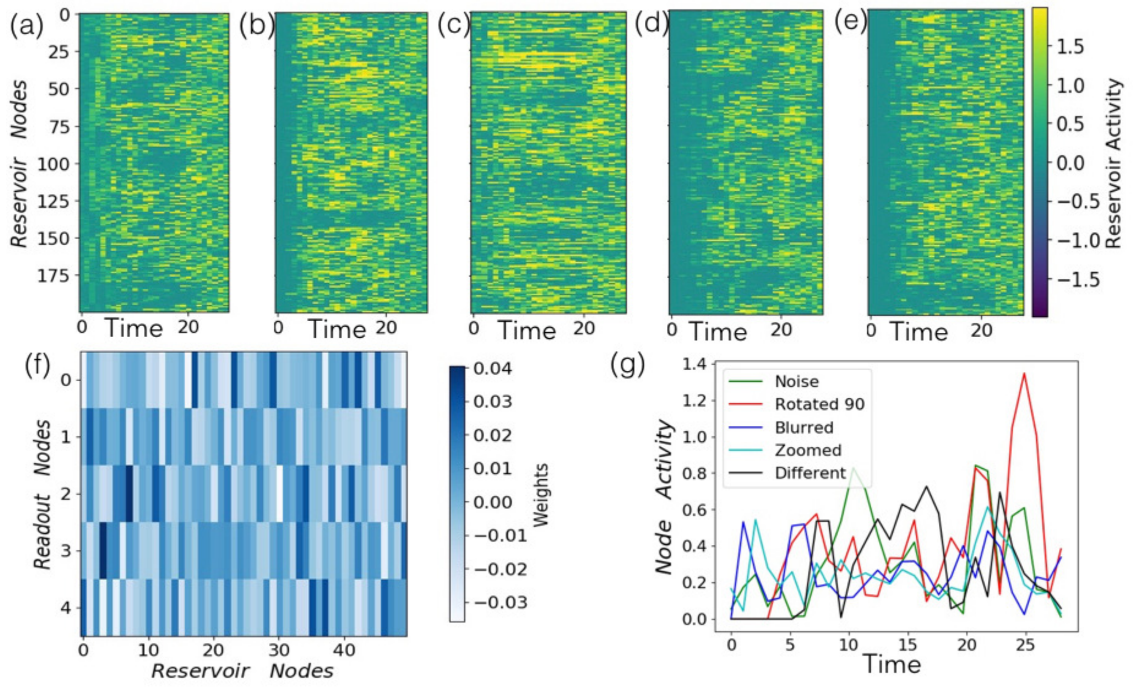


Figure 2.12: Reservoir activity for the dual reservoir architecture. (a), (b), (c), (d), (e) show the differential reservoir activity of 200 nodes over 28 timesteps for input relationships noise, rotated, zoomed, blurred and different respectively. (f) shows the output weight matrix(W^{out}) for 50 reservoir nodes. (g) shows activity of a random node for all output labels over 28 timesteps. N_R: 1000, $\gamma = 0.5$, sparsity= 0.9.

For completion, we plot the reservoir activity, i.e., averaged reservoir system state trajectory corresponding to our five relationships applied to the MNIST dataset, output weights, and single node activity. Fig. 2.11 and 2.12 show plots of activity in the single reservoir and dual reservoir architecture respectively. We see that the individual node (f) itself doesn't encode any decipherable information. However each output label (a,b,c,d,e) has a slightly different signature in reservoir space.

2.7 Loss and Accuracy of SNNs

In Fig. 2.13 we plot the training loss and accuracy for the base SNN (4 layers), deep SNN (8 layers), and convolutional SNN for the two tasks of identifying rotation operator in MNIST and identifying similar visual scenes from a moving camera. Since training data is small, losses converge fairly quickly over epochs. The optimizer Adadelta, that employs a variable learning rate, was used in training.

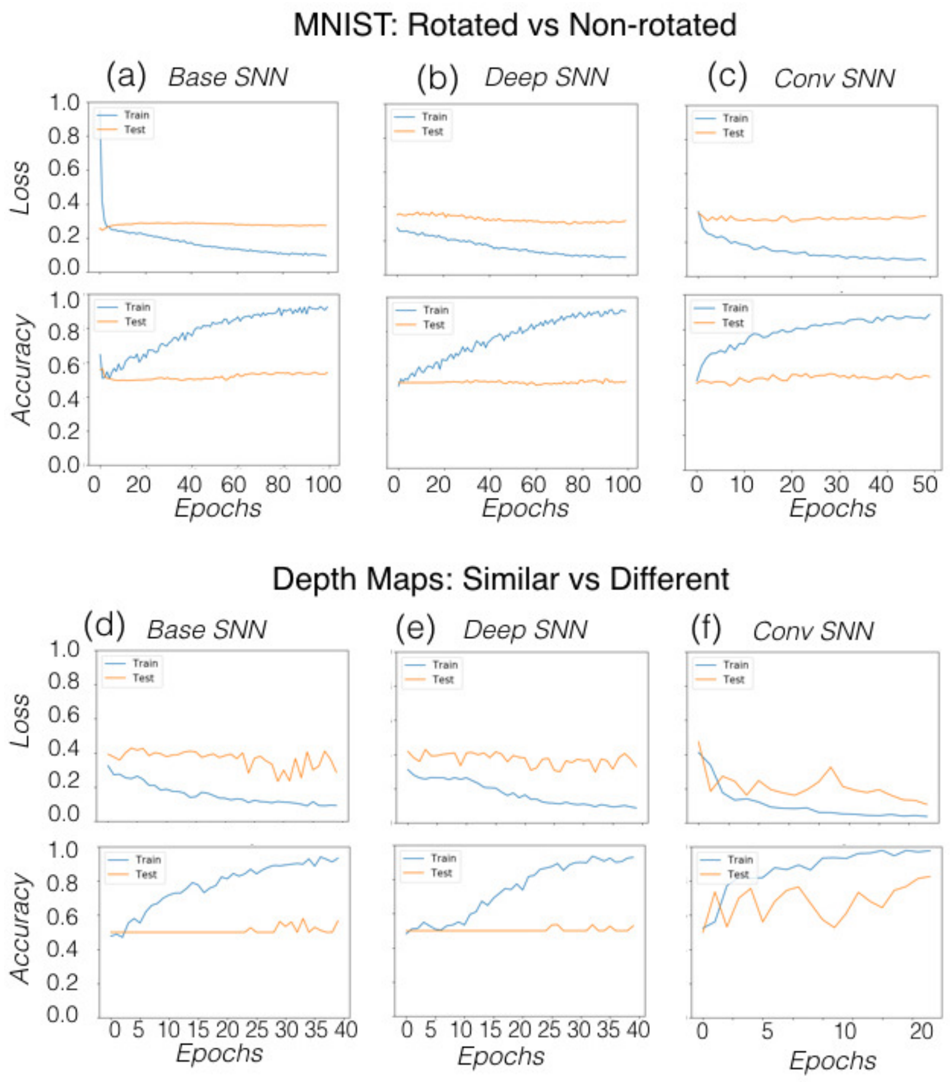


Figure 2.13: Plot of training loss and accuracy for (a&c) base siamese network, (b&d) deep siamese network, and (c&f) convolutional siamese network.

Chapter 3: Separation of Chaotic Signals by Machine Learning

This chapter is currently submitted to the journal Chaos and is on arxiv: Sanjukta Krishnagopal, Edward Ott, Michelle Girvan, and Brian Hunt. "Separation of Chaotic Signals by Reservoir Computing"

3.1 Introduction

The problem of extracting a signal from ambient noise has had wide applications in various different fields such as signal processing, weather analysis [49], medical imaging [50], and cryptography[51]. We consider the related (but potentially more difficult) problem of separating two or more similar signals that are linearly combined. This is a version of the cocktail party problem, i.e., how do people at a cocktail party separate out and focus on the voice of the person they are interested in listening to from a combination of several voices that reaches their ear. In our version of the problem, signals are generated by chaotic processes. If the equations governing these processes are known, the problem can be attacked for example, by using chaos synchronization [52, 53, 54, 55]. We consider instead an approach that relies only on data. Our problem is similar to that of blind source separation for chaotic signals [56, 57, 58], but the latter problem typically assumes that multiple linear combinations of the signals are being measured, with the number of independent measurements at least as large as the number of signals. In contrast, our method requires only one linear combination of the signals to be measured after training is complete. For training, we assume that we have finite-time samples of the separate component signals, and our method learns from these samples to separate subsequently measured combination signals into their components. We also note that Wiener obtained an optimal linear solution for the signal separation problem (the 'Wiener filter'). In contrast, the technique we use is fundamentally nonlinear, and, as we will show, can significantly outperform the linear technique.

Machine learning techniques have been very successful in a variety of tasks such as image classification, video prediction, voice detection etc. [4, 59, 60]. Recent work in speech separation includes supervised machine learning techniques such as deep learning [61], support vector machines [62], as well as unsupervised methods such as non-negative matrix factorization [63]. Our approach is based on a recurrent machine learning architecture originally known as Echo State Networks (originally proposed in the field of machine learning) [10] and Liquid State Machines (originally proposed in the field of computational neuroscience) [11], but now commonly referred to as Reservoir Computing [12]. Reservoir computing has been applied to several real-world problems such as prediction of chaotic signals [3], time-series analysis [64], similarity learning [65], electrocardiogram classification [66], short-term weather forecasting [67] etc. We expect that, although our demonstrations in this paper are for Reservoir Computing, we expect similar results could be obtained with other types of machine learning using Recurrent Neural Network architectures, like LSTM and Gated Recurrent Units [68]; however, based on results in [68], we expect that these architectures will have a significantly higher computational cost for training than reservoir computing in the cases we consider.

We train a Reservoir Computer (RC) as follows. Training data consists of an input time series and a desired output time series. The RC consists of the input layer, the reservoir, and the output layer. The input time series is processed by the input layer as well as the reservoir; the resulting reservoir states are recorded as a vector time series. Then, linear regression is then used to find an output weight matrix (the output layer) that fits the reservoir state to the desired output. The internal parameters of the input layer and the reservoir are not adjusted to the training data, only the output weights are trained.

In this article, we input a linear combination of two chaotic signals to the RC and train it to output an estimate of one of the signals. In the simplest case, which we describe in section 3.3, the ratio between the amplitudes of the signals is known in advance. In section 3.4, we consider the case in which this ratio is unknown. In this case, we first train a RC to estimate this ratio which we call the mixing fraction, and then train another RC to separate the chaotic signals given the ratio. We demonstrate our results on signals from the Lorenz system in the chaotic regime.

We describe our implementation of reservoir computing in section 3.2.1 and review the Lorenz system in section 3.2.2. We compare our results with an approximation to the Wiener filter, computed by estimating the power spectra of the signals using the same training data we use for the RC. If computed from the exact spectra, the Wiener filter is the optimal linear seperator for uncorrelated signals (see

Appendix .1). Motivated in part by this comparison, we consider three scenarios is section 3.3: separating signals with different evolution speeds on the same attractor, signals with different Lorenz parameters, and signals with both the parameters and evolution speed perturbed in such a way that the spectra of the signals almost match. We present our conclusions in section 3.5, a brief summary of which are as follows: (1) the RC is a robust and computationally inexpensive chaotic signal separation tool that outperforms the Wiener filter, (2) we use a dual-reservoir computer mechanism to enable separation even when the amplitude ratio of the component signals in the mixed signal is unknown. Here, the first RC estimates a constant parameter, the mixing fraction (or amplitude ratio), whereas the second RC separates signals given an estimated mixing fraction.

3.2 Methods

We consider the problem of estimating two scalar signals $s_1(t)$ and $s_2(t)$ from their weighted sum $u(t) = \beta_1 s_1(t) + \beta_2 s_2(t)$. We normalize $s_1(t)$, $s_2(t)$ and $u(t)$ to have mean 0 and variance 1. We assume $s_1(t)$ and $s_2(t)$ to be uncorrelated, in which case our normalization implies, $\beta_1^2 + \beta_2^2 = 1$. Let $\alpha = \beta_1^2$, so that

$$u(t) = \sqrt{\alpha} s_1(t) + \sqrt{1-\alpha} s_2(t) \tag{3.1}$$

We call α the mixing fraction; more precisely it is the ratio of the variance of the first component of $u(t)$ to the variance of $u(t)$.

In section 3.3, we assume that the value of α is known. In section 3.4, we consider the case where α is unknown. In both sections, we assume that limited-time samples of $s_1(t)$ and $s_2(t)$ are available, say for $0 \leq t \leq T$, and we use these samples to train the reservoir. Our methods require no knowledge of the processes that generate $s_1(t)$ and $s_2(t)$, but we assume that these processes are stationary enough that the training samples are representative of the components of future instances of u(t). For our numerical experiments, we generate $s_1(t)$ and $s_2(t)$ from the Lorenz family of chaotic dynamical systems (see section 3.2.2). More generally, the same method can be used if $s_2(t)$ is a combination of multiple signals and/or noise.

3.2.1 Reservoir Computer

There are many variations in implementation; in this paper we adopt the Echo State Network approach of reservoir computing proposed by Jaeger [10]. The reservoir computer has three components (Fig. 1), a linear input layer with M_i scalar input (one for each component of the M-dimensional input signal $\boldsymbol{u}(t)$), a recurrent, nonlinear reservoir network with N dynamical reservoir nodes driven both by inputs as well as by delayed feedbacks from the other reservoir nodes, and a linear output layer with M_o scalar outputs, as shown in Fig. 3.1. We describe a method for general M_i and M_o, but in our experiments we will always use $M_i = M_o = 1$, with $s(t)$ equal to either $s_1(t)$ or $s_2(t)$, or (in section 3.4), the constant α.

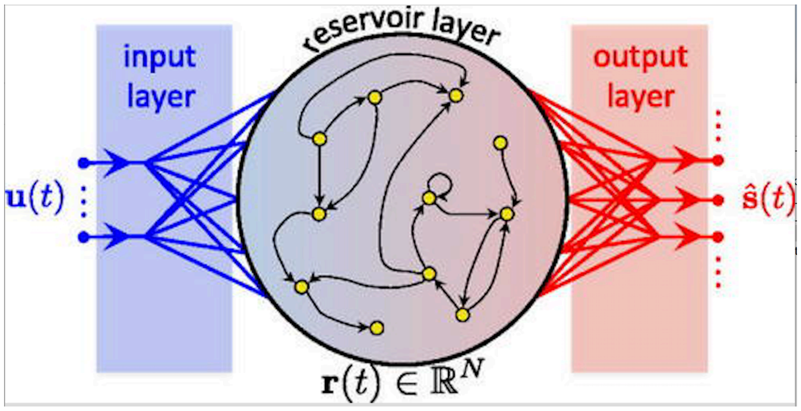


Figure 3.1: Reservoir architecture with input state at time t denoted by $\boldsymbol{u(t)}$, reservoir state by $\boldsymbol{r(t)}$, and output state by $\boldsymbol{\hat{s}(t)}$. The output layer is trained so that $\boldsymbol{\hat{s}(t)}$ approximates the desired output signal $\boldsymbol{s(t)}$.

3.2.1.1 Input Layer

The input layer is described by a $N \times M_i$ matrix W^{in} where elements are randomly chosen between to be a uniform distribution between $[-k, k]$, where k is a hyper-parameter to be chosen later.

3.2.1.2 Reservoir Layer

The reservoir can be though of as a dynamical system with state vector $\boldsymbol{r}(t)$ at time t given by :

$$\boldsymbol{r}(t + \Delta t) = (1 - a)\boldsymbol{r}(t) + a \tanh\left(\boldsymbol{W}^{\text{in}}\boldsymbol{u}(t) + \boldsymbol{W}^{\text{res}}\boldsymbol{r}(t) + b\right) \tag{3.2}$$

The state of the reservoir at time t is given by $\boldsymbol{r}(t)$, of size N. The notation $tanh(\dots)$ with a vector argument is defined as the vector whose components are the hyperbolic tangents of the corresponding components of the argument vector. The leakage parameter a, which is bounded between $[0, 1]$, determines the speed at which the input affects or leaks into the reservoir. Both a and the bias magnitude b are hyper-parameters. The recurrent connection weights $\boldsymbol{W}^{\text{res}} \in \mathbb{R}^{N\times N}$ are initialized randomly between -1 and 1; then, $\boldsymbol{W}^{\text{res}}$ is normalized by a multiplication of all its components by a constant chosen so that the spectral radius (maximal absolute eigenvalue of $\boldsymbol{W}^{\text{res}}$) λ, which is another hyper-parameter. A discussion of the effect of spectral radius on performance is presented in [42]. Typically N is much larger than M_i, so that the reservoir transforms the input from the input space into a much higher dimensional reservoir space. The sparsity sp of the reservoir layer is a measure of how sparsely connected the nodes are. Sparsity of zero means all-to-all coupling, and sparsity of one means no connections at all.

3.2.1.3 Output Layer

After running the reservoir (Eq. 3.3) for a transient time period -100 to 0, we form the reservoir state trajectory, $\boldsymbol{R}$, is formed by concatenating the reservoir state vectors (the state of all reservoir nodes) at every timestep $\boldsymbol{r}(t)$ corresponding to the input $u(t)$ as follows:

$$\boldsymbol{R} = [\boldsymbol{r}(1), \boldsymbol{r}(2), \ldots, \boldsymbol{r}(T)] \tag{3.3}$$

Thus, $\boldsymbol{R}$ is an augmented matrix of size $N \times T$ where T is the training time, i.e., number of time steps for which training data is available. During training, the output weights are found by mapping the reservoir state trajectory to the desired output layer representation $\boldsymbol{S} = [\boldsymbol{s}(1), \boldsymbol{s}(2), \ldots, \boldsymbol{s}(T)]$ over T samples. Only the weight matrix $\boldsymbol{W}^{\text{out}}$ is optimized during training such that the the mean square error $E(\hat{s})$ between the output of the reservoir and the target signal s is minimized. The reservoir output $\hat{\boldsymbol{s}}\boldsymbol{(t)}$ is obtained through the output weights as follows:

$$\hat{\boldsymbol{s}}(t) = \boldsymbol{W}^{\text{out}}\boldsymbol{r}(t) \tag{3.4}$$

$\boldsymbol{W}^{\text{out}} \in \mathbb{R}^{M_o \times N}$ where M_o is the dimensionality of the readout layer.

Ridge Regression (or Thikonov regularization) [47], is used for fitting, thus making the system robust to overfitting and noise. Ridge regression minimizes squared error while regularizing the Euclidian norm of the weights as follows:

$$J(\boldsymbol{W}^{\text{out}}) = \eta \|\boldsymbol{W}^{\text{out}}\|^2 + \|\boldsymbol{W}^{\text{out}}\boldsymbol{R} - \hat{\boldsymbol{s}}\|^2. \tag{3.5}$$

where η is a regularization constant, which is also a hyper-parameter.

Once training is done, the RC predicts state variables for times $t \geq T$ through $\boldsymbol{W}^{\text{out}}$ and Eq. 3.4.

3.2.2 Data: Lorenz system

Our examples are based on the Lorenz equations [69]:

$$\begin{aligned} \dot{x} &= \sigma(y - x) && (3.6) \\ \dot{y} &= -xz + \rho x - y && (3.7) \\ \dot{z} &= xy - \beta z && (3.8) \end{aligned}$$

The Lorenz attractor is a strange attractor that arises in these system of equations. The Lorenz system is known to be chaotic for the parameter values: $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$ [70], and appears to be chaotic for other parameter values we use in this article. We generate trajectories using the 4^{th} order Runge-Kutta method with step size 0.01, and we sample them every five steps, i.e., 0.05 time units.

