Structure, Dynamics and Self-Organization in Recurrent Neural Networks: From Machine Learning to Theoretical Neuroscience

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr.rer.nat.)

im Fachgebiet

Informatik

vorgelegt von

Diplom Ingenieur Pau Vilimelis Aceituno
geboren am 01.02.1989 in Lleida, Spanien

‘

Die Annahme der Dissertation wurde empfohlen von:
Professor Dr. Jürgen Jost (Universität Leipzig)
Professor Dr. Benjamin Grewe (Eidgenössische Technische Hochschule Zürich)

‘

Die Verleihung des akademischen Grades erfolg mit Bestehen der Verteidigung
am 19.06.2020 mit dem Gesamptprädikat magna cum laude

Bibliographische Daten

Structure, Dynamics and Self-Organization in Recurrent Neural Networks:
From Machine Learning to Theoretical Neuroscience
(Struktur, Dynamik und Selbstorganisation in rekurrente neuronalen Netzen: Vom maschinellen Lernen zur theoretischen Neurowissenschaft)
Vilimelis Aceituno, Pau
Universität Leipzig, Dissertation, 2019
126 Seiten, 21 Abbildungen, 172 Referenzen

Abstract

At a first glance, artificial neural networks, with engineered learning algorithms and carefully chosen nonlinearities, are nothing like the complicated self-organized spiking neural networks studied by theoretical neuroscientists. Yet, both adapt to their inputs, keep information from the past in their state space and are able of learning, implying that some information processing principles should be common to both. In this thesis we study those principles by incorporating notions of systems theory, statistical physics and graph theory into artificial neural networks and theoretical neuroscience models.

The starting point for this thesis is Reservoir Computing (RC), a learning paradigm used both in machine learning [JH04] and in theoretical neuroscience [MNM02]. A neural network in RC consists of two parts, a reservoir a directed and weighted network of neurons that projects the input time series onto a high dimensional space and a readout which is trained to read the state of the neurons in the reservoir and combine them linearly to give the desired output. In classical RC, the reservoir is randomly initialized and left untrained, which alleviates the training costs in comparison to other recurrent neural networks. However, this lack of training implies that reservoirs are not adapted to specific tasks and thus their performance is often lower than that of other neural networks. Our contribution has been to show how knowledge about a task can be integrated into the reservoir architecture, so that reservoirs can be tailored to specific problems without training.

We do this design by identifying two features that are useful for machine learning: the memory of the reservoir and its power spectra. First we show that the correlations between neurons limit the capacity of the reservoir to retain traces of previous inputs, and demonstrate that those correlations are controlled by moduli of the eigenvalues of the adjacency matrix of the reservoir. Second, we prove that when the reservoir resonates at the frequencies that are present on the desired output signal, the performance of the readout increases.

Knowing the features of the reservoir dynamics that we need, the next question is how to impose them. The simplest way to design a network with that resonates at a certain frequency is by adding cycles, which act as feedback loops, but this also induces correlations and hence memory modifications. To disentangle the frequencies and the memory design, we studied how the addition of cycles modifies the eigenvalues in the adjacency matrix of the network. Surprisingly, the shape of the eigenvalues is quite beautiful [ARS19] and can be characterized using random matrix theory tools. Combining this knowledge with our result relating eigenvalues and correlations, we designed an heuristic that tailors reservoirs to specific tasks and showed that it improves upon state of the art RC in three different machine learning tasks.

Although this idea works in the machine learning version of RC, there is one fundamental problem when we try to translate to the world of theoretical neuroscience: the proposed frequency adaptation requires prior knowledge of the task, which might not be plausible in a biological neural network. Therefore the following questions are whether those resonances can emerge by unsupervised learning, and which kind of learning rules would be required.

Remarkably, these resonances can be induced by the well-known Spike Time-Dependent Plasticity (STDP) combined with homeostatic mechanisms. We show this by deriving two self-consistent equations: one where the activity of every neuron can be calculated from its synaptic weights and its external inputs and a second one where the synaptic weights can be obtained from the neural activity. By considering spatio-temporal symmetries in our inputs we obtained two families of solutions to those equations where a periodic input is enhanced by the neural network after STDP. This approach shows that periodic and quasiperiodic inputs can induce resonances that agree with the aforementioned RC theory.

Those results, although rigorous, are expressed on a language of statistical physics and cannot be easily tested or verified in real, scarce data. To make them more accessible to the neuroscience community we showed that latency reduction, a well-known effect of STDP [SMA00] which has been experimentally observed [MQW00], generates neural codes that agree with the self-consistency equations and their solutions. In particular, this analysis shows that metabolic efficiency, synchronization and predictions can emerge from that same phenomena of latency reduction, thus closing the loop with our original machine learning problem.

To summarize, this thesis exposes principles of learning recurrent neural networks that are consistent with adaptation in the nervous system and also improve current machine learning methods. This is done by leveraging features of the dynamics of recurrent neural networks such as resonances and correlations in machine learning problems, then imposing the required dynamics into reservoir computing through control theory notions such as feedback loops and spectral analysis. Then we assessed the plausibility of such adaptation in biological networks, deriving solutions from self-organizing processes that are biologically plausible and align with the machine learning prescriptions. Finally, we relate those processes to learning rules in biological neurons, showing how small local adaptations of the spike times can lead to neural codes that are efficient and can be interpreted in machine learning terms.

Acknowledgements

During the past three years I have been extremely lucky to be supervised by Jürgen Jost, who has been a flexible and insightful advisor whose ideas and support made this work possible, and who taught me how to think conceptually and motivated me to use intuitive geometric intuitions when possible.

None of the results presented here would have been possible without the patience and work of many collaborators. Specially valuable where the discussions and detailed revisions with Yang-Yu Liu and Gang Yan, without whom I could not have completed the research on Echo State Networks. Similarly, Masud Ehsani helped refine and prune many of the present arguments on synaptic plasticity. Finally, I am also thankful to Henning Schommerus and Tim Rogers, whose knowledge of random matrix theory was instrumental to understand the subtleties of large networks.

I have also benefited greatly from the scientific interactions with some of the more experienced members of the institute. The discussions with Tobias, Slava, Ivan, Guido, Nihat and Matteo where extremely instructive and taught me not only how to solve specific problems, but also how to think and behave as a researcher. On the other direction, I have to thank the Laura State and George Sivulka, two promising graduate students which worked under my supervision and showed me the limits of my own perspectives.

Besides the purely scientific interactions, I have also benefited from the help of administrative staff that shielded me from most of the non-scientific problems related to my thesis. I am specially grateful to the Antje Vandenberg and the Library staff for making the work of a scientist really about science.

This thesis would not have been possible without the funding of the Max Planck Society, both from the Max Planck Institute for Mathematics in the Sciences and from the Max Planck School of Cognition. I should also mention ”la Caixa” foundation, which sponsored my stay in Boston and gave me the chance to come back to research.

Besides a dive into research, the past three years where a fascinating personal experience. In this regard, I was fortunate to enter a friendly and open group of researchers who became friends. A special thanks goes to Paolo, Sharwin, Renan, Kostas, Caio, Gerardo, Raffaella, Felix, Marzieh, Claudia, Carolin and Zack.

Finally, even the most pleasant work interactions can sometimes be stressful and frustrating. For their unwavering support and help through my Ph.D. I have to thank my parents, my brother Miquel and my girlfriend Svenja.

Contents

List of Figures

List of Tables

Abbreviations

ESN Echo State Network

LIF Leaky Integrate and Fire

LTD Long Term Depression

LTP Long Term Potentiation

NRMSE Normalized Root Mean Squared Error

PSD Power Spectral Density

RC Reservoir Computing

STDP Spiking Time-Dependent Plasticity

Part I

Introduction

Scope of this thesis

1 Motivation

How organisms and computers are able to take noisy and messy information from the environment and eventually make inferences or predictions, and even find explanations for the processes they are exposed to is a fascinating question that motivated this thesis. The fact that the best modern learning systems are inspired by the way our own brains are structured makes the study of neural networks the obvious choice to approach this problem.

Yet, just because we call both artificial learning systems and biological models neural networks, does not mean that both are equivalent. Despite the use of the term ”neuron”, most modern research in neural networks for machine learning and computational neuroscience differ at many levels. Machine Learning scientists and engineers use abstract neurons that have simple and differentiable activation functions in a discrete time setting typically with feed-forward networks [Bis95] that can be taught to perform some task with learning algorithms such as backpropagation, while computational neuroscientists use complex neuron models [HH53, Lap07b, Izh04] with heavily recurrent architectures [GKNP14, Spo10] whose only goal is to reproduce activity from biological observations, with learning being enforced by local activity-driven synaptic learning rules [DA01, SMA00].

Thus we must be careful to remain at the intersection of the two fields to develop theories and concepts that are compatible with either. This does not imply that the specific systems that we will study must always be biologically inspired or machine learning based, but rather that any result must have a crystal clear interpretation in either case. The features that we will be maintaining through this thesis is the large number of neurons, the predominance of recurrent connections and the fact that computation will be performed by the dynamics of the neural activity.

2 Approach and tools

Those questions can be addressed from many different perspectives. Since this is a Ph.D. done in a mathematics institute, it is natural to take a theoretician's approach, using formalisms and abstractions to obtain principles and derive new measures and equations. Given my own background, most of the tools will draw heavily from control theory and systems engineering, complemented with some notions from statistical physics to deal with the large number of neurons that we will be using. That being said, those tools are sometimes obscure, so we will use geometric arguments as much as possible to make the results more intuitive.

An overview of neural computation

3 Brief history

The neuron doctrine – the concept that the nervous system is composed of individual neurons– was put forward by Ramón y Cajal in 1888 [LMBA06]. This led to the development of modern neuroscience where neurons would be characterized and modeled using ideas from electromagnetism [Lap07a]. Understanding the behaviour of neurons would eventually led to the question of how can network of neurons serve as a substrate for the mind able to reason and learn.

The first theory of how neurons compute was proposed by Warren McCulloch (a neuroscientist) and Walter Pitts (a logician) in 1943, where a neuron was proposed to compute by adding its inputs and performing a step function on that addition [MP43]. This theory would inspire the field of cybernetics, which gained traction after the book of Norbert Wiener [Wie48] in which self-regulating mechanisms are used as a unifying concept to a variety of fields, including neuroscience and computer science. On that same decade, Donald Hebb proposed that neurons could learn by reinforcing synapses that generate a spike, effectively presenting the first biologically plausible learning rule [Heb05].

Soon after the famous study from Hodgkin and Huxley [HH53]showed that neurons could be described by a small set of mathematical equations, which paved the way for first simulation of a neural network by Nathaniel Rochester [RHHD56]. The computerization of neurons eventually led to the first workshop on artificial intelligence [MMRS06] and later to the development of the perceptron by Frank Rosenblatt, an abstract network of neurons that could learn to classify patterns [Ros58].

At this point, however, the unfulfilled promises of artificial intelligence [Ola96] and the book of Minsky and Papert [MS69] clarifying the limitations of a single layer perceptron, effectively halting research on artificial intelligence in general in the so-called ”winter of AI”.

Naturally, the halt in AI did not have an effect in the study of neuroscience from a computational perspective, the prime example being the works of Hubel and Wiesel which showed how neurons in the visual cortex computed edge and

movement detection [HW62]. Similarly, some of

Research on computation with neural networks resurfaced on the eighties, when John Hopfield improved popularized a neural network proposed by Little [Lit74] which showed associative memory using Hebb's learning rule [Hop82]. On the same year, the publication of David Marr's book building analogies between neuroscience and computer science based on its levels of analysis brought computational approach to the spotlight. By the end of that decade, Computational Neuroscience had its first conference [Sch93].

During the following decade, theoretical and computational neuroscientists started exploring the synaptic mechanisms of learning [GKvHW96, BP98], machine learning researchers developed efficient methods for training neural networks [HS97, LBB+98]. However, the approaches where very different and the two fields diverged.

For most of the twentieth century, mainstream neuroscience research removed around targeted experiments trying to test a specific hypothesis by direct observation. This changed as computers became more powerful, widespread and data could be easily digitalized and shared. Thus computer based techniques became the stepping stone of some of the flagship projects in neuroscience such as the Blue Brain Project [Mar06], which provided the first large-scale simulation of parts of the brain, the Human Connectome Project [SL15] which creates a large atlas of connections between neurons. This computer-based approach also permeated into conceptual works trying to understand the principles of neural computation, with two prime examples being the Neural Engineering Framework [EA04] and Liquid State Machines [MNM02].

4 Current Trends and Relevant Approaches

Currently, the undeniable feats of machine learning have reignited the trend of comparing trained artificial neural networks (ANN) to the biological ones, providing examples of successful cross-pollination between the two fields [GBFK19].

The prime example is found on the visual cortex, where the best machine learning models for image understanding [HZRS15] have features such as hierarchical multi-level organization reminding of the ventral visual system [Li14]. Furthermore, their activations are also system, with Gabor-like filters appearing [GvG15] similarly to the edge detection neurons in the V1 region of the visual cortex [HW62], a similarity that also goes to V4 [YD16] and IT [KRK14], and even to the behaviour level in terms of errors made [KGGM16].

Beyond the visual system, tools from dynamical systems, have been leveraged to make by a direct comparison between recorded biological neural activity and trained recurrent neural networks, illustrating that the dynamics – if not the

training – do indeed coincide for the motor cortex [SCKS15]. Another example of those similarities is on navigation tasks, where the Neural Engineering Framework was used to create a model of navigation that already hinted to the existence of grid cells before their discovery [CE05], and more modern trained networks do show the equivalent of place and grid cells [KF17].

On an even more abstract level, the theory of cognitive spaces [BGMD18] proposes that human reasoning works by using compact spaces where the dimensions are given by multiple features to make inferences or classify unknown features. Although rarely implemented with a neural network substrate, this idea is well known in the machine learning community under the name of support vector machines [SV99].

Besides computer-based approaches, the development of very large-scale integration chip design has been used to emulate biological neural systems with electrical components [Mea89], and this would eventually lead to the development of neuromorphic electronics, which try to imitate neural architectures and functions [NVS16] with hardware implementations, on the hopes of overcoming the raising costs of training and running artificial neural networks [SGM19, AH18], and provide cheap neuroscience simulators [DSG+13].

Finally, the approach that will be most relevant for this thesis is the use of large recurrent networks of neurons as high-dimensional projections of inputs has been proposed in both fields under the name of reservoir computing [JH04, MNM02], with local unsupervised learning rules inspired by the brain have been found to bring healthy dynamics to the recurrent neural network [SWV+08]. On the opposite direction, there is also a relevant line of research in which synaptic learning rules are studied by the effects that they have on large recurrent neural networks with biologically inspired features, and how those rules are interpretable in terms of machine learning [LPT09].

5 Limitations, Caveats and Controversies

One of the main issues with the use of artificial neural networks as models for the brain – and vice versa – is that the level of abstraction at which we are working is not always clear. Indeed, there are many differences between artificial and natural neural networks at the implementation level. The most clear one is the use of temporal codes in some systems [Tho90], which do not have an analogy in computer-run systems. Even if we assume rates and abstract temporal codes, artificial systems have simple neurons that are replicated exactly across multiple layers and whose dynamics depend only on its inputs and outputs, while biological systems combine heterogeneous mechanisms working at multiple spatial and temporal layers, from short time plasticity to synaptic adaptation or neuromodula-

tors. The differences also extend to training mechanisms, which are almost always supervised for artificial neural networks [Bis95], but are typically assumed to be unsupervised for most of the nervous system [HSP99] or based on reinforcement learning [Doy00b] or a combination of both [Doy00a]. Here we must mention that on an abstract level the boundaries between unsupervised and supervised learning can be fairly thin [IGB19, DS12a, Ger18].

At the functional level, there are also undeniable differences between both systems. Artificial neural networks can outperform humans at complex board games [SHM+16] using strategies that are not similar to those of humans [LWY+16]. On the other side, humans are able to do tasks that are impossible for artificial networks [LUTG17], and even when the later do equal or surpass humans, they typically need much more data [XT07] and can be subject to adversarial examples [NYC15], which humans are not.

Part II

Tailoring Artificial Recurrent Neural Networks

The Echo State Network Framework

Reservoir computing is a computational paradigm that uses the dynamics of an input-driven system to perform computations. The idea emerged from recurrent neural networks research, both in machine learning [JH04] and biological models [MNM02], and eventually derived into a wide range of physical and virtual systems [TYH+19, Ver09]. The key feature of this approach is the reservoir, a high-dimensional non-linear dynamical system which projects the input into a high-dimensional space where linear regression or classification can be applied.

In the first part of this thesis we will focus on ESN, a particular variant of RC designed for machine learning tasks involving time series processing [JH04] where the reservoir is a large network of neurons. Owing to its simplicity, flexibility and empirical success, ESN and its variants have attracted intense interest during the last decade [Jae01b, Jae02], and have been applied to many different tasks such as electric load forecasting [DS12b], robotic control [PAGH03], epilepsy forecasting [BSVS08], stock price prediction [LYS09], grammar processing [TBCC07], and many others [VLS+10, NS12, Cou10, PHG+18].

Over the last decade, a plethora of studies have focused on finding good reservoir networks. Those studies fall broadly into two categories. First, for specific tasks, systematical parameter searches provide some improvement over classical Monte Carlo reservoir selection [FL11, JBS08, DZ06, Lie04, RIA17], but remain costly and do not offer a significant performance improvement or better understanding. Second, some authors have explored networks with some particular characteristics that make them desirable, typically with long memory [FBG16, RT12, SWL12] or "rich" dynamics [OXP07, BOL+12], although the desirability of those traits is typically task-specific. Here we focus on a "mechanistic" understanding of reservoir dynamics, but instead of trying to find reservoirs with predefined features, we design reservoirs whose dynamics are tailored to specific problems.

6 The ESN Framework

The basic ESN architecture is depicted in Fig. 1. With different coefficients (weights), the input signal and the predicted output from the previous time step are sent to all neurons in the reservoir. The output is calculated as a linear combination of the neuron states and the input. At each time step, each neuron updates its state according to the current input it receives, the output prediction and its neighboring neurons' states from the previous time step. Formally, the discrete-time dynamics of an ESN with N neurons, one input and one output is governed by

$$\mathbf{x}(t) = f(\mathbf{W}\mathbf{x}(t-1) + \mathbf{w}_{\text{in}}u(t) + \mathbf{w}_{\text{ofb}}y(t-1)), \quad (1)$$

$$y(t) = \mathbf{w}_{\text{out}} \begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix}, \quad (2)$$

where $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^\top \in \mathbb{R}^N$ denotes the state of the N neurons at time t, $u(t) \in \mathbb{R}$ is the input signal, the vector $\begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix} \in \mathbb{R}^{N+1}$ represents the concatenation of $\mathbf{x}(t)$ and $u(t)$, and $y(t) \in \mathbb{R}$ is the output at time t. There are various possibilities for the nonlinear function f, the most common ones being the logistic sigmoid and the hyperbolic tangent [Bis06]. Without loss of generality we choose the latter in this work. The matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix of the reservoir network describing the fixed wiring diagram of N neurons in the reservoir. There is a rich literature on the conditions that the matrix $\mathbf{W}$ must fulfill [Jae07, YJK12, BY06, GTJ12]. Here we adopt a conservative and simple condition that the reservoir must be a stable dynamic system. The vector $\mathbf{w}_{\text{in}} \in \mathbb{R}^N$ captures the fixed weights of the input connections, which we draw from a uniform distribution in the interval $[-1, 1]$. The vector $\mathbf{w}_{\text{ofb}} \in \mathbb{R}^N$ denotes the fixed weights of the feedback connections from the output

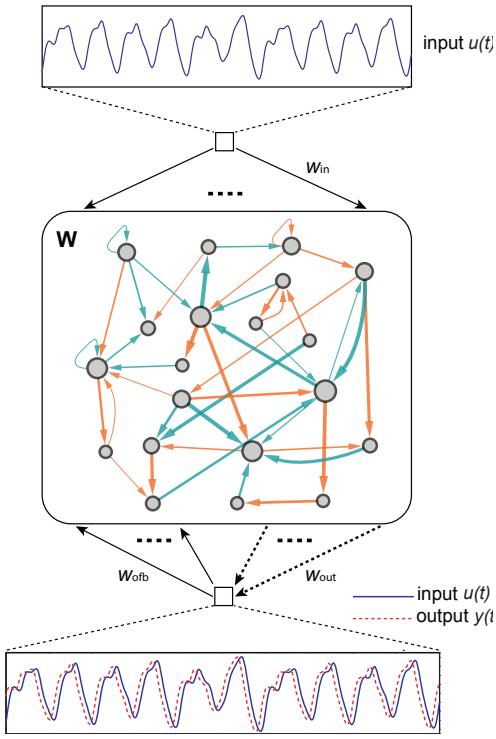


Figure 1: **The basic schema of an ESN.** The input signal $u(t)$ goes to each neuron in the reservoir with input weights w_{in}, the neurons send their states to their neighbors according to the matrix $\mathbf{W}$, and the contribution of each neuron to the output $y(t)$ is collected by w_{out}. The reservoir network may have self-loops, and can have both excitatory (yellow) and inhibitory (gray) synaptic connections.

to the N neurons, which can induce instabilities if chosen carelessly and may be zero in some tasks [Jae02]. Finally, the row vector $\mathbf{w}_{\text{out}} \in \mathbb{R}^{1\times(N+1)}$ represents the trainable weights of the readout connections from the N neurons and the input to the output.

A key feature of ESN is that $\mathbf{W}$, $\mathbf{w}_{\text{in}}$ and $\mathbf{w}_{\text{ofb}}$ are all predetermined before the training process, and only the weights of the readout connections $\mathbf{w}_{\text{out}}$ are modified to $\mathbf{w}^*_{\text{out}}$ during the training process:

$$\mathbf{w}^*_{\text{out}} = \arg\min_{\mathbf{w}_{\text{out}}} \sum_{t=t_0}^{t_0+T} (y(t) - \hat{y}(t))^2, \tag{3}$$

where t_0 is the starting time, T is the interval of the training, and $\hat{y}(t)$ is the target output obtained from the training data. In other words, $\mathbf{w}^*_{\text{out}}$ is the linear regression weights of the desired output $\hat{y}(t)$ on the extended state vector $\begin{pmatrix}\mathbf{x}(t)\\ u(t)\end{pmatrix}$, which can be easily solved (see SI Sec. I. for details). Hence, $\mathbf{w}^*_{\text{out}}$ captures the underlying mechanism of the dynamic system that produces the training data. Indeed, the right choice of $\mathbf{w}^*_{\text{out}}$ can be used to forecast, reconstruct or filter nonlinear time series.