3.3 Results: Generalization Separation with Known Mixing Fraction

We calculate the error between the trained reservoir output $\hat{s}(t)$ and the desired signal $s_1(t)$ as follows. The mean square reservoir error E_R is:

$$E_R = \frac{< (s_1 - \hat{s})^2 >}{\min_\zeta < (s_1 - \zeta u)^2 >} \tag{3.9}$$

where $s_1 \in [x_1, y_1, z_1]$, i.e., one of the components of the signal (typically we use $s_1 = x_1$). ζ scales the input so that the denominator indicates the root mean square error in the absence of any processing by the reservoir.

The Wiener filter [1] is known to be the best possible *linear* solution to our signal separation problem, and is also commonly used in tasks such as signal denoising, image deblurring etc. A detailed explanation of the Wiener filter is given in Appendix .1. In this article, we always compare the RC with a Wiener filter, which uses a window of 500 for estimating the spectrum. The mean square Wiener error measure E_W is:

$$E_W = \frac{< (s_1 - \hat{s}_w)^2 >}{\min_\zeta < (s_1 - \zeta u)^2 >} \tag{3.10}$$

where $\hat{s}_w$ is the output of the Wiener filter.

3.3.1 Separating Lorenz signals with different parameters

In this section, the reservoir input consists of a combination of two x-component signals from Lorenz systems with different parameter values. The signal x_1 has parameters $\boldsymbol{p}_1 = \{\sigma = 10, \rho = 28, \beta = 8/3\}$. The signal x_2 has parameters $\boldsymbol{p}_2 = 1.20 \times \boldsymbol{p}_1$.

Several parameters related to the reservoir system must be chosen for running experiments. These parameters include reservoir size N, spectral radius γ, leakage parameter a, and the length of training signal. In Fig. 3.2, we vary each of these parameters individually while keeping the others constant in order to identify an appropriate set of values.

Fig. 3.2 shows the performance of the reservoir in separating the x component of the s_1 trajectory for varying parameters given $\alpha = 1/2$. The reservoir errors are averaged over 10 random initializations of the reservoir. The error bars denote standard error, i.e., the standard deviation across l random initializations over $\sqrt{l}$. We observe a downward trend in error with increasing reservoir size, which seems

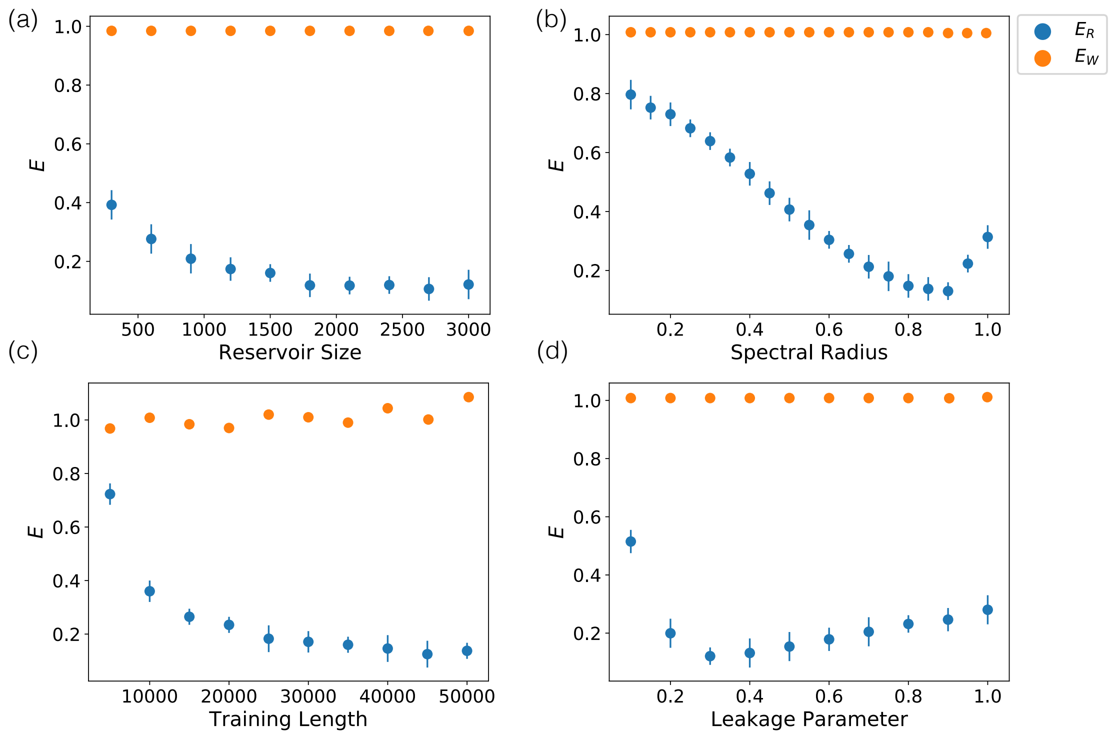


Figure 3.2: Reservoir error E_R and Wiener error E_W over a test length of 5000 timesteps for (a) varying reservoir size N, (b) spectral radius λ, (c) training length and (d) leakage parameter a. All parameter values except the parameter being varying: $\Delta t = 0.05$, sparsity $sp = 0.95$, $N = 2000, \lambda = 0.9, a = 0.3, k = 0.13$ training length $= 50000, \alpha = 0.5$, $\boldsymbol{p}_2 = 1.2\boldsymbol{p}_1$.

to saturate at about 2000 nodes. Panel (a) shows that a reservoir size of $N = 2000$ gives a reasonable trade-off between performance and computational efficiency. Panel (b) shows that $\lambda = 0.9$ seems to result in the best separation, with all other parameters constant. As seen in panels (c,d), the optimal leakage parameter a and training length are $0.3, 50000$ respectively. Other parameters chosen are input strength normalization $k = 0.13$ such that the input does not saturate the tanh curve. These parameter values are used for the rest of this paper unless mentioned otherwise.

Fig. 3.3 (a) presents the performance of the reservoir as a function of the

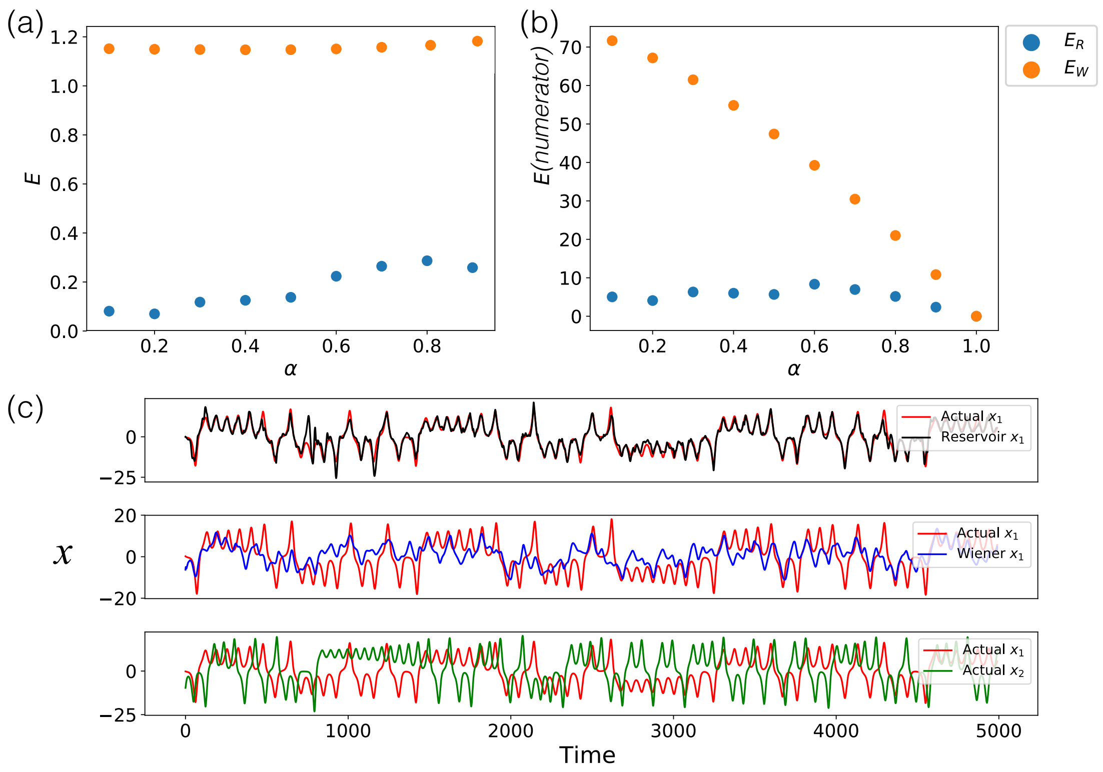


Figure 3.3: (a) Reservoir (blue) and Wiener (orange) error over a test length of 5000 timesteps across mixing fraction α for two Lorenz signals with the same speed but parameters $\boldsymbol{p}_1 = 1.2 \times \boldsymbol{p}_2$. (b) shows the numerator only of the error measures E_R, E_W. (c) For a mixing fraction $\alpha = 0.5$, (top) time-series plot of actual x_1 and reservoir predicted output $\hat{s}$, (middle) time-series plot of actual x_1 and Wiener predicted output $\hat{s}_w$, (bottom) actual x_1 and x_2. $E_R = 0.15, E_W = 1.14$. $\Delta t = 0.05$, sparsity $sp = 0.95$, $N = 2000, \lambda = 0.9, a = 0.3, k = 0.13$, training length $= 50000$, and for (c), $\alpha = 0.5$.

mixing fraction. We observe that the reservoir computer consistently outperforms the Wiener filter. However, as α increases, the error increases as well. This can be attributed to the denominator in the the error measure tending to zero as alpha tends to one. Fig. 3.3 (c) shows the estimated separated x_1 trajectory along with the actual x_1 in the testing phase, and the actual x_1, x_2 trajectories. The top panel demonstrates that the RC prediction does indeed match the actual x_1 accurately. The middle panel demonstrates that the best linear filter, the Wiener filter, is comparitively worse than the RC at estimating the chaotic signal x_1.

3.3.2 Separating Lorenz signals with different speed

In this section, the reservoir input consists of a combination of two signals from the x component of the Lorenz system with the same parameter values $\boldsymbol{p}_2 = \boldsymbol{p}_1$, but with different speeds, i.e., the right hand side of Eq. 3.8 for the Lorenz system corresponding to x_1 is multiplied by a speed fraction (ratio of speeds) η. The reservoir parameters remain the same as those found in sec. 3.3.1.

Fig. 3.4 (a) presents the performance of the reservoir computer as a function of the mixing fraction for $\eta_1 = 1.2\eta_2$ respectively, where η_i is the speed of the i^{th} signal. The error seems to have an increasing trend with α, as in the case with different parameters as in section 3.3.1. This can be attributed to the denominator in the error measure tending to zero as alpha tends to one. The non-normalized error is plotted in (b), and here we can see that as α tends to one, error tends to zero, as it should. The RC consistently outperforms the Wiener filter. As seen in Fig. 3.4 (c), the Wiener prediction is much poorer than the reservoir prediction of the chaotic signal x_1.

3.3.3 Separating Lorenz signals with matched power spectras

Linear signal denoising/separation methods are based on difference in spectra. Here we study the z component of two signals, z_1, z_2 with $\boldsymbol{p}_2 = 1.1 \times \boldsymbol{p}_1$ and $\eta_2 = 0.9 \times \eta_1$, where $\boldsymbol{p}_1 = \{\sigma = 10, \rho = 28, \beta = 8/3\}$, and $\eta_1 = 1$. Fig. 3.5 (a) shows a log plot of

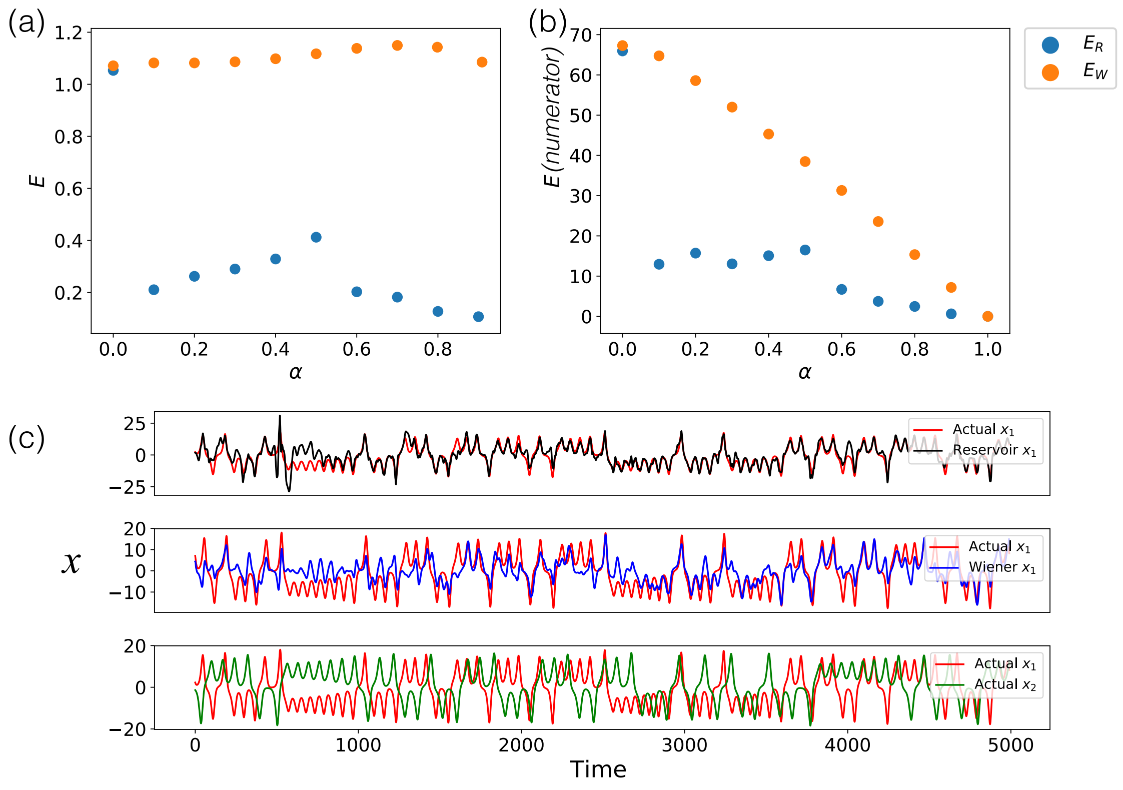


Figure 3.4: (a) Reservoir (blue) and Wiener (orange) error over a test length of 5000 timesteps across mixing fraction α for two Lorenz signals with the same parameters but speeds $\eta_1 = 1.2 \times \eta_2$. (b) shows the numerator only of the error measures E_R, E_W. (c) For a mixing fraction $\alpha = 0.5$, (top) time-series plot of actual x_1 and reservoir predicted output $\hat{s}$, (middle) time-series plot of actual x_1 and Wiener predicted output $\hat{s}_w$, (bottom) actual x_1 and x_2. $E_R = 0.51, E_W = 1.02$. $\Delta t = 0.05$, sparsity $sp = 0.95$, $N = 2000, \lambda = 0.9, a = 0.3, k = 0.13$, training length $= 50000$, and for (c) $\alpha = 0.5$.

the Power Spectral Density (PSD) ϕ_{z1z1}, ϕ_{z2z2} of the two signals z_1, z_2 respectively as a function of frequency Ω. The PSD of the z component of the Lorenz system has a distinct peak (unlike the x, y components), followed by a much smaller peak. For this reason we plot the PSD of the z signals to demonstrate spectra matching even though we separate the corresponding x signals (which may conceivably be a harder problem to solve since the x switches between positive and negative sides chaotically, and the RC has to learn when to switch correctly). We observe, in panel (a) that the peaks do indeed line up and the spectra match fairly well. Fig. 3.5(b) plots the reservoir and Wiener errors as a function of α for the spectra matched case. For a case where the spectra of the two individuals are indistinguishable, a spectra-based filter such as the Wiener filter will be unable to separate the signals. Fig. 3.5(c,d) show the reservoir predictions for the x and z component of the Lorenz system respectively for $\alpha = 0.5$. We observe, that the RC is indeed, able to separate the signals, even when their spectra match, unlike other state-of-the-art spectra-based signal separation methods like the Wiener filter (see Appendix .1).

3.4 Generalization Separation with Unknown Mixing Fraction

Often, as in the case of the cocktail party problem, the ratio of amplitudes in a mixed signal may not be known. In this section, we describe a methodology to separate signals without knowledge of the mixing fraction α. We found that training a single reservoir computer to separate signals for a wide range of α values was unsuccessful. So here we present a two step method: First, train a single RC to identify the mixing fraction (output α given a mixed signal). As before, we assume that we have access to training data for the individual signals. Using this kind of data along with known mixing fractions, we can first train a single RC to identify the mixing fraction of a mixed signal with an unknown mixing fraction. We then train a second RC to separate the signals using this estimated α as in section 3.3.

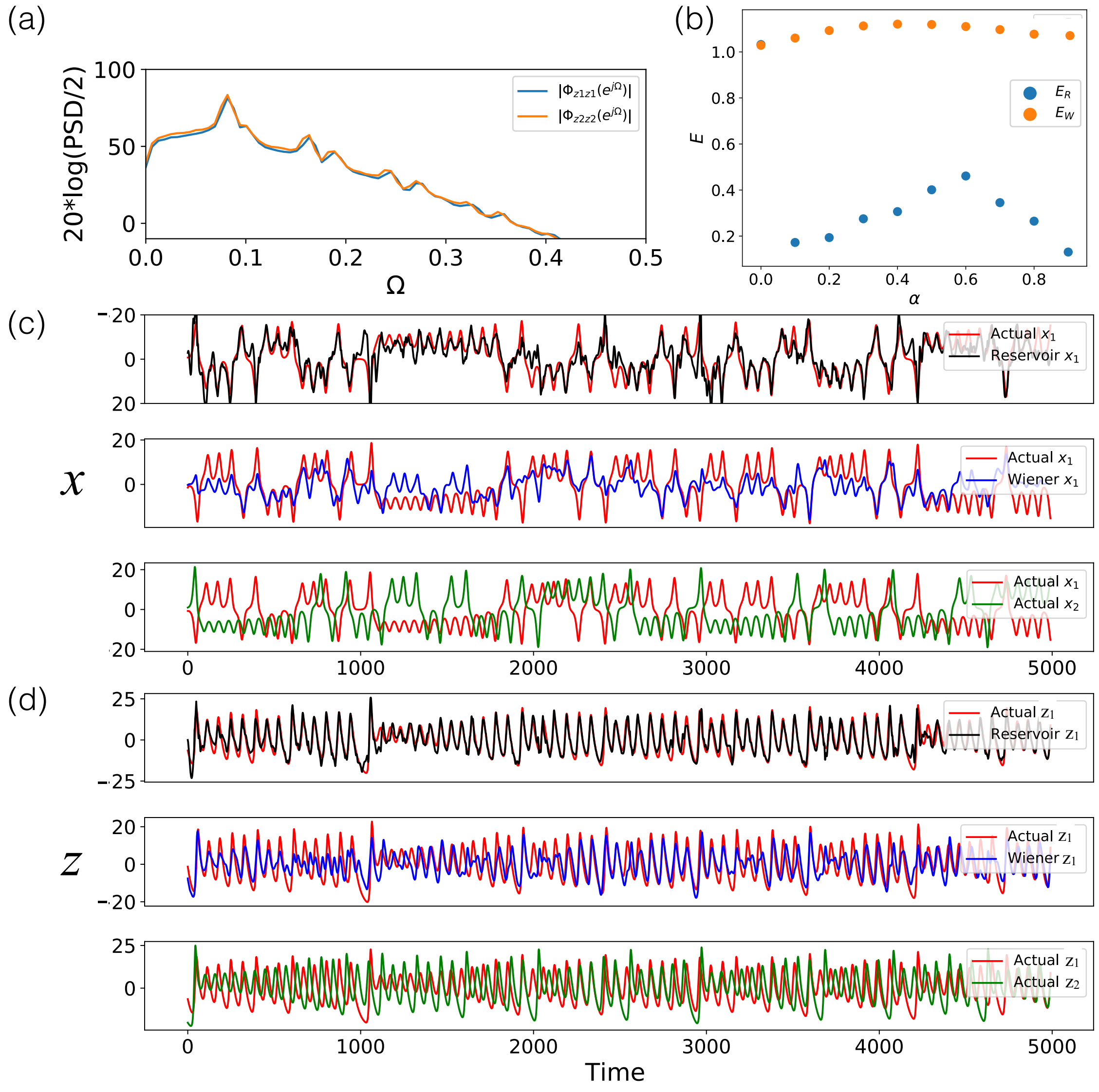


Figure 3.5: (a) Power spectral density of z_1, z_2 across frequency Ω, on a scaled log-plot to demonstrate a clear match in PSD respectively. (b) shows Reservoir and Wiener error (E_R, E_W) for separation of spectra matched Lorenz x signals as a function of α. (c,d) For a mixing fraction $\alpha = 0.5$, (top) time-series plot of actual x_1, z_1 respectively and reservoir predicted outputs, (bottom) actual inputs x_1, x_2 and z_1, z_2 respectively. for (c), $E_R = 0.48, E_W = 1.02$ and for (d), $E_R = 0.26, E_W = 1.14$. $t = 0.05$, sparsity $sp = 0.95$, $N = 2000, \lambda = 0.9, a = 0.3, k = 0.13$ training length $= 50000, \alpha = 0.5, \eta_1 = 1.1 \times \eta_2, \boldsymbol{p}_2 = 1.1 \times \boldsymbol{p}_1$.

3.4.1 Estimation of the Mixing Fraction Parameter

In training, the first RC is given mixed signals over a range of discrete values of α (0 to 1 in intervals of 0.05) and trained to output a constant α value. In testing, the constant predicted α are averaged over the testing time. However, learning a fixed value from a chaotic signal is non-trivial. We find that the trained RC always has a tendency to predict α closer to mean α (0.5) when fit on both the training and the test data, i.e., overestimate small actual α and underestimate large actual α. To correct for this tendency, we use the training dataset to construct a mapping from the predicted α to the desired α via a fit to a third order polynomial function. We then apply this same function to the reservoir-predicted α value(s) for the test data, in order to obtain corrected values.

Fig. 3.6 illustrates the performance of our approach for separating a mixed signal of x-components of two Lorenz systems with parameter values that differ by 20%. Fig. 3.6 (top) shows the reservoir-estimated α vs the actual α and Fig. 3.6 (bottom) shows the corrected α estimate (calculated using a third-order polynomial fit to the points in 3.6 (top)). After making the correction, our method accurately predicts α for both the training and test data.

Once the RC predicts an estimate of the mixing fraction, a second RC can be trained on that value of mixing fraction to separate chaotic Lorenz signals.

We note also that similar to this method for estimating the mixing fraction parameter, RC along with output correction can be used for parameter estimation from data in other systems as well.

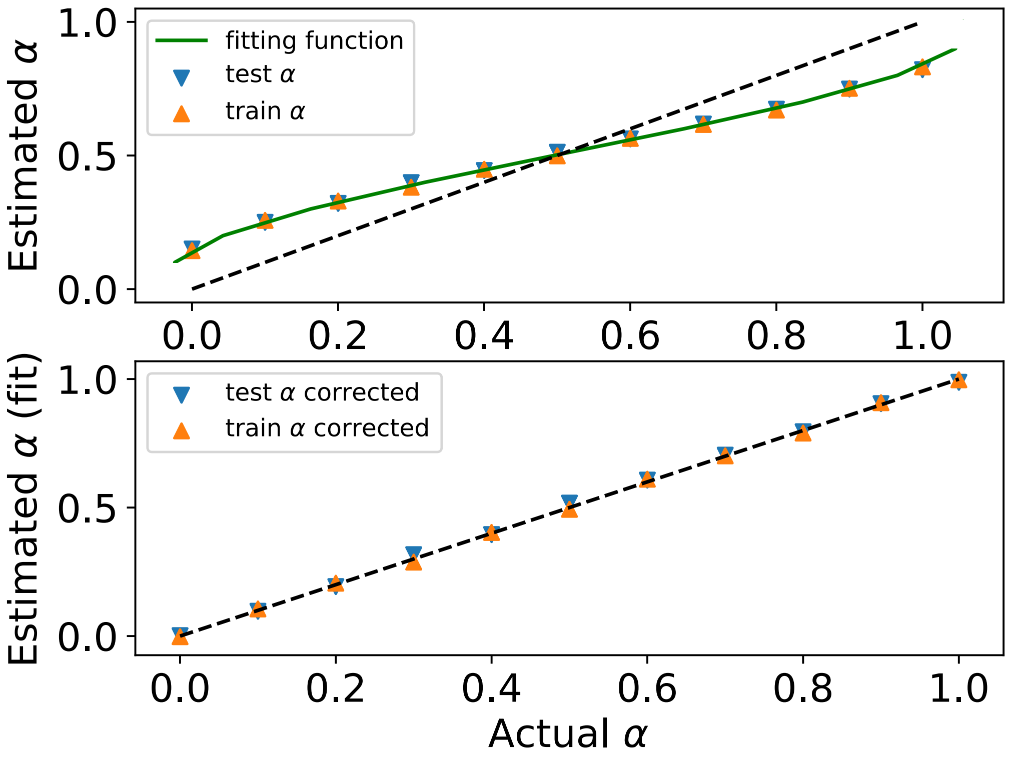


Figure 3.6: (top) Plot of the actual α used in the mixed signal vs the reservoir predicted α for both the training (orange) and test (blue) dataset. The green curve is the third degree polynomial fitting function between 0 and 1. The black dashed line represents the diagonal for reference. (bottom) The corrected test and train RC-estimated mixing fractions (after fitting to the third degree polynomial). The Lorenz signals being separated have the following characteristics $\boldsymbol{p}_2 = 1.2\boldsymbol{p}_1, \eta_1 = \eta_2$. Here $\Delta t = 0.05$, sparsity $sp = 0.99$, $N = 1000, \lambda = 0.9, a = 0.3, k = 0.13$ such that input signal has unit standard deviation, training and testing length $= 50000$ for each training $\alpha = [0, 0.1, 0.2, \ldots, 0.9, 1]$.

3.4.2 Interpolating between trained reservoir computers

Often, the RC may have computational and consequently training constraints, i.e., it may not be feasible to train a RC on each value of predicted α obtained. For instance, RCs in hardware have training constraints. In such cases, RCs can be pre-trained on discrete values of α. Any intermediate predicted α estimate obtained can then be used for separation by interpolating between the two nearest trained

α RCs. Here, interpolating between RCs means interpolating between their trained output weights. What spacing of α is appropriate for training? A mixed signal with predicted $\alpha = q$ can be separated by using the following output weight matrix $\boldsymbol{W_q^{out}}$ if q is in between two discrete trained values of α (q_- and q_+).

$$\boldsymbol{W_q^{out}} = \frac{q - q_-}{q_+ - q_-}\boldsymbol{W_{q+}^{out}} + \frac{q_+ - q}{q_+ - q_-}\boldsymbol{W_{q-}^{out}} \tag{3.11}$$

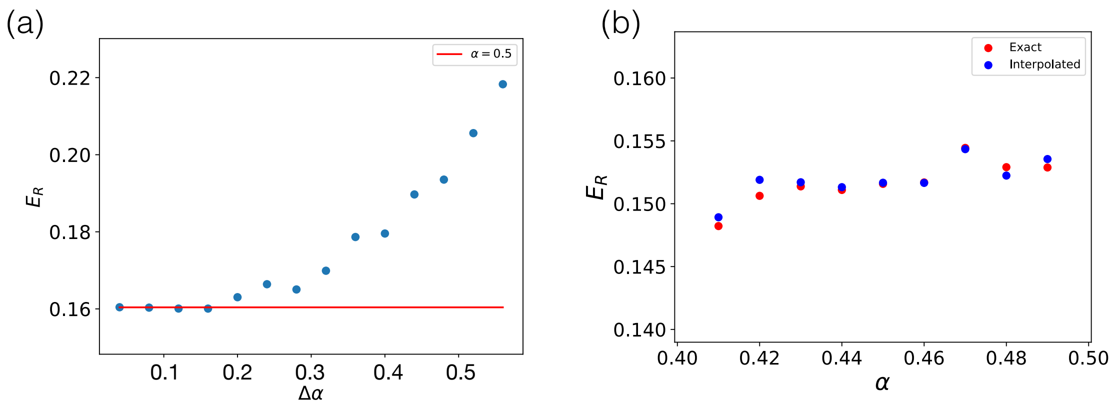


Figure 3.7: Reservoir errors on separating Lorenz signals with $\alpha = 0.5$. (red) RC trained on $\alpha = 0.5$, (blue) average of RCs trained on $\alpha = 0.5 \pm \Delta\alpha/2$ (b) Errors in interpolation between predictions by independent RCs trained on $\alpha = [0.4, 0.5]$ (blue) and RC trained on the exact $0.4 < \alpha < 0.5$ (orange); training length = 50000, training length =5000. $\boldsymbol{p}_2 = 1.2\boldsymbol{p}_1, \eta_2 = \eta_1$, $\Delta t = 0.05$, sparsity $sp = 0.95$, $N = 1000, \lambda = 0.9, a = 0.3, k = 0.13$

In Fig. 3.7, (a), we plot the reservoir error E_R for $\alpha = q = 0.5$ in orange, and for the average of the $\boldsymbol{W^{out}}$ matrices of reservoirs trained on $\alpha \pm \Delta\alpha/2$. We notice that interpolating between discrete RCs with a spacing of $\Delta\alpha = 0.1$ is well within the range of negligible error compared to training on individual predicted αs. Hence, one only needs to train on αs in intervals of 0.1 ($\alpha = [0, 0.1, 0.2, \ldots, 0.9, 1.0]$). Fig. 3.7, (a) shows the reservoir error for RCs trained individually on $\alpha = [0.40, 0.41, 0.42...0.49, 0.50]$ compared with the reservoir error obtained by averaging the $\boldsymbol{W^{out}}$ matrices of RCs trained on $\alpha = 0.4, 0.5$ appropriately as in Eq. 3.11. We observe that the results practically coincide, and that the method is fairly robust to errors in α prediction. Successful interpolation between RCs trained on discrete mixing fractions drastically reduces the training time and computation without compromising on quality of signal separation. In fact, we only need to train on 11 distinct mixing fractions to be able to separate Lorenz signals mixed in any proportion. Thus we are able to generalize our chaotic signal separation technique to cases where the mixing fraction is unknown.

3.5 Conclusion and Future Work

In this article, we used a type of machine learning called reservoir computing for separation of chaotic signals. We demonstrated the ability to separate signals for several cases where the two signals are obtained from Lorenz systems with different parameters. We compared out results with the Wiener filter, which is the optimal linear filter and whose coefficients can be computed from the power spectra of the signals (see Appendix .1). Spectra-based methods, naturally, perform poorly at signal separation if the spectra of the signals to be separated are indistinguishable. By contrast, the RC performs reasonably well even when the two signals that are mixed have very similar spectra. Our results were significantly better than the Wiener filter calibrated from the same training data for all the scenarios we considered.

Often, in signal separation applications such as the cocktail party problem, the mixing fraction (amplitude ratio) of the signals to be separated is unknown. We introduce a RC-based way to separate signals with an unknown parameter, in our case the mixing fraction. The first step of this generalized method is estimating a mixing fraction for a given signal. Estimation of a constant valued parameter from a temporal signal is a problem of broad interest, with applications such as weather prediction, predicting parameters of flow, equation modeling etc. We find that after time-averaging its trained output, the RC tends to skew the estimated parameter towards the mean of the parameter values used during training. By fitting a mapping function that corrects the averaged output for the training data, we are also able to approximately correct the test output. This method of introducing an additional non-linear 'correction' to the learned output weights may be useful for predicting constant outputs in other dynamic machine learning systems.

In some cases, training on a wide range of mixing fractions may not be possible, due to the need for quick separation of signals and limited training capacity (e.g. in hardware applications). Hence, we study the robustness of the RC, and the ability to use interpolated RCs pre-trained at discrete mixing fractions. We demonstrate that the RCs need only be trained at a coarse grid of mixing fractions in order to accurately separate Lorenz signals with arbitrary mixing fractions. Our results are robust to errors in the prediction of mixing fraction. Hence, in situations where computational resources are constrained, RCs can be trained on discrete mixing fractions, and interpolation between these RCs can be used to accurately separate chaotic signals for intermediate values of mixing fraction.

Here we have demonstrated the ability of reservoir computing to act as an efficient and robust method for separation of chaotic signals. The dynamical properties of the reservoir make it a prime candidate for further exploration of chaotic signal processing. An interesting future direction might be to study alternate RC architectures such as parallel RCs for more complex signal extraction problems.

Acknowledgment

This research was supported through a DoD contract under the Laboratory of Telecommunication Sciences Partnership with the University of Maryland, and, in part, by NSF award DGE-1632976.

.1 Wiener Filter

The Wiener filter [1] was designed to separate a signal from noise, but it can be used to (imperfectly) separate any two signals with different power spectra. For simplicity, we formulate the filter here in continuous time, as a linear noncausal filter with infinite impulse response (IIR) for a scalar signal. Later we describe how we compute the filter in practice, in discrete time with finite impulse response.

Let $u(t)$ be the combined signal – the input to the filter – and let $s(t)$ be the component signal that is the desired output of the filter.

A noncausal IIR filter can be written as a convolution

$$h * u(t) = \int_{-\infty}^{\infty} h(\tau)u(t-\tau)d\tau \tag{12}$$

where $h(\tau)$ is the impulse response function of the filter. (We assume that $|h(\tau)|$ has finite integral; then if $u(t)$ is bounded, so is $h * u(t)$.) Of all such filters, the Wiener filter is the one that minimizes the mean-square error $\langle (h * u(t) - s(t))^2 \rangle$ between the filter output and the desired output.

Similar to linear least-squares in finite dimensions, the minimizing function h_W can be related to the auto-covariance and cross-covariance functions

$$C_{uu}(\tau) = \langle u(t-\tau)u(t) \rangle, \qquad C_{us}(\tau) = \langle u(t-\tau)s(t) \rangle. \tag{13}$$

Specifically, setting the first variation (*i.e.*, the first derivative in the calculus of variations) of the mean-square error equal to zero yields

$$h_W * C_{uu}(\tau) = C_{us}(\tau). \tag{14}$$

Equation (14) can be solved in the frequency domain, where convolution becomes multiplication. Let H_W, P_{uu}, and P_{us} be the Fourier transforms of h, C_{uu}, and C_{us} respectively; then $H(\omega)P_{uu}(\omega) = P_{us}(\omega)$, and

$$H_W(\omega) = \frac{P_{us}(\omega)}{P_{uu}(\omega)}. \tag{15}$$

We interpret equation (15) based on the fact that $P_{uu}(\omega)$ and $P_{us}(\omega)$ are respectively the power spectral density of $u(t)$ and the cross spectral density of $s(t)$ and $u(t)$, by an appropriate version of the Wiener-Khinchin theorem. In both Wiener's application and our application, the component signals $s(t)$ and $s = u(t)-$

$s(t)$ are uncorrelated, in the sense that their cross-covariance is identically zero. In this case, $C_{us}(\tau) = C_{uu}(\tau)$ and $P_{us}(\omega) = P_{uu}(\omega)$. Thus, the transfer function $H_W(\omega)$ of the Wiener filter at frequency ω is the fraction of the power of $u(t)$ at that frequency attributable to the component signal $s(t)$.

In practice, we sample the signals $s(t)$ and $u(t)$ at discrete intervals of $\Delta t = 0.05$. We estimate their spectral densities using the Welch method, averaging the estimates on overlapping segments of length 500 samples, using the discrete Fourier transform (DFT) and the Hann window on each segment. We then apply the inverse DFT to the resulting discrete estimate of H_W, yielding an impulse-response vector h_w of length 500. We convolve h_w with the sampled signal to compute the Wiener filter.

Chapter 4: Encoding of a Chaotic Attractor in a Reservoir Computer: A Directional Fiber Investigation

This chapter is accepted for publication as part of the conference proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN): Sanjukta Krishnagopal, Garrett Katz Michelle Girvan, and James Reggia. "Encoding of a Chaotic Attractor in a Reservoir Computer: A Directional Fiber Investigation," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, 2019.

4.1 Introduction

Neural network architectures have been very successful in a variety of tasks such as image classification [4], video prediction, voice detection [59] etc. While feed-forward networks have had tremendous success in recent years, recurrent architectures are often found to be better suited to tasks that have a temporal aspect, for instance, speech recognition, video classification [60, 71] etc. There have simultaneously been a number of rapid advances in recurrent machine learning architectures such as Echo State Networks (ESNs) (originally proposed in the field of machine learning) [10] and Liquid State Machines (LSMs) (originally proposed in the field of computational neuroscience) [11], commonly falling under the term Reservoir Computing [12]. In this work, we use the term reservoir computing to refer to the ESN class.

A good example of a chaotic system with a temporal element for a hard machine learning problem is the Lorenz system [69]. In 1963, Edward Lorenz modeled a system of equations for his study on the predictability of certain atmospheric flows. The Lorenz system consists of three quadratically coupled ordinary differential equations. The Lorenz system was the first example of a physically interesting system of equations which was observed to have a strange attractor. It is a chaotic system, and is known to have exactly three fixed points in its chaotic regime.

Often, when studying the dynamics of a physical system, one only has access to a limited set of measurements of the state variables from which one must attempt to understand the dynamical properties encoded by the underlying equations. In this work, we use measurements of an inherently chaotic Lorenz system trajectory during an initial time to infer the underlying dynamics characterizing a dynamical system, and to predict future measurement in subsequent time. To this end, in our work, we use Reservoir Computing, a dynamical machine learning system that nonlinearly transforms input data in a reproducible way in order to serve as a resource for information processing.

A Reservoir Computer (RC), because of its dynamical properties and simple training needs, may inherently be better suited for learning temporal signals. RCs are particularly appealing for learning of a chaotic temporal signal (a hard machine learning problem since two signals that start near each other may evolve along entirely different trajectories) because of their dynamics that may allow for encoding of the latent representation of the measured variables. Several recent studies have been done using RCs for the learning of chaotic signals [3, 37]. While other recurrent architectures, like LSTMs, Gated Recurrent Units (GRUs) etc., may also offer dynamical properties enabling time-series prediction, due to their complex structure and training, they often require comparatively larger training data, are more computationally expensive, and may not have the dynamical properties that allow for learning chaotic signals that RCs offer. Study of the dynamics of the RC during learning allows us to explore and interpret how the Lorenz system is encoded in the network during limited exposure to measurements of the state variables.

Several recent works have explored encoding of information in the dynamics of a machine learning network. In the ground-breaking work of Hopfield in [72], dynamics in neural network phase space is dominated by attractors; an idea extended to several architectures since. The dynamical system of the Recurrent Neural Network (RNN) is left running until it ends up in one of its several attractors. In [30], RNNs that are actively generating, or passively being driven by different input patterns, end up in unique dynamical states called 'conceptors' that are characterized by the input pattern. However, training of RNNs is difficult due to problems like exploding or vanishing gradient. RCs overcome this problem by training only the output weights. RCs offer a convenient solution to some of the problems with RNNs, while offering many of the same advantages. RCs also offer easy scalability, since only the output weights are trained, while the recurrent connections within the reservoir are fixed randomly. Applications of RCs include many real world phenomena such as weather prediction [73], stock market forecasting [74], image similarity learning [65], etc. and have been implemented in hardware [75, 76, 77].