Note that there is a rich literature on methods to improve the ESN performance such as using regularization in the computation of $\mathbf{w}^*_{\text{out}}$ [Jae02], controlling the input weights [SWL12] or changing the dynamics of the neurons [LJ09]. Those results, while relevant and important for applications, are tangential to our study. Therefore in this work we will use the simplest version of the ESN as presented above.

7 Performance Measurement

In the literature of ESN it is common to forecast time series [JH04]. To be consistent with the previous literature we use the Normalized Root Mean Squared Error (NRMSE), as a metric of forecasting error

$$\sigma = \sqrt{\frac{\sum_{t=t_0}^{t_0+T}(y(t) - \hat{y}(t))^2}{T \cdot \text{var}(u(t))}}. \tag{4}$$

This metric is a normalization of the classical root mean squared error. The normalization is necessary in this case to avoid having different values if the signal is multiplied by a scalar. This is particularly important for ESN because one of the parameters that is usually tunned [Jae02] is the scaling of the input vector $\mathbf{w}_{\text{in}}$ or the input signal $u(t)$. The parameter t_0 is used to describe when we start to count

the performance, since it is also common to ignore the inputs during the initialization phase [Jae02], which is taken here as the full initialization steps given for each task (see details in the subsequent sections). The parameter T is simply the number of time-steps considered, which we take here as the full count of all points except the initialization phase in each testing time series.

The NRMSE as presented here is obviously not a good metric for classification tasks where the target variable is discrete. In order to have a comparable metric for ESN performance, we use the failure rate in classification tasks such as the Spoken Arabic Digit Recognition, which is similar to a Hamming distance. Note that having 10 digits implies that the failure rate with random guesses is 0.9, therefore a failure rate of 0.3 is well below it.

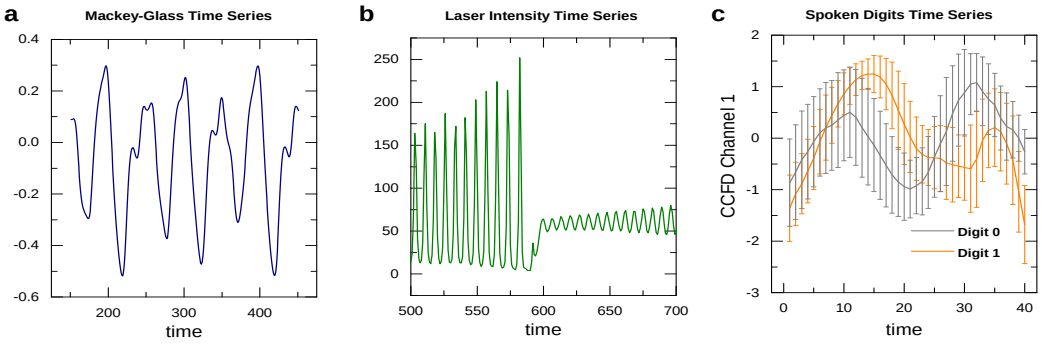


Figure 2: **Plots of the time series for each task**: The Mackey-Glass time series with 500 points (left), the Laser Intensity time series with 300 points (center) and the average value of the first mel-frequency cepstral coefficient (MFCC) Channel of the first Spoken Arabic Digit, with the error bars represent standard deviations over the training dataset (right).

7.1 Forecasting Mackey-Glass time series

Forecasting Mackey-Glass time series is a benchmark task to test the performance of ESN [JH04]. The Mackey-Glass time series follows the ordinary differential equation [JH04]:

$$\frac{ds(t)}{dt} = \beta \frac{s(t-\tau)}{1+s(t-\tau)^n} - \gamma s(t),$$

where β, γ, τ, n are real positive numbers. We used the parameters $\beta = 0.2$, $\gamma = 0.1$, $\tau = 17$, $n = 10$ in our simulations. The discrete version of the equation uses a time step of length $h = 0.1$. For each time series we generated $\frac{\tau}{h} = 170$ uniformly distributed random values between 1.1 and 1.3 and then followed the equations. The first 1000 points were considered as initialization steps, which did

not fully capture the time series dynamics and were thus discarded. For training and testing we used time series of 10,000 points, but in both cases the first 1000 states of the reservoir were considered as initialization steps and were thus ignored for training and testing. For an ESN with 1000 neurons and an optimized memory, the forecasting performance for this setting is close to its maximum value, thus the addition of short cycles will have a small effect. In order to show the interest of our contribution, we normalized the signal to have mean zero and variance of one and we added Gaussian white noise with $\sigma = 0.05$, and the forecasting was done using reservoirs of 100 neurons, average degree $\langle k \rangle = 10$ and spectral radius of $\alpha = 0.85$, and the output was feed back to the reservoir through the vector $\mathbf{w}_{\text{ofb}}$ where every entry is independently drawn from a uniform distribution on the interval $[-1, 1]$. The ESN was trained to forecast one time-step, and then we used this readout to forecast 84 time-steps in the future by recursively feeding the one-step prediction of $s(t+1)$ into the ESN as the new input. Although it is possible to directly predict $s(t+k)$ from the state of the reservoir at time t, empirically it has been observed the performance improves if the readout is trained to perform a one-step prediction and then feeding back the result [Jae02, JH04].

7.2 Forecasting Laser Intensity time series

The Laser Intensity time series [HAW89, HKAW89] was obtained from the Santa Fe Institute time series Forecasting Competition Data. It consists of 10,093 points, which we normalized to have an average of zero and an standard deviation of one, and were filtered with a Gaussian filter of length three and standard deviation of one. The forecasting was done using reservoirs of 100 neurons, average degree $\langle k \rangle = 10$ and spectral radius of $\alpha = 0.9$, without feedback so $\mathbf{w}_{\text{ofb}} = 0$. Here we forecasted one time-step. We used 1,000 points of the time series for initialization, 4,547 for training and 4,546 for testing.

7.3 Spoken Arabic Digit Recognition

The Spoken Arabic Digits [HB10] dataset was downloaded from the [HB10] from the UCI Machine Learning Repository [Lic13]. This dataset consists of 660 recordings (330 from men and 330 from women) for each of the ten digits and 110 recordings for testing. Each recording is a time series of varying length encoded with MCCF [Mer76] with 13 channels. While using the first three channels gave a better performance, here we use only the first channel, which is akin to a very lossy compression. We normalized this time series to have average of zero and a standard deviation of one, and a length of 40. Since in most cases we had less than 40 points, we computed the missing values by interpolation. The classification procedure was done using the forecasting framework. We collected the

reservoir states from all the training examples of each digit and computed $\mathbf{w}_{\text{out}}$ as did in the previous forecasting tasks. In the testing we collected the states and computed the forecasting performance σ for each of the 20 cases. We classified the time series as the digit that yielded the lowest forecasting error. Then we calculate the failure rate as the number of misclassified recordings divided by the total number of recordings in the training set. We used reservoirs of 100 neurons, average degree $\langle k \rangle = 10$ and spectral radius of $\alpha = 1$, without feedback ($\mathbf{w}_{\text{ofb}} = 0$). Note that in our simulations we find that for this particular task the ESN performance is not drastically affected by the output feedback.

Memory in ESN: From Reservoir Dynamics to Structure

The success of ESN in tasks such as forecasting time-series comes from the ability of its reservoir in retaining memory of previous inputs [CLL12]. Therefore, one of the first problems to address consists on relating the structure and dynamics of the reservoir with its memory.

8 Measuring Memory Capacity

In RC literature, the main metric for memory is the memory capacity defined as follows [Jae01a]:

$$M = \sum_{\tau=1}^{\tau_{\max}} M_\tau, \tag{5}$$

with

$$M_\tau = \max_{\mathbf{w}_{\text{out}}^\tau} \frac{\text{cov}^2(r(t-\tau), \mathbf{w}_{\text{out}}^\tau \mathbf{x}(t))}{\text{var}(r(t-\tau))\text{var}(\mathbf{w}_{\text{out}}^\tau \mathbf{x}(t))}. \tag{6}$$

Here $r(t)$ is a random variable drawn from a standard normal distribution $\mathcal{N}(0,1)$, serving as a random input, 'cov' represents the covariance. Note that $\mathbf{w}_{\text{out}}^\tau$ is equivalently obtained as a minimizer of the difference between $y_\tau(t)$ and $r(t-\tau)$ for any delay $\tau \in [1, ..., \tau_{\max}]$, with $\tau_{\max}$ chosen so that $M_{\tau_{\max}} \approx 0$.

To get a first impression of how M depends on the network structure of the reservoir we will measure M_τ for different network architectures:

- Erdös-Rényi (ER) random graphs where the degree distribution follows a Poisson distribution and the weights are drawn from a normal distribution and varying spectral radii of $\mathbf{W}$.
- Erdös-Rényi random graphs with weights drawn from a power law distribution (PL) with exponenet $\beta \in [2, 5]$ but normalized to have a spectral radius $\alpha = 1$.

- Scale-Free (SF) networks, whose degree distribution follows a power law. The degree heterogeneity is given by the degree exponent $\gamma \in [2, 6]$, and the weights are drawn from a Gaussian distribution, also normalized to have a spectral radius $\alpha = 1$.
- Random Regular (RR), or graphs where every node has the same degree $\langle k \rangle$ and a spectral radius $\alpha = 1$.

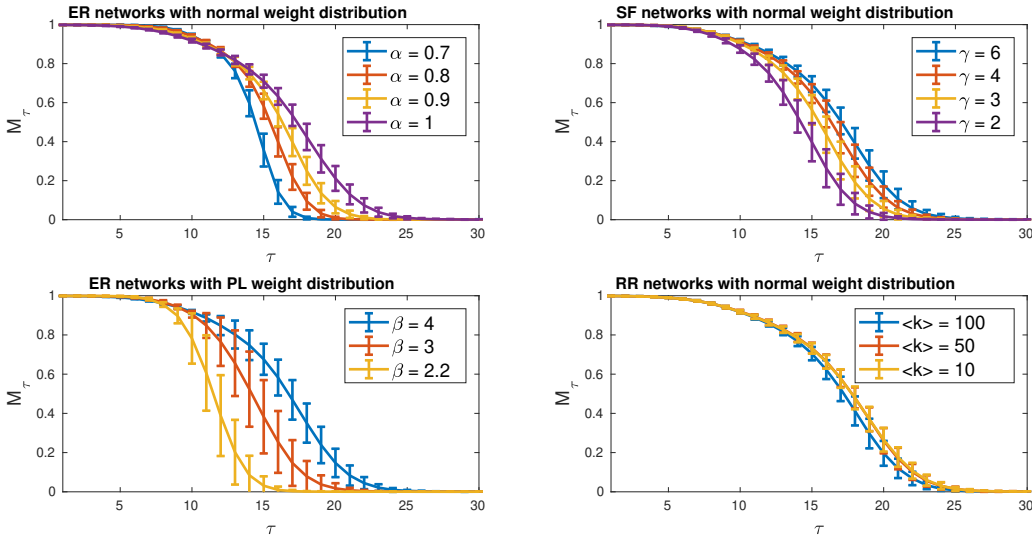


Figure 3: **Memory Capacity decay for various network architectures**. The ability of the reservoir to retrieve previous inputs decays as with τ, the delay with which we try to retrieve them. Each network has 400 neurons and the average degree is $\langle k \rangle = 20$ and is fed with 4,000 inputs randomly sampled from a normal distribution. Each curve shows the average result over 100 trials and the error bars are the standard deviation. The spectral radius $\alpha = 1$, except in ER networks where it varies as marked in the legend. The plots show M_τ, while the total memory capacity M is simply the area under the curve.

The results from Fig. 3 show that the parametrization of the memory is not entirely intuitive. On one side, the effects of the spectral radius are well understood [Jae01a]: the larger the spectral radius, the slower do perturbations decay, and thus the longer an input is stored in the reservoir state. However, other parameters such as degree heterogeneity or the tail strength of the weight distribution also affect the memory capacity, while some network parameters that are typically important – such as degree – do not always have an appreciable effect.

The first intuition that we can obtain from this is rather simple: when the state of every neuron depends on many diverse variables, the memory is high, while

when each neuron depends on only few values the memory is low. The strength of the spectral radius modulates the relative effects of the last input $u(t)$ versus the previous state of the reservoir, such that when the spectral radius is high, $\mathbf{x}(t)$ depends highly on the previous reservoir state – which is a high-dimensional variable –, while a low spectral radius implies that most of the variance in $\mathbf{x}(t)$ is imposed by the low-dimensional input $u(t)$. Conversely, the degree heterogeneity controls the ratio of shared inputs between neurons, as a high heterogeneity implies that most neurons will get inputs from a very small set of highly connected neurons, thus there are few variables that drive $\mathbf{x}(\mathbf{t})$. Similarly, heavy tails on the weight distribution imply that most neurons have one or two neighboring neurons that largely control their state, as opposed to having a multitude of relevant neighbors. Finally, the effect – or lack thereof – of the degree distribution seems to contradict our intuition, unless we consider that $\langle k \rangle = 10$ already implies that on average every neuron is subject to 11 inputs, one for every neighboring neuron and one given by the systems' input $u(t)$.

9 Memory and Dynamics

9.1 A formal link between memory and correlations

To explain in more detail this relationship we will use an upper bound which connects the variance of the reservoir across different directions with the memory capacity. Intuitively, the state of the reservoir encodes previous inputs that must be extracted linearly, hence more linearly independent neurons will be able to encode more information about the past.

Before starting we need to note a couple of properties in our reservoirs, namely that they are contracting. This implies that if we take two input sequences $\mathbf{r^1}$, $\mathbf{r^2}$ differing only at a given time t, the difference in the reservoir state space follows

$$\|\mathbf{x}^1(t) - \mathbf{x}^2(t)\| > \|\mathbf{x}^1(t+1) - \mathbf{x}^2(t+1)\| \quad \forall t. \tag{7}$$

Similarly, reservoirs always have the fading memory property, which implies that in the same context,

$$\lim_{i \to \infty} \|\mathbf{x}^1(t+i) - \mathbf{x}^2(t+i)\| = 0, \tag{8}$$

which can be explained informally by saying that the reservoir eventually forgets its inputs. This explains the decay in M_τ and it is a necessary property of all reservoirs [Maa11, Jae01a, YJK12].

Setting the problem

We start by noticing that the linear nature of the projection vector $\mathbf{w}_{\text{out}}$ implies that we are treating the system as

$$\mathbf{x}(t) = \sum_{k=0}^{\infty} \mathbf{a}_k r(t-k) + \boldsymbol{\varepsilon}_r(t) \tag{9}$$

where the vectors $\mathbf{a}_k \in \mathbb{R}^N$ are correspond to the linearly extractable effect of $r(t-k)$ onto $\mathbf{x}(t)$ and $\boldsymbol{\varepsilon}_r(t)$ is the nonlinear contribution of all the inputs onto the state of $\mathbf{x}(t)$.

Notice that there are two perspectives here: on one side, the readout extracts the best linear approximation of past inputs with a noise-like term, and on the other it can be interpreted as a Taylor expansion around an undefined point where the first order corresponds to the first term and the non-linear behavior to the other expansion terms. This separation between linear and non-linear behavior has been thoroughly studied [DVSM12, GHS08] and the general understanding is that linear reservoirs have longer memory, but nonlinearity is needed to perform interesting computations. Here we will not try to leverage or bypass this trade-off, but rather we will show that for a fixed ratio of the non-linearity, the more decorrelated the neurons are the higher the memory.

To maintain this trade-off between linear and non-linear behavior, we will assume that the distribution of the linear and non-linear strengths are fixed. This can be achieved if we impose the probabilities of the neuron states do not change, meaning that the mean, variance and other moments of the neuron outputs are unchanged and hence the strength of the nonlinear effects is unchanged.

A first constraint can also be obtained from the maintained strength of the linear side of Eq. 9

$$\text{var}\left(\sum_{\tau=1}^{\infty} \mathbf{a}_\tau r(t-\tau)\right) = c \tag{10}$$

where c is a constant.

If we are allowed to shift the linear side of Eq. 9, the natural choice of $\mathbf{a}_\tau$ to maximize the memory capacity would be to impose $a_\tau > 0 \iff \tau > N$, and also make the vectors $\mathbf{a}_\tau$ orthogonal to each other, so that the input at time $t - \tau_1$ does not interfere with the effect of an input at time $t - \tau_2$. This can be introduced into Eq. 10 and we obtain

$$\text{var}\left(\sum_{\tau=1}^{\infty} \mathbf{a}_\tau r(t-\tau)\right) = \sum_{\tau=1}^{N} \text{var}(r(t-\tau))\|\mathbf{a}_\tau\|^2 = \sum_{\tau=1}^{N} \|\mathbf{a}_\tau\|^2 = c \tag{11}$$

This leaves us with a straightforward choice for the readout vector, namely

$$\mathbf{w}^\tau = \mathbf{a}_\tau, \tag{12}$$

which we can plug into the memory capacity to obtain

$$M_\tau^* = \frac{\text{cov}^2(r(t-\tau), \mathbf{a}_\tau \mathbf{x}(t))}{\text{var}(\mathbf{a}_\tau \mathbf{x}(t))} = \frac{\text{cov}^2(r(t-\tau), \|\mathbf{a}_\tau\|^2 r(t-\tau) + \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle)}{\text{var}(\|\mathbf{a}_\tau\|^2 r(t) + \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t))}$$
$$= \frac{\|\mathbf{a}_\tau\|^4 \left[\text{cov}(r(t-\tau), r(t-\tau)) + \text{cov}(r(t-\tau), \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle)\right]^2}{\|\mathbf{a}_\tau\|^2 \text{var}(r(t-\tau)) + \text{var}(\langle \mathbf{a}_\tau r(t), \boldsymbol{\varepsilon}_r(t)\rangle) + \|\mathbf{a}_\tau\|^2 \text{cov}(r(t-\tau), \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle)} \tag{13}$$

where M_τ^* is the maximum memory that can be achieved by shifting the linear side of Eq. 9 but maintaining the linear-nonlinear ratio of variance. We must recall the definition of $\boldsymbol{\varepsilon}_r(t)$ in Eq. 9, which implies that any correlation between its projection on $\mathbf{a}_\tau$ and $r(t)$ would imply that part of $r(t-\tau)$ is projected linearly onto the non-linear component. This necessarily means that

$$\text{cov}(\langle r(t-\tau), \langle \mathbf{a}_\tau, \boldsymbol{\varepsilon}_r(t)\rangle) = 0, \tag{14}$$

hence our previous equation becomes

$$M_\tau^* = \frac{\|\mathbf{a}_\tau\|^2}{\|\mathbf{a}_\tau\|^2 + \text{var}\left(\langle \frac{\mathbf{a}_\tau}{\|\mathbf{a}_\tau\|}, \boldsymbol{\varepsilon}_r(t)\rangle\right)}, \tag{15}$$

and for the sake of simplicity we will name $\|\mathbf{a}_\tau\|^2 = a_\tau$, and $\text{var}\left(\langle \frac{\mathbf{a}_\tau}{\|\mathbf{a}_\tau\|}, \boldsymbol{\varepsilon}_r(t)\rangle\right) = \varepsilon_\tau$, leaving us with

$$M_\tau^* = \frac{a_\tau}{a_\tau + \varepsilon_\tau}, \tag{16}$$

where a_τ is the squared modulus of the linear projection of the input $r(t-\tau)$ on the reservoir state and ε_τ is the variance in that direction induced by the non-linear terms in that same direction.

Hence the new problem that we must solve is to maximize

$$\sum_{\tau=1}^{N} M_\tau^* = \sum_{\tau=1}^{N} \frac{a_\tau}{a_\tau + \varepsilon_\tau} \tag{17}$$

subject to the constraint

$$\sum_{\tau=1}^{N} a_\tau = c, \tag{18}$$

with an order on the coefficients arising from the contractiveness or the reservoir

$$a_{\tau+1} < a_\tau \tag{19}$$

and under the assumption that we do not change the nonlinear effects on any direction, hence ε_τ will be fixed.

Finally, we shall also note that we will assume that M_τ^* is always a monotonically decreasing variable that goes from $M_1^* \approx 1$ to $M_N^* \approx 0$, as we observed in the plots on Fig. 3. By noting that

$$M_\tau^* = \frac{1}{1 + \frac{\varepsilon_\tau}{a_\tau}}, \tag{20}$$

this assumption implies that $q_\tau = \frac{\varepsilon_\tau}{a_\tau}$ goes from $r_1 \approx 0$ to $r_N \to \infty$, hence ε_τ starts being much smaller than a_τ and decreases much slower than a_τ.

Optimizing the linear projections

Now the problem is to find how a change in the distribution of $[a_1, a_2, ..., a_N]$ would affect the value of M^*. Our approach will be to show that the fastest a_τ decreases, the lower M^*.

We will show that there is one case, namely $a_\tau \propto \varepsilon_\tau$ which has higher M^* than an other setting where a_τ decreases faster. Since we know that the values of a_τ must decrease faster than ε_τ, our best arrangement of the strength of the linear projection of $r(t)$ is to try to make a_τ decrease as slowly as possible.

With a fixed ratio $a_\tau = \chi\varepsilon_\tau$, the upper bound on the memory is

$$M^* = \sum_{\tau=1}^{N} \frac{a_\tau}{a_\tau + \varepsilon_\tau} = \sum_{\tau=1}^{N} \frac{1}{1 + \frac{\varepsilon_\tau}{a_\tau}} = N\frac{1}{1+\chi}. \tag{21}$$

Now we will split the sequence of M_τ^* into two subsequences, one with $\tau < k$ where the values of a_τ will be increased by a factor β_+ and another one with $\tau > k+1$ where the values will be decreased by β_-. This leads us to the new memory capacity bound,

$$M^* = \sum_{\tau=1}^{k} \frac{\beta_+}{\beta_+ + \chi} + \sum_{\tau=k+1}^{N} \frac{\beta_-}{\beta_- + \chi} = k\frac{\beta_+}{\beta_+ + \chi} + (N-k)\sum_{\tau=k+1}^{N} \frac{\beta_-}{\beta_- + \chi} \tag{22}$$

which for simplicity we will normalize to obtain

$$\frac{M^*}{N} = m^* = \frac{k}{N}\frac{\beta_+}{\beta_+ + \chi} + \frac{N-k}{N}\frac{\beta_-}{\beta_- + \chi} = \phi\frac{\beta_+}{\beta_+ + \chi} + (1-\phi)\frac{\beta_-}{\beta_- + \chi}. \tag{23}$$

where $\phi = \frac{k}{N}$, and m^* is just a normalized memory capacity bound.