We are interested in understanding how RC dynamics encode for the chaotic Lorenz system from limited measurements. To this end, we study a basic dynamical property, i.e., location and stability of fixed points within the high dimensional reservoir, comparing them to the known attractors of the Lorenz system. To identify fixed points in reservoir space, we use 'directional fibers', a recently introduced mathematical technique for enumerating distinct solutions of smooth multivariate non-linear equations [2]. In the context of dynamical systems, a directional fiber is the set of all points that flow in the same fixed direction. Different fixed directions will determine different directional fibers. Not all directional fibers are path connected, but every fiber does contain all fixed points, and an individual fiber can be numerically traversed to enumerate many of those fixed points. [78] explored the fixed points of Hopfield nets using directional fibers. Directional fibers are explained in more detail in the following sections.

In particular, we are interested in exploring the way in which the Lorenz system (and its corresponding fixed points) are encoded in reservoir space, exploring the basin of attraction of the learned Lorenz fixed-point attractors through the position of any additional non-Lorenz fixed points, as well as exploring the effects that these fixed points have in the consequent prediction of state variables. In what follows, we outline the methods, including a description of the chaotic Lorenz system in Section 4.2.1, an illustration of the reservoir models and relevant reservoir computing concepts in Section 4.2.2, as well as an overview of the mathematical directional fibers concept in Section 4.2.3. We then present the results of our computational experiment of applying the directional fibers to the reservoir learned on the Lorenz system. In Section 4.3.1, we present the reservoir's ability to predict the chaotic Lorenz time series after learning. In Section 4.3.2, we present the extracted fixed points in the reservoir dynamics and their stability. We then present, in Section 4.3.3, the dependence of the results on several parameters of the reservoir as well the directional fiber. Additionally, we classify the fiber-identified fixed points into those learned from the Lorenz system and spurious fixed points. These spurious fixed points are not approximations to the Lorenz fixed points; they are an inherent property of the trained reservoir (reservoir with trained output weights). Lastly we investigate the effects of spurious fixed points on prediction error in 4.3.4.

4.2 Methods

4.2.1 Lorenz system

The Lorenz attractor is a strange attractor that arises in a system of equations characterizing a simple model for atmospheric convection. In the early 1960s, Lorenz discovered the chaotic behavior of a simplified 3-dimensional system of coupled differential equations [69], now known as the Lorenz 63 equations:

$$\begin{aligned}\frac{\mathrm{d}X}{\mathrm{d}t} &= \sigma(Y - X)\\ \frac{\mathrm{d}Y}{\mathrm{d}t} &= -XZ + \rho X - Y\\ \frac{\mathrm{d}Z}{\mathrm{d}t} &= XY - \beta Z\end{aligned} \tag{4.1}$$

The Lorenz system is in the chaotic regime for the following values: $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$.

The set of vector valued variables $\overrightarrow{u} = (X, Y, Z)$ evolves over time according to equations 4.1. The Lorenz system goes through several bifurcations. At the above parameter values, it inherently has three fixed points z_{fp} [79] given by: $[0,0,0], [br, br, \rho - 1], [-br, -br, \rho - 1]$ where $br = \sqrt{\beta \times (\rho - 1)}$. For the above mentioned parameter values, the origin is an unstable fixed point. Additionally, the other two (non-origin) critical points go through a Hopf bifurcation to become unstable as well.

We are interested in the case where $\overrightarrow{u}$ is available during a time period $[0, \ldots, T]$. We do not assume knowledge about the form of the function f that evolves the state variables in discrete time, $\overrightarrow{u(t+1)} = f(\overrightarrow{u(t)})$, but seek to use a reservoir computing approach to learn it from $\overrightarrow{u}[0, \ldots, T]$ in discrete time intervals δt.

4.2.2 Reservoir Computing

There are many variations in implementation; in this paper we adopt the Echo State Network class of the most basic reservoir computing technique proposed by Jaeger [10]. A significant advantage of using RCs over conventional machine learning architectures is that only the output weights are trained, making it relatively computationally inexpensive. In this work, when we refer to a 'training', we refer to the training of the output weights and not the internal recurrent connections. The reservoir computer has three components (Fig. 1), an input layer with N_u input nodes (one for each component of $\overrightarrow{u}$), a recurrent, nonlinear reservoir network with

N_R dynamical reservoir nodes driven both by inputs as well as by delayed feedback (at the previous time step) from the other reservoir nodes, and a linear output layer with N_s output nodes, as shown in Fig. 4.1.

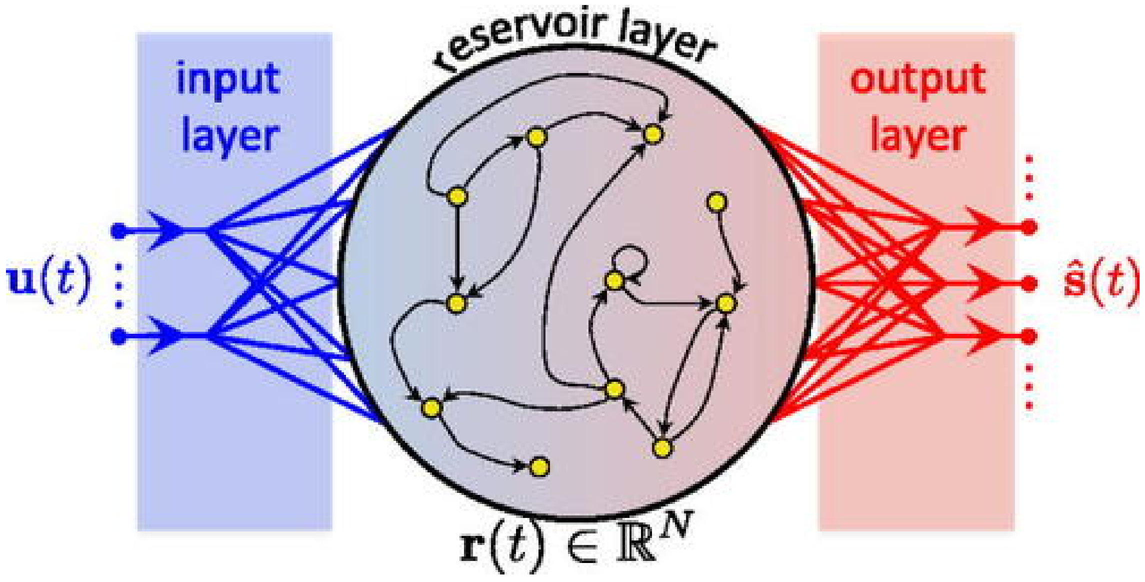


Figure 4.1: Reservoir architecture with input state at time t denoted by $\overrightarrow{u(t)}$, reservoir state by $\overrightarrow{r(t)}$ and output state by $\overrightarrow{s(t)}$.

4.2.2.1 Input Layer

The Lorenz system is a three dimensional system. At every time-point a three dimensional vector ($\overrightarrow{u(t)}$) is input to the reservoir through the input weights W^{in}. For the input layer, the i^{th} of the N_u input signals is connected to N_R/N_u reservoir nodes with connection weights in the i^{th} column of W^{in}. Each reservoir node receives input from exactly one input signal. The nonzero elements of W^{in} are randomly chosen from a uniform distribution between $[-\Sigma, \Sigma]$. Σ is chosen to keep the reservoir input between $[-1, 1]$.

4.2.2.2 Reservoir Layer

The reservoir can be thought of as a dynamical system where the reservoir is described by a reservoir state vector $\overrightarrow{r(t)}$ at time t given by :

$$\overrightarrow{r(t+1)} = \tanh\left(W^{\text{in}} \cdot \overrightarrow{u(t)} + W^{\text{res}} \cdot \overrightarrow{r(t)}\right) \tag{4.2}$$

The input weights matrix $W^{\text{in}} \in \mathbb{R}^{N_R \times N_u}$, where N_R is number of nodes in the reservoir and N_u is the dimension of the input vector $\overrightarrow{u(t)}$ (N_u is three for the three dimensional Lorenz system). The activity of the reservoir at time t is given by $\overrightarrow{r(t)}$, of size N_R. The recurrent connection weights $W^{\text{res}} \in \mathbb{R}^{N_R \times N_R}$ are initialized randomly between -1 and 1. Only the output weights are trained. The notation tanh() with a vector argument is defined as the vector whose components are the hyperbolic tangents of the corresponding components of the argument vector.

W^{res} is normalized so that the spectral radius (maximal absolute eigenvalue of W^{res}) is set to λ. A discussion of the effect of spectral radius on performance is presented in [42]. The reservoir is a dynamical system that transforms the input from the input space into a much higher order reservoir space. Hence, the reservoir reaches its optimal performance even when the W^{res} is sparse. A sparsity of p implies that $100 \times p\%$ of the entries are randomly chosen to be zero. A key point is that the reservoir layer serves as an active medium driven by inputs $\overrightarrow{u(t)}$ where each reservoir node has a different nonlinear response to its inputs, so that for $N_R >> 1$ we can hope that a wide variety of desired outputs can be approximated by a linear combination of the N_R dimensional reservoir nodal response states.

4.2.2.3 Output Layer

The reservoir state trajectory, R, is formed by concatenating the $N_R \times 1$ reservoir state vectors (the state of all reservoir nodes) at every time step $\overrightarrow{r(t)}$ as follows:

$$R = \overrightarrow{r(0)} \oplus \overrightarrow{r(t=1)} \oplus \ldots \oplus \overrightarrow{r(t=T)} \tag{4.3}$$

R is an augmented matrix of size $N_R \times T$ where T is the number of time steps for which state variable data is available. During training, the output weights are found by mapping the reservoir state trajectory to the expected output layer representation (S). For the task of prediction of the state variable evolution, $S = \overrightarrow{u}[1, \ldots, T+1]$. Only weight matrix W^{out} is optimized during training such that it minimizes the mean squared error $E(s, S)$ between the output of the reservoir s and the target signal S. The reservoir output is:

$$s = W^{\text{out}} R \tag{4.4}$$

$W^{\text{out}} \in \mathbb{R}^{N_s \times N_R}$ where N_s is the dimensionality of the readout layer.

Ridge Regression (or Thikonov regularization) [47], which adds an additional small cost to least square error, is used for fitting, thus making the system robust to overfitting and noise. Ridge regression minimizes squared error while regularizing the norm of the weights as follows:

$$J(W^{\text{out}}) = \zeta||W^{\text{out}}||^2 + \sum_i ((W^{\text{out}})^T R_i - S_i)^2. \tag{4.5}$$

where ζ is a small regularization constant.

Once learning is done, the reservoir predicts state variables for times $t = T+1$ onwards using the trained output weights W^{out}. The output weights convert the reservoir system state trajectories $\overrightarrow{r(t)}$ into the reservoir output $\overrightarrow{s(t)}$ (whose values are the prediction Lorenz state variables $\overrightarrow{u(t+1)}$).

4.2.3 Directional Fibers

Directional fibers are mathematical objects that can be used for systematic root-finding in smooth, non-linear, multivariate equations [2]. Let $f(r) = \mathbf{0}$ be any such equation, where f is a smooth function and $\mathbf{0}$ is an appropriately sized vector containing all zeros. Given a constant direction vector $c \in \mathbb{R}^N - \{\mathbf{0}\}$, the directional fiber of c under f is defined as:

$$\gamma^{(c)} = \{r \in \mathbb{R}^N : f(r) \text{ is a scalar multiple of } c\}. \tag{4.6}$$

In other words, directional fibers are like typical mathematical fibers, except they are the pre-image of a *direction* rather than a single point in the co-domain.

Under mild conditions on f given in [2], it can be shown that almost every choice of c (up to a measure zero set) yields a directional fiber that is a one-dimensional manifold. Hence, directional fibers can typically be numerically traversed. Moreover, by definition, any directional fiber contains every root: roots occur precisely where $f(r)$ is the trivial zero multiple of c. Consequently, numerical traversal of a directional fiber can be used to systematically enumerate distinct roots. The method is not strictly global, because some choices of c may result in fibers that are not path connected, and traversing one connected component will fail to identify roots on another connected component. Nevertheless, it has been found in practice that directional fiber traversal is often an effective means for locating many roots in certain systems. More mathematical details on directional fiber traversal are given in the appendix.

In this work, we apply directional fibers to the special case $f(r) = \tanh(Wr) - r$, where r is the reservoir state vector as described in the previous section, and $W = W^{\text{res}} + W^{\text{in}}W^{\text{out}}$ consolidates the autonomous reservoir dynamics in prediction mode when output is fed back into the input layer, i.e, $u = W^{\text{in}}W^{\text{out}}r$ in (4.2). In this case, f computes the *change* in reservoir state after one time-step. When this change is zero, r is a fixed point, so the roots of f are identically the fixed points of the reservoir system.

Note that in this application, f is an odd function: $f(-r) = -f(r)$. As a corollary, reservoir dynamics will necessarily be symmetric about the origin. In particular, the fixed points will come in $\pm$ pairs. The result is that training on the Lorenz attractor, which is restricted to a half-space where $z > 0$, will result in a reservoir that models the original attractor but also a symmetric 'phantom' attractor that is reflected through the origin. Initial conditions in the appropriate half-space will typically stay in that half-space, so the phantom attractors do not impact prediction and can be viewed as redundant. However, there may also be spurious fixed points *within* the appropriate half-space, which potentially *do* interact with prediction. These are explored further in our experiments reported below.

4.3 Results

We construct a reservoir computer following Section 4.2.2 in order to predict the Lorenz 63 system.

4.3.1 Prediction of Lorenz Time Series

Fig. 4.2 shows the performance of the reservoir in predicting $\overrightarrow{u} = [X, Y, Z]$ trajectories for varying spectral radii λ. We observe that the reservoir is remarkably successful in predicting a chaotic signal for several time steps after training. From Fig. 4.2, $\lambda = 0.9$ seems to result in the best prediction. This agrees with studies [80] that encourage setting the spectral radius close to but under 1 to stay close to the boundary of chaos. The training size was chosen to be above the saturation threshold following [36]. This is lower than the training set sizes typically used in machine learning. Fig. 4.2 (d) plots the breaking point (point at which prediction fails) as a function of reservoir size. We observe an upward trend in break point with reservoir size, however it seems to saturate at about 700 nodes. However, the directional fiber algorithm is a computationally intensive method. For the purpose of investigation of reservoir dynamics, a reasonably performing reservoir is sufficient. A reservoir size of

300 gives a reasonable trade-off between performance ($\sim$ 10 Lorenz cycles accurately predicted) and computational efficiency. The reservoir yields accurate short-term predictions and then deviates from the actual Lorenz trajectories, which is expected since any small error grows exponentially in a system with chaotic dynamics like the Lorenz system. Although long-term prediction deviates from the actual trajectory, the dynamics appears to resemble that of the original Lorenz system.

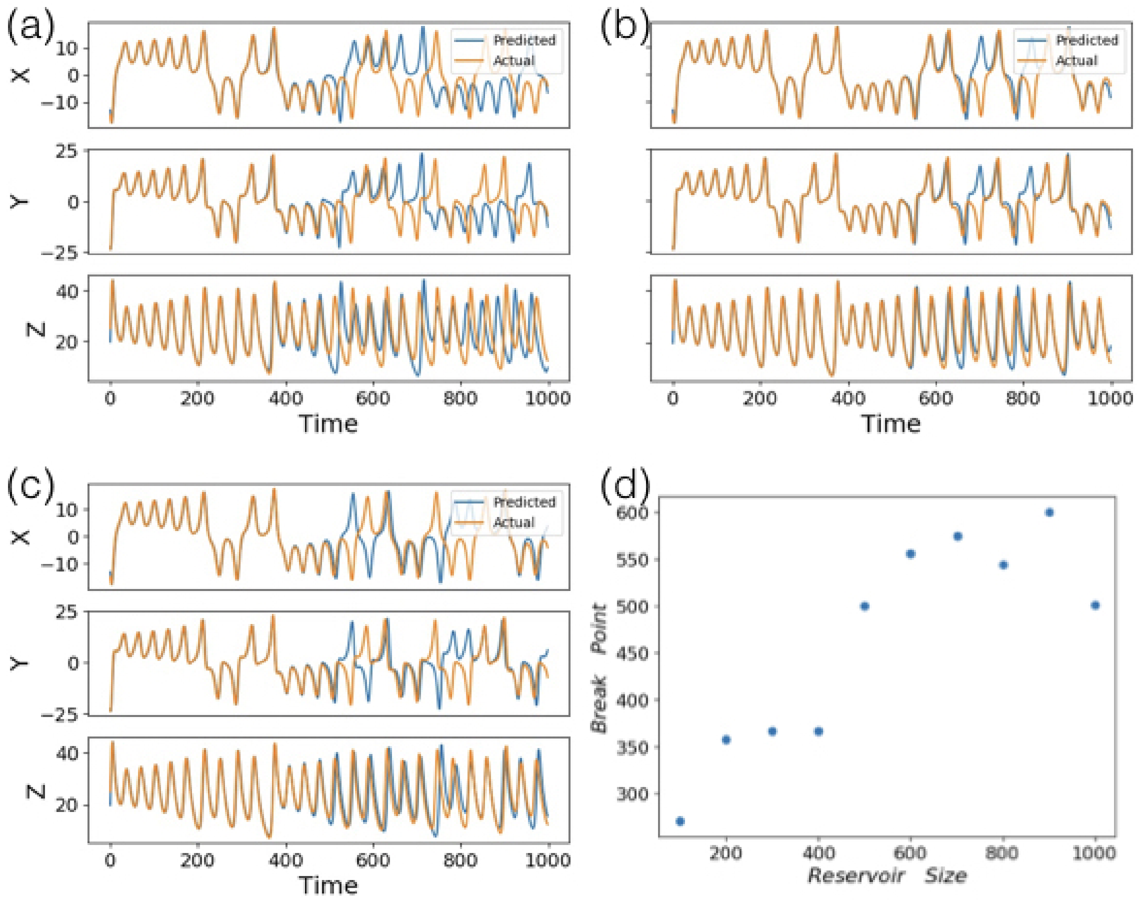


Figure 4.2: (a,b,c) Reservoir performance in predicting Lorenz time series for 1000 timesteps with spectral radius of $0.5, 0.9, 1.2$ respectively and break point (point at which prediction is above an error of 20% of the Lorenz signal amplitude for 7 consecutive timesteps) at 486, 663, 531 timesteps respectively. (d) is plot of the break point as a function of the reservoir size. $\Delta t = 0.01$, sparsity $p = 0.99$, $\eta = 10^{-6}$, training length $= 10000$, (a,b,c) $N_R = 500$, (d) $\lambda = 0.9$.

4.3.2 Extracting Fixed Points in The Reservoir

As seen in Fig. 4.2, though long-term predictions of the reservoir deviate from the trajectory, they have the same statistical pattern of a trajectory from a Lorenz system. This can be thought of as the prediction deviating from the actual trajectory and following a different trajectory from the same Lorenz system corresponding to a different initial condition. We are interested in the question of whether the reservoir learns some representation of the underlying equations of the Lorenz system. In order to explore this further, we study a fundamental dynamical property of dynamical systems, fixed points, using the directional fiber.

Fig. 4.3 shows a 2D view of the actual Lorenz and reservoir-predicted fixed points. The blue (solid) and orange (dashed) lines show the actual Lorenz trajectory and reservoir-predicted trajectory respectively. The mean square prediction error is defined as $E = \frac{||s-u||_2}{||u||_2}$, where $u = (X, Y, Z)$ is the actual trajectory (blue) and s is the predicted state evolution (orange); in this plot, $E = 0.52$ across a test time of 2000 steps. The directional fiber finds fixed points in the high dimensional reservoir space. These reservoir fixed points (z_R) are then projected into the Lorenz space and plotted here for easy viewing. The Lorenz projected fixed points (z_L) are obtained as follows $z_L = W^{\text{out}} z_R$. The z_L that overlap with the original fixed points of the Lorenz system, and their negatives (due to the symmetry considerations explained above) are plotted in blue and are unstable, in agreement with the stability of the fixed points of the Lorenz system in the chaotic regime. The z_L corresponding to spurious fixed points and their negatives are plotted in orange, and are all unstable.

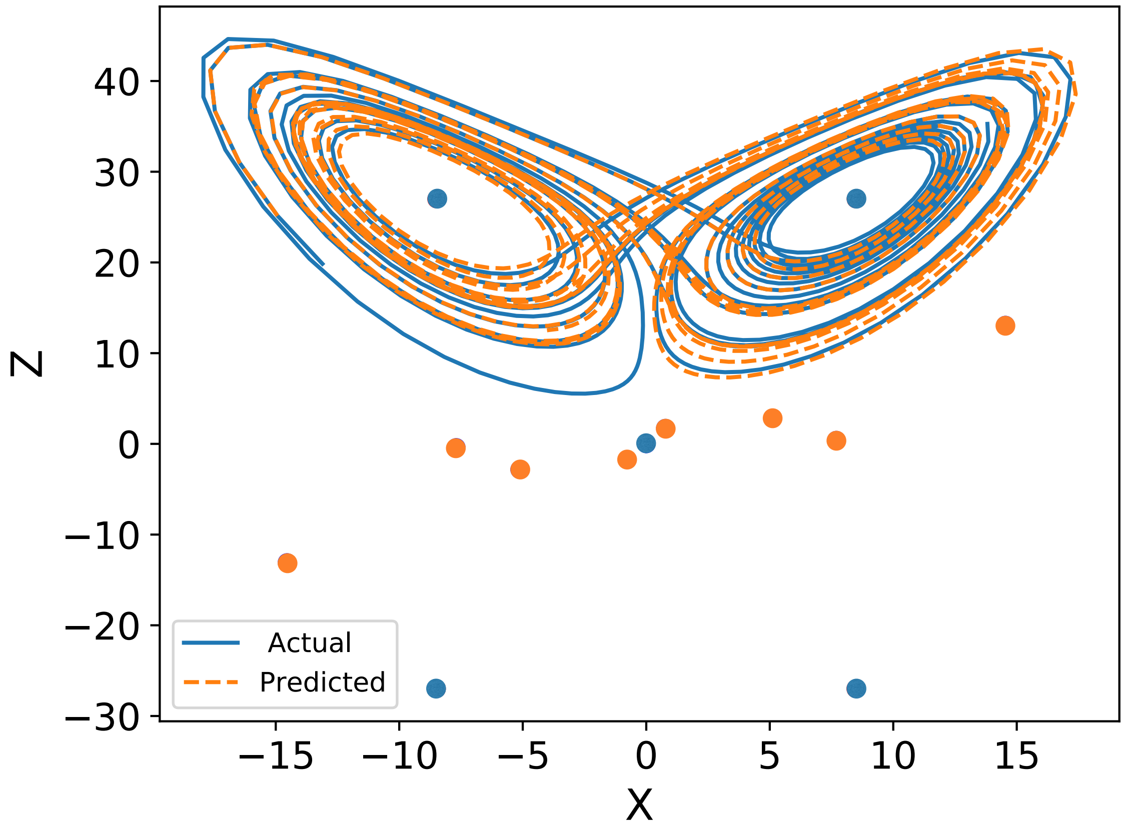


Figure 4.3: 2D view of the reservoir-predicted and actual Lorenz trajectory for 1000 timesteps in X-Z space. All fixed points identified by the directional fiber are projected onto Lorenz space. Of these, the ones that coincide with the Lorenz fixed points are marked in blue (all blue points are superposed over fiber-identified orange points). $\Delta t = 0.01$, sparsity $p = 0.99$, $\eta = 10^{-6}$, training length $= 10000$, $N_R = 300$, $\lambda = 0.9$, break point $=539$, E=0.52.

The stability of the z_R is calculated through identifying the magnitude of the largest eigenvalue of the Jacobian of the reservoir system. The Jacobian (J) of the reservoir system characterized by (4.2) at the i^{th} point z_{Ri} is given by:

$$J = DW \tag{4.7}$$

where $D_{ii} = 1 - \tanh^2(Wz_{Ri})$ is a diagonal matrix and $W = W^{\text{res}} + W^{\text{in}}W^{\text{out}}$. If all eigenvalues of J have an absolute value less than one, then z_{Ri} is asymptotically stable, if any eigenvalue (v) of J has $|v| > 1$, then z_{Ri} is asymptotically unstable. The directional fiber finds fixed points in pairs, i.e., $\pm z_R$. Since the hyperbolic tangent is an odd function, for each point, the directional fiber (according to the analysis in Section 4.2.3) finds both the fixed points and the negative of the fixed points, i.e., it is symmetric.

In Fig. 4.3, we find that the inherent Lorenz fixed points z_{fp} are embedded in the high dimensional reservoir space. These are the three veridical fixed points mentioned in Section 4.2.1, i.e., trivial fixed point and the two fixed points inside the two Lorenzian loops. A subset of the reservoir fixed points z_R when projected into Lorenz space as z_L coincide with z_{fp}. Additionally, the stability of these fixed points in reservoir space matches the stability of the original z_{fp}. This shows that the reservoir, in learning the Lorenz trajectory, also learns dynamical properties of the underlying system, as evidenced by the fixed points. The directional fiber also finds the negative of these fixed points $-z_{fp}$ for the reasons stated above, however all negative fixed points can be considered redundant for this analysis. While Fig.

4.3 shows one case, we find that the reservoir continues to embed the Lorenz fixed points for different runs and with varying parameters. However, the position and number of the additional spurious fixed points that do not correspond to the Lorenz system vary across runs. It is also worth noticing that all spurious fixed points (along with their negatives) are consistently found to be unstable across experiments. The distribution of the spurious fixed points gives us insight into the basin of attraction of the reservoir learned Lorenz system.

4.3.3 Distribution of Spurious Fixed Points

In an attempt to understand the role that the spurious fixed points play in learning, we study the effect of parameters that affect the location and number of fixed points, and the error of the reservoir prediction upon perturbation as it approaches a spurious fixed point. Table 4.1 tabulates across several runs the break point (bp), spectral radius γ_W of matrix W, total number of independent (excluding negatives) fixed points identified by the directional fiber (N_{fp}), average distance in reservoir space of the independent fixed points from the origin $|d|$ and distance in reservoir space of the closest fixed point to the predicted trajectory at the break point $d_{\min}(s(bp))$. Values are tabulated over independent runs with $\Delta t = 0.01$, sparsity $p = 0.99$, training length $= 10000$, $N_R = 300$, $\lambda = 0.9$.

From Table 4.1, we notice no obvious correlation between any of the quantities in the table. We gain a sense of the approximate basin of attraction of the chaotic attractor from the average distance in reservoir space of the independent fixed points

bp	γ_W	N_{fp}	$\lvert d \rvert$	$d_{\min}(s(bp))$
379.	2.02	6.	13.16	1.82
502.	1.77	8.	17.22	1.99
500.	3.15	6.	21.03	4.85
362.	1.9	8.	27.26	10.35
538.	3.72	11.	13.5	3.63
312.	3.76	6.	19.9	11.24
362.	3.16	5.	26.46	7.76
498.	1.6	6.	25.07	1.37
315.	6.21	3.	12.37	1.34

Table 4.1: Prediction break point for different properties of the reservoir and identified fixed points

from the origin $|d|$. However, in particular, we notice no obvious correlation between break point and the distance of the trajectory from the nearest spurious point at break point ($d_{\min}(s(bp))$) or break point and the overall number of independent fixed points (these include the Lorenz projected fixed points, but not the negatives on any of the fixed points in the Lorenz space). Thus, the results that we have presented here so far do not support the conclusion that the fixed points have an obvious role to play, although we revisit this question in Section 4.3.4.

In order to further investigate the distribution of the fixed points in Lorenz space, we plot histograms of the x, y, z locations of the independent fixed points in Fig. 4.4. These histograms plot the distribution of the spurious fixed points (excluding negatives and the fixed points corresponding to z_{fp}) projected in Lorenz space z_L. We ran several tests with different random initializations of the Lorenz time-series as well as the reservoir and input weights, however we show only 3 representative plots here.

We find, in Fig. 4.4, that there is a distinct pattern in the location of the spurious fixed points around the Lorenz attractor. The spurious points have a visi-

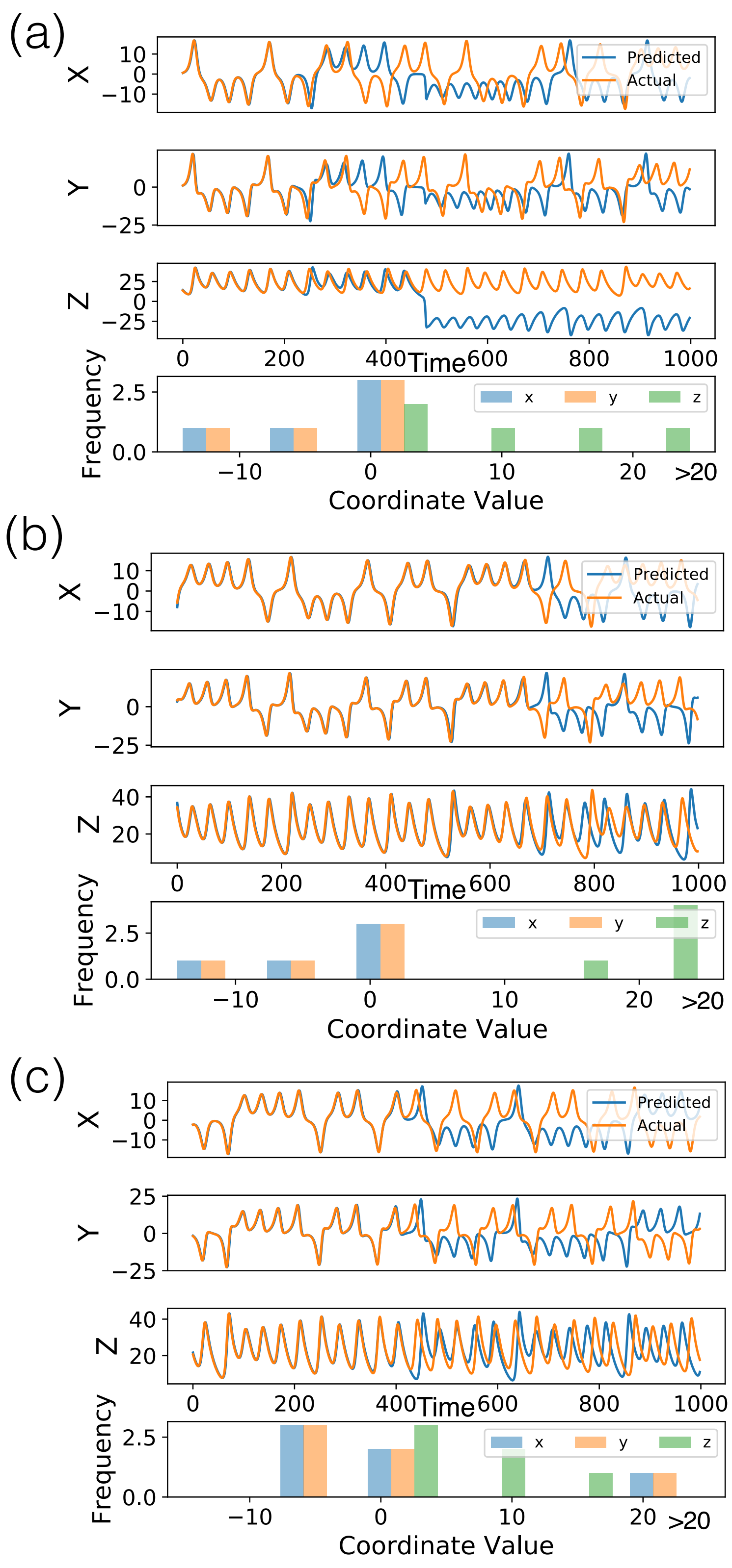


Figure 4.4: Predicted and actual X,Y,Z trajectories along with corresponding distribution of spurious fixed points for three random cases in (a,b,c). $\Delta t = 0.01$, sparsity $p = 0.99$, $\eta = 10^{-6}$, training length = 10000, $N_R = 300$, $\lambda = 0.9$.

ble symmetry in the X, Y coordinates. Additionally, the independent spurious fixed points display a tendency towards mostly negative X, Y coordinates corresponding to positive Z coordinates. We would like to mention that for all positive-negative pairs found by the directional fiber, we discard the point with the negative Z coordinate, since both the training, as well as the Lorenz attractor lies in the positive Z plane. These trends (symmetry in X, Y) display a resemblance to the characteristics of the Lorenz trajectories and fixed points. Thus, the spurious fixed points must be a result of training the output weights explicitly on the Lorenz system. Lastly, we want to draw attention to clustering of spurious fixed points with X, Y coordinates close to 0, in most cases. In Fig. 4.4, and in other experiments not shown here, we observe that the break point always occurs at $(X, Y, Z) \sim (0, 0, Z)$. This correlates directly with the density of spurious points. Hence, while there seems to be no global influence of the spurious fixed point on prediction (as evidenced by Table 4.1), there seems to be a qualitative relationship between location of break point and distribution of spurious points.

4.3.4 Effect of Spurious Fixed Points in Prediction

To investigate this further, Fig. 4.5 plots the distance to the nearest spurious point at each time point. For simplicity and ease of understanding, we demonstrate cases with at most one independent spurious point nearest to each 'wing' of the Lorenz attractor, and at least one independent spurious point in the entire Lorenz space. This is done to ensure the effects of an arbitrary spurious point can be studied independent of secondary effects due to other spurious points in the vicinity.

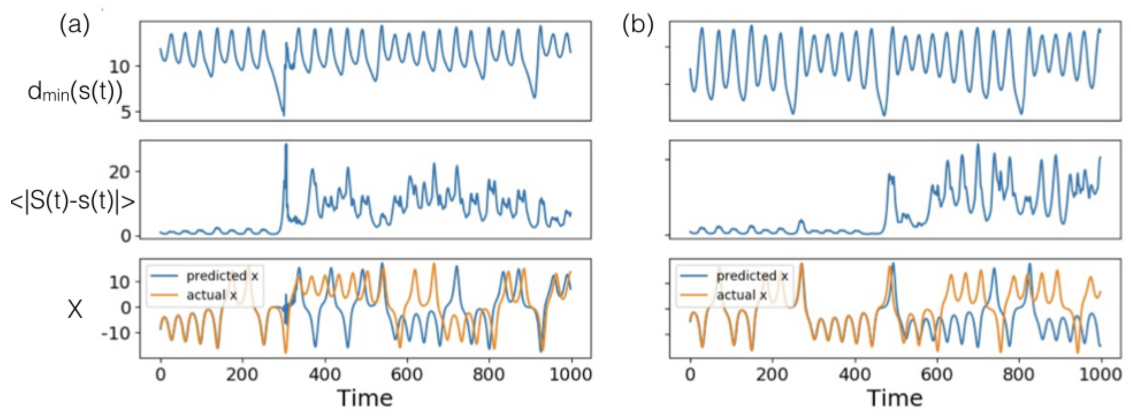


Figure 4.5: Two cases with different randomizations are shown in (a,b). The first row shows the distance at each time point from the reservoir state corresponding to the predicted trajectory to the nearest spurious point in reservoir space $d_{\min}(s(t))$. The second row shows the prediction error between the actual trajectory (S) and the predicted trajectory (s) at each time point averaged across the X, Y, Z directions. The third row shows the corresponding predicted and actual X trajectory. $\Delta t = 0.01$, sparsity $p = 0.99$, $\eta = 10^{-6}$, training length $= 10000$, $N_R = 300$, $\lambda = 0.9$.

We observe that in regions where the predicted orbit does multiple loops in the same wing, consecutive loops move closer to the spurious fixed point. This is evidenced by correlating regions of activity within the same loop in Fig. 4.5 row 3, with the corresponding error in reservoir space in row 1. We would like to explicitly note here that the error in row 1 is not the euclidian distance between predicted and actual vectors in Lorenz space, but in the high dimensional reservoir space. This phenomenon of cycling closer to the spurious points holds true both in the phase when prediction is accurate, as well as in the region where prediction fails. This indicates that the spurious fixed points (which, as established in the section above, are a function of the training data) are learned during the training phase to be located such that the actual Lorenz orbit trajectory is matched through successive minimization of the distance to the spurious point. Being chaotic, however, eventually prediction diverges rapidly from the actual trajectory as small errors blow up over time. However, the learned property of approaching spurious points in consecutive loops in the same wing holds true. Thus, the spurious points do indeed play a role in allowing the very high dimensional reservoir to mimic the dynamics of the relatively much lower dimensional Lorenz system through guiding the dynamics onto the attractor. The spurious points, their location and distribution, depend on the region of the Lorenz time-series chosen as input. They can be thought of as restricting the basin of attraction of the reservoir to model the Lorenz system.

4.4 Conclusion and Future Work

In this paper we have used reservoir computing to investigate prediction of chaotic time series. We found that the reservoir encodes dynamical properties of the system when trained on only a limited amount of time series data from the system. We demonstrated that the reservoir does indeed encode the dynamics of the system by probing its fixed points through a mathematical tool called directional fibers. We also showed that the high dimensional reservoir space, upon training of output weights, develops an encoding of the Lorenz fixed points, along with their corresponding stabilities. Additionally, the directional fibers also find non-Lorenz spurious fixed points in reservoir space on the outside of the projected attractor region. We investigated the statistics of these points that indicate that they are characteristic of the Lorenz system. Lastly, we explored the effect the spurious points have on training, and identify that the spurious points are distributed so as to minimize distance in reservoir space of consecutive actual cycles to the spurious points. Consequently, reservoir predictions also follow the same pattern of minimizing distance to spurious points in consecutive loops. Hence, the spurious points indicate the basin of attraction of the Lorenz system representation learned by the driven reservoir. This allowed us to gain insight into the ways by which a three dimensional system is constrained and encoded in a several hundred dimensional system.

From a dynamical system perspective, our findings indicate that the reservoir dynamics, upon being driven by the Lorenz system and after training of the reservoir outputs, literally mimic the dynamics of the driving system. Additionally, the reservoir represents other dynamical properties that allow it to project the driving system into a higher dimensional space in a constrained manner. From a computational perspective, the reservoir is fast since only the output weights are being trained and the reservoir is sparsely connected. However other reservoir-like architectures with training of recurrent connections in addition to output weights [81] may be of interest to some readers. We see the strength of our work as lying in not only its ability to learn a chaotic system, but also our ability to explain this in terms of the fixed points distribution through the novel directional fibers tool. Contrary to the black box approach, this relates to new ideas in explainable Artificial Intelligence, a topic that is attracting a lot of attention. Interesting future directions would be a thorough investigation of other dynamical properties such as limit cycles that the reservoir may use for encoding dynamics. Other interesting future work could use the directional fibers in conjunction with machine learning approaches to improve them through an understanding of their dynamic encoding.

4.5 Appendix: Directional Fiber Details

The directional fiber is the set of all points r where $f(r)$ is a scalar multiple of a constant direction c. Numerical traversal of a directional fiber can be made precise by making that scalar multiple explicit at every point. To that end, we define the function $F^{(c)}(r, \alpha) : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ as follows:

$$F^{(c)}(r, \alpha) = f(r) - \alpha c \tag{4.8}$$

where α is the scalar multiple. Then we can define the 'lifted directional fiber' $\Gamma^{(c)}$ as the set of points (r, α) satisfying an implicit equation:

$$\Gamma^{(c)} = \{(r, \alpha) \in \mathbb{R}^N \times \mathbb{R} : F^{(c)}(r, \alpha) = \mathbf{0}\}. \tag{4.9}$$

The subset of points where $\alpha = 0$ are precisely the roots of f. Where convenient we can recast (r, α) as a vector $\hat{r} \in \mathbb{R}^{N+1}$ whose first N coordinates are r and whose last coordinate is α. The i^{th} row of the Jacobian matrix $DF^{(c)}(\hat{r})$ can be viewed as the gradient of one coordinate F_i with respect to $\hat{r}$. Moving in the direction of this gradient is the fastest way to make F_i non-zero. Conversely, moving *orthogonally* to all of these gradients is the best way to keep F equal to zero (i.e., to maintain the implicit equation defining the lifted directional fiber). Hence the tangent vector to the fiber at a point $\hat{r}$, which we denote z, satisfies $DF^{(c)}(\hat{r})z = \mathbf{0}$. As in [2], this can be made fully rigorous via the Inverse Function Theorem. Furthermore,

[2] also shows that by Sard's Lemma, if $Df(r)$ is full rank at every fixed point, then $DF^{(c)}(\hat{r})$ is full rank everywhere along the fiber, for almost every choice of c (up to a measure zero subset). Hence a unique tangent direction will typically be well-defined.