Note that the function m^* is still subject to the constraints presented in Eq. 18 and Eq. 19. This yields

$$\beta_+ \sum_{\tau=1}^{k} a_\tau + \beta_- \sum_{\tau=k+1}^{N} a_\tau = c, \tag{24}$$

by introducing the variable

$$\gamma = \frac{\sum_{\tau=1}^{k} a_\tau}{c\sum_{\tau=1}^{N} a_\tau}, \tag{25}$$

where $\gamma < 1$, and

$$1 - \gamma = \frac{\sum_{\tau=k+1}^{N} a_\tau}{c\sum_{\tau=1}^{N} a_\tau}. \tag{26}$$

Now we can rewrite Eq. 18 to obtain the new constraint

$$\beta_+\gamma + \beta_-(1-\gamma) = 1. \tag{27}$$

Now we can introduce another variable $q = \frac{\beta_+}{\beta_-}$ which determines the difference between β_+ and β_-. As our main point is to show that a faster decrease of a_τ would decrease m^*, which in this particular case translates to

$$\frac{\partial m^*}{\partial q} < 0, \tag{28}$$

which would imply that β_+ is larger than β_- and hence a_τ decreases faster than ε_τ at $\tau = k$ and equally at any other τ.

We can now compute the derivative

$$\begin{aligned}\frac{\partial m^*}{\partial q} &= \phi\frac{\partial}{\partial\beta_+}\left(\frac{\beta_+}{\beta_+ + \chi}\right)\frac{\partial\beta_+}{\partial q} + (1-\phi)\frac{\partial}{\partial\beta_-}\left(\frac{\beta_-}{\beta_- + \chi}\right)\frac{\partial\beta_-}{\partial q} \\ &= \phi\frac{\chi}{(\beta_+ + \chi)^2}\frac{\partial\beta_+}{\partial q} + (1-\phi)\frac{\chi}{(\beta_- + \chi)^2}\frac{\partial\beta_-}{\partial q}\end{aligned} \tag{29}$$

where the first term is positive and the second is negative – because the memory capacity for $\tau < k$ will increase when β_+ grows and vice-versa. Thus our main goal is to prove that

$$\phi\frac{\chi}{(\beta_+ + \chi)^2}\frac{\partial\beta_+}{\partial q} < -(1-\phi)\frac{\chi}{(\beta_- + \chi)^2}\frac{\partial\beta_-}{\partial q}. \tag{30}$$

Since $\beta_+ > 1 > \beta_-$,

$$\frac{\chi}{(\beta_+ + \chi)^2} < \frac{\chi}{(\beta_- + \chi)^2}, \tag{31}$$

hence we only need to prove that

$$\phi\frac{\partial\beta_+}{\partial q} < -(1-\phi)\frac{\partial\beta_-}{\partial q}. \tag{32}$$

To do so we will use the constraint from Eq. 27. By setting $\beta_+ = \beta_- q$ we can solve both equations and obtain

$$\begin{aligned}\beta_+ &= \frac{1}{\gamma}\frac{q}{q+\frac{1-\gamma}{\gamma}} \\ \beta_- &= \frac{1}{\gamma}\frac{1}{q+\frac{1-\gamma}{\gamma}}.\end{aligned} \tag{33}$$

Therefore,

$$\begin{aligned}\frac{\partial\beta_+}{\partial q} &= \frac{\frac{1-\gamma}{\gamma}}{(q+\frac{1-\gamma}{\gamma})^2} \\ \frac{\partial\beta_-}{\partial q} &= -\frac{1}{(q+\frac{1-\gamma}{\gamma})^2},\end{aligned} \tag{34}$$

which we can plug directly into Eq. 32, which simplifies to

$$\begin{aligned}&\frac{\phi}{\gamma}\frac{\frac{1-\gamma}{\gamma}}{(q+\frac{1-\gamma}{\gamma})^2} < \frac{1-\phi}{\gamma_+}\frac{1}{(q+\frac{1-\gamma}{\gamma})^2} \\ &\iff \phi(1-\gamma) < (1-\phi)\gamma \\ &\iff \phi < \gamma,\end{aligned} \tag{35}$$

which is true by the definitions of γ and ϕ, and the fact that a_τ is decreasing.

Now we have proven that whenever we can take an index k and increase a_τ for $\tau > k$ and decrease a_τ for $\tau < k$, the memory capacity bound M^* decreases. We shall now use this result to prove that whenever

$$\frac{\epsilon_{\tau+i}}{\epsilon_\tau} < \frac{a_{\tau+i}}{a_\tau} \quad \forall i, \tau > 0, \tag{36}$$

the memory capacity bound M^* is lower than when $\varepsilon_\tau \propto a_\tau$.

We do so by starting with the case $\varepsilon_\tau \propto a_\tau$ and we will change it step by step to a series of a'_τ that fulfills Eq. 36. This is a simple iterative procedure. We start with $k = 1$, then select $q = \frac{\beta_+}{\beta_-}$ such that $a_k\beta_+ = a'_k$ under the constraint from Eq. 27. Then we fix a_1 and we have the same problem for a_τ starting at $\tau \geq 2$. This process can be repeated until $k = N$, at which point the memory bound m^* will be lower because every modification with index k lowered it.

Note that we have used a_τ in our argument, hence we obtained that the distribution of a_τ should be as homogeneous as possible. However, our result can also be stated for $a_\tau + \varepsilon_\tau$, because the non-linear effects of the reservoir on every direction are unchanged and the series ε_τ is also a decreasing one.

Correlations as constraints on the variance

From the previous section we know that the memory bound increases when the variance along the projections of the input into the reservoir state become more homogeneous. This can be expressed in terms of the state space of the reservoir. Intuitively, the variances at directions $\mathbf{a}_\tau$ must fit into the variances of the state space, and since we already established orthogonality of the projections, those variances must be along orthogonal directions. Since our goal is to have a variance as homogeneous as possible along the directions of $\mathbf{a}_\tau$, we need variance that is as homogeneous along orthogonal directions. Finally, this homogeneity is reduced when we add correlations between neurons.

We shall recall that we started our discussion by assuming that the probability distribution of neuron states is unchanged, which would ensure that the strength of the nonlinearity is not altered. This implies that the distribution of variances of the neurons is fixed. If we start by having zero correlations, we can start by setting

$$a_\tau + \varepsilon_\tau = \text{var}\left(x_{\text{sort}(\tau)}(t)\right) \tag{37}$$

where sort(τ) is the operation that finds the neuron with the τth largest variance in the reservoir. In other words, we associate every neuron to one direction of $\mathbf{a}_\tau$, with the constraint that the variances along those directions are ordered, hence we associate $\mathbf{a}_1$ to the neuron with highest variance, $\mathbf{a}_2$ to the second and so on.

The distribution of $a_\tau + \varepsilon_\tau$ in that particular case is then given by the distribution of var $(x_n(t))$. If the correlations are not zero, however, we need a new family of vectors which preserves orthogonality across the covariance matrix $\mathbf{C}$. This is given by the eigenvectors of $\mathbf{C}$, which implies that we are using Principal Component Analysis as presented in Appendix 3. In that framework, the new variances are given by the eigenvalues of the covariance matrix, $\lambda_n(\mathbf{C})$. Naturally, when the correlations are zero, the eigenvalues correspond to the entries of the diagonal, which in our case are the variances as in the previous case.

Hence we have to now work on the distribution of the eigenvalues of the covariance matrix. Specifically, we would want to show that increasing the correlations between neurons increases the inhomogeneity of the eigenvalues, which would decrease our memory bound M^*. A simple way to quantify this inhomogeneity is the mean with respect to the square root of the raw variance, which is given by

$$\nu = \frac{\sum_{n=1}^{N} \lambda_n^2(\mathbf{C})}{\left(\sum_{n=1}^{N} \lambda_n(\mathbf{C})\right)^2}, \tag{38}$$

where $\lambda_n(\mathbf{C})$ is the nth eigenvalue of $\mathbf{C}$. To get an intuition of how this metric reflects the inhomogeneity, consider the case of two eigenvalues $\lambda_1,\ \lambda_2$; when

$\lambda_1 = \lambda_2$ –very homogeneous – then $\nu = \frac{1}{2}$, but when $\lambda_1 > 0, \lambda_2 = 0$ – very inhomogeneous–, then $\nu = 1$. For $N \gg 1$ the perfectly homogeneous case approach zero but the perfectly inhomogeneous one is still one.

We can compute $\left(\sum_{n=1}^{N} \lambda_n(\mathbf{C})\right)^2$ by using th relationship between trace and eigenvalues [Str93],

$$\sum_{n=1}^{N} \lambda_n(\mathbf{C}) = \text{tr}\left[\mathbf{C}\right] = \sum_{n=1}^{n} \text{var}\left(x_n(t)\right) \tag{39}$$

which is constant by the assumption that the probability distributions of the neuron activities are fixed. Hence we can focus on the value of $\sum_{n=1}^{N} \lambda_n^2(\mathbf{C})$. This is easily done by noting that

$$\mathbf{C}^k v_n(\mathbf{C}) = \lambda_n(\mathbf{C})\mathbf{C}^{k-1} v_n(\mathbf{C}) = \lambda_n^k(\mathbf{C}) v_n(\mathbf{C}) \tag{40}$$

where $v_n(\mathbf{C})$ and $\lambda_n(\mathbf{C})$ are, respectively the nth eigenvector and eigenvalue of $\mathbf{C}$. If we plug this into the relationship between trace and eigenvalues we obtain

$$\sum_{n=1}^{N} \lambda_n^2(\mathbf{C}) = \text{tr}\left[\mathbf{C}^2\right], \tag{41}$$

which we can compute by decomposing the square of covariance matrix and obtain

$$\sum_{n=1}^{N} \lambda_n^2(\mathbf{C}) = \sum_{n=1}^{N}\sum_{m=1}^{N} \mathbf{C}_{nm}\mathbf{C}_{mn} = \sum_{n=1}^{N} \text{cov}\left(x_n(t), x_m(t)\right)^2. \tag{42}$$

Which obviously grows when the neurons become correlated. Hence, the inhomogeneity measured by ν grows.

Example

The bound developed in previous sections might seem a bit artificial and far from the standard practice of reservoir computing, particularly the notion that we can adapt the linear part of the dynamics as we want. To make it more understandable and to show that the bound is indeed sharp we will present a simple example where all our assumptions are easily verified and the bounds are sharp.

For this we consider a line of neurons with the input on the first one. That is,

$$\mathbf{W}_{ij} = \begin{cases} w & \iff j = i + 1 \\ 0 & \text{otherwise,} \end{cases} \tag{43}$$

with $w < 1$ to keep the contractivity and the input is only sent to the first neuron, meaning that $\mathbf{w}_{\text{in}} = [w_{\text{in}}, 0, 0, ...]$. If we let $w_{\text{in}} \ll 1$, then the network is effectively linear, because the hyperbolic tangent is almost an identity around 0. Then the reservoir state becomes

$$\mathbf{x}(t) \approx w_{\text{in}} \left[u(t), wu(t-1), ..., w^N u(t-N)\right] \tag{44}$$

which corresponds to the case where $\varepsilon_\tau \approx 0$, and the previous input can be easily recovered, and this gives us $M = M^* = N$, which is the memory maximum [DVSM12, Jae01a]. If we increase the value of w_{in}, the reservoir is described as

$$\mathbf{x}(t) = \left[\tanh\left(w_{\text{in}} u(t)\right), \tanh\left(w \tanh\left(w_{\text{in}} u(t-1)\right)\right), ...\right] \tag{45}$$

where a readout can be obtained for each delay but the nonlinearity of repeatedly applying the hyperbolic tangent makes the memory harder and harder to recover, so the factor $\frac{\varepsilon_\tau}{a_\tau}$ grows. Naturally, here $M = M^*$.

In either case, the covariance matrix is diagonal because the inputs $r(t),\ r(t-\tau)$ are uncorrelated. Notice that, if we add connections between neurons or if we feed inputs with some alternative $\mathbf{w}_{\text{in}}$, then we would be increasing the correlations because there would be more neurons feed by the same input. This can only decrease the memory, because a reservoir with a line of neurons achieves its maximum memory capacity [DVSM12, WLS04].

10 Correlations and network spectra

Having related the memory capacity of an ESN and the dynamics of its reservoir, the subsequent step is to relate the structure of the network to the aforementioned dynamics.

The intuition here follows from the notion that the spectral radius sets the time that a perturbation – the input – will affect the system [Jae01a]. While the spectral radius fails to account for the effects of degree heterogeneity or fat tails on the weight shown in Fig. 3, the notion that the eigenvalues of the adjacency matrix of the network affect the memory is still valid.

Consider a network where all eigenvalues are zero, where there would be no edges. The state of each neuron will only depend on the last input, and the $M_\tau = 0$ for all $\tau \geq 1$. If we add one non-zero eigenvalue to the matrix, the state of every neuron now depends on two variables, the current input and the projection of the previous state onto the single eigenvector. This idea can be extended to include more and more eigenvalues, with their moduli accounting for the speed at which the variance of previous states contracts. Eventually, every neuron will have many variables on which it will depend, and therefore it will be less correlated –on average – to other neurons, which will depend differently on the same variables.

This intuition can be easily checked by studying the eigenvalues of the networks whose memory was measured in Fig. 3. The results, plotted in Fig. 4, show that indeed this is the case, as both parameters α, γ and β pull more eigenvalue moduli close to zero, while $\langle k \rangle$ does not.

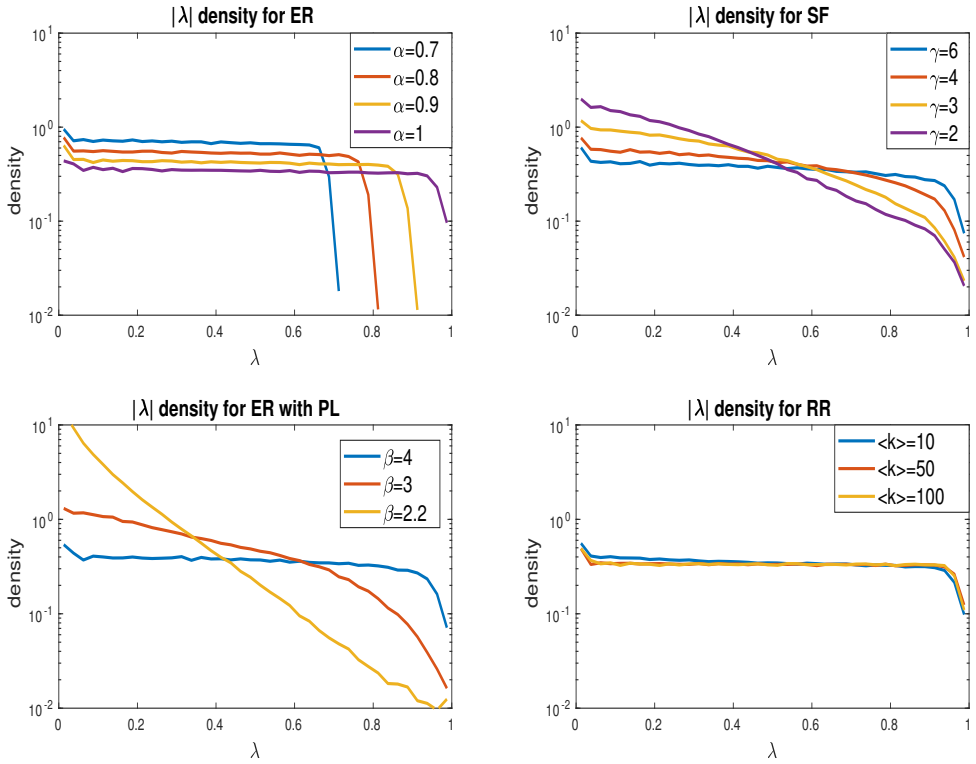


Figure 4: **Eigenvalue densities of the random networks studied in Fig. 3**. Each line shows the density of eigenvalues at a distance λ from the origin. 200 realizations of a network. The network size is 1000, the edge weights are drawn from a normal distribution except when in the ER networks with PL. Unless explicitly stated in the sub-figure legend, $\langle k \rangle = 50$ and the spectral radius is $\alpha = 1$.

10.1 Formal relationship between eigenvalues and correlation

We must now show that the larger the eigenvalues of $\mathbf{W}$, the lower the correlations.

To do so we will first linearize the system presented in Eq. 1, which gives us

$$\mathbf{x}(t) = \mathbf{W}\mathbf{x}(t-1) + \mathbf{w}_{\text{in}}u(t). \tag{46}$$

This linearization might seem unjustified, as a key requirement of a reservoir is that it must be non-linear [Jae02] to provide the necessary diversity of computations that a practical RC implementation requires. However, here we are interested in the memory capacity, which is maximized for linear reservoirs [Jae02, WLS04]. That is, by studying a linear system we are implicitly deriving an upper bound on the memory, similarly to the approach taken in the control-theoretical study of the effect of the spectral radius [Jae01a]. Finally, note that this linearizion is within the parameters of the ESN from Eq.1, as it would suffice to set $\|\mathbf{w}_{\text{in}}\| \ll 1$.

Given that our system is linear, we can formulate the state of a single neuron $x_i(t)$ as

$$x_i(t) = \sum_{k=0}^{\infty} \left(\mathbf{W}^k \mathbf{w}_{\text{in}}\right)_i u(t-k) = \sum_{k=0}^{\infty} a_{i,k} u(t-k) = \langle \mathbf{a}_i, \mathbf{u}_t \rangle \tag{47}$$

where $\mathbf{u}_t = [u(t), u(t-1), u(t-2), ...]$ and the coefficients of the vector $\mathbf{a}_i = [a_{i,0}, a_{i,1}, ...]$ are given by

$$a_{i,k} = \left(\mathbf{W}^k \mathbf{w}_{\text{in}}\right)_i, \tag{48}$$

namely the effect that the kth last input $u(t-k)$ will have on $x_i(t)$.

We can then plug this into the covariance between two neurons,

$$\text{cov}\,(x_i, x_j) = \lim_{T\to\infty} \frac{1}{T} \sum_{q=t}^{t+T} \langle \mathbf{a}_i, \mathbf{u}_q \rangle \langle \mathbf{a}_j, \mathbf{u}_q \rangle. \tag{49}$$

Here it is useful to decompose each summand

$$\begin{aligned} \langle \mathbf{a}_i, \mathbf{u}_q \rangle \langle \mathbf{a}_j, \mathbf{u}_q \rangle &= \left(\sum_{k=0}^{\infty} a_{i,k} u(q-k) \right) \left(\sum_{l=0}^{\infty} a_{j,l} u(q-l) \right) \\ &= \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} a_{i,k} a_{j,l} u(q-k) u(q-l), \end{aligned} \tag{50}$$

and combining now the limit and adding over all indexes q we obtain

$$\begin{aligned} \text{cov}\,(x_i, x_j) &= \lim_{T\to\infty} \frac{1}{T} \sum_{q=t}^{t+T} \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} a_{i,k} a_{j,l} u(q-k) u(q-l) \\ &= \sum_{k=0}^{\infty} \sum_{l=0}^{\infty} a_{i,k} a_{j,l} \lim_{T\to\infty} \frac{1}{T} \sum_{q=t}^{t+T} u(q-k) u(q-l), \end{aligned} \tag{51}$$

and given that $u(t)$ is a random time series with zero autocorrelation and variance of one,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{q=t}^{t+T} u(q-k) u(q-l) = \begin{cases} 1 & \iff l = k \\ 0 & \text{otherwise} \end{cases} \tag{52}$$

This gives us

$$\text{cov}\,(x_i, x_j) = \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} a_{i,k}a_{j,l}\delta(k-l) = \sum_{k=0}^{\infty} a_{i,k}a_{j,k} = \langle \mathbf{a}_i, \mathbf{a}_j \rangle, \tag{53}$$

and similarly, we can compute the variance of x_i,

$$\text{var}\,(x_i) = \text{cov}\,(x_i, x_i) = \langle \mathbf{a}_i, \mathbf{a}_i \rangle = \|\mathbf{a}_i\|^2. \tag{54}$$

We can plug the previous two formulas into the Pearson's correlation coefficient between two nodes i, j as:

$$P_{ij} = \frac{\langle \mathbf{a}_i, \mathbf{a}_j \rangle}{\|\mathbf{a}_i\|\|\mathbf{a}_j\|} = \cos(\angle(\mathbf{a}_i, \mathbf{a}_j)) \tag{55}$$

which is the same as the cosine distance between vectors $\mathbf{a}_i$ and $\mathbf{a}_j$.

The next step is thus to write $\mathbf{a}_i$ as a function of the eigenvalues of $\mathbf{W}$. To do so, we note that the state of a neuron can be written as

$$\mathbf{x}(t) = \sum_{k=0}^{\infty} \mathbf{W}^k \mathbf{w}_{\text{in}} u(t-k) = \sum_{k=0}^{\infty} \left(V \mathbf{\Lambda}^k V^{-1}\right) \mathbf{w}_{\text{in}} u(t-k) \tag{56}$$

where V is the matrix eigenvectors of $\mathbf{W}$ and $\mathbf{\Lambda}$ the diagonal matrix containing the eigenvalues of $\mathbf{W}$. When we obtain

$$x_i(t) = \sum_{k=0}^{\infty}\sum_{n=1}^{N} \lambda_n^k \langle v_n^{-1}, \mathbf{w}_{\text{in}} \rangle (v_n)_i u(t-k), \tag{57}$$

where v_n and v_n^{-1} are, respectively, the left and right eigenvectors of $\mathbf{W}$. Notice that as long as the network given by $\mathbf{W}$ is drawn from an edge-symmetric probability distribution – meaning that $\Pr[\mathbf{W}_{ij} = a] = \Pr[\mathbf{W}_{ji} = a] \ \forall a \in \mathbb{R}$– and is self averaging then v_n and v_n^{-1} are vectors drawn from the same distribution.