Let θ denote the magnitude of a single numerical step along the fiber, starting from a point $\hat{r}^{(0)} \in \Gamma^{(c)}$. In order to step a distance θ in the direction of the tangent vector z, while simultaneously remaining on the directional fiber, one must solve the equation

$$\begin{bmatrix} F^{(c)}(\hat{r}^{(\theta)}) \\ z^\top(\hat{r}^{(\theta)} - \hat{r}^{(0)}) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \theta \end{bmatrix} \tag{4.10}$$

for $\hat{r}^{(\theta)}$, where $\hat{r}^{(\theta)}$ will be the new point along the fiber after the step. As long as θ is not too large, then $\hat{r}^{(\theta)}$ can reliably be found by a local solver such as Newton's method, using $\hat{r}^{(0)}$ as an initial seed. In practice, θ can be kept fixed during traversal, or it can be computed adaptively by the automated method in [2], which rigorously guarantees that the numerical traversal stays within machine precision of the true mathematical fiber.

Chapter 5: Identifying and predicting Parkinsons disease subtypes through trajectory clustering via bipartite networks

This chapter is under review at the journal Plos One and is on arxiv: Sanjukta Krishnagopal, Rainer Von Coelln, Lisa M. Shulman, and Michelle Girvan. "Identifying and Predicting Parkinson's Disease Subtypes through Trajectory Clustering via Bipartite Networks." arXiv preprint arXiv:1906.05338 (2019).

5.1 Introduction

Parkinsons disease (PD) is the most common neurodegenerative movement disorder, affecting an estimated 7-10 million people worldwide [82]. The cause of PD is unknown, and the disease course is variable with age of onset and rate of progression differing across the population [83]. Furthermore, the clinical presentation is variable, with a broad range of possible motor and non-motor symptoms [84]. Based on these differences, multiple PD subtypes have been proposed, based on clinical intuition or unbiased data-driven approaches like cluster analysis [85]. Disease subtypes are likely to be related to the underlying etiology, treatment responsiveness and prognosis, and will therefore be useful tools to improve PD research, direct existing therapies, and counseling of patients regarding prognosis [86, 87].

There is currently no consensus on Parkinson's subtypes that are biologically valid and clinically relevant, and the best approach for identifying such subtypes remains elusive [88]. Lack of integration of the longitudinal data for a large number of variables, and lack of data-based prognosis are limitations of existing approaches.

Network medicine [89, 90, 91, 92, 93] offers a promising approach for untangling the complexities due to multiple influences on disease via analysis of interconnections within data. For example, studies of the human disease network (i.e. the diseaseome) [93], in which diseases are linked if they share one or more associated genes, are useful for identifying disease pathways and predicting other disease-related genetic variants [91]. With a few exceptions, most network medicine studies have focused on biomolecular data [93, 94, 95, 96] rather than the complexities of clinical phenotypic assessments, and disease subtyping based on disease progression patterns is relatively unexplored [97, 98]. Additionally, a large number of genetic variants have been identified as risk factors to develop PD. Recently, evidence has emerged that the same genetic risk variants also determine certain clinical features of the disease, including disease progression [99, 100].

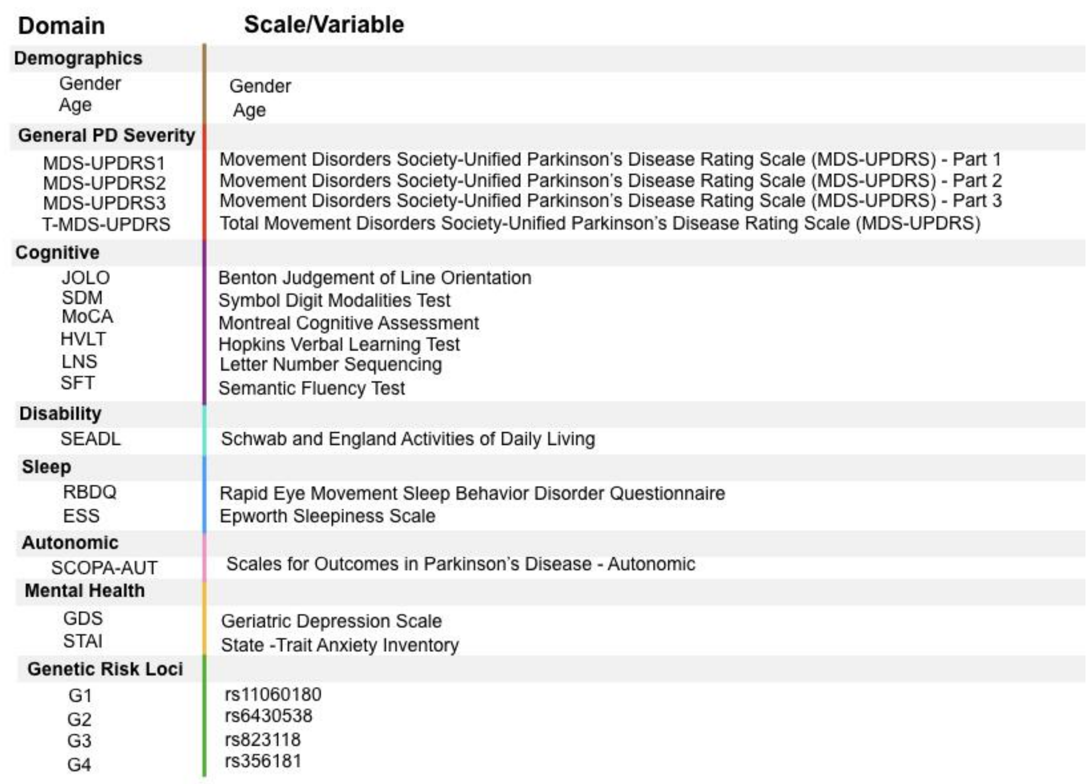
Technological advances in data processing and storage capacity have enabled development of large clinical datasets, containing longitudinal clinical and biological data. In this work we use data from the Michael J. Fox Foundations Parkinsons Progression Markers Initiative (PPMI), a worldwide study to establish a comprehensive set of clinical, imaging and genetic data (http://www.ppmi-info.org). Such datasets require sophisticated data-driven approaches for effective extraction and analysis of clinically relevant information. Data-driven methods are typically applied to diseases in two ways: disease-specific, i.e., identifying disease subtypes and variable progression patterns from large scale patient data, and patient-specific, i.e., predicting disease subtype and trajectory in the individual patient based on their data. Our work incorporates both these perspectives and presents a network science method that not only identifies disease subtypes using diverse types of patient data (e.g., genetic and clinical variables), but also is predictive. We present our results based on a PD dataset, however this method is easily applied to other chronic medical conditions.

To provide an intuitive data-driven solution that is both disease- and patient-centric, we develop a novel Trajectory Profile Clustering (TPC) algorithm to identify PD subtypes through similarities in patterns of progression. One of the advantages of our algorithm is that it identifies communities in heterogenous multi-variate datasets by accounting for similarities in 'trajectories' or change in variable profiles in addition to similarities in the variable values themselves. Additionally, we demonstrate the predictive ability of our algorithm on a test/validation cohort of new patients. We also explore inclusion of four PD genetic variants in our approach, to demonstrate its capacity to simultaneously incorporate clinical, demographic, and genetic information. Thus, TPC is a data-driven algorithm that can incorporate different types of data such as genetic, clinical etc., different weighting schemes for different variables, and also possess predictive power, making it a unique and useful tool for clinicians in the study of multivariate, progressive disease datasets. While network medicine has been implemented with some success in the study of diseases, our method, to the best of our knowledge, presents a new and easily generalizable way that acts as a robust subtype identifier by accounting for disease ‘progression’ patterns in addition to variable severity. This work is aimed at the gap between the computational methodologies developed by network and data scientists and the clinical experience of health professionals.

5.2 Materials and methods

5.2.1 Description of data

Data used in the preparation of this article were obtained from the Parkinsons Progression Markers Initiative (PPMI) database (www.ppmi-info.org/data). The data consists of patient variable values across 5 years (baseline year data extracted from the year 1 dataset and years 1,2,3,4). Of the 430 patients at baseline in this dataset, 314 patients remained in year 4. Once patients with incomplete data were excluded, 194 patients remained in our analysis. Twenty percent of this population (number of individuals $n = 39$) was kept as a test/validation dataset. The remainder of the patients ($n = 155$) formed the training dataset that was used in the algorithm to identify PD subtypes. The data included demographics (age in year 4, gender), clinical variables from six clinical domains (General PD Severity, Disability, Cognition, Autonomic Function, Sleep, and Mental Health) and 4 PD genetic variants (Fig 5.1).



Domain	Scale/Variable
Demographics	
Gender	Gender
Age	Age
General PD Severity	
MDS-UPDRS1	Movement Disorders Society-Unified Parkinson's Disease Rating Scale (MDS-UPDRS) - Part 1
MDS-UPDRS2	Movement Disorders Society-Unified Parkinson's Disease Rating Scale (MDS-UPDRS) - Part 2
MDS-UPDRS3	Movement Disorders Society-Unified Parkinson's Disease Rating Scale (MDS-UPDRS) - Part 3
T-MDS-UPDRS	Total Movement Disorders Society-Unified Parkinson's Disease Rating Scale (MDS-UPDRS)
Cognitive	
JOLO	Benton Judgement of Line Orientation
SDM	Symbol Digit Modalities Test
MoCA	Montreal Cognitive Assessment
HVLT	Hopkins Verbal Learning Test
LNS	Letter Number Sequencing
SFT	Semantic Fluency Test
Disability	
SEADL	Schwab and England Activities of Daily Living
Sleep	
RBDQ	Rapid Eye Movement Sleep Behavior Disorder Questionnaire
ESS	Epworth Sleepiness Scale
Autonomic	
SCOPA-AUT	Scales for Outcomes in Parkinson's Disease - Autonomic
Mental Health	
GDS	Geriatric Depression Scale
STAI	State -Trait Anxiety Inventory
Genetic Risk Loci	
G1	rs11060180
G2	rs6430538
G3	rs823118
G4	rs356181

Figure 5.1: Data includes two demographic variables, outcome variables from six clinical domains, and four genetic single nucleotide polymorphisms.

5.2.2 Trajectory Profile Clustering algorithm

Our Trajectory Profile Clustering (TPC) algorithm is designed to group together patients based on the similarities of their disease trajectories. The algorithm proceeds as follows:

1. *Create bipartite networks connecting individuals to variables*: At a single point in time t (e.g., baseline, year 1, year 2, etc.) we construct an $N \times V$ bipartite graph indicating the connections between individuals and disease variables, where N is the number of individuals in the population and V is the total

number of variables measured, as illustrated in Fig 5.2. For M time points, we can represent the set of these bipartite graphs as an $N \times V \times M$ multidimensional array, where X_{ivt} gives the value of individual is disease variable v at time t.

2. *Transform data so that higher values are associated with disease progression*: For each variable, we determine its direction. If higher values of the variable are associated with greater disease severity then direction for the v^{th} variable is given by $d_v = +1$, otherwise $d_v = 1$. For our data, clinical variables JOLO, SDM, MOCA and SEADL are the only ones with $d_v = 1$, We then define a new $N \times V \times M$ multi-dimensional array Y such that $Y_{ivt} = d_v X_{ivt}$.

3. *Construct patient trajectory profiles*:For each patient i, we construct a $V \times M$ trajectory profile matrix, T^i. The matrix entries of T^i are calculated as follows:

   - For non-binary variables:

$$\begin{aligned} T^i_{vt} &= 1 \text{ if } Y_{ivt} > \theta_v \\ &= 0 \text{ otherwise.} \end{aligned} \tag{5.1}$$

   where θ_v is the threshold for variable v, and is chosen to be 50 percentile in this manuscript. All individuals possess all disease variables. However, we threshold the connections to only connect the individual with the disease variables for which they have a high enough severity. This thresholding is why patients aren't shown to be connected to all variables in Fig 5.2.

- For binary variables:

  For gender: $T^i_{vt} = 1$ if the patient is male, $T^i_{vt} = 0$ otherwise *For genetic risk loci*: $T^i_{vt} = 1$ if patient contains single nucleotide polymorphisms (SNP) v, $T^i_{vt} = 0$ otherwise. Each SNP is treated as independent

4. *Create a patient-patient network with connections based on trajectory similarity*: After having defined the trajectory profile matrix T^i for each individual i, we create a patient-patient network P of all patients in the training set with an adjacency matrix given by:

$$P_{ij} = \sum_{v,t} (T^i_{vt} \equiv T^j_{vt}). \tag{5.2}$$

   In other words, P_{ij} gives the number of matrix entries for which T^i has the same value as T^j. This formulation implies that variables are equally important. If we wanted to have different weights across variables and across the years, we can instead set $P_{ij} = \sum_{v,t} w_{vt}(T^i_{vt} \equiv T^j_{vt})$ where w_{vt} is the weight of variable v at time t.

5. *Cluster the network to identify communities/subtypes*: We then perform Louvain community detection [101] to maximize the Newman-Girvan modularity function [102] on the uni-partite network defined by the weighted matrix P.

The number of communities that maximizes the modularity is picked. This allows us to cluster trajectory profiles, and hence patients, into communities (subtypes).

6. *Construct aggregate profiles to characterize each community/subtype*: We average the trajectory profiles of all patients in each community C^l to obtain the community/subtype profile S^l, a $V \times M$ matrix with elements defined by

$$S^l_{vt} = \frac{\sum_{i \in C^l} T^i_{vt}}{N_l U_{v0}} \tag{5.3}$$

where N_l is the total number of individuals in community C^l. U_{v0} is a normalization constant that represents the average value for variable v in the baseline: $U_{v0} = \frac{\sum_i T^i_{v0}}{N}$, and again 0 denotes the baseline year.

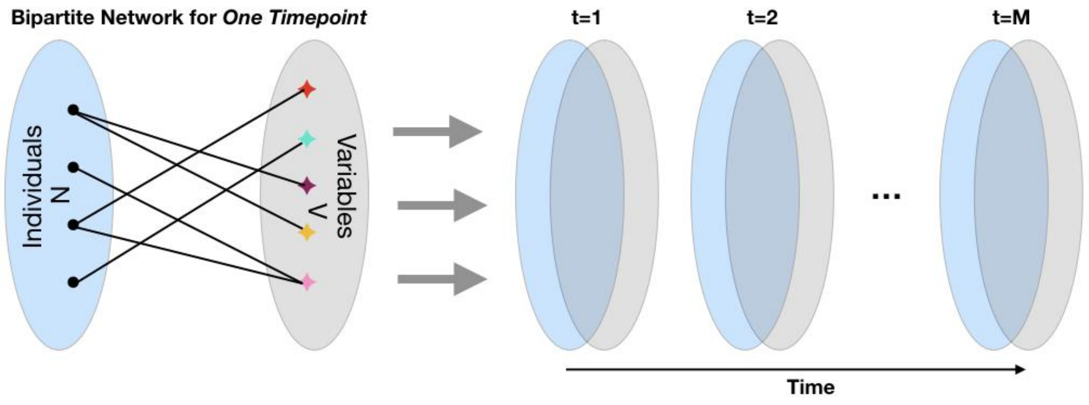


Figure 5.2: (left) An illustration of an individual-variable bipartite graph at one timestep. (right) Set of bipartite graphs across time.

5.2.3 Prediction scheme for test patients

From baseline data, we predict the community/subtype that an individual test patient (patient whose data was not used in identifying the PD subtypes) belongs to. We then check whether the test patient is still aligned with the same community/subtype after 4 years to demonstrate the utility of our baseline prediction.

To predict test patient is subtype from his/her baseline profile, we find the community (subtype) C^l whose baseline community profile, with elements S^l_{v0}, has the smallest Euclidean distance from the patients baseline profile. In other words, l is chosen to minimize the distance

$$d^{il}_0 = \sqrt{\sum_v (T^i_{v0} - S^l_{v0})^2}. \tag{5.4}$$

Does the patients trajectory match the subtypes trajectory? We then investigate the quality of the subtype/community baseline prediction at a later time t by calculating the patients subtype/community C^l such that l is chosen to minimize the distance between the community profile and the patients profile at time t:

$$d^{il'}_t = \sqrt{\sum_v (T^i_{vt} - S^{l'}_{vt})^2}. \tag{5.5}$$

The prediction accuracy is then defined as the fraction of test patients for which the subtype identification (l) from the baseline matches the subtype identification (l) at a later time t.

5.3 Results

5.3.1 TPC algorithm for PD subtype identification

In this section, we present the disease subtypes (communities) identified by our method from the training patient data. Maximizing Newman-Girvan modularity on the patient-patient trajectory profile network gives us three distinct subtypes, i.e., three is the optimal number of subtypes for this data indicated by the modularity measure.

The darkness of the shade of grey of a continuous variable in a year denotes the fraction of the subtype population that has a value above the median of the total population baseline for that variable. The darkness of the shade of a grey for a binary variable is the fraction of the subtype population containing that variable (male in the case of the variable gender). For some scales or tests, a higher score implies a healthier/less severely affected patient (such as the Montreal Cognitive Assessment), while for other scales, the opposite is true (higher score = greater severity). Therefore, in step one of our algorithm we normalized the data, so that for all variables except for the genetic and demographic variables, a higher score is associated with greater severity of that variable and a deeper shade of grey.

5.3.2 Description of the subtypes

As seen in Fig 5.3, subtype 1 is small ($n = 37$), largely female and relatively young. Subtype 2 is the largest ($n = 62$), highly male, and shows a clearly different cognitive profile with better scores on some cognitive variables (SFT, HVLT, LNS), but worse scores on others (JOLO, SDM, MoCA). It is also largely male and older, with greater severity in variables describing general PD severity, sleep, autonomic function and mental health. Subtype 3 is fairly large ($n = 53$),with a milder disease course (less disease progression from baseline to year 4) on measures of general PD severity, mental health, autonomic function and sleep. Interestingly, the six cognitive variables split into two sub-categories (Cog I: JOLO, SDM, MoCA and Cog II: SFT, HVLT, LNS) especially in subtypes 1 and 2. The bottom panel in Fig 5.3 shows the profile of the total population. Since the threshold variable severity in an individual is set with respect to the median of the total population at baseline, the total population baseline profile for all variables has a value close to 0.5 (i.e., 50% of the total population at baseline has a value of 1 for any variable, and the other half has a value of 0). Fluctuations of the baseline total population value around 0.5 may occur if the precise value of a variable in the baseline year for multiple people coincides with the variable median of the entire population.

5.3.3 Early prediction of patient subtypes

In addition to identifying PD subtypes, our method predicts the individual patient subtype years in advance. In this section the test patient cohort ($n = 39$) is used

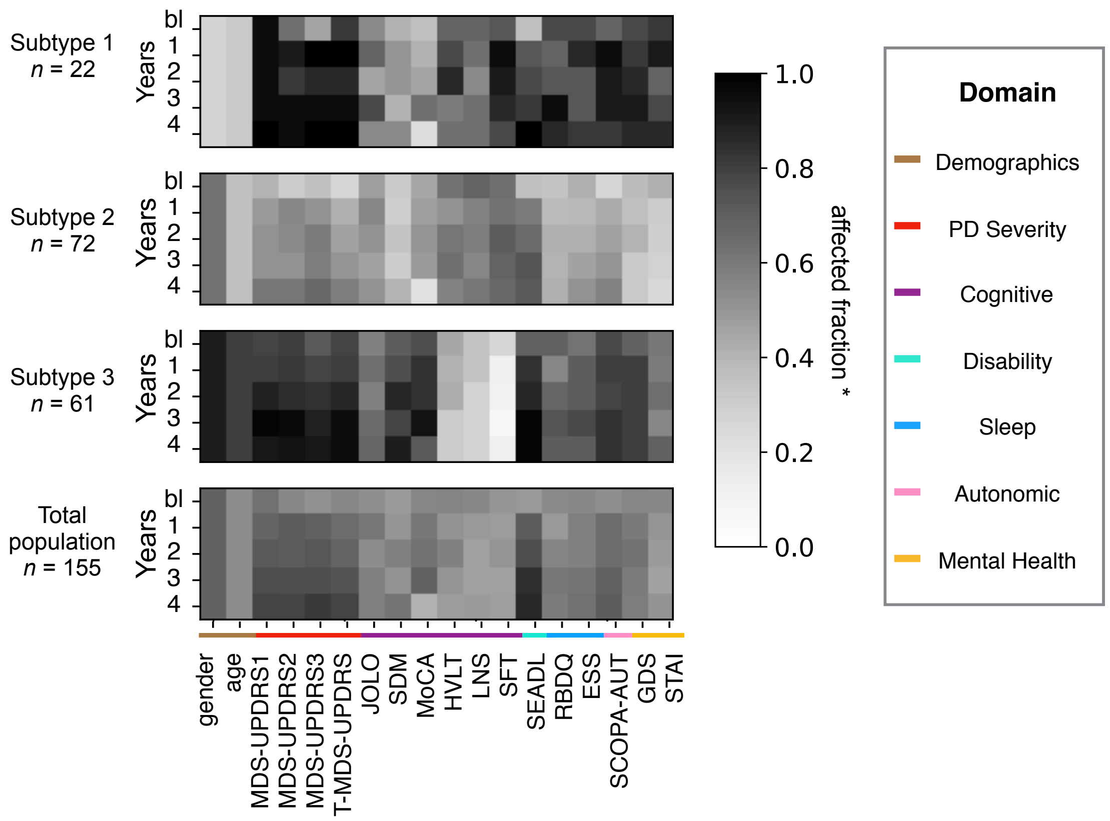


Figure 5.3: Subtypes/communities identified by our algorithm: top three panels show three subtype/community profiles (average profile of all patients in the subtype). Subtypes identified by the algorithm containing fewer than 10 patients are not shown. The bottom panel shows the total population profile. The shade of grey indicates the affected fraction, i.e, fraction above baseline median in the direction of disease progression for the continuous variables, and fraction that is male for gender. n is the number of patients in the subtype. The variable names are listed below the panels (See Fig 5.1 for description).

to assess the accuracy of early prediction of disease subtype. Data from these test patients is not used in the identification of the subtypes. Fig 5.4 shows the prediction of future PD subtype based on baseline data for 39 test patients that run across the horizontal axis. The top panel shows the Euclidian distance between the baseline profile of a patient and the baseline profile of each subtype (subtypes are shape coded). The subtype with which the patient has minimum baseline distance is the ‘predicted subtype, and is marked in red. Patients are organized from left to right in order of decreasing confidence, i.e., from minimum to maximum distance of the patients baseline profile with the predicted subtype baseline profile. The remaining panels represent subsequent years, plotting the distance between the patient profile and subtype profiles in corresponding years. The red coding of predicted community makes it easy to track across the years. Finally, in year 4, we assess the accuracy of our predictions. Our algorithm identifies subtypes based on disease progression, i.e., the patterns of disease variables as they evolve through time. One of the contributions of our method is that the subtypes identified take temporal evolution into account. Hence, if our algorithm were successful in subtype prediction in test patients, we would expect the patients to fall in the same subtype through all the years. If the subtype with minimum distance to the patient in year 4 is the same as the originally predicted subtype, then we consider our prediction to be successful for that patient. In other words, if in year 4 the red subtype for a patient is below the black subtypes then our prediction is successful. For the newly diagnosed (test) PD patients in the PPMI dataset, our algorithm predicts the PD subtype after 4 years of disease progression with 82% accuracy.

5.3.4 Incorporating genetic data into the TPC algorithm

Genetic variants are increasingly recognized as important determinants of disease subtype and disease progression and prognosis. As an exploratory objective, we investigated the integration of genetic variants (single nucleotide polymorphisms, or SNPs) in previously identified PD risk loci into our TPC-based approach. Each

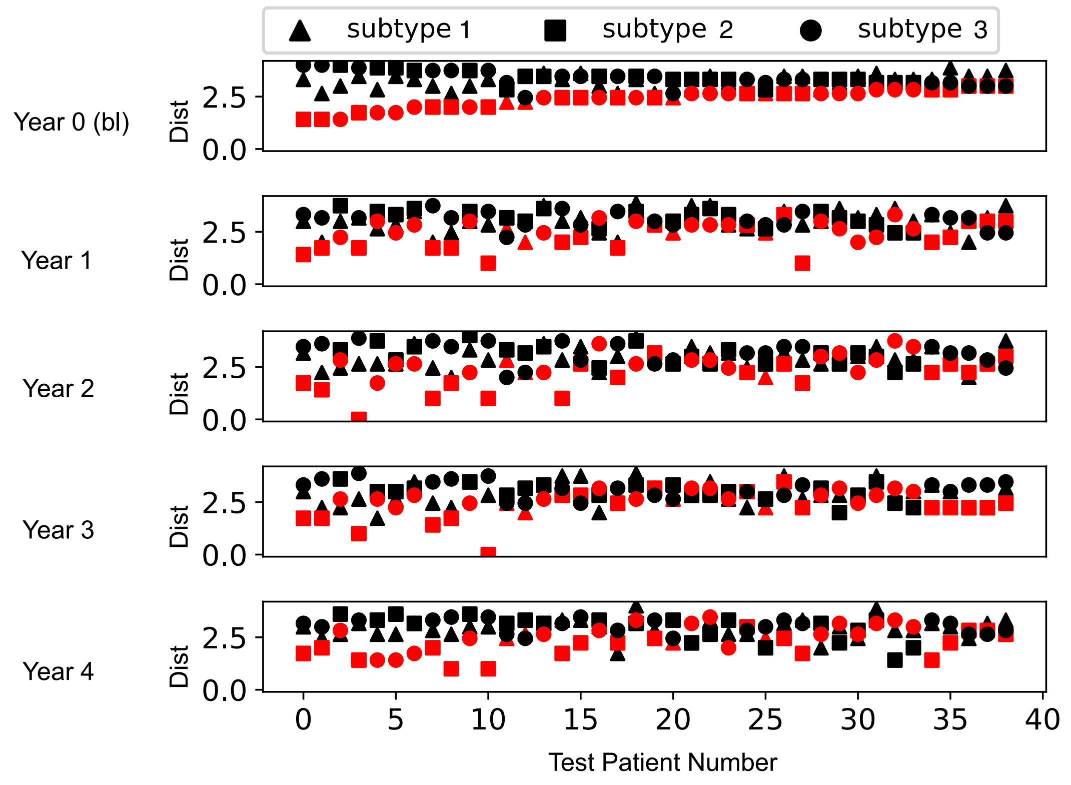
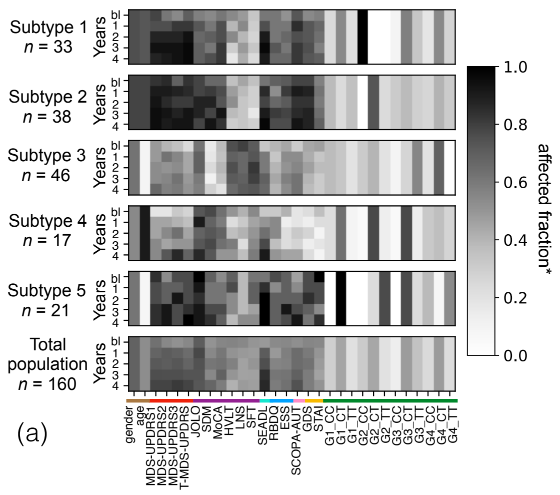


Figure 5.4: The i^{th} panel (row) shows the distance between the test patient i^{th} year profile and the i^{th} year subtype profile (shape coded). The predicted subtype for each individual (subtype with minimum baseline-year distance) is colored red to allow for tracking across the years (panels). Prediction accuracy in year 4 is 82%. Data includes 39 test patients and 18 clinical variables across 5 time points: baseline (bl) or year 0 + years 1,2,3,4).

patient has 2 copies for each piece of genetic information, and there are by definition 2 variants for each SNP. Hence, there are 3 possible combinations of the 2 variants for each of the genetic risk loci. PPMI contains information for 28 such SNPs for each patient. As a proof of principle, we selected 4 of those 28 SNPs to be included in our approach as a proof-of-principal. Ideally, this selection should be based on established pathogenic/biological significance of the genetic variants in question. For one of those SNPs (rs356181/2), an association with PD motor features has recently been described [100], making this an obvious choice for our study. None of the other SNPs have been associated with any identifiable clinical variations in PD. In the absence of any established genotype-phenotype correlation for the other risk loci, the selection of the additional 3 SNPS to be included in our study remained arbitrary. We therefore picked the additional 3 SNPs based on their high minor allele frequency, so that all 3 possible combinations of the 2 genetic variants were present in sufficient numbers in our study population of 194 subjects.

Fig 5.5 shows the five subtypes identified when genetic data is introduced. Five is the optimal number of subtypes that maximizes the network modularity measure for the data containing clinical and genetic information. The plots Fig 5.5(a,b) are organized in the same way as Fig 5.3 and 4 respectively. In Fig 5.5(a), the darkness of the shade of grey of a variable in a year denotes the fraction of the subtype that has a value above the baseline median of the total population of that variable. Subtypes 1 (top) and 2 have relatively similar clinical profiles, with the difference being in their genetic profile. The CC allele of G2, and CT allele of G3 and G4 are frequent in subtype 1, and the CT allele of G2is frequent in subtype 2. Subtypes 1,2 and 3 have alternate sets of cognitive variables dominating, a feature also observed in Fig 5.3. Subtype 3 and 4 have the mildest disease progression profile. Subtype 2 has a larger population ($n = 46$), and has the largest proportion of young patients. Subtype 4 has less psychiatric symptomatology and behavioral variables but intermediate motor (MDS-UPDRS3). Each of the subtypes has a distinct genetic profile. When genetic data is added to the analysis, prediction of patient subtype in the test group shows an accuracy of 64%. This prediction is significantly less accurate than prediction when using only the clinical data.



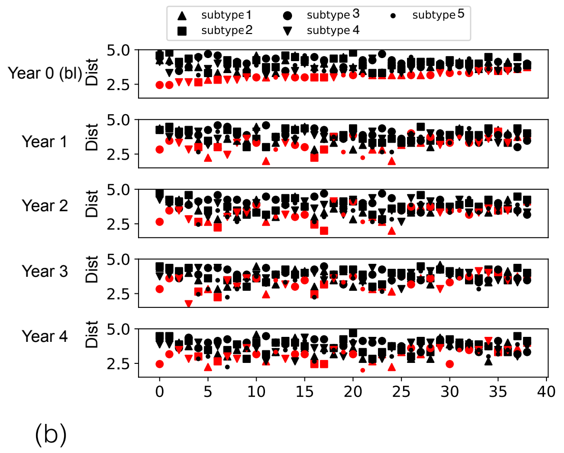


Figure 5.5: (a) Top five panels show five average community (subtype) profiles C, identified by our TPC algorithm. Subtypes identified by the algorithm containing less than 10 patients are not shown. The bottom panel shows the total population profile. The legend is a measure of the affected fraction, i.e, fraction above baseline median in the direction of disease progression for the continuous variables, and fraction that is male and fraction containing the genetic SNP for gender and genetic variables respectively. n is the number of patients in the community. (b) The m^{th} panel shows the distance between the test patient m^{th} year profile and the m^{th} year profile of the subtypes (shape coded). The predicted subtype for each individual (subtype with minimum baseline distance) is colored red to allow for tracking across the years (panels). Prediction accuracy in year 4 is 64%. Data includes 39 test patients and 18 clinical variables across 5 time points: baseline (bl) or year 0 + years 1,2,3,4.

5.4 Discussion

Multidimensional clinical datasets are valuable resources that are not used to their full potential due to the analytic challenges of diverse biomarkers and outcome variables. We describe development of a method to identify disease subtypes based on the pattern of progression of multidimensional clinical data including demographics, clinical variables, and genetics. We then validate our method by measuring the accuracy of subtype prediction in individual patients based on baseline clinical and genetic variables. The disease subtypes are characterized by patterns of progression of the clinical variables. The concordance between our agnostic results with the domain-structure of the variables supports our approach. For instance, in the analysis on identifying PD subtypes, in spite of treating the variables independently, all the variables in the General PD severity domain show similar trends in each subtype. For example, subtypes 1 and 3 have high progression of all PD severity variables and subtype 2 has a low progression of all PD severity variables. Variables of other domains such as Sleep, Mental Health and Cognition show similar trends within a subtype (although Cognitive shows an additional layer of differentiation into Cog I and Cog II).

Our predictions of the future subtype of individual patients in the test sample based on their baseline data, shows good accuracy in predicting disease subtypes four years later (82% for clinical data and 64% for clinical+genetic data). The explanation for the reduction in predictive accuracy with addition of genetic data may be due to: 1) the inclusion of a very limited number of genetic risk loci, 2) that SNPs are more potent predictors of risk in PD than the clinical phenotype, 3) that genetic data isnt predictive of PD subtype within the 4-year time frame of our data or 4) that the genetic data has a large variance in the population, thus requiring a larger dataset for long-term prediction (the larger number of subtypes found by our method may indicate this). Nonetheless, this exploratory work successfully demonstrates the inclusion of genetic data in this approach. Other biomarkers (i.e. serologic and cerebrospinal fluid biomarkers) can also be easily integrated into our analysis. Our algorithm is likely to benefit from more extensive datasets with larger populations.

A number of studies have identified PD subtypes based on baseline characteristics [103, 104, 105]. In this work, we used the longitudinal data to identify disease subtypes. In other words, our method accounts for both disease variable values as well as their progression patterns. To our knowledge, this is a novel approach. The baseline features of individual patients in a test cohort were then used to predict their future disease trajectory (prognosis). Our study represents an innovative network-based data-driven approach, that has advantages over previous methods by taking full advantage of large heterogenous, longitudinal datasets.

Genetic factors likely play a major role in sdetermining PD subtypes, however, few data-driven algorithms for suptype identification exist that incorporate genetic data. Two recent studies have developed models of PD progression based on clinical, demographic and genetic data at baseline, using hierarchical cluster analysis and a Bayesian multivariate predictive inference platform, respectively, to identify PD subtypes that show significant differences in their rate of progression over time [88, 99]. Only one of these studies assessed the predictive accuracy of their model by applying the coefficient of determination as a measure of overall explanatory power in an independent validation cohort [99].

Our trajectory clustering method works with various types of data including clinician- and patient-reported outcome measures, genetic alleles, physical performance measures, as well as diverse results from diagnostic investigations. This first approach uses demographics, clinician- and patient-reported data, and genetic data. In this analysis, each genetic SNP and clinical variable is treated independently and allotted the same weight. Our algorithm allows for variable weightings, where each domain and SNP is assigned a chosen weight. However, this raises the question of how the weighting would be decided. For example, if we had allotted equal weights to one hundred SNPs in our analysis in addition to the 18 clinical variables, the genetic information would dominate the algorithm, and affect the resulting communities. On the other hand, different weighting strategies may be preferable based on the study aims. For example, if the main objective is to identify disease subtypes based on motor vs. cognitive function, one could allot equal cumulative weight to the motor and cognitive domains.

A strength of our algorithm, which is also a caveat, is that it is entirely data-driven. The level of severity of each variable relative to the baseline median is used to normalize all variables, as opposed to the absolute value of the variable. This is done so as to readily compare changes in different variables. A notable example is the clinical variable, SEADL (a disability scale). SEADL is a relatively insensitive scale in the early years of PD since there is little functional disability in the years following diagnosis. Yet, in our analysis SEADL shows high progression (darker shade in later years) in Fig 5.3 and 5.5(a). It is important to note that this dark shade isnt indicative of the absolute severity. It only tells us that a larger fraction of the population in the later years has SEADL values above the baseline mean (which may be low to begin with). The limitations of our work are as follows: it is entirely data-driven, and must be used in practice in conjunction with medical expertise. We make several choices (thresholds, choice of weighting scheme of the network, variables to include etc.). While such a data-driven method is entirely agnostic, the decisions on these choices must be made with care, and may be different for different applications/datasets. Lastly, like any data-driven method, the robustness of the method is proportional to the quantity of data. Hence, while this method is suitable for heterogenous, time-varying datasets such as the PPMI dataset, there may be other datasets that have large gaps in data collection, inconsistent times of acquiring data, too much variation in data or simply too little data, that our method may not be suitable for.

Our approach is innovative, adaptable, and clinically relevant. PD subtyping [106] is an area of active research but there are currently no clinically prognostic analyses in use for the management of PD. Application of an approach like ours for subtype identification as a predictive model of PD progression allows for identification of the clinical significance of PD subtypes based not only on baseline features, but also on complex progression of multiple variables over time. It will help the neurologist improve clinical management of individual patients. For example, such a model may prompt the clinician to pursue an earlier, more aggressive management for those patients for whom the model predicts a more rapid disease progression ('precision medicine'). It may also guide the neurologist to probe in more detail for specific symptoms (e.g., cognitive impairment) predicted in individual patient based on their subtype that could otherwise go unnoticed (and hence untreated) for longer in a less targeted patient evaluation. Finally, prediction of disease progression will improve prognostic counseling, a problem commonly encountered by clinicians, by bringing to attention disease variables that are predicted to develop over the course of the disease. A natural extension of this work involves implementing the method for datasets in other chronic medical conditions. Another interesting direction involves extending the TPC algorithm to incorporate and compare other network clustering approaches, such as multi-layer network clustering [107]. Other future directions include studies of the effect of treatment on progression of disease variables, and predicting modifications of algorithm-identified subtypes as a consequence of different treatments.

5.5 Data Availability

Data was obtained from PPMI. PPMI a public-private partnership is funded by the Michael J. Fox Foundation for Parkinsons Research.

Chapter 6: Conclusion

In this thesis we have used concepts from dynamical systems and network science to identify patterns in complex data and make predictions. We have demonstrated this on different types of data - images, videos, chaotic signals, and multi-variate disease data.

Using Reservoir Computing as a machine learning model, we have identified patterns/relationships in pairs of images in a generalizable way with limited data and inexpensive training. We use this to make predictions on relationships between images from classes that are unseen. We see the strength of our work as lying not only in its demonstration of the utility of RCs for generalization using small datasets, but also in our ability to interpret the generalization ability through clustering of the dynamics of the reservoir state.

RCs are particularly well suited for processing temporal signals. Signal separation is a difficult problem, especially when the signals are chaotic and the sources are fewer than the numbers of signals being mixed. The cocktail party problem, where a party-goer hears several voices at once but is able to separate out a single voice, is easily done by the human brain and yet is a much harder in artificial systems. In our work, we solved a chaos version of the cocktail party, i.e., separation of chaotic signals generated from the Lorenz system, using an RC. Often, in signal separation applications, the mixing fraction (amplitude ratio) of the signals to be separated is unknown. Our RC-based method can separate signals with an arbitrary unknown amplitude ratio. This is particularly useful in applications such as blind source separation and noise removal from unknown sources with unknown parameters. Several state-of-the-art signal separation techniques are spectra-based, for instance, the Wiener filter, which can be shown to be the best linear filter in the limit of infinite training. Spectra-based methods, naturally, will perform poorly at signal separation if the spectra of the signals to be separated are very similar. Contrarily, the reservoir performs reasonably well even when the two signals that are mixed have very similar spectra, and outperforms the Wiener filter calibrated from the same data for all the scenarios we considered.