The λ_n^k terms present in the previous equation can be used as a new vector basis,

$$x_i(t) = \sum_{n=1}^{N} \langle v_n^{-1}, \mathbf{w}_{\text{in}} \rangle (v_n)_i \langle \boldsymbol{\lambda}_n, \mathbf{u}_t \rangle = \sum_{k=0}^{\infty}\sum_{n=1}^{N} \lambda_n^k b_{i,n} u(t-k), \tag{58}$$

where $\boldsymbol{\lambda}_n = [1, \lambda_n, \lambda_n^2, ...]$ and $b_{i,n} = \langle v_n^{-1}, \mathbf{w}_{\text{in}} \rangle (v_n)_i$. By simple identification from Eq. 47, we find that

$$(\mathbf{a}_i)_k = \sum_{n=1}^{N} \lambda_n^k b_{i,n}. \tag{59}$$

Thus every coefficient of $\mathbf{a}_i$ is a sum of many terms. Specifically, every term is a multiplication of $b_{i,n}$, which are all independent as they refer to the projections of $v_{n,i}$ into $\mathbf{w}_{\text{in}}$ and the values of λ_n, whose phase – which we assume to be uniformly distributed on $[0, 2\pi]$– ensures that $(\mathbf{a}_i)_k$ is uncorrelated with $(\mathbf{a}_i)_{k+1}$.

We will now proceed to cast the distribution of $\mathbf{a}_i$ as a uniform distribution of points defining an ellipsoid. By the central limit theorem, the values of $\mathbf{a}_i$ are independent random variables drawn from a normal distribution with zero mean and whose variance decreases with the index k. Therefore the distribution of $\mathbf{a}_i$ is given by

$$\prod_{k=0}^{\infty} e^{-\frac{(\mathbf{a}_i)_k^2}{s_k^2}} \tag{60}$$

where s_k is a decreasing function of k. Thus all the points with probability e^{-r^2} are given by the surface

$$\sum_{k=0}^{\infty} \frac{a_k^2}{s_k^2} - r^2 = 0 \tag{61}$$

which are ellipsoids of infinite dimension and axes $\frac{s_k}{r}$. Furthermore, we are only interested in the angular coordinates of the points in the ellipsoid, not on their distance to the origin. Thus we can project every one of those surfaces into an ellipsoid with axis

$$\boldsymbol{s} = \left[1, \frac{s_2}{s_1}, \frac{s_3}{s_1}, ...\right]. \tag{62}$$

Note that, even though the ellipsoid has infinite dimensions, the length of the axes decreases exponentially due to the factor λ_n^k. Therefore, it has finite volume and it can be approximated by an ellipsoid with finite dimensions.

Now we have that the vectors $\mathbf{a}_i$ are, ignoring their length, uniformly distributed on an ellipsoid with axis $\boldsymbol{\sigma}$. Then

$$\lim_{N\to\infty} S = \frac{1}{2} \int_{E_{\boldsymbol{s}}} \int_{E_{\boldsymbol{s}}} \cos^2(\angle(p, q)) dp dq \tag{63}$$

where the integrals are taken over $E_{\boldsymbol{s}}$, the ellipsoid with axes $\boldsymbol{s}$, and the half factor comes from counting every pair only once.

If we now change to spherical coordinates we will find that the two vectors can be expressed as

$$p = [r_p, \phi_1^p, \phi_2^p, ...]$$
$$q = [r_q, \phi_1^q, \phi_2^q, ...],$$

where ϕ_1^p is the angle of p on the plane given by the first and second axis, ϕ_2^p the plane by the first and third axis and so on. The cosine between the two vector is

then

$$\cos(\angle(p,q)) = \prod_{k=1}^{\infty} \cos(\phi_k^p - \phi_k^q). \tag{64}$$

Thus we can write the integral from Eq. 63 as

$$\lim_{N\to\infty} S = \frac{1}{2} \prod_{k=1}^{\infty} \int_0^{2\pi} \cos^2\left(\phi_k^p - \phi_k^q\right) \mu_{\phi_k}(\phi_k^p) \mu_{\phi_k}(\phi_k^p) d\phi_k^p d\phi_k^q \tag{65}$$

where $\mu_{\phi_k}(\phi)$ is the probability density function of the difference angle $\phi_k^p - \phi_k^q$.

We will show that this integral decreases when the values s_k increase. To do so, it is helpful to consider the extreme cases to get an intuition: when the semi-minor axis is zero, then we have a line, and all the points in the line have an angle between them either of zero or π, and thus a squared cosine of one. Conversely, when the semi-minor axis is maximal it equals the semi-major one and we have a circle, and the average squared cosine becomes $1/2$. Those are the two extreme values and thus the squared cosine decreases as the ellipse becomes more similar to a circle.

To make this argument more precise, we start by finding the density $\mu_{\phi_k}(\phi_k^p)$. This density is found by taking a segment of differential length dl_S on the sphere of radius one and then compare it with the length covered in the ellipse dl_E. This gives us

$$\begin{aligned}\mu_{\phi_k}(\phi) \propto \frac{dl_E}{dl_S} &= \frac{\| \cos(\phi - d\phi) - \cos(\phi), s_k^2(\sin(\phi - d\phi) - \sin(\phi))\|}{\phi + d\phi - \phi} \\ &= \sqrt{\sin^2(\phi) + a_k^2 \cos^2(\phi)} = \sqrt{1 - (1 - s_k^2)\cos^2(\phi)}.\end{aligned}$$

To fully evaluate the previous integral we would need to normalize μ_{ϕ_k} and then evaluate the integral as a function of s_k. However, we would take a simpler approach and note that s_k controls the homogeneity of μ_{ϕ_k}: the larger s_k is (within the interval $[0,1]$), the more the mass of probability is concentrated on the area around $\phi \sim 0$ and $\phi \sim \pi$.

Furthermore, we note that the squared cosine has the following periodicities

$$\cos^2(\theta) = \cos^2(\pi + \theta) = \cos^2(\pi - \theta) = \cos^2(2\pi - \theta),$$

thus, when we integrate over the angle ϕ we can take advantage of the four-fold symmetry and integrate only on the interval $[0, \pi/2]$. Thus we only need to study the integral

$$\int_0^{\frac{\pi}{2}} \cos^2\left(\phi_k^p - \phi_k^q\right) \mu_{\phi_k}(\phi_k^p) \mu_{\phi_k}(\phi_k^p) d\phi_k^p d\phi_k^q, \tag{66}$$

and by using $\sin^2(\theta) + \cos^2(\theta) = 1$, we can recast the previous integral as

$$1 - \int_0^{\frac{\pi}{2}} \sin^2\left(\phi_k^p - \phi_k^q\right) \mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p) d\phi_k^p d\phi_k^q. \tag{67}$$

In the interval $\phi \in [0, \pi/2]$ the squared sine can be seen a metric between two angles. Therefore the term

$$\int_0^{\frac{\pi}{2}} \sin^2\left(\phi_k^p - \phi_k^q\right) \mu_{\phi_k}(\phi_k^p)\mu_{\phi_k}(\phi_k^p) d\phi_k^p d\phi_k^q \tag{68}$$

is nothing else than an average distance between the points which have a density given by μ_{ϕ_k}. Thus, the more the density is homogeneous, the larger the distance and vice-versa. Putting it all together, s_k controls the homogeneity of μ_{ϕ_k}, and the homogeneity of μ_{ϕ_k} controls the terms on Eq. 65. Specifically, increasing s_k decreases S.

The last thing to mention is that the values of $|\lambda_n|$ control s_k, as they give the variance to $(\mathbf{a}_i)_k$ in Eq. 59. Thus, the larger $|\lambda_n|$ the higher s_k and the lower S. By the negative correlation between M and S, increasing the values of $|\lambda_n|$ should increase the memory.

11 A structural proxy for Memory Capacity

The previous argument highlighted the relationship between the eigenvalues of $\mathbf{W}$ and S, but it is far from practical. Ideally, we would like to have a single number that can be used to design reservoirs.

Given that the moduli of all eigenvalues of $\mathbf{W}$ contribute to the memory, we take their average:

$$\langle|\lambda|\rangle = \frac{\sum_{i=1}^{N} |\lambda_i|}{N}, \tag{69}$$

where λ_i's are the eigenvalues of the matrix $\mathbf{W}$.

The numerical simulations presented in Fig5 show that indeed $\langle|\lambda|\rangle$ is negatively correlated with S.

The correlation of $\langle|\lambda|\rangle$ with S and therefore with M indicate that $\langle|\lambda|\rangle$ indeed reflects the memory capacity of the reservoir. As opposed to M and S, $\langle|\lambda|\rangle$ is much easier to compute and is solely determined by the reservoir network. This offers a simple measure to quantify the ESN memory capacity that does only depend on the network structure. $\langle|\lambda|\rangle$ is consistent with the effects of scaling the adjacency matrix to tune the spectral radius [JLPS07a] and it extends to network topologies with a fixed spectral radius from Fig.3. This also explains two recent studies in which it was found that ring networks and orthogonalized networks have

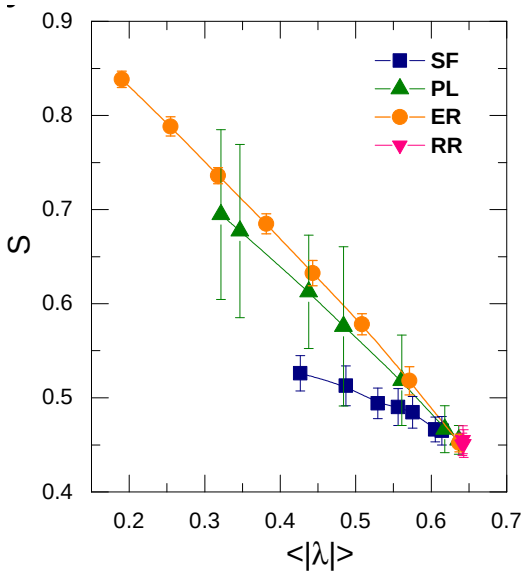


Figure 5: **Neuron state correlation S vs average eigenvalue modulus** $\langle|\lambda|\rangle$. We created reservoirs of 400 neurons and sequences of 4000 random inputs uniformly distributed in the interval $[-1, 1]$ as inputs. For every network we plotted the correlations S generated versus the value of $\langle|\lambda|\rangle$. The ER curve is given by classical reservoirs having networks given by Erdös-Rényi random graphs with weights drawn from a Gaussian distribution and varying spectral radii. The SF curve (blue) corresponds to scale-free networks where the degree heterogeneity is given by the degree exponent $\gamma \in [2, 6]$, with more heterogeneous networks rendering higher S and lower $\langle\lambda\rangle$. The PL curve (green) is calculated from Erdös-Rényi random graphs with weights drawn from a power-law (PL) distribution with varying exponent $\beta \in [2, 5]$, with lower β rendering higher S and lower $\langle\lambda\rangle$. All networks have a spectral radius $\alpha = 1$, except the ER random graphs where each point corresponds to a spectral radius increasing from 0.2 to 1. The ER random graphs are plotted with various spectral radius to show the impact of spectral radius.

high memory capacities [RT12, FI16], as both networks have large eigenvalues with respect to their spectral radii.

Since ESN is fundamentally a machine learning tool, any results should be supported by studying its effects on the ESN performance. To demonstrate the validity of $\langle|\lambda|\rangle$ as a proxy measure for memory, we tested ESN performance for the three tasks presented earlier with a wide range of network topologies and parameters in Fig. 6.

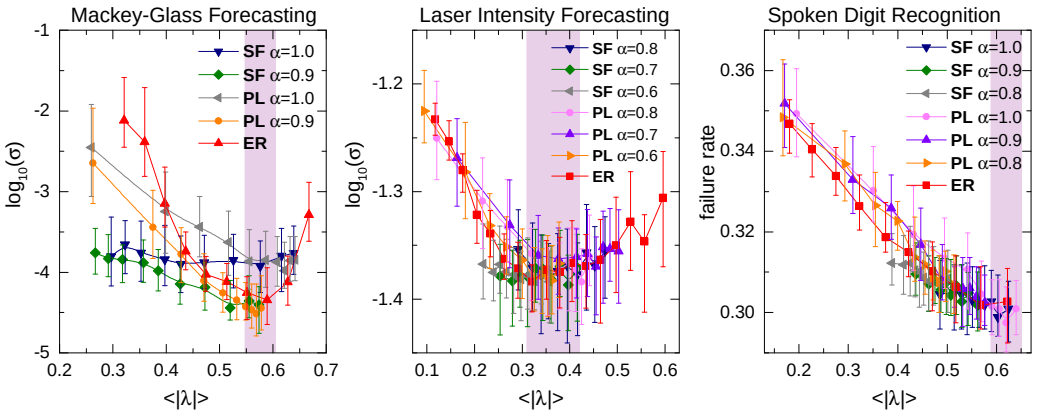


Figure 6: **ESN performance explained by** $\langle|\lambda|\rangle$. The plot shows the ESN performance as a function of the average eigenvalue modulus $\langle|\lambda|\rangle$ for forecasting the Mackey-Glass time series (left), the Laser Intensity time series (center), and classificatio of Spoken Arabic Digits (right). For each task, we use scale-free networks (SF), Erdős-Rényi random graphs with homogeneous link weights (ER), and Erdős-Rényi random graphs whose link weights follow a power-law distribution (PL) as reservoirs (see SI Sec. II for the network generation algorithms). The SF and PL reservoirs have various spectral radii α, chosen to be around the optimal value of α for the Erdős-Rényi case. For each parameter set of each network type we created 200 ESN realizations, and then all the points obtained were grouped in 10 bins containing the same number of points. We plotted their median $\langle|\lambda|\rangle$ against their median performance: σ from Eq.4 for forecasting; and the failure rate for the classification. The error bars correspond to the upper and lower quartile respectively. Each ESN realization corresponds to a reservoir with $N = 1000$ neurons and average degree $\langle k\rangle = 50$ for Mackey-Glass; and $N = 100$, $\langle k\rangle = 10$ for Laser Intensity and Spoken Digit recognition. The three panels show that, regardless of topology or spectral radius, all networks have their optimal performance when the average eigenvalue module, $\langle|\lambda|\rangle$, is within the intervals $[0.55, 0.6]$ for Mackey-Glass, $[0.3, 0.4]$ for Laser Intensity, or $[0.6, 0.7]$ for Spoken Arabic Digit Recognition, which are highlighted in pink.

12 Discussion

The results from Fig. 6 highlight the generality of our argument, as it seems to work for every network and every topology, but also a more basic idea: that a reservoir should have a certain amount of memory for a given task, and not more. If the memory is insufficient, the readout does not have enough information to complete its task and the performance will be poor, as it happens when $\langle|\lambda|\rangle$ is small. Conversely, if the memory is too high, the network is spending ”resources”

– here meaning volume in the state space – trying to remember inputs that are irrelevant for the task, and thus its performance will decrease when $\langle|\lambda|\rangle$ is too large.

An important consequence of the derivation presented here is that the relationship between memory and structure goes through the correlations between neurons. This implies that the same arguments presented here can be reused in more complicated dynamical systems such as photonic reservoirs [LSB$^+$12, BSVdS19], where the nonlinearity of the nodes is not monotonic and thus the stability cannot be used as a criteria for memory. In such cases, our argument of limiting the memory by the correlations still works, hence tunning the correlations would still adapt the memory of the reservoir.

However, the key result from this chapter remains that the metric $\langle|\lambda|\rangle$ can serve as a proxy for memory, thus if we need to constraint our network architecture we can find the optimal memory fist – by scaling the spectral radius in an ER network – and then generate our constrained network in such a way that the value of the memory is unchanged. This will be necessary in the following chapters, where we will tailor reservoirs to specific problems in a way that would change the distribution of their eigenvalues.

Frequency adaptation for Echo State Networks

The need for a narrow interval for $\langle|\lambda|\rangle$, as opposed to simply maximizing the memory, is in line with the general rule that specialization is key to performance. This insight, which resembles in spirit the no-free-lunch theorem, indicates that a reservoir should be selected depending on the task at hand. In this section we follow this intuition using notions from signal processing, specifically the frequency spectrum decomposition.

The Fourier transform has been an extremely useful tool in time-series processing and dynamic systems modeling [CG89, BZA01, Wei94]. Any signal or time series can be expressed in terms of its spectrum, which reflects the decomposition of signals into sinusoids with different frequencies [SS90]. This type of characterization usually offers valuable insights into the nature or underlying dynamics of the system, and has been exploited for various applications [Ell13]. In this section we leverage it to adapt reservoirs to specific tasks or problems.

The intuition that we will use here is that every neuron can be seen as a filter that extracts some features from the input time series. If the reservoir extracted the right features, the ESN performance would improve. Our goal is then to show that a convenient family of features to look for corresponds to the frequency response of the neurons. To do so, we will first show how the problem of selecting a good reservoir can be mapped into the problem of selecting neurons with the right frequencies by a formal argument, then show that this frequencies can be imposed onto the reservoir by adding cycles to it and finally show a simple heuristic to improve ESN performance.

13 A geometric approach to ESN training

The gist of our argument relies on a geometric interpretation of the linear regression: the target time series is a point in a space of dimension T, where every coordinate is the value of the target time series; the neurons are also points in that same space, and the linear regression simply finds the point in the affine linear

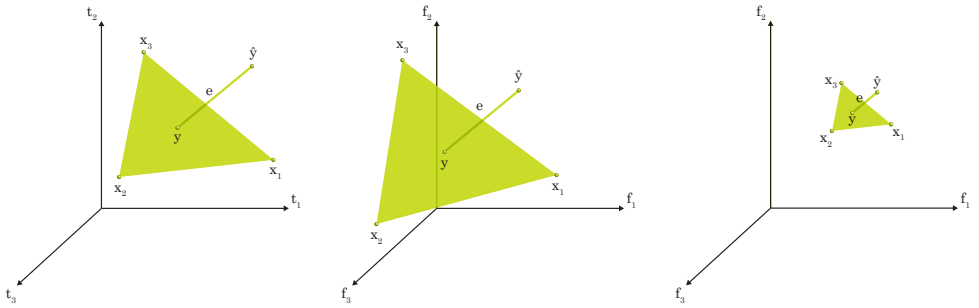


Figure 7: **Sketch of the frequency adaptation argument**: Given an input and output time series, a randomly generated reservoir consists of N non-linear projections of the input time series, which can be written as N vectors in a space of dimension T (left). This picture and the distances between points do not change in the Fourier domain (center). However, in that domain it will be possible to force the neurons to approach the target.

subspace spanned by all the neuron time series that is closest to the target point. Naturally, by having the neurons closer to the target, the distance between the hyperplane and the target – the training error – decreases. The problem here is that getting the neurons closer to the target is typically difficult, as the time series of the neurons are intertwined [PMB13]. Our insight is that the distances in the time domain are the same as the distances in the Fourier domain – by Parseval's theorem– thus we can reduce the training error by making the reservoir resonate at specific frequencies.

We start by noting that the ESN training from Eq. 3 is the minimization of a Euclidean distance. More specifically, the squared training error

$$\|\mathbf{e}\|^2 = \sum_{t=t_0}^{T+t_0} e^2(t) = \sum_{t=t_0}^{T+t_0} (y(t) - \mathbf{w}_{\text{out}}\mathbf{x}(t))^2 \quad = \|y - \sum_{i=1}^{N} (\mathbf{w}_{\text{out}})_i x_i\|^2 \qquad (70)$$

is a squared distance, where $\|\cdot\|$ is the euclidean norm and $\mathbf{e}$ is the vector of errors, which inhabits the space of training time series. In this space, y is the target point, where every value of $y(t)$ corresponds to the coordinate of y at dimension t. The time series of the neurons x_i are also points in that space with $x_i(t)$ being their corresponding coordinates. Then, $\hat{y}$ is the linear combination of neuron points that is closest to y.

Naturally, this implies that a change of basis does not affect the training error as long as the new basis is orthonormal

Having this geometrical interpretation of the ESN training we can already get an intuitive understanding of how the reservoir network should be selected: we

should sample x_i to be as close as possible to the target y, then the error – the minimum distance between the hyperplane spanned by x_i and the target – should also be reduced.

To make this argument more precise, we will rotate the coordinate frame so that the first dimension of our space is in the exact direction of y. We will note the points in this coordinate by the superindex P, so that x_i^P are the new points and $y^P = [\|y\|, 0, 0, ..., 0]$.

In this new basis, we can rewrite the error as

$$\|e\|^2 = \|y^P(0) - \sum_{i=1}^{N}(\mathbf{w}_{\text{out}})_i x_i^P(0)\|^2 + \|\sum_{i=1}^{N}(\mathbf{w}_{\text{out}})_i x_i^P(j)\|^2. \tag{71}$$

To obtain a bound we impose the constraint that the first term must be equal to zero, so

$$\mathbf{w}_{\text{out}}^* = \arg\min_{\mathbf{w}_{\text{out}}^*} \|\sum_{i=1}^{N}(\mathbf{w}_{\text{out}})_i x_i^P(j)\|^2 \quad s.t. \quad y^P(0) - \sum_{i=1}^{N}(\mathbf{w}_{\text{out}}^*)_i x_i^P(0) = 0, \tag{72}$$

where $\mathbf{w}_{\text{out}}^*$ is the new readout. Note that we are not proposing a new training goal, as this constraint is just a mathematical trick to bound the training error.

Naturally, this extra constraint can only make the minimization problem harder, so

$$\|e\|^2 \leq \|e^*\|^2 = \sum_{j=2}^{T-1}\left[\sum_{i=1}^{N}(\mathbf{w}_{\text{out}}^*)_i x_i^P(j)\right]^2. \tag{73}$$

Furthermore, we can also use the constraint to bound $\mathbf{w}_{\text{out}}$ through the Cauchy-Schwarz inequality,

$$(y^P(0))^2 = \|y\|^2 = \left(\sum_{i=1}^{N}(\mathbf{w}_{\text{out}}^*)_i x_i^P(0)\right)^2 \leq \|\mathbf{w}_{\text{out}}^*\|^2 \sum_{i=1}^{N}(x_i^P(0))^2 \tag{74}$$

which gives us the inequality

$$\|\mathbf{w}_{\text{out}}^*\|^2 \geq \frac{\|y\|^2}{\sum_{i=1}^{N}(x_i^P(0))^2}. \tag{75}$$

Now we can plug this into Eq.73,

$$\|e\|^2 \leq \sum_{j=2}^{T-1}\left[\sum_{i=1}^{N}(\mathbf{w}_{\text{out}}^*)_i x_i^P(j)\right]^2 \leq \|y\|\frac{\sum_{j=2}^{T-1}\left[\sum_{i=1}^{N} x_i^P(j)\right]^2}{\sum_{i=1}^{N}(x_i^P(0))^2}. \tag{76}$$

This equation gives us an upper bound for our training error. This bound simply states that if the points are very large in coordinate 0 of the new reference frame, and very small in all the other coordinates, then the training error will improve. This is not surprising, as the coordinate 0 is aligned with the target; our bound simply states that if the time series of the neurons are very similar to the target, then calculating the target from the neurons is easy.

Furthermore, since the rotation preserves the symmetry, the signs of $x_i^P(0)$ are uncorrelated with the signs of $x_i^P(j)$,

$$\sum_{i=1}^{N} x_i^P(j) \leq \mathcal{N}(0, N s^2_{x^P \perp y}) \tag{77}$$

where $s^2_{x^P \perp y}$ is the variance of $x_i^P(j)$ for $j > 1$, and the zero mean comes from the symmetry of the distribution. As the neurons are independently drawn and thus have all the same variance, then

$$\sum_{j=2}^{T-1} \left[\sum_{i=1}^{N} x_i^P(j) \right]^2 = N s^2_{x^P \perp y} \sum_{j=2}^{T-1} \mathcal{N}(0,1)^2 \tag{78}$$

where $\sum_{j=2}^{T-1} \mathcal{N}(0,1)^2$ is the chi-squared distribution with $T-1$ degrees of freedom. When T is large, we can use the central limit theorem, and thus

$$\|e\|^2 \leq \|y\| \frac{T N s^2_{x^P \perp y}}{N s^2_{x^P \| y}} = \|y\| T \frac{s^2_{x^P \perp y}}{s^2_{x^P \| y}}. \tag{79}$$

where $s^2_{x^P_{\| y}}$ is the variance of $x_i^P(0)$.

Note that the factor T appears because we did not normalize $\|e\|^2$ by the number of entries in the time series. imagine that we concatenate the same time series twice, both the training and target points. Naturally, $\mathbf{w}_{\text{out}}$ would be the same, but the squared distance to between y and $\hat{y}$ would have doubled.

The bound in Eq. 79 is simple to understand but it is hardly helpful: designing a reservoir so that the neuron time series would approach those of the target is a high-dimensional problem with many coupled variables, so we would not expect this to be an easy task; indeed, training recurrent neural networks is a known hard problem [PMB13]. Our main insight here is that this design is much simpler in the Fourier domain.

By Parseval's theorem [Par06]

$$\|e\|^2 = \sum_{t=t_0}^{t_0+T} (y(t) - \mathbf{w}_{\text{out}} \mathbf{x}(t))^2 \quad = \sum_{f=0}^{T} |\mathcal{F}\left[y - \mathbf{w}_{\text{out}} \mathbf{x}\right](\omega)^2, \tag{80}$$

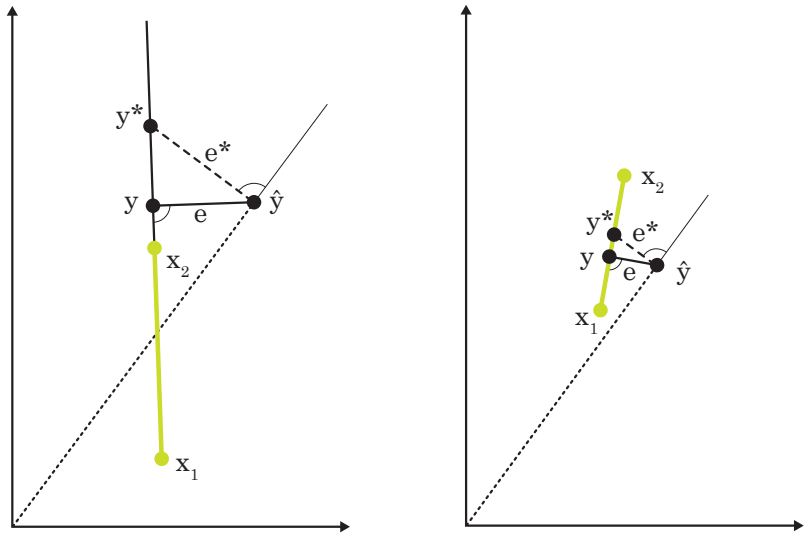


Figure 8: **Sketch of the geometric bound**: The target output is approximated by its orthogonal projection into the hyper plane formed by the neurons, with the training error $\|e\|$ being the distance. The constrained error $\|e^*\|$ instead is the projection that is orthogonal to $\hat{y}$, which is obviously larger. Both errors decrease when the neurons are closer to $\hat{y}$, with $\|e^*\|$ serving as an upper bound for $\|e\|$. Notice that the argument works for any orthonormal basis of the time series, thus it is more general than the frequency adaptation used here.

where $\mathcal{F}$ is the Fourier transform. Furthermore, the Fourier transform preserves linear operations, so

$$\|e\|^2 = \sum_{f=0}^{T} \left(|\mathcal{F}[y](\omega) - \sum_{i=1}^{N} (\mathbf{w}_{\text{out}})_i . \mathcal{F}[x_i](\omega)| \right)^2, \tag{81}$$

Then we can plug this in Eq. 79, specifically on the quotient

$$\frac{s^2_{x^P \perp y}}{s^2_{x^P \| y}} \tag{82}$$

which is fairly easy to interpret in terms of signal processing: $s^2_{x^P \| y}$ is the projection of the power spectral density of the distribution of x_i on the target time series, and $s^2_{x^P \perp y}$ is the orthogonal power spectral density. In other words, ESNN performance improves if the PSD of the variables x_i approaches that of the target

time series y. This is quite a natural result, since the power spectral density is often used as a measure of how distant time series are [Wei94]; thus we are simply saying that if the time series of the variables x_i are similar to the target, then the readout will work better.

14 Altering the PSD of the reservoir's neurons

Knowing that altering the reservoir's PSD may increase the ESN's performance, we can deliberately generate reservoirs with specific frequency bands. To achieve that, we add feedback loops with delay L in our neurons, encoded as cycles of length L in the network. Note that this can be considered as a simple extension of classical Infinite Impulse Response filters from Signal Processing [Ell13]. We account for the number and strength of those cycles by using the following measure:

$$\rho_L = \mathbb{E}\left[W^L\right] = \frac{\sum_{n=1}^{N}\left(W^L\right)_{nn}}{N} \tag{83}$$

which accounts for the number of cycles of length L because the Lth power of the adjacency matrix of a graph corresponds to the adjacency matrix of the graphs of paths of length L.

Before giving a mathematical study of how those frequencies are enhanced we will show numerically that a reservoir with those cycles enhances different families of frequencies by feeding white noise –which has all frequencies in equal measure – to reservoirs with different ρ_L values, as shown in Fig. 9.

14.1 Cycles, resonances and eigenvalues in linear systems

The notion of resonance or frequency enhancement that we used can be expressed in a dynamical systems setting, where a reservoir resonates at a frequency $\frac{1}{L}$ if

$$\mathbf{x}(t) \sim a\mathbf{x}(t-L) \sim a^2\mathbf{x}(t-2L)... \tag{84}$$

for $a > 0$. Note that the similarity should only hold for $t-L, t-2L, ...$, meaning that a the system state will be close to its previous state L timesteps in the past and not similar to it otherwise.

To show how the addition of cycles can generate those resonances, we will start by considering a simple linear – or in this case, linearized – dynamical system defined by

$$\mathbf{x}(t) = \mathbf{W}\mathbf{x}(t-1) + \mathbf{w}_{\text{in}}u(t) = \sum_{k=0}^{\infty}\mathbf{W}^k\mathbf{w}_{\text{in}}u(t-k) \tag{85}$$

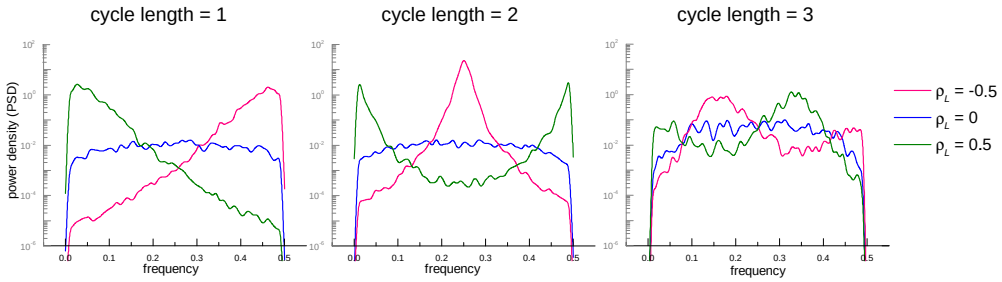


Figure 9: **Frequency domain representation of reservoir activity for reservoirs with cycles**. We plotted the average PSD of the reservoirs' neuron states for reservoirs with various ρ_L when using a random Gaussian input with zero mean and variance of one. In each panel we plot the average PSD of 500 reservoirs with 400 neurons and connectivity 0.05. The length of cycles added into the reservoir is 1, 2 and 3. Any reservoir with $\rho_L > 0$ consists mostly of low pass filters and will therefore enhance the frequencies close to 0, while $\rho_L < 0$ will enhance the complementary ones.

where the hyperbolic tangent has been replaced by an identity, similar to a linearization around the origin. Expanding $\mathbf{W}$ into its eigenvalues and eigenvectors gives us

$$\mathbf{x}(t) = \sum_{k=0}^{\infty} \mathbf{W}^k \mathbf{w}_{\text{in}} u(t-k) = \sum_{n=1}^{N} v_n \langle v_n^{-1}, \mathbf{w}_{\text{in}} \rangle \sum_{k=0}^{\infty} \lambda_n^k u(t-k) \tag{86}$$

where v_n and v_n^{-1} are the left and right eigenvectors of $\mathbf{W}$ respectively and λ_n the eigenvalues.

With this decomposition, it is easy to see that in order to obtain the approximation from Eq. 84, we would like to have

$$\sum_{n=1}^{N} \lambda_n^k = \begin{cases} a & \iff k = L \\ 0 & \iff k \neq L. \end{cases} \tag{87}$$

The trick to obtain that is to realize that the eigenvalues live in the complex domain, so we can use them as vectors to satisfy Eq. 87. As a simple illustrative example, we will consider a system of $N = 3$ neurons forming a cycle of length three, which is defined by the matrix

$$\mathbf{W} = \begin{pmatrix} 0 & w_{12} & 0 \\ 0 & 0 & w_{23} \\ w_{31} & 0 & 0 \end{pmatrix} \tag{88}$$

The eigenvalues of this matrix are the three cubic roots of $w_{12}w_{23}w_{31}$, thus fulfilling Eq. 87. The three eigenvalues form an equilateral triangle centered at zero. When we go back to Eq. 84, we find that in this particular system,

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{w}_{\text{in}}u(t) + \mathbf{W}\mathbf{w}_{\text{in}}u(t-1) + \mathbf{W}^2\mathbf{w}_{\text{in}}u(t-2) \\ &+ w_{12}w_{23}w_{31}\mathbf{w}_{\text{in}}u(t-3) + \mathbf{W}^4\mathbf{x}(t-4) \\ &\sim w_{12}w_{23}w_{31}\left[\mathbf{w}_{\text{in}}u(t-3) + \mathbf{W}\mathbf{x}(t-4)\right] = w_{12}w_{23}w_{31}\mathbf{x}(t-3),\end{aligned} \tag{89}$$

which implies that $a = w_{12}w_{23}w_{31}$.

The simple example above illustrates the general relationship between eigenvalues and resonances: when the linear system resonates with period L, its dominant eigenvalues have phases of $\frac{2\pi k}{L}$, for $k = \{0, 1, ..., L-1\}$ if the cycles have positive feedback and $\frac{2\pi(k+1)}{L}$ if the feedback is negative. This is simply because then the dominant eigenvalues to the power L will be real, while they will not have any alignment when taken to a power which is not a multiple of L.

14.2 Eigenvalues of random matrices with cycles

With this general intuition in place, we will now proceed to study the eigenvalue support of large random matrices with an overabundance of cycles of length L. We will do so by first establishing a connection between cycles and the eigenvalue distribution of the network's adjacency matrix and then studying the symmetries and extrema of the eigenvalue distribution.

Before proving anything, however, we will generate some of those random networks and show their eigenvalues. The algorithm to generate them takes as parameters the number of neurons N; the number of edges E; $c_L \in [0, 1]$, which is the portion of edges that are dedicated to cycles and must be a rational fraction with quotient E, a probability distribution for the weights P that is symmetric around zero and has variance of one, and $s \in \{-1, 1\}$, which corresponds to the feedback sign.

Step-1: Create $\frac{c_L E N}{2L}$ permutations of L numbers randomly picked from 1 to N without replacement. Each permutation corresponds to L nodes that will be connected to form a cycle.

Step-2: For each cycle, assign random weights drawn from P.

Step-3: For each cycle, if the sign of the product of the edge weights is not the same as s, multiply the last edge by -1. This process generates an adjacency matrix $\mathbf{W}_{\text{c}}$.

Step-4: To complete the connectivity, draw random empty entries from the matrix and fill them with random weights drawn from P.

Step-5: Normalize the matrix by $\sqrt{\frac{N}{E}}$.

Before continuing, we should notice that in the previous algorithm or figure we have not considered the case of $L = 1$. The reason is that using purely the network structure it would lead to inconsistencies: if $Ec_1 > N^{-1}$, which would be common as $N \to \infty$, it would give a network where every node would have more than one edge connecting to itself. This does not make sense in our framework so for the reservoir generation we will simply use

$$\mathbf{W} = (1 - r_1)\mathbf{W_r} + c_1 I \tag{90}$$

where $\mathbf{W_r}$ is a random network and I an identity matrix. Here the strength of the cycles of length $L = 1$ comes not from their number, but from their weights given by c_1.

When we plot the eigenvalues of those networks we find a surprising result, namely that after appropriate normalization, the eigenvalues of such a matrix in the limit of $N \to \infty$ are within the support given by

$$z(\varphi) = e^{-i\varphi} + \rho_k e^{i(L-1)\varphi}. \tag{91}$$

This support, which was derived in a collaboration with Tim Rogers and Henning Schomerus [ARS19] is a generalization of the elliptic law of random matrices [Gir86]. Although it is a very interesting result which was obtained in the scope of this thesis, the proofs and derivations were mostly developed by Tim Rogers (for the case of sparse matrices) and Henning Schommerus (for the case of dense matrices). We will thus use a simpler method that will not get the exact support but that does not require knowledge of cavity methods or free probability.

Our derivations follow the logic of the method of moments, which is classically used to derive the Wigner's semicircle law [Fei12]. Since we rely heavily in combinatorial arguments, we will take P to be a binary distribution that associates weights of either 1 or -1 to every weight with equal probability. As exposed in [ARS19], the actual distribution of the weights is irrelevant to obtain the eigenvalue support, so we shall just use the most convenient one.

We begin by defining the normalized weight of the cycles of length L

$$\rho_L = \frac{1}{N} \sum_{c \in C_L} w_c \tag{92}$$

where C_L is the set of cycles of length L, and $w_c = \prod_{e \in c} w(e)$, the multiplication of weights of the edges e in cycle c. Note that this includes cycles where edges or nodes are visited multiple times, as opposed to simple cycles where every node

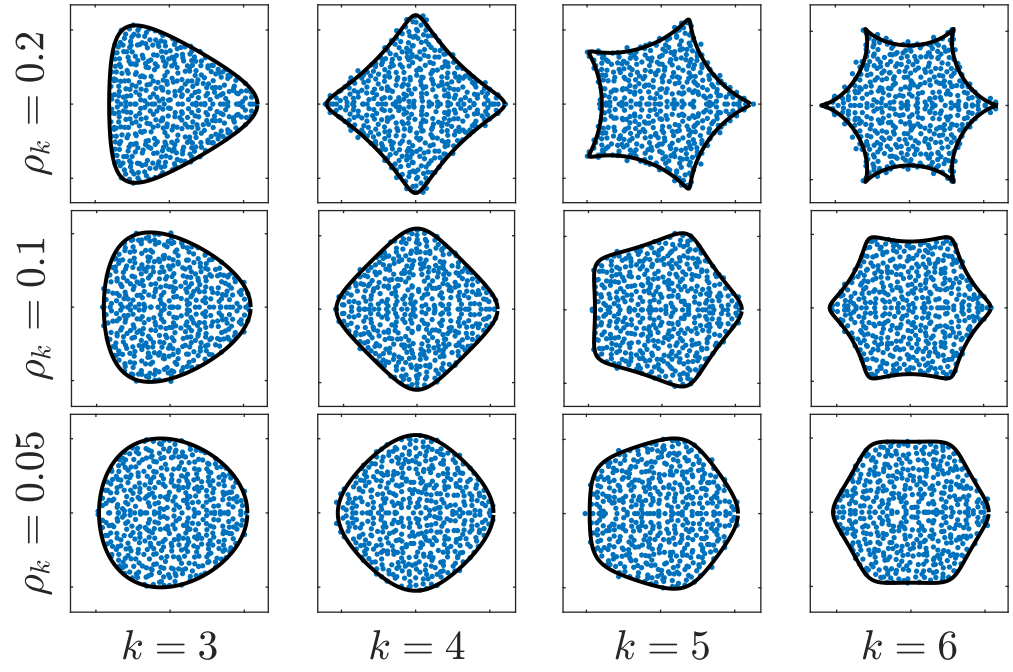


Figure 10: **Eigenvalues of matrices with cyclic correlations with theyr support**: Hypotrochoid curves given by Eq. 91 (black lines) bounding the eigenvalue spectra (blue dots) of random matrices with correlations $\overline{\mathrm{Tr}\mathbf{W}^k} = \rho_k$, for different cycle lengths k. Note that we did not include $k = 1$ nor $k = 2$, because those are well-characterized cases.

and edge can only be visited once. Since the value $\sum_{c \in C_L} w_c$ is given by the entries of the power of the graphs' adjacency matrix $\mathbf{W}$ [GY05], we can obtain ρ_L from the adjacency matrix $\mathbf{W}$ by the formula

$$\rho_L = \frac{1}{N} \mathrm{tr}\left[\mathbf{W}^L\right] \tag{93}$$

where $\mathrm{tr}\left[\cdot\right]$ is the trace operator. The trace of a matrix equals the sum of its eigenvalues [Str93], therefore ρ_L can be written as

$$\rho_L = \frac{1}{N} \sum_{n=1}^{N} \lambda_n^L \tag{94}$$

where λ_n is the nth eigenvalue of the adjacency matrix M.

Equation 94 is particularly interesting as $N \to \infty$. For this limit we can consider the eigenvalues of a matrix as random i.i.d. values sampled from a probability density function $p(\lambda)$ in the complex plane $\mathbb{C}$. For a random graph sampled

from a probability distribution with a fixed ρ_L, this value corresponds to a moment of the eigenvalue distribution,

$$\mu_L = \int_{\mathbb{C}} p(\lambda)\lambda^L d\lambda = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \lambda_n^L = \rho_L. \tag{95}$$

As the number of nodes goes to infinity, the values of ρ_L converge to their expectations, and since only the cycles of length τ have non-zero expectations.

$$\lim_{N\to\infty} \rho_L = \begin{cases} w^L \dfrac{F_L}{N} & \iff L \equiv 0 \mod \tau \\ 0 & \text{otherwise} \end{cases} \tag{96}$$

Where C_L is the number of cycles of length L that were added as feedback loops. Since ρ_L are the moments of $p(\lambda)$, the previous equation says that the moments which are not multiples of τ tend to zero. We will use this to study the symmetries in the eigenvalues, but first we need to consider the rotated density function $p(e^{\theta i}\lambda)$. The moments of this distribution can be computed through a simple change of variable,

$$\mu_L^\theta = \int_{\mathbb{C}} p(e^{\theta i}\lambda)\lambda^L d\lambda = \int_{\mathbb{C}} p(\lambda)\lambda^L e^{L\theta i} d\lambda = e^{L\theta i}\int_{\mathbb{C}} p(\lambda)\lambda^L d\lambda = e^{L\theta i}\mu_L. \tag{97}$$

Then the moments of $p(\lambda)$ and $p(e^{\theta i}\lambda)$ are equal under the following conditions,

$$\mu_L = \mu_L^\theta = e^{L\theta i}\mu_L \iff \begin{cases} \theta \in \left\{0, \dfrac{2\pi}{L}, \dfrac{4\pi}{L}, ..., \dfrac{2(L-1)\pi}{L}\right\} \\ \mu_L = 0. \end{cases} \tag{98}$$

This condition is fulfilled for large graphs with abundant cycles of length τ.

Furthermore since the eigenvalues are bounded, the equality of moments implies that $p(\lambda) = p(e^{\theta i}\lambda)$, so

$$p(\lambda) = p(e^{\theta i}\lambda) \tag{99}$$

for $\theta \in \left\{0, \frac{2\pi}{\tau}, ..., \frac{2(\tau-1)\pi}{\tau}\right\}$. In geometric terms, this means that $p(\lambda)$ has τ rotational symmetries in the complex plane.

Within the constraints of this symmetries, we would still need to know how ρ_τ affects the eigenvalue distribution. Since the entries of our adjacency matrix are i.i.d. and their magnitude decreases sub-exponentially, the probability distributions are uniform [BYY14] within a closed curve. This means that increasing ρ_τ decreases the surface in the angular range $\theta \in \left[\frac{\pi}{k}, \frac{2\pi}{k}\right]$ while increasing it in

$\theta \in \left[0, \frac{\pi}{k}\right]$. Intuitively, $\rho_\tau > 0$ accounts for the number of cycles, and every cycle is a feedback loop, therefore the linear system described by $\mathbf{W}$ has many cycles with positive feedback, so it resonates at the frequency $\frac{1}{\tau}$ and its corresponding harmonics. This resonant frequency corresponds to the phases of the dominant eigenvalues, meaning that when ρ_τ increases the eigenvalues with the phases $\frac{2\pi k}{\tau}$ for $k \in \{0, 1, ..., \tau - 1\}$ increase their moduli while those with phases $\frac{2\pi(k+1)}{\tau}$ decrease it. For $\rho_\tau < 0$, the argument is the same, but the eigenvalues that are dampened are those with phases $\frac{2\pi k}{\tau}$ and the enhanced ones correspond to phases $\frac{2\pi(k+1)}{\tau}$.

14.3 Dealing with hyperbolic tangents

Studying reservoir dynamics through the spectrum of their networks can be extremely helpful, but it ignores the fact that reservoirs must be non-linear. Thus, we might be tempted to question whether rewiring the reservoir to include cycles will enhance frequencies. We could just refer to the numerics as presented in Fig. 9, but we will instead provide an argument that will also highlight some of the problems of dealing with the specific reservoirs used in ESN, where the nonlinearity is an hyperbolic tangent as in Eq. 1. Note that this will be easily generalizable to any reservoir whose neurons have monotonic activation functions.

Naturally, it is difficult to deal with nonlinearities and large system sizes simultaneously, so in this section we will simplify the large size problem and we study how a single neuron changes its power spectral density when it is embedded in a cycle. Consider a neuron with adjacent connections whose state is defined by the equation

$$x_i(t) = \tanh\left((\mathbf{w}_{\text{in}})_i u(t) + \sum_{j \in \mathcal{S}_i} w_{ji} x_j(t-1)\right), \tag{100}$$

where $\mathcal{S}_i$ are the presynaptic neighbors of i. For simplicity, we set $u(t)$ to be a random normal variable independently sampled at every time.

Sparse random large graphs are locally tree-like [Wor99, BKM10], meaning that typically all the values of $x_j(t-1)$ are only remotely connected to x_i and among themselves. This implies that we can effectively treat the input to the neuron

$$r_i(t) = (\mathbf{w}_{\text{in}})_i u(t) + \sum_{j \in S_i} w_{ji} x_j(t-1) \tag{101}$$

as a random variable, which in our case has mean zero and bounded variance.

Now we embed the neuron into a positive cycle of finite length L, meaning that there is a sequence of $L-1$ weights $w_{ij_1},\ w_{j_1 j_2},\ w_{j_2 j_3}, ...,\ w_{j_{L-1} i}$ such that

$$w_c = w_{ij_1} \cdot w_{j_1 j_2} \cdot ... \cdot w_{j_{L-1} i} > 0, \tag{102}$$

and proceed to use the mean value theorem to $x_i(t+\tau)$,

$$\begin{aligned} x_i(t+\tau) &= \tanh'\left(\xi_i(t+\tau)\right) r_i(t+\tau), \\ r_i(t+\tau) &= w_{j_{L-1}i} x_{j_{L-1}}(t+\tau) + (\mathbf{w}_{\text{in}})_i u(t+\tau) + \sum_{k \in S_i - \{j_{L-1}\}} w_{ki} x_k(t+\tau-1) \end{aligned} \tag{103}$$

where $\text{sign}\left[\xi_i(t+\tau)\right] = \text{sign}\left[x_i(t+\tau)\right]$ and $\tanh'$ is the derivative of $\tanh$. To simplify our notation, we will write

$$s_i(t) = (\mathbf{w}_{\text{in}})_i u(t+\tau) + \sum_{k \in S_i - \{j_{L-1}\}} w_{ki} x_k(t+\tau-1), \tag{104}$$

which are randomly sampled from a symmetric probability distribution, so their expectation over all samples of $\mathbf{W}$, even including w_c, is zero. The same expansion can be applied to any node in our cycle, so for $l < L$,

$$\begin{aligned} x_{j_l}(t+\tau-l) &= \tanh'\left(\xi_{j_l}(t+\tau-l)\right) r_{j_l}(t+\tau-l) \\ r_{j_l}(t+\tau-l) &= w_{j_l i} x_{j_l}(t+\tau-l-1) + s_{j_l}(t+\tau-l). \end{aligned} \tag{105}$$

Then by recursively expanding $x_{j_l}(t-l)$,

$$\begin{aligned} x_i(t+\tau) = &\prod_{l=0}^{L-1} w_{l,l+1} \tanh'\left(\xi_{j_l}(t+\tau-l)\right) x_i(t) \\ &+ \sum_{l=0}^{L-1} s_{j_l}(t+\tau-l) \prod_{k=0}^{l} \tanh'\left(\xi_{j_l}(t+\tau-l)\right) \end{aligned} \tag{106}$$

where we used the convention $i = j_0 = j_L$. We can already see that the value of $x_i(t+\tau)$ is now coupled to the value of $x_i(t)$. We can now compute this new correlation

$$\begin{aligned} \mathbb{E}_{|w_c}\left[c_i(\tau)\right] = &\mathbb{E}_{|w_c}\left[\prod_{l=0}^{L-1} w_{l,l+1} \tanh'\left(\xi_{j_l}(t+\tau-l)\right) x_i^2(t)\right] \\ + &\mathbb{E}_{|w_c}\left[x_i(t) \sum_{l=0}^{L-1} s_{j_l}(t+\tau-l) \prod_{k=0}^{l} \tanh'\left(\xi_{j_l}(t+\tau-l)\right)\right]. \end{aligned} \tag{107}$$

By the symmetry of the parameters of the network, the sign of s_{jl} is uncorrelated with x_i if we average over all the possible values of $\mathbf{W}$, so if we set $\tau = L$ we are left with

$$\begin{aligned} \mathbb{E}_{|w_c}\left[c_i(L)\right] &= \mathbb{E}_{|w_c}\left[\prod_{l=0}^{L-1} w_{l,l+1} \tanh'\left(\xi_{j_l}(t+L-l)\right) x_i^2(t)\right] \\ &= w_c \mathbb{E}_{|w_c}\left[\tanh'\left(\xi_{j_l}(t+L-l)\right) x_i^2(t)\right]. \end{aligned} \tag{108}$$

Note that $\tanh'(x) = \left(1 - \tanh^2(x)\right) > 0$, therefore

$$\mathbb{E}_{|w_c}\left[c_i(L)\right] > 0. \tag{109}$$

This implies that by having cycles, the neurons will have, on average, a positive autocorrelation with time-delay L.

Finally, the autocorrelation function gives the PSD by the Wiener-Khinchin Theorem [Khi34, Wie30],

$$s_i(f) = \sum_{k=-\infty}^{\infty} c_i(k) e^{-2\pi i f k}, \tag{110}$$

thus if we select $f = n/L$,

$$\mathbb{E}_{|w_c}\left[s_i\left(\frac{n}{L}\right)\right] > 0, \tag{111}$$

for $n \in \mathbb{Z}$.

The gist of the argument presented here is that since neurons are filters randomly sampled from a probability distribution that is not biased towards positive or negative feedback, the average autocorrelation is zero. By adding cycles, we are forcing some of those autocorrelations to be –on expectation– different from zero, which is equivalent to changing frequencies.

An important insight from the derivation presented here is that by using the mean value theorem we loose information about the actual value of the autocorrelation. That is, while in linear systems we know *how much* the frequencies will be modified, in non-linear systems we do not, having to conform ourselves with knowing which frequencies will be enhanced or dampened.

Improving reservoirs

Now that we know how to alter the frequencies of the reservoir while keeping its memory intact, the next natural step is to apply this knowledge and design reservoirs adapted to a specific task.

Before working on the frequency adaptation it is important to consider the memory capacity of the reservoirs. As we have seen before, having cycles in the adjacency matrix change the distribution of eigenvalues, specifically creating some extreme values that are larger than in the original, non-adapted adjacency matrix [Ace18]. This obviously affects the spectral radius as well as $\langle|\lambda|\rangle$. Since we expect that the memory capacity requirement for a forecasting task is not affected by adapting the PSD, it is reasonable to use $\langle|\lambda|\rangle$ as the normalization variable for the weights of reservoir matrix, instead of the spectral radius [Jae02]. We can thus use the optimal value of $\langle|\lambda|\rangle$ from the data in Fig. 6 where $c_L = 0$ before adapting the PSD.

We start by simply taking different reservoirs with varying fractions of cycles and compare values of c_L against the performance of the reservoir in that task.

As we see in Fig. 11, the performance of ESN depends heavily on the fraction of cycles c_L. To understand them better, it is useful to compare the optimal c_L values with the PSD of the time series. A simple example is given by the Mackey-Glass time series: Fig. 12 shows that for $c_L > 0$, the reservoir's average PSD response is enhanced for the frequencies close to 0, which is exactly the regime where the spectrum of the Mackey-Glass Time Series is concentrated. Consistent with our hypothesis, positive c_L improves the ESN performance on forecasting the Mackey-Glass Time Series (Fig. 11.a, d, g), while negative c_L decreases it. Similarly, the Spoken Arabic Digits are also dominated by frequencies close to 0, and thus the performance of ESN improves when $c_L > 0$ (Fig. 11c,f,i). As for the Laser Intensity Time Series, its dominating frequencies are around 0.13, 0.27 and 0.38, thus ESN is improved when the response of the reservoir enhances those frequencies. As shown in Fig. 12.e,h, this happens when $c_L < 0$ for $L = 2, 3$. Indeed, as shown in Fig. 11e,h, negative c_L (for $L = 2, 3$) improves the ESN performance. For the case of $L = 1$, we observed in Fig. 12 that the three peaks cannot be all enhanced simultaneously by setting r_1 to be either positive or

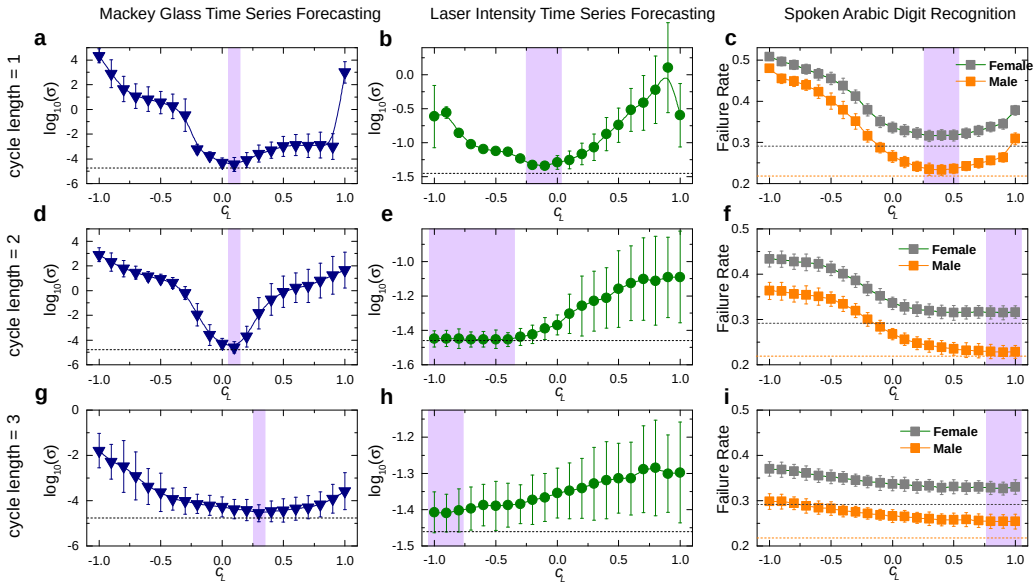


Figure 11: **Improving ESN through frequency adaptation**. ESN performance σ vs ρ_L, for the tasks of Mackey-Glass Forecasting (a, d, g), Laser Intensity Forecasting (b, e, h), and Spoken Arabic Digit Recognition (c, f ,i). The length of cycles added into the reservoir is 1 in (a-c), 2 in (d-f) and 3 in (g-i). Every point corresponds to the median performance –measured by σ from Eq.4 in the (a, b, d, e, g, h) and by the failure rate in (c, f, i)– over 200 ESN realizations with the error bars corresponding to upper and lower quartiles. The dashed lines correspond to the performances obtained for each task by creating reservoirs combining cycles of various lengths. Each ESN realization corresponds to a reservoir with $N = 1000$ neurons and average degree $\langle k \rangle = 50$ for (a) and $N = 100$, $\langle k \rangle = 10$ for (b) and (c). In all cases we see that combining cycles of different lengths can bring substantial improvements to ESN performance.

negative. Instead, setting $r_1 = 0$ would yield the optimal performance. This is what we observed in Fig11.b.

The immediate conclusion is that the reservoir should be designed to enhance the frequencies present in the target signal. A simple way of achieving this is to obtain the frequency responses of the reservoir for an unstructured signal (white Gaussian noise), and then select the parameters of the reservoir for which the frequencies match the target signal better. Based on those considerations we designed a simple heuristic algorithm to find the optimal values of c_L's

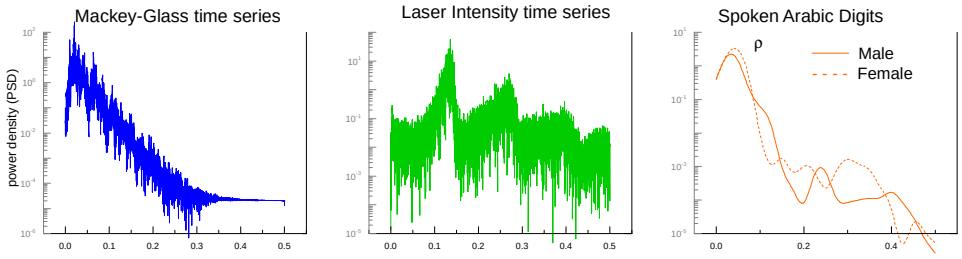


Figure 12: **Frequency plot of the Time Series**: We plot the power spectral density of the time series used. The Mackey-Glass (left) is a low-frequency signal, meaning that it evolves smoothly and thus most of its power is on low frequencies. The Laser Intensity has three frequency peaks in the middle of its spectrum, making it a comparatively high-frequency signal as observed by the fact that it has very large fast shifts. Finally, the Spoken Arabic Digits is also a low frequency signal on average.

15 Generate adapted reservoirs

In the tasks described in the main text, different tasks require different reservoir parameters. Specifically, for a given maximum cycle length value L there is one combination of ρ_l, $\forall l \leq L$ and a value of $\langle|\lambda|\rangle$, which optimizes the ESN performance. In this section we present the heuristic that we use to find those parameters.

The first step is to tune the memory for the current task. We take a very simple approach, where we simply try many spectral radii on the interval $[0, 1]$ and pick the one that gives the best performance on a classical ER network. Once found, we calculate the corresponding $\langle|\lambda|\rangle$ and we will use that for the rest of the process (see Supplementary Information Sec. I).

Since the reservoirs that we use are not linear, we need to characterize their frequency response for various values of ρ_l for all the $l \leq L$ considered. This response, that we denote $\hat{R}(\rho_l, L)$ is computed by generating Gaussian noise with the same variance and mean as the original signal and use it as an input for the reservoir. Then apply the Fast Fourier Transform to the neurons' states and average over all neurons. As the reservoirs are generated randomly, it is necessary to average those responses over multiple reservoir instances. For a given L we use the following heuristic to find the optimal combination of cycles with size no greater than L.

Step-1: Compute the Fourier transform of the input signal and keep the vector of absolute values $\hat{s} = [\hat{s}(0), \hat{s}(2), ...\hat{s}(f_S)]$, where f_S is half the sampling frequency.

Step-2: Compute the scalar product $\langle \hat{s}, \hat{R}(\rho_l, l) \rangle$ for all ρ_l, l, and select the ρ_l that maximizes it for each l.

Step-3: Test the performance of an ESN with the values of ρ_l found in the previous step. If the performance is lower than in the default case of $\rho_l = 0$, do not optimize with regard to that length.

Step-4: For all values of ρ_l where the cycle length is allowed and which fill the condition $\|\rho\|_1 \leq 1$, select the one that maximizes $\sum_l \langle \hat{s}, \hat{R}(\rho_l, l) \rangle$.

As shown in Fig. 11 (black dashed lines), this heuristic does indeed further improve the ESN performance. Intuitively, the explanation for our approach is that a reservoir is a set of coupled filters that extract features from the input signal, and our heuristic simply selects filters that focus on the family of features that are present in the signal, thus increasing the ability of the reservoir to capture relevant information.

16 Discussion

The heuristic and network generation algorithms proposed improve the performance of an ESN beyond that of traditional reservoirs with relatively simple and fast procedures. This has obvious advantages from an engineering point of view, as we can obtain better performances with smaller reservoirs for any task in which an Echo State Network might be used. This is particularly interesting for the neuromorphic computing community, as they typically resort to reservoir computing to implement learning algorithms in their chips [TRA+17, VdSBS17, TYH+19].

Our approach is in stark contrast to previous results in which the reservoir network was selected to cover uniformly all the frequency spectrum [OXP07], creating reservoirs that generalize well across multiple tasks, but failing to specialize in a particular one. Note that adding cycles of length one is similar to the Leaky reservoirs used previously [JLPS07b], but here we show that those reservoirs are not better simply because they increase the memory –as this could had been achieved by scaling the weight matrix – but rather because the leaky term adapts the neurons to the dynamics of the time series. Thus our results provide a theoretical backbone and a direct generalization of leaky reservoirs.

An important limitation of the work outlined here is that the cycles cannot have arbitrary long lengths, because the frequency enhancement becomes weaker and weaker with growing L (see Fig. 12). This comes from the random matrix theory finding, as high ρ_L are difficult to generate without lowering the average degree to a point where the network would be disconnected [ARS19].

However, a recent line of work has expanded leaky reservoirs to multi-layer architectures [GMP18,GMP17], finding that adding layers can refine low-frequency adaptation, just as stacking up filters can further refine frequency adaptation. While those studies relay on the PSD to adapt the reservoir architecture, they focus on obtaining rich reservoir dynamics and low-frequency features, ignoring possible intermediate or high frequencies as well as the supervised frequency selection that we use here. Thus an encouraging continuation of this work would be to use hierarchical architectures to have reservoirs that can be adapted in a supervised way to very precise PSDs.

It is important to note that we are not advocating for hand-tuning reservoir topologies for specific tasks, but rather to raise the point that notions from classical signal processing can help us understand and improve recurrent neural networks, either through selection of appropriate initial topologies in a pre-training stage, or by designing learning algorithms that account for the principles outlined here. Given that most current learning strategies such as backpropagation focus on adapting single weights, we are convinced that many new learning algorithms can be created by focusing on network-level features; after all the heuristic presented here is extremely simple and could be refined in multiple ways.

Part III

Self-Organized Activity in Spiking Neural Networks

Spiking Neural Networks and Spiking Time-Dependent Plasticity

Knowing how to design neural networks that are tuned to a particular task does not necessarily tell us anything about biological neural networks. Indeed, we would not expect that the heuristic and network algorithms that we proposed and used are implemented by biology, as this would require knowing the task at hand before wiring the networks. However, the principles behind them might still provide us insight into how biology adapts, and in the following chapters we will show how some of the notions used in signal processing can be implemented by well-known biological rules. Before digging into that problem it is necessary to understand the models and assumptions underlying biological neural networks, which is the topic of this chapter.

17 Spiking neural networks

The main feature of a neuron is the capacity to receive information through electrical impulses, combine this information and send impulses to other neurons. However, the neurons – or neuron models – that neuroscientists and machine learning engineers use are very different. While in machine learning neurons are nodes in a network that take a linear combinations of their inputs and apply a nonlinear function at every time step [Ros57, Bis95], biological neurons are very complicated cells that can have extremely rich dynamics [HH53, Izh04] and they behave continuously in time without an actual clock to tell them when to process or not to process information.

Since real biological neuron dynamics can be very complicated, the study of biological networks of neurons often starts by making some simplifying assumptions [HGMJ06, BHR14]. A common starting point is to assume that neurons are individual units with a single state variable whose state depends only on their direct inputs from other neurons, and they communicate through short discrete signals called spikes that are sent to other neurons when a certain value of their state variable is reached.

This still gives us a whole family of possible models, depending on how the activity of neurons at some point in time is described – either as a instantaneous rate or a collection of individual spikes –. In this thesis we will oscillate between those cases depending on the precise problem at hand; when modeling how neural populations change their activity we will consider instantaneous rates and compute averaged quantities, while the study of how single neurons modify their spike trains will relay on individual spikes. This does not mean that the neurons considered would be different depending on our problem, but rather that they correspond to different levels of description. In fact, our simulations will all be done with the same neuron model and the transition between different levels of description will be made explicit when necessary.

To be more precise, the model that we will use for simulations and some of our analysis will be the Leaky Integrate and Fire (LIF) model with a refractory period [Lap07b]. In this model, the state of a neuron at a given time is described by its membrane potential $v(t)$, which evolves according to the equation

$$\tau_m \frac{dv(t)}{dt} = -(v(t) - v_0) + u(t), \tag{112}$$

where $\tau_m = 10ms$, $v_0 = -70mV$. $u(t)$ is the input to the neuron at time t. When the membrane potential reaches a certain threshold $v_{th} = -50mV$, the neuron "fires" or "spikes", meaning that it emits a pulse of current – the spike – that will be sent to other neurons in the form of a delta function. After firing, the membrane potential is reset to its resting state v_0 and kept frozen at this value for a fixed period of time called the refractory period $t_{ref} = 1ms$.

The firing of a neuron generates pulses of current that arrive at other neurons, which in turn update their membrane potentials. If neuron a receives the spikes of neuron b we will say that there is a synapse going from the second to the first. The receiving neuron is called postsynaptic and the sending neuron is the presynaptic one. This synapse is characterized by a weight w_{ab} and a delay d_{ab} which correspond, respectively, to the gain and the latency that the pulse of neuron a goes through before arriving at b.

At this point it is useful to discuss the input $u(t)$. As mentioned before, the input to a neuron is often given by spikes from other neurons, which would give us

$$u(t) = \sum_k w_k \delta(t - t_k), \tag{113}$$

where w_k is the weight of the kth spike and t_k its arrival time to the neuron. Note that we can also aggregate small contributions from many other neurons forming a smooth function of time or a stochastic variable. In this case, $u(t)$ might be better described by the statistics of this variable, such as the expected value and the variance.

Finally, we shall note that the activity of neuron populations does not need to be described by a set of spike times. In many cases we are interested in a coarser description and then it is better to use variables such as the instantaneous firing rate – the average number of spikes per time unit – instead the exact spike times [GKNP14]. Furthermore, if we are considering time scales much larger than τ_m, the exact evolution of the inner neuron state is irrelevant and we can focus on the coarse neuron activity which is given by

$$x(t) = f\left(u(t)\right), \tag{114}$$

where f is a smooth function with lower bounded by zero – meaning that the neuron cannot fire less than zero spikes – and upper bounded by $\frac{1}{t_{ref}}$, because the neuron needs $1ms$ between spikes.

18 Synaptic Plasticity

One of the most fascinating feats of the nervous system is its ability to learn. However, in contrast to the supervised, error-based learning that we took in the first part of this thesis the networks of neurons that we observe in biology evolve in an unsupervised way, meaning that they do not have an explicit notion of error, but rather adapt to the input they receive and their inner dynamics. In this section we will present some of the most common rules and models governing this adaptation.

18.1 Spike Time-Dependent Plasticity

Despite the rate-centered approach that we –and many others– use to analyze network dynamics, the fact that neurons communicate by spikes with precise times has crucial consequences in terms of adaptation. The most prominent rule that accounts for this is known as Spiking Time-Dependent Plasticity (STDP) [SG10, GKvHW96].

In STDP the weight of a connection is modified depending on the time interval between pairs of pre- and post-synaptic spikes. For every pair the weight of the synapse is modified according to the equations

$$\Delta_{STDP} w(\Delta t) = \begin{cases} A_+(w) e^{-\frac{|\Delta t|}{\tau_s}} & if\ \Delta t \geq 0 \\ A_-(w) e^{-\frac{|\Delta t|}{\tau_s}} & if\ \Delta t < 0 \end{cases} \tag{115}$$

where $\Delta t = t_{post} - t_{pre}$ is the time difference between the postsynaptic spike and the presynaptic one. The constant τ_s establishes the timescales of the STDP and is typically very short in comparison to *behavioral* timescales, meaning that

most processes involving STDP happen over long time intervals. The coefficients $A_+ > 0$ and $A_- < 0$ are coefficients that determine the relative strengths of the Long Term Depression (LTD) and Long Term Potentiation (LTP) processes, and both of them are very small so that the STDP works on very long timescales. In subsequent sections we will refer to $A_\pm$ to indicate both A_+ and A_- depending on the sign of Δt.

Stability

As many other rules used in neuroscience, STDP has the problem of being unbounded and potentially unstable [DA01]. That is, we need to have some mechanism that prevents synaptic weights from growing to infinity.

We will take two different approaches to this. When we are considering neurons that get input spike trains whose origin is not relevant, we will only require the weights to be upper bounded. This is the simplest approach to study biologically plausible input-output function for neurons. The model is based on previous works [KH00, VRBT00], and consist on taking A_+ and A_- as

$$\begin{aligned} A_+(w) &= \eta_+(w_{\max} - w), \\ A_-(w) &= -\eta_-(w - w_{\min}). \end{aligned} \tag{116}$$

However, sometimes we have large networks of neurons with recurrent synapses and then our goal is not only to ensure a finite synaptic weight, but also to have stable neural activity. This can be done by different means, for example by having an homeostatic term [TN04] of the form

$$\Delta_H w = -\gamma(r)w \tag{117}$$

where γ is a monotonically increasing positive function of r, the firing rate of the neuron. In our simulations this will be a simple linear dependency, but other rules are possible. Note that this term also keeps the weights finite, so we also obtain the weight stability.

Full Plasticity Model

When we integrate all the previous components we end up with the following model which is valid for a the time interval $[t, t+T]$,

$$\Delta w = -\gamma\left(\frac{S_{post}}{T}\right) + \sum_{s_{pre}}^{S_{pre}} \sum_{s_{post}}^{S_{post}} A_\pm(w) e^{-\frac{|\Delta t|}{\tau_s}} \tag{118}$$

where $S_{pre},\ S_{post}$ are the number of pre- and post-synaptic spikes in the interval $[t, t+T]$ respectively. Note that in any case we will assume $|\Delta w| \ll 1$, so that

the evolution of the weight in any single input presentation is small and the weight evolves smoothly.

Furthermore, depending on the case at hand we will use different parameter regimes. When considering recurrent neural networks we will be concerned with neuron rates but we can ignore weight saturations, hence we will work on the regime where $\gamma(r) > 0$ and $A_+(w)$ and $A_-(w)$ are constant. Conversely, when we consider single neurons getting a train of spikes without recurrent connections we will not be concerned with rates directly so that $\gamma(r) \sim 0$, but add a saturation term to $A_\pm(w)$.

18.2 Excitation-Inhibition

We must note that the input need not to be always positive. Many neurons – and their corresponding synapses – are inhibitory, meaning that they lower the membrane potential of the target neuron, reducing the chance that it will fire.

This can have a variety of strange effects on the dynamics of a neural network. For instance, it might be crucial to impose oscillatory regimes [PSvRN13], or shape receptive fields [SG02, DA01]. More recently it has been shown that the balance between excitation and inhibition is a key factor in explaining the irregular firing that biological neurons exhibit [Bru00]. Thus in studies we should be aware of the parameter regime in which we are working. In this section we will show that for our purposes, any parameter regime in which excitation is either stronger than inhibition or precisely balanced, thus covering a wide range of possible regimes.

LIF neurons subject to excitatory or balanced inputs

For now we will consider $u(t)$, the input variable to a single neuron as a stationary stochastic variable. In particular, for a single leaky integrate and fire neuron, the output firing rate is given by the time it takes for the membrane potential to cross the threshold. The average inter-spike-interval decreases as the mean of $u(t)$ increases – meaning that excitation is larger than inhibition – or, if the mean is kept at zero – meaning a balanced regime –, if the variance of $u(t)$ increases. To illustrate this we simulated a LIF neuron subject to a Gaussian input in Fig. 13. The firing rate starts at zero, then starts to grow and eventually saturates and it does so for both the mean and variance of $u(t)$. This implies that we can use the so-called balanced networks [Bru00], where the amount of inhibition is similar to the amount of excitation, and still model the activation of networks of neurons by a monotonic function bounded from above and below.

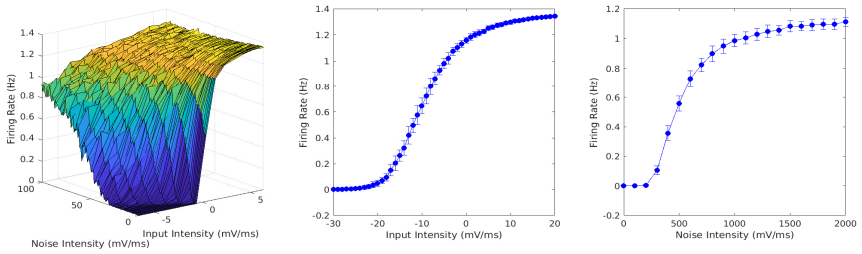


Figure 13: **Leaky Integrate-and-Fire as a smooth function**: Here we plot the firing rate of a LIF neuron as function of the parameters of a Gaussian input. The left plot presents the average of firing rate for an input presented for 5 seconds and with the two variables being the mean and standard deviation of a Gaussian input. The middle and right plots correspond to cuts through that surface on the respective directions, with the center one being the mean firing rate for a standard deviation of $40V/s$ and the right one corresponding to varying levels of noise for a mean input of $-5V/s$.

Balance-preserving plasticity

The STDP rule presented before is well characterized and widely used, but mostly for excitatory neurons. However, in biological systems, inhibitory synapses are also present and have plasticity [VFD$^+$13]. Naturally, this might compromise the effects of STDP, as an inhibitory synapse that gets potentiated could counteract the effects of excitatory STDP. For instance, it might decrease the membrane potential and thus increase the latency of the postsynaptic neuron [EJL15]. Our goal in this section is to find the parameter regime in which the presence of inhibitory plasticity gives qualitatively the same effect for the STDP. Specifically, we will try to find the parameters of STDP such that the probability of generating a postsynaptic spike at time t increases if that time is shortly before a postsynaptic spike and decreases if that time is shortly after a postsynaptic spike.

Intuitively, in a network with random connectivity for both inhibitory and excitatory the mean inputs will still follow –on average – the potentiating tendency of excitatory STDP as long as the STDP in inhibitory synapses is weaker than the STDP in excitatory ones. The question is then to find a way of measuring "how much weaker" it has to be. Following the observation from Fig. 13 that the activity of a neuron will increase either by increasing the mean or the variance of the input, we will find parameters such that the inhibitory and excitatory STDP cancel each other. This is in line with the balanced neural networks that have been shown to reproduce biological observations [Bru00].

To maintain this balance, the potentiation of excitatory synapses must be compensated by the potentiation of inhibitory synapses. Potentiating all synapses but

maintaining the average input leads to the increase in fluctuations of the membrane potential, meaning that the membrane potential preceding a postsynaptic spike would change more around the average.

The approach that we will take is to start by assuming a fixed postsynaptic spiking time and then we will study how the presynaptic input is modified by STDP. This will lead to modified average membrane potentials, and thus modified spike trains.

Consider a single postsynaptic spike at time t_{pos}. For simplicity we will only treat the membrane potential at $t < t_{post}$, as the case where $t > t_{post}$ case follows the same logic but with depression instead of potentiation. This membrane potential is given by

$$v(t) = \sum_{t_k<t} w_k e^{-\frac{t-t_k}{\tau_m}}, \tag{119}$$

and for now $v(t) < v_{th}$. Now we wonder what happens when the weights w_k change, by using only information about the distribution from which the list of (w_k, t_k) was sampled.

The subsequent step is to find the conditions that guarantee that $\mathbb{E}[\Delta_r v(t)]$ increases. A sufficient condition for this to happen is to have

$$\mathbb{E}[\Delta_r u(t)] = \Delta_r \mathbb{E}[u_e(t)] - \Delta_r \mathbb{E}[u_i(t)] > 0, \ \forall t < t_{post} \tag{120}$$

where $\mathbb{E}[u_e(t)]$ is the expected input to the neuron at time t, and $\mathbb{E}[u_i(t)]$, $\mathbb{E}[u_i(t)]$ is simply its decomposition in inhibitory and excitatory inputs, which gives us

$$\begin{aligned} \mathbb{E}[u_e(t)] &= \rho_e \int_0^\infty \mu_{w_e}(w,t)dw \\ \mathbb{E}[u_i(t)] &= \rho_i \int_0^\infty \mu_{w_i}(w,t)dw \end{aligned} \tag{121}$$

where ρ_e, ρ_i are the rates of incoming spikes and $\mu_{w_e}(w,t), \mu_{w_i}(w,t)$ the probabilities of the weights associated to time t.

Thus, to maintain the condition from Eq. 120 we must ensure that the parameters μ_{w_e}, μ_{w_i}, η_+^e, η_+^i, $w_{\min}^e$, $w_{\min}^i$ are such that

$$\rho_e \int_0^\infty \Delta w_e(r)\mu_{w_e}(w,t)dw > \rho_i \int_0^\infty \Delta w_i(r)\mu_{w_i}(w,t)dw, \tag{122}$$

where $\Delta w(r)$ are given by the STDP Eq. 115 over many repetitions –counted by r – of the input spike train. We will now find a parameter regime in which this holds by finding its boundary. In other words, we are interested on the parameter set in which

$$\rho_e \int_0^\infty \Delta w_e(r)\mu_{w_e}(w,t)dw = \rho_i \int_0^\infty \Delta w_i(r)\mu_{w_i}(w,t)dw. \tag{123}$$

Note that it is not enough to find two weight distributions μ_{w_e}, μ_{w_i} where

$$\rho_e \int_0^\infty A_+^e(w_e)\mu_{w_e}(w,t)dw = \rho_i \int_0^\infty A_+^i(w_i)\mu_{w_i}(w,t)dw, \tag{124}$$

because this would only work for the first input repetition. We have to ensure that even after STDP changes the distribution, the equality holds. A simple way to achieve this is to set the inhibitory and excitatory parameters to be equal. It is obvious that if the probability distributions of weights, input rates and STDP parameters are the same, then the change in input will affect the inhibitory and excitatory synapses in the same way. However, we know that this is not the case, as there are typically fewer inhibitory synapses than excitatory ones. Thus, we modify this symmetry to include rescaling, meaning that we have the ratio

$$\alpha = \frac{\rho_i}{\rho_e} \tag{125}$$

that is also intrinsic to the probability distributions

$$\alpha\mu_{w_i}(\alpha x, t) = \mu_{w_e}(x,t)\ \forall x, t. \tag{126}$$

and the STDP parameters

$$\alpha A_+^i(\alpha x) = A_+^e(x)\ \forall x. \tag{127}$$

By a simple change of variable we can show that, if those properties are satisfied,

$$\begin{aligned}\rho_e \int_0^\infty A_+^e(x)\mu_{w_e}(x,t)dx &= \frac{1}{\alpha}\rho_i \int_0^\infty \alpha A_+^i(\alpha x)\alpha\mu_{w_i}(\alpha x,t)\frac{1}{\alpha}d(\alpha x) \\ &= \rho_i \int_0^\infty A_+^i(y)\mu_{w_i}(y,t)dy.\end{aligned} \tag{128}$$

Furthermore, if we take a pair of inhibitory and excitatory weights such that $w_e = \alpha w_i$ we have that after STDP,

$$\begin{aligned}\alpha w_i \to \alpha(w_i + A_+^i(w_i)e^{-\frac{|t-t_{post}|}{\tau_s}}) &= \alpha w_i + \alpha A_+^i(\alpha w_e)e^{-\frac{|t-t_{post}|}{\tau_s}} \\ &= w_e + A_+^e(w_e)e^{-\frac{|t-t_{post}|}{\tau_s}} \leftarrow w_e,\end{aligned} \tag{129}$$

meaning that the weight probability changes in such a way that

$$\begin{aligned}&\mu'_{w_e}\left(x + A_+^e(x)e^{-\frac{|t-t_{post}|}{\tau_s}}, t\right) = \mu_{w_e}(x,t) \\ &= \alpha\mu_{w_i}(\alpha x, t) = \alpha\mu'_{w_i}\left(\alpha\left(x + A_+^i(x)e^{-\frac{|t-t_{post}|}{\tau_s}}\right), t\right),\end{aligned} \tag{130}$$

where μ'_{w_e} and μ'_{w_i} are the weight distributions after STDP has acted once. Thus, if Eq. 126 holds at some point, it will also hold for all subsequent iterations of the input spike pattern.

Thus, we have found a set of conditions that satisfy Eq. 126 at $r = 0$ and for any subsequent $r > 0$ for the case where the postsynaptic spike does not change during r repetitions.

Notice that the self-consistency of this condition does not make any assumptions about the learning constant or Δt dependent term on STDP, it only requires that the expected increase in excitatory input is matched by the expected increase in inhibitory input.

The same argument also applies when the postsynaptic spike advances. Consider the change of weights at time t with a postsynaptic spike initially triggered at time t_{post} for r' repetitions, and then changes in the presynaptic spike train change the firing time so that the postsynaptic spike is triggered at t'_{post} with $t < t'_{post}, t_{post}$. In that case the shape of Eq. 129 remains the same for every repetition, is only that the coefficient of change will be $e^{-\frac{|t-t_{post}|}{\tau_s}}$ for $r < r'$ and $e^{-\frac{|t-t'_{post}|}{\tau_s}}$ for $r \geq r'$.

Now we have a large set of parameters in which latency reduction is expected to happen. Any STDP parameters for which $\alpha A^i_+(\alpha x) < A^e_+(x)$ combined with Eq. 126, or distribution of weights with $\alpha\mu_{w_i}(\alpha x, t) < \mu_{w_e}(x, t)$ with Eq. 127, or both cases combined.

It is worth noticing that the case when all the equalities Eq. 126 and Eq. 127 are met we would still expect the latency to decrease. The reason is that even if

$$\mathbb{E}\left[\Delta_r v(t)\right] = \mathbb{E}\left[\Delta_r u(t)\right] = \Delta_r \mathbb{E}\left[u(t)\right] = 0, \tag{131}$$

the variance of $v(t)$ increases. More explicitly,

$$\begin{aligned} \Delta_r \text{Var}\left[v(t)\right] &= \Delta_r \int_{-\infty}^{t} \text{Var}\left[u(t)\right] dt \\ &= \Delta_r \int_{-\infty}^{t} \mathbb{E}\left[u^2(t)\right] - \Delta_r \int_{-\infty}^{t} \mathbb{E}\left[u(t)\right]^2 dt, \end{aligned} \tag{132}$$

and since $\Delta_r \mathbb{E}\left[u(t)\right] = 0$, we can write

$$\begin{aligned} \Delta_r \text{Var}\left[v(t)\right] &= \int_{-\infty}^{t} \Delta_r \mathbb{E}\left[u^2(t)\right] dt \\ &= \int_{-\infty}^{t} \Delta_r \mathbb{E}\left[u_e^2(t)\right] dt + \int_{-\infty}^{t} \Delta_r \mathbb{E}\left[u_i^2(t)\right] dt \end{aligned} \tag{133}$$

where the term $\mathbb{E}\left[u(t)\right]^2 = 0$ by the symmetry of the weights and it is maintained at zero by the symmetry of the STDP. Since we are only concerned with $t < t_{post}$,

STDP potentiates both inhibitory and excitatory synapses, so

$$\Delta_r \mathbb{E}\left[u_i^2(t)\right], \Delta_r \mathbb{E}\left[u_e^2(t)\right] > 0 \tag{134}$$

and therefore the variance increases.

Notice that the conditions stated do can also be used for $t > t_{post}$, as it suffices to set

$$\alpha A_-^i(\alpha x) = A_-^e(x)\ \forall x. \tag{135}$$

and replace all $A_+^e,\ A_+^i$ for $A_-^e,\ A_-^i$ in the subsequent equations and they would still hold. Naturally the difference is that the synapses get depressed rather than potentiating, but the equilibrium condition is kept. This implies that $\Delta_r \mathbb{E}\left[u(t)\right] = 0$ and

$$\Delta_r \mathbb{E}\left[u_i^2(t)\right], \Delta_r \mathbb{E}\left[u_e^2(t)\right] < 0, \tag{136}$$

which is still in agreement with the general principle of STDP, namely to increase the firing probability before a postsynaptic spike and decrease it afterwards.

Naturally, if the variance of a certain distribution increases, then the probability of reaching a value higher than some threshold $-v_{th}-$ also increases, which is precisely what we observed in Fig. 13.

The approach outlined here can be also used for other STDP kernels. While the symmetry in the excitatory and inhibitory STDP kernels might not exist for some choices of inhibitory and excitatory plasticity, the approach can, in principle, still be used by simply adding a bias on the number or weight of excitatory synapses that would compensate the asymmetry in the STDP kernel.

Self-Organized Signal Enhancement

From Sec. 15 in Chapter II we know that adapting the network structure to enhance the frequencies present in the reservoir would be advantageous from a machine learning perspective. In a more general setting, we would think that exposing a network to a signal should make the network represent the signal better. In this section we will show that these phenomena of signal improvement can be implemented with the well-known rules presented before using simulations and an analytical approach based on two self-consistency equations relying neural activity and its weights.

As a introductory example, we can consider a single neuron with two autapses – synapses to the neuron itself – with different delays presented in Fig. 14. One of the autapses has a delay of $100ms$ while the other has a delay of $200ms$, on the orders of magnitude od non-myelinated central axons [SW12]. If the neuron is externally forced to fire every $200ms$, the autapse with delay two has its presynaptic spikes coincide with postsynaptic ones, and thus will get reinforced, while the one with delay of one second would not be affected, on account that the pre- and postsynaptic spikes are too far apart to undergo plasticity. This leads to a neuron with a very strong autapse with delay two, which is effectively a resonant dynamical system with frequency $\frac{1}{2}\ Hz$.

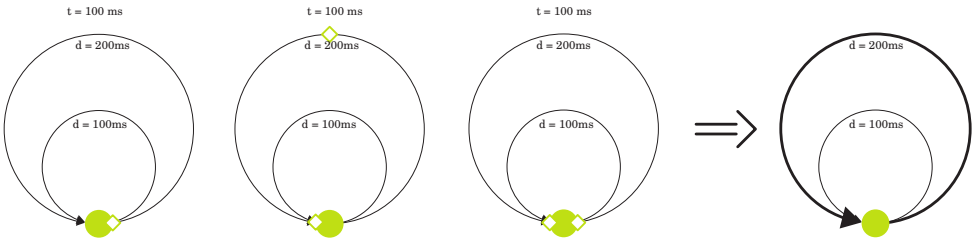


Figure 14: **Schema of the emergence of resonances**: A minimal example of a neural network that induces resonances. The three leftmost schemas show the evolution movement of a spike through the autapses and the rightmost plot the resulting structure of the network. A neuron

The previous schema is simple to understand but biologically implausible; synapses are typically very fast and autapses are extremely rare, so we will instead use large networks of neurons where the spikes will travel through a large network with short synapses instead of few large synapses. This will require having a network where neurons are active at different points in time, and thus the receptive fields of the neurons must correspond to different phases of the input.

To ease our computations we will use periodic signals with symmetries among neurons and only excitatory synapses. Specifically, we will have the external input being

$$u_n(t) = u_m(t + \Delta t_{mn}) = u_n(t + T) \tag{137}$$

where T is the period of the signal, and Δt_{mn} simply encodes the difference of timing between neurons n and m. Alternatively, we can think of this external input to every neuron as a periodic signal with neurons having different receptive fields.

This external input, which was our signal in Section 6 modifies the activity of the neurons with the equation

$$x_n(t) = f\left(u_n(t) + \sum_{m=1}^{N} \mathbf{W}_{mn} x_m(t)\right) \tag{138}$$

where f is a monotonically increasing function with upper and lower bounds as shown in Fig. 13. This is just a coarse representation of the LIF neurons that represents the probability of a spike at time t –sometimes called the instantaneous firing rate– when the time scale of the input is much larger than the time scale of the neuron.

18.3 Signal-To-Noise Ratio

Before looking into the STDP adaptation we shall clarify what we would consider an improvement of the input representation. The main idea behind the geometric bound derived to adapt ESN to a task and is associated heuristic is to make the network activity reflect the features of the input while keeping the nonlinear variations that make the reservoir useful. Focusing on the linear part turns out to give relevant improvements on ESN performance, thus we will show how this is implemented by STDP. Specifically, we will take the Signal-to-Noise Ratio interpretation of Eq. 79, meaning that we are looking at how much the neural activity grows with respect to the background noise.

The ratio can grow through two mechanisms: by increasing the activity, with constant background noise, or by decreasing the background noise. In either case, it is worth noticing that the shape of the input should be maintained, as otherwise

the nonlinearity of the reservoir would be also modified – and it is not immediately clear how or when that is a good thing.

19 Derivation of self-consistency equations

Given the activities of two neurons we can then apply the STDP equation, which gives us

$$\Delta w_{mn} = -\gamma(r_n) w_{mn} + \int_{-\infty}^{\infty} \int_{0}^{T} x_m(t) x_n(t + \Delta t) A_{\pm}(w_{mn}) e^{\frac{\Delta t}{\tau_S}} dt d\Delta t. \quad (139)$$

Note that this equation has some parameters that change as the weights evolve, such as the rate r_n or the activities of the neurons. Here we are not interested in the evolution of those, but rather in their final values, so we will treat this equation together with Eq. 138 as a system at equilibrium. This implies that r_n is constant, $x_n(t)$ is fixed and Δw_{mn} is zero, thus

$$\mathbf{W}_{mn} = \frac{1}{\gamma(r_n)} \int_{-\infty}^{\infty} \int_{0}^{T} x_m(t + \Delta t) x_n(t) A_{\pm}(w_{mn}) e^{-\frac{|\Delta t|}{\tau_S}} dt d\Delta t. \quad (140)$$

Furthermore, since we have the $\gamma(r_n)$ homeostatic term we do not need to have bounds on the weights, so we will drop that dependency on $A_{\pm}$.

If we notice that the timescale of the STDP is very small so that $\tau_S \to 0$ we can simplify the integral

$$\begin{aligned} &\int_{-\infty}^{\infty} x_n(t + \Delta t) A_{\pm} e^{\frac{\Delta t}{\tau_S}} d\Delta t = \\ &\quad \int_{-\infty}^{0} x_n(t + \Delta t) A_{-} e^{\frac{\Delta t}{\tau_S}} d\Delta t + \int_{0}^{\infty} x_n(t + \Delta t) A_{+} e^{-\frac{\Delta t}{\tau_S}} d\Delta t \\ &\quad \approx \kappa_S x_n(t) + \kappa_A \dot{x}_n(t) \end{aligned} \quad (141)$$

where $\dot{x}_n$ is the derivative of x_n and the constants κ_A and κ_S are constants that account for the symmetric and asymmetric parts of the STDP kernel,

$$\begin{aligned} \kappa_A &= \int_{0}^{\infty} A_{+}(w_{mn}) e^{-\frac{\Delta t}{\tau_S}} d\Delta t + \int_{-\infty}^{0} A_{-}(w_{mn}) e^{\frac{\Delta t}{\tau_S}} d\Delta t \\ \kappa_S &= \int_{0}^{\infty} A_{+}(w_{mn}) e^{-\frac{\Delta t}{\tau_S}} d\Delta t - \int_{-\infty}^{0} A_{-}(w_{mn}) e^{\frac{\Delta t}{\tau_S}} d\Delta t, \end{aligned} \quad (142)$$

and therefore we can compute the final weights by

$$\begin{aligned} w_{mn} &= \lfloor \frac{1}{\gamma(r_n)} \int_{0}^{T} x_m(t) \left[\kappa_S x_n(t) + \kappa_A \dot{x}_n(t)(t) \right] dt \rfloor \\ &= \lfloor \beta_S \langle x_m, x_n \rangle + \beta_A \langle x_m, \dot{x}_n \rangle \rfloor \end{aligned} \quad (143)$$

where $\beta_S,\ \beta_A$ correspond to the values of $\kappa_S,\ \kappa_A$ normalized by $\gamma(r_n)$, and $\langle , \rangle$ corresponds to the correlation over the interval $[0, T]$. The brackets $\lfloor \cdot \rfloor$ denote the fact that the weights cannot be negative; even if we were to use balanced networks with inhibitory synapses, the weights would simply decay to zero.

At this point it is convenient to modify the notation and associate to every neuron a spatial variable θ such that we can write the input as

$$u_n(t) = u(t, \theta_n) = u(t - c\theta_n) \tag{144}$$

where $u(t - c\theta_n)$ is a periodic function that corresponds to the input with different delays due to the neurons position.

Naturally this leads to a similar formulation in terms of the neuron activity,

$$x_n(t) = x(t, \theta_n) = x(t - c\theta_n). \tag{145}$$

Here it is important to note that by using this type of notation where the position is only relevant in terms of time we are imposing a spatial symmetry that must be justified.

In a network where the neurons have different connections, we cannot claim that the activity of one neuron is equal to the activity of another neuron with a shift. This is simply because their inputs from other neurons might differ. Thus, what we are stating here is that not only do neurons obtain the same input with a phase change, but also that the matrix $\mathbf{W}$ is built so that

$$\mathbf{W}_{mn} = \mathbf{W}_{m'n'} \quad \forall m - n = m' - n' \mod N. \tag{146}$$

Here we shall note that most biological networks of neurons are sparse, meaning that most pairs of neurons are not connected. In small networks, this would imply that the input to a single neuron is not given by its expectation, as we will assume here. The main justification here is the sheer number of connections that every neuron has, which is typically assumed to be on the order of 10.000 [HA16]. This implies that we can be fairly sure the input to a neuron follows the law of large numbers.

Furthermore, the number of neurons can also be assumed to be large, implying that even if the phases θ_n are randomly drawn from a uniform probability, the whole phase space will be uniformly covered, again by the law of large numbers.

Another side effect of the symmetries imposed is that we can now describe the activity of the whole network by a single equation

$$x(t) = f\left(u(t) + \int_0^T \mathbf{w}(\Delta\theta) x(t + c\Delta\theta) d\Delta\theta\right). \tag{147}$$

Notice that we are talking about a periodic input, and the phase of each neuron is only given by the phase of that neuron in temporal units that are arbitrary. Thus, we will set $c = 1$ and now we can exchange the integral by a convolution, so that

$$x(t) = f\left(u(t) + \left[x * \mathbf{w}\right](t)\right) \tag{148}$$

where $*$ is the convolution operator.

We can also use our new notation and the normalization of c to rewrite Eq. 143

$$\begin{aligned}\mathbf{w}(\Delta\theta) &= \beta_A \int_0^T x(t)\dot{x}(t+\Delta\theta)dt + \beta_S \int_0^T x(t)x(t+\Delta\theta)dt \\ &= \beta_A \left[x * \dot{x}\right](\Delta\theta) + \beta_S \left[x * x\right](\Delta\theta).\end{aligned} \tag{149}$$

Eq. 148 and Eq. 149 determine the final weights and activity of a neural network subject to the input $u(t)$ after STDP has modified the synapses.

Now we would like to know if they can implement a network adaptation such as the ones presented in Section 15 given an input signal $u(t)$.

20 Solution I: Linearization and Sinusoids

The first problem in Eq. 148 is that we have the non-linear function $f(\cdot)$, rendering closed-form solutions complicated. The first approach to deal with such situations is to linearize Eq. 148,

$$x(t) = \varrho\left[u(t) + \int_0^T \mathbf{w}(\Delta\theta)x(t + c\Delta\theta)d\Delta\theta\right] \tag{150}$$

where ϱ is the derivative of f around the mean input to a single neuron,

$$\varrho = \frac{\partial f}{\partial u}\left(\bar{u} + \bar{x}\int_0^T \mathbf{w}(\Delta\theta)d\Delta\theta\right) \tag{151}$$

with $\bar{x},\ \bar{u}$ are the mean activity and input respectively. Note that the derivative of f is taken with respect u, although it can be taken with respect to any of the inputs to a neuron. The means can be used because the linearization implies that the fluctuations are small, but in any case the main point is that ϱ is constant.

Now that the system consists only of linear equations with convolutions whose solutions are periodic, we can notice that all the operations involved can be written in the Fourier domain,

$$\begin{aligned}\mathcal{F}\left[x\right](\omega) &= \varrho\left(\mathcal{F}\left[u\right](\omega) + \mathcal{F}\left[x\right](\omega)\mathcal{F}\left[\mathbf{w}\right](\omega)\right) \\ \mathcal{F}\left[\mathbf{w}\right](\omega) &= \left(\beta_A\omega + \beta_S\right)\mathcal{F}\left[x\right]^2(\omega),\end{aligned} \tag{152}$$

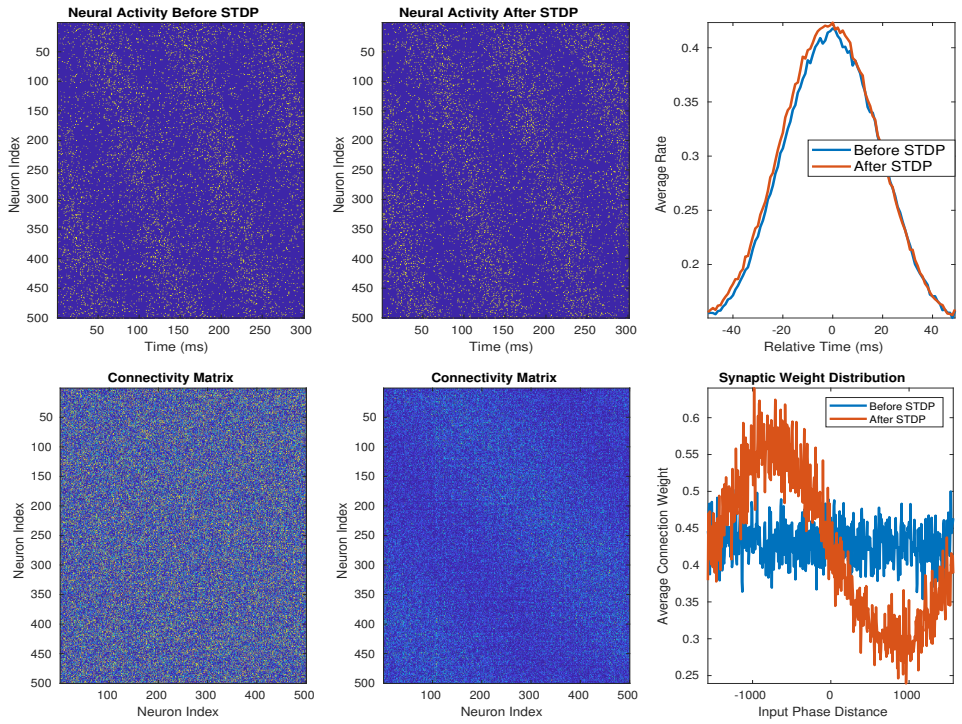


Figure 15: **Evolution of a circulant peak of activity**: We simulated 400 neurons and feed each one with a sinusoid with phases uniformly distributed across the full circle for all neurons. The upper left and upper center plot correspond to raster plots of neural spikes, and the upper right is the neural activity averaged over ten periods and over all neurons with the phase being centered with respect to their inputs. The lower left and center plots are the matrix weights where brighter color represents higher weights, and the right lower plot is the average vector $\mathbf{w}$. After repeating the input many times, the STDP changes the weights which take the shape of a sinusoid (lower row), and the activity is shifted to be slightly advanced and higher (upper row). The low effect of the weight is simply a result of the parameters of the neurons and weights, which do not allow large weights without generating instabilities through the STDP

which we can merge to obtain

$$\mathcal{F}[x](\omega) = \varrho\left(\mathcal{F}[u](\omega) + \mathcal{F}[x]^3(\omega)\left(\beta_A\omega + \beta_S\right)\right) \tag{153}$$

This last equation has the advantage that it gives us a direct connection between the input and the activity. More specifically, if we put a single sinusoid with period T as an input, the Fourier transform consists of two deltas at $\omega = \pm 2\pi$. Thus, the previous equation is zero everywhere except at the position of the deltas, and there it yields a depressed cubic equation that can be solved analytically.

Even though we have an analytical solution, it is worth noticing that the values of $\varrho,\ \beta_A,\ \beta_S$ must be estimated numerically as they depend nonlinearly on the input statistics to each neuron. Furthermore, there are severe constraints on the parameters of the STDP that arise from the stability of the plasticity and depend on the number of neurons and synapses. Thus, in the simulations presented in Fig. 15 we observe that there is only a small modification of the input activity.

That limitation does not imply that our previous results were useless. First, because the results relate the general shape of the STDP kernel to qualitative modifications on the activity, namely the shift in signal phase linked to the strength and sign of β_A that arises from the two solutions for $\pm\omega$. Second, because we can obtain the shape of $\mathbf{w}$ as a sinusoid which in our case leans towards the asymmetry and thus gives us weights that are sinusoids with a phase difference in the interval $\left[0, \frac{\pi}{2}\right]$, mostly toward the end of it as shown in Fig. 15.

21 Solution II: Sparsity and Binary Activity

The second approach to this problem to deal with Eq. 148 is to take advantage of the particular the nonlinearity of $f(\cdot)$ – with upper and lower saturation – and take a signal – $u(t)$ – that is encoded as a sequence of very sharp symbols with small overlapping, and then let STDP converge into a neural network where each of those symbols "prepares" the activation of the next.

The intuition here is that we have a signal consisting in two levels of activity: the peak and the background activity. To improve the signal we only need to push those two levels apart, meaning that the neurons that would be active should have a rate as high possible and those that are not meant to be active should be as silent as possible. This can be seen as a long sequence of symbols where only a small population of neurons should be active for every symbol and with the populations having very little overlap; then, the causal associations promoted by STDP will create a network of neural populations where the populations associated to each symbol have strong connections towards the subsequent population and thus prepare it to receive an input and fire.

In more practical terms, we are using a simple input which consists of a very narrow peak of current concentrated on very few neurons, and a constant background activity corresponding to the probability of a neuron firing without input. This peak corresponds to a mollified Dirac of activity, which we can plug into Eq. 149,

$$\mathbf{w}(\theta) = \left\lfloor \beta_A \left[\tilde{\delta}(\theta) * a\tilde{\delta}(\theta - \epsilon) - \tilde{\delta}(\theta) * a\tilde{\delta}(\theta + \epsilon) \right] + \beta_S \left[\tilde{\delta}(\theta) * \tilde{\delta}(\theta) + c \right] \right\rfloor \tag{154}$$

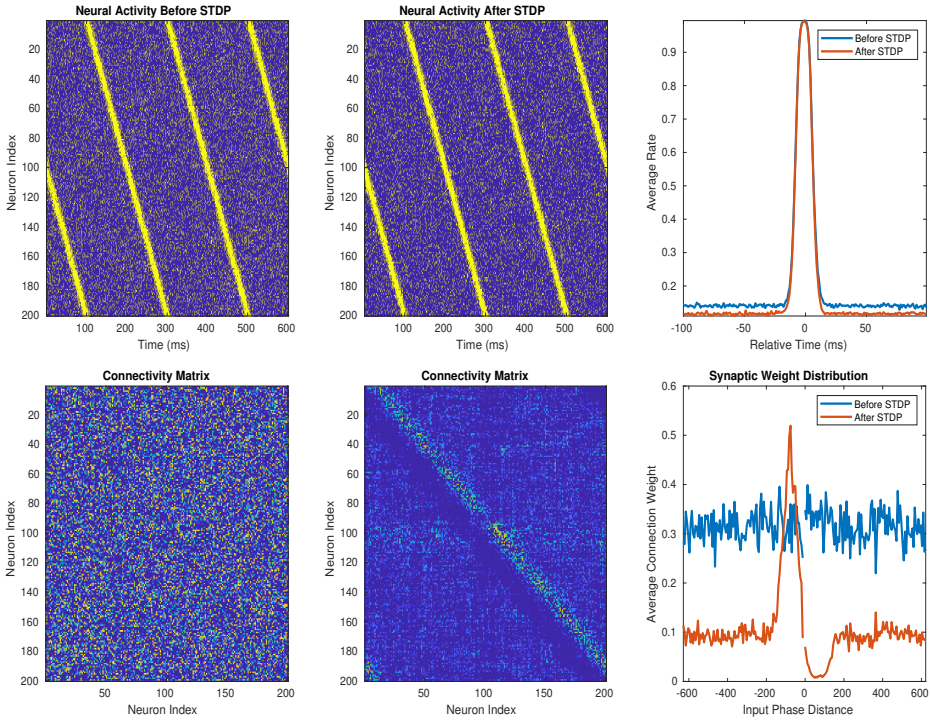


Figure 16: **Evolution of a circulant peak of activity**: We simulated 500 neurons with random connections and imposed an activity $u(t) \propto \exp\left(-\frac{t^2}{2\tau_u}\right)$ where the time constant τ_u is on the order of magnitude of the leak time constant so that the activity of neurons with phase $t - c\delta\theta$ affects the activity of neurons at t. This activity modifies the initially random network into a circulant-like network, where every neuron connects only to its left neighbors (lower panels), which in turn reduces the noise in the network by decreasing the amount of input at non-active times and increasing the input at the time when the neurons should be active (upper panels).

where $\tilde{\delta}$ is the mollified Dirac function, whose derivative is approximated by $a\tilde{\delta}(\theta - \epsilon) - a\tilde{\delta}(\theta + \epsilon)$, with ϵ being a very small number and a being a scaling that depends on the exact mollification, and the c corresponds to the probability of two neurons randomly firing at similar times thus strengthening their connections by β_S. Assuming that the mollified version behaves similarly to a Dirac, we can apply the convolutions and obtain

$$\mathbf{w}(\theta) \approx \left\lfloor \beta_A a\tilde{\delta}(\theta - \epsilon) - \beta_A\tilde{\delta}(\theta + \epsilon) + \beta_S\tilde{\delta}(\theta) + c\beta_S \right\rfloor . \tag{155}$$

In our case $\beta_A \gg \beta_S > 0$, and the weights cannot be negative, the shape of the weights remains constant with value $\beta_S c$ for most θ, then a peak with height

$a\beta_A$ at $-\epsilon$ and a drop to zero at ϵ. Furthermore, with those weights the activity gets reinforced because the neurons at phase θ promote the activity of neurons at phase $\theta + \delta\theta$, thus the activity increases due to neighboring neurons is mostly arriving at the time when the neuron is getting the high level input. This leads to an increased signal to noise ratio, which is shown in Fig. 16

22 Discussion

In this chapter we derived solutions for the evolution of synaptic activity under periodic inputs that use differential Hebbian learning under a self-consistency set-up. In this way we have shown that the features of Echo State Networks that were important in the first part of the thesis can emerge out of synaptic plasticity rules that are well known in the field of neuroscience. In particular, we saw that increased signal-to-noise ratios appear in both the linear and the sparse solution, showing that the signal processing notions can provide neuroscience insight.

The networks proposed here differ from the Echo State Networks used in earlier chapters in the sense that the two solutions presented require very specific receptive fields. Therefore, we cannot use the input as we did in Echo State Networks, but must instead project it into the network in such a way that different neurons are active at different times.

Although the self-consistency equations used before can in principle be solved numerically, the fact that the parameters $\beta_A,\ \beta_S$ are unknown makes such numerical approaches problematic. However, the limitations outline here are general to theoretical neuroscience, as biological neural networks are typically heterogeneous and estimating the parameters of its components is an arduous task. The results here should therefore be taken qualitatively, in the sense that we would expect STDP to adapt networks of neurons to their input in such a way as to increase the signal to noise-ratio and maybe induce an advancement of the neural activity with respect to the input.

Another remark to this chapter is that we have only proposed the two simplest solutions compatible with Eq. 148 and Eq. 149, but naturally there are other possible options. A straightforward approach would be to increase the multiplicity of the solution, for instance by having multiple isolated sinusoids, or multiple peaks of activity per neuron, in such a way that the general shape of the solution stays the same but neurons have activities composed by multiple receptive fields.

Optimizations of the neural code

The self-organized dynamics that we explored in the previous cases gave us a way to compute the activity and weights that a recurrent spiking neural network will converge to. While this follows naturally from the machine learning theory presented in the first chapters, it is not necessarily optimal for studying neural codes, as the receptive fields of any neuron might not follow one of the two prescribed receptive fields. Furthermore, this only describes the activity towards which the neural activity converges, but does not state anything about its evolution.

We would thus like to study how the postsynaptic spikes evolve under general assumptions and while doing so gain insight into the neural code.

23 Evolution of a single postsynaptic spike

We start by showing how STDP can change individual postsynaptic spikes by reducing their latencies and their number. We will start by presenting simple scenarios with excitatory inputs in which both effects are easy to illustrate, then show how inhibitory synapses can be added to the model, and finally show that those effects can appear in random input spike trains by presenting simulations. It is worth noticing that the time windows in this section are on the order of τ_s and the number of repetitions of each input pattern will be small.

23.1 Latency Reduction

If a fixed train of presynaptic spikes is repeated very often, then the spikes that arrive before the postsynaptic spike get reinforced. This implies that the postsynaptic spike might then be triggered earlier [SMA00, GKvHW96]. When this happens, the refractory period of the postsynaptic neuron would prevent a second spike on the original spiking site. However, when the postsynaptic spike happens earlier and earlier, it might lead to a proliferation of spikes by having a new spike appear at the time of the original postsynaptic spike. Following previous literature [SMA00, AN00, KGH01], to prevent this effect, we assume that long

term depression – the weakening of synaptic weights – is stronger than long term potentiation – the strengthening of postsynaptic weights.

This is easy to understand in a simple scenario: Considering a very long, excitatory presynaptic spike train which generates a single postsynaptic spike at some time t_0. The postsynaptic spike will advance through the spike train, and after some repetitions it will be triggered one presynaptic spike earlier. After this advancement is repeated many times, the postsynaptic spike is triggered at time t_∞, very far (in time) from the place where it was first triggered, so that

$$t_\infty \ll t_0. \