Chaotic systems show high sensitivity to initial conditions and are difficult to predict and separate, especially with only data, without knowledge of the underlying system. To understand why the RC is able to succesfully predict chaotic signals, we explored its dynamics with a mathematical attractor-finding tool called 'directional fiber'. We found that reservoir encodes dynamical properties (such as attractors) of the system it is trained on even when trained with only a limited amount of time series data. Additionally, the reservoir learns 'spurious' fixed points in the high dimensional reservoir space. Proximity to the spurious fixed points during testing causes prediction to fail. This allowed us to gain insight into the ways by which a high (several hundred) dimensional system uncovers patterns and enables prediction through projecting data from a chaotic three dimensional system.

Unconvering patterns in complex datasets with many variables whose interactions evolve with time may benefit from a network approach. We studied medical data from Parkinson's disease, a progressive multi-variate disease influenced by several interacting variables of different types including demographics, clinical variables, and genetics. We developed a network-based algorithm to identify disease subtypes based on the pattern of progression of multidimensional data. We then validated our method by measuring the accuracy of subtype prediction in test patients. We were able to predict Parkinson's subtype in test patients with 74% accuracy 4 years in advance. Our method is applicable to other multidimensional temporal datasets as well, and provides a data-driven way of uncovering clinically relevant disease patterns.

We use approaches from network science, machine learning and dynamical systems to uncover patterns in complex data. This relates to new ideas in explainable data-driven mathematical modeling and artificial intelligence, topics that continues to receive traction. An interesting direction would be to explore different reservoir-like architectures that may be better suited for different types of data. Another promising direction could be to study synchronization patterns in the nodes of machine learning systems like RCs and their role in learning. Lastly, a possible extension of our network algorithm with potential tangible impact could be to extend to other datasets and incorporate other network clustering approaches into our algorithm, such as multi-layer network clustering.

Bibliography

[1] Norbert Wiener. Extrapolation, interpolation, and smoothing of stationary time series, vol. 2, 1949.

[2] Garrett E Katz and James A Reggia. Using directional fibers to locate fixed points of recurrent neural networks. *IEEE transactions on neural networks and learning systems*, 29(8):3636–3646, 2018.

[3] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2):024102, 2018.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, 2012.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[6] Bo Du, Wei Xiong, Jia Wu, Lefei Zhang, Liangpei Zhang, and Dacheng Tao. Stacked convolutional denoising auto-encoders for feature representation. *IEEE transactions on cybernetics*, 47(4):1017–1027, 2017.

[7] Zachary Chase Lipton. A critical review of recurrent neural networks for sequence learning. *CoRR*, abs/1506.00019, 2015. URL `http://arxiv.org/abs/1506.00019`.

[8] Wenhao Zhu, Tengjun Yao, Jianyue Ni, Baogang Wei, and Zhiguo Lu. Dependency-based siamese long short-term memory network for learning sentence representations. *PloS one*, 13(3):e0193919, 2018.

[9] Yu Zhang, William Chan, and Navdeep Jaitly. Very deep convolutional networks for end-to-end speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4845–4849. IEEE, 2017.

[10] Herbert Jaeger. The" echo state" approach to analysing and training recurrent neural networks. 148, 01 2001.

[11] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.*, 14(11), November 2002. doi: 10.1162/089976602760407955.

[12] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[13] Pierre Enel, Emmanuel Procyk, Ren Quilodran, and Peter Ford Dominey. Reservoir computing properties of neural dynamics in prefrontal cortex. *PLOS Computational Biology*, 12(6):1–35, 06 2016. doi: 10.1371/journal.pcbi.1004967. URL `https://doi.org/10.1371/journal.pcbi.1004967`.

[14] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. 521:436–44, 05 2015.

[15] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, volume 16, pages 2786–2792, 2016.

[16] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 378–383. IEEE, 2016.

[17] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4): 98, 2015.

[18] Masanobu Inubushi and Kazuyuki Yoshimura. Reservoir computing beyond memory-nonlinearity trade-off. *Scientific Reports*, 7(1):10199, 2017.

[19] Nuno Maarico Da Costa and Kevan A.C. Martin. The proportion of synapses formed by the axons of the lateral geniculate nucleus in layer 4 of area 17 of

the cat. *The Journal of Comparative Neurology*, 516(4):264–276, 2009. ISSN 1096-9861. doi: 10.1002/cne.22133. URL `http://dx.doi.org/10.1002/cne.22133`.

[20] Stefan Haeusler and Wolfgang Maass. A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1):149–162, 2007. doi: 10.1093/cercor/bhj132. URL `http://dx.doi.org/10.1093/cercor/bhj132`.

[21] Wolf Singer Danko Nikolic, Stefan Haeusler and Wolfgang Maas. Temporal dynamics of information content carried by neurons in the primary visual cortex. NIPS'06, 2006.

[22] C. van Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996. ISSN 0036-8075. doi: 10.1126/science.274.5293.1724. URL `http://science.sciencemag.org/content/274/5293/1724`.

[23] Rune Rasmussen, Mogens H Jensen, and Mathias L Heltberg. Chaotic dynamics mediates brain state transitions, driven by changes in extracellular ion concentrations. *Cell Systems*, 5(6):1–13, 2017.

[24] Peter J. Urcuioli, Edward A. Wasserman, and Thomas R. Zentall. Associative concept learning in animals: Issues and opportunities. *Journal of the Experimental Analysis of Behavior*, 101(1):165–170, 2014. ISSN 1938-3711. doi: 10.1002/jeab.62. URL `http://dx.doi.org/10.1002/jeab.62`.

[25] Martin Giurfa, Shaowu Zhang, Arnim Jenett, Randolf Menzel, and Mandyam V. Srinivasan. The concepts of 'sameness' and 'difference' in an insect. 410: 930–933, 03 2001.

[26] Samuel C. Andrew, Clint J Perry, Andrew B Barron, Katherine Berthon, Verónica Peralta, and Ken Cheng. Peak shift in honey bee olfactory learning. *Animal Cognition*, 17:1177–1186, 2014.

[27] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[28] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[29] J. J. Hopfield. Neurocomputing: Foundations of research. chapter Neural Networks and Physical Systems with Emergent Collective Computational Abilities. 1988. ISBN 0-262-01097-6.

[30] H. Jaeger. Controlling recurrent neural networks by conceptors. 2014. doi: arXiv:1403.3369.

[31] Yonggang Qi, Yi-Zhe Song, Honggang Zhang, and Jun Liu. Sketch-based image retrieval via siamese convolutional neural network. In *Image Process-*

ing (ICIP), 2016 IEEE International Conference on, pages 2460–2464. IEEE, 2016.

[32] Cheng Zhang, Wu Liu, Huadong Ma, and Huiyuan Fu. Siamese neural network based gait recognition for human identification. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2832–2836. IEEE, 2016.

[33] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017.

[34] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[35] Akshay Mehrotra and Ambedkar Dukkipati. Generative adversarial residual pairwise networks for one shot learning. *arXiv preprint arXiv:1703.08033*, 2017.

[36] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[37] Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott. Reservoir observers: Model-free inference of unmeasured

variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102, 2017.

[38] Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. Information processing capacity of dynamical systems. *Scientific reports*, 2: 514, 2012.

[39] Nils Schaetti, Michel Salomon, and Raphaël Couturier. Echo state networks-based reservoir computing for mnist handwritten digits recognition. *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 484–491, 2016.

[40] Jean Bullier. Integrated model of visual processing. *Brain research. Brain research reviews*, 36 2-3:96–107, 2001.

[41] Emmanuelle Volle, Sam J. Gilbert, Roland G. Benoit, and Paul W. Burgess. Specialization of the rostral prefrontal cortex for distinct analogy processes. *Cerebral Cortex*, 20(11):2647–2659, 2010. doi: 10.1093/cercor/bhq012.

[42] Mustafa C Ozturk, Dongming Xu, and José C Príncipe. Analysis and design of echo state networks. *Neural computation*, 19(1):111–138, 2007.

[43] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer*

Vision and Pattern Recognition (CVPR'06), volume 2, 2006. doi: 10.1109/CVPR.2006.100.

[44] Hossein Rahmani, Ajmal Mian, and Mubarak Shah. Learning a deep model for human action recognition from novel viewpoints. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):667–681, 2018.

[45] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1386–1383. IEEE, 2017.

[46] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. 9: 2579–2605, 11 2008.

[47] Francis Wyffels, Benjamin Schrauwen, and Dirk Stroobandt. Stable output feedback in reservoir computing using ridge regression. In *Proceedings of the 18th International Conference on Artificial Neural Networks*, ICANN '08, 2008. ISBN 978-3-540-87535-2.

[48] Shishir B. Vengamoorthy G.K. Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks*, 2009.

[49] Jayanta Basak, Anant Sudarshan, Deepak Trivedi, and MS Santhanam. Weather data mining using independent component analysis. *Journal of Machine Learning Research*, 5(Mar):239–253, 2004.

[50] Marco Congedo, Cedric Gouy-Pailler, and Christian Jutten. On the blind source separation of human electroencephalogram by approximate joint diagonalization of second order statistics. *Clinical Neurophysiology*, 119(12): 2677–2686, December 2008. doi: 10.1016/j.clinph.2008.09.007. URL `https://hal.archives-ouvertes.fr/hal-00343628`.

[51] Nikolaos Doukas and Nikolaos V Karadimas. A blind source separation based cryptography scheme for mobile military communication applications. *WSEAS Trans. Commun*, 7(12):1235–1245, 2008.

[52] Arturo Buscarino, Luigi Fortuna, and Mattia Frasca. Separation and synchronization of chaotic signals by optimization. *Phys. Rev. E*, 75:016215, Jan 2007. doi: 10.1103/PhysRevE.75.016215. URL `https://link.aps.org/doi/10.1103/PhysRevE.75.016215`.

[53] Lev S Tsimring and Mikhail M Sushchik. Multiplexing chaotic signals using synchronization. *Physics Letters A*, 213(3-4):155–166, 1996.

[54] TL Carroll and LM Pecora. Using multiple attractor chaotic systems for communication. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 9 (2):445–451, 1999.

[55] Paolo Arena, Arturo Buscarino, Luigi Fortuna, and Mattia Frasca. Separation and synchronization of piecewise linear chaotic systems. *Phys. Rev. E*, 74: 026212, Aug 2006. doi: 10.1103/PhysRevE.74.026212. URL `https://link.aps.org/doi/10.1103/PhysRevE.74.026212`.

[56] Yang Jianning and Fang Yi. Blind separation of mixing chaotic signals based on ica using kurtosis. In *2012 International Conference on Computer Science and Service System*, pages 903–905. IEEE, 2012.

[57] Masahiko Kuraya, Atsushi Uchida, Shigeru Yoshimori, and Ken Umeno. Blind source separation of chaotic laser signals by independent component analysis. *Optics Express*, 16(2):725–730, 2008.

[58] Lü Shan-Xiang, Wang Zhao-Shan, Hu Zhi-Hui, and Feng Jiu-Chao. Gradient method for blind chaotic signal separation based on proliferation exponent. *Chinese Physics B*, 23(1):010506, 2013.

[59] Simon Leglaive, Romain Hennequin, and Roland Badeau. Singing voice detection with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 121–125. IEEE, 2015.

[60] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*, 2017.

[61] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

[62] Kun Han and DeLiang Wang. A classification based approach to speech seg-

regation. *The Journal of the Acoustical Society of America*, 132(5):3475–3483, 2012.

[63] Mikkel N Schmidt and Rasmus K Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Ninth International Conference on Spoken Language Processing*, 2006.

[64] John B Butcher, David Verstraeten, Benjamin Schrauwen, CR Day, and PW Haycock. Reservoir computing and extreme learning machines for non-linear time-series data analysis. *Neural networks*, 38:76–89, 2013.

[65] Sanjukta Krishnagopal, Yiannis Aloimonos, and Michelle Girvan. Similarity learning and generalization with limited data: A reservoir computing approach. *Complexity*, 2018, 2018.

[66] Miguel Angel Escalona-Morán, Miguel C Soriano, Ingo Fischer, and Claudio R Mirasso. Electrocardiogram classification using reservoir computing with logistic regression. *IEEE Journal of Biomedical and health Informatics*, 19(3): 892–898, 2014.

[67] Aida A Ferreira, Teresa Bernarda Ludermir, Ronaldo RB de Aquino, Milde MS Lira, and Otoni Nóbrega Neto. Investigating the use of reservoir computing for forecasting the hourly wind speed in short-term. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1649–1656. IEEE, 2008.

[68] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and

Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[69] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[70] Warwick Tucker. A rigorous ode solver and smale's 14th problem. *Foundations of Computational Mathematics*, 2(1):53–117, 2002.

[71] Lionel Pigou, Aäron Van Den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre. Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. *International Journal of Computer Vision*, 126(2-4):430–439, 2018.

[72] 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.

[73] Da Liu, Jilong Wang, and Hui Wang. Short-term wind speed forecasting based on spectral clustering and optimised echo state networks. *Renewable Energy*, 78:599–608, 2015.

[74] Xiaowei Lin, Zehong Yang, and Yixu Song. Short-term stock price prediction based on echo state networks. *Expert Systems with Applications*, 36(3, Part 2):7313 – 7317, 2009. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.

2008.09.049. URL http://www.sciencedirect.com/science/article/pii/S0957417408006519.

[75] 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.

[76] Yingyezhe Jin, Yu Liu, and Peng Li. Sso-lsm: A sparse and self-organizing architecture for liquid state machine based neural processors. In *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 55–60. IEEE, 2016.

[77] Bon Woong Ku, Yu Liu, Yingyezhe Jin, Sandeep Samal, Peng Li, and Sung Kyu Lim. Design and architectural co-optimization of monolithic 3d liquid state machine-based neuromorphic processor. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[78] Baram Sosis, Garrett E. Katz, and James A. Reggia. Learning in a continuous-valued attractor network. In *Proceedings of the 17th International Conference on Machine Learning and Applications (ICMLA 2018)*. IEEE, 2018.

[79] Paul Manneville and Yves Pomeau. Intermittency and the lorenz model. *Physics Letters A*, 75(1-2):1–2, 1979.

[80] David Verstraeten, Benjamin Schrauwen, Michiel d'Haene, and Dirk

Stroobandt. An experimental unification of reservoir computing methods. *Neural networks*, 20(3):391–403, 2007.

[81] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[82] Tamara Pringsheim, Nathalie Jette, Alexandra Frolkis, and Thomas DL Steeves. The prevalence of parkinson's disease: A systematic review and meta-analysis. *Movement disorders*, 29(13):1583–1590, 2014.

[83] A.J. Lees, J. Hardy, and T Revesz. Parkinson's disease. *Lancet*, 374(9691), 2009.

[84] Thomas Foltynie, Carol Brayne, and Roger A Barker. The heterogeneity of idiopathic parkinson's disease. *Journal of neurology*, 249(2):138–145, 2002.

[85] Rainer von Coelln and Lisa M Shulman. Clinical subtypes and genetic heterogeneity: of lumping and splitting in parkinson disease. *Current opinion in neurology*, 29(6):727–734, 2016.

[86] Connie Marras and Anthony Lang. Parkinson's disease subtypes: lost in translation? *J Neurol Neurosurg Psychiatry*, 84(4):409–415, 2013.

[87] Mary Ann Thenganatt and Joseph Jankovic. Parkinson disease subtypes. *JAMA neurology*, 71(4):499–504, 2014.

[88] Seyed-Mohammad Fereshtehnejad, Yashar Zeighami, Alain Dagher, and

Ronald B Postuma. Clinical criteria for subtyping parkinson's disease: biomarkers and longitudinal progression. *Brain*, 140(7):1959–1976, 2017.

[89] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, 12(1):56, 2011.

[90] César A Hidalgo, Nicholas Blumm, Albert-László Barabási, and Nicholas A Christakis. A dynamic network approach for the study of human phenotypes. *PLoS computational biology*, 5(4):e1000353, 2009.

[91] Plamen Ch Ivanov, Kang KL Liu, and Ronny P Bartsch. Focus on the emerging new fields of network physiology and network medicine. *New journal of physics*, 18(10):100201, 2016.

[92] Stephen Y Chan and Joseph Loscalzo. The emerging paradigm of network medicine in the study of human disease. *Circulation research*, 111(3):359–374, 2012.

[93] Kwang-Il Goh, Michael E Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

[94] Pascal Braun, Edward Rietman, and Marc Vidal. Networking metabolites and diseases. *Proceedings of the National Academy of Sciences*, 105(29):9849–9850, 2008.

[95] Arunachalam Vinayagam, Travis E Gibson, Ho-Joon Lee, Bahar Yilmazel, Charles Roesel, Yanhui Hu, Young Kwon, Amitabh Sharma, Yang-Yu Liu, Norbert Perrimon, et al. Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets. *Proceedings of the National Academy of Sciences*, 113(18):4976–4981, 2016.

[96] Jose A Santiago and Judith A Potashkin. A network approach to clinical intervention in neurodegenerative diseases. *Trends in molecular medicine*, 20 (12):694–703, 2014.

[97] SJG Lewis, Thomas Foltynie, Andrew D Blackwell, Trevor W Robbins, Adrian M Owen, and Roger A Barker. Heterogeneity of parkinson'fs disease in the early clinical stages using a data driven approach. *Journal of Neurology, Neurosurgery & Psychiatry*, 76(3):343–348, 2005.

[98] Peter J Castaldi, Jennifer Dy, James Ross, Yale Chang, George R Washko, Douglas Curran-Everett, Andre Williams, David A Lynch, Barry J Make, James D Crapo, et al. Cluster analysis in the copdgene study identifies subtypes of smokers with distinct patterns of airway disease and emphysema. *Thorax*, 69(5):416–423, 2014.

[99] Jeanne C Latourelle, Michael T Beste, Tiffany C Hadzi, Robert E Miller, Jacob N Oppenheim, Matthew P Valko, Diane M Wuest, Bruce W Church, Iya G Khalil, Boris Hayete, et al. Large-scale identification of clinical and genetic predictors of motor progression in patients with newly diagnosed parkinson's

disease: a longitudinal cohort study and validation. *The Lancet Neurology*, 16 (11):908–916, 2017.

[100] Christine A Cooper, Nimansha Jain, Michael D Gallagher, Daniel Weintraub, Sharon X Xie, Yosef Berlyand, Alberto J Espay, Joseph Quinn, Karen L Edwards, Thomas Montine, et al. Common variant rs356182 near snca defines a parkinson's disease endophenotype. *Annals of clinical and translational neurology*, 4(1):15–25, 2017.

[101] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[102] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[103] Seyed-Mohammad Fereshtehnejad, Silvia Rios Romenets, Julius BM Anang, Véronique Latreille, Jean-François Gagnon, and Ronald B Postuma. New clinical subtypes of parkinson disease and their longitudinal progression: a prospective cohort comparison with other phenotypes. *JAMA neurology*, 72 (8):863–873, 2015.

[104] Michael Lawton, Yoav Ben-Shlomo, Margaret T May, Fahd Baig, Thomas R Barber, Johannes C Klein, Diane MA Swallow, Naveed Malek, Katherine A Grosset, Nin Bajaj, et al. Developing and validating parkinson's disease sub-

types and their motor and cognitive progression. *J Neurol Neurosurg Psychiatry*, 89(12):1279–1287, 2018.

[105] Ali H Rajput, Michele L Rajput, Leslie W Ferguson, and Alex Rajput. Baseline motor findings and parkinson disease prognostic subtypes. *Neurology*, 89 (2):138–143, 2017.

[106] Mikko Kivela, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[107] Seyed-Mohammad Fereshtehnejad and Ronald B Postuma. Subtypes of parkinson's disease: what do they tell us about disease progression? *Current neurology and neuroscience reports*, 17(4):34, 2017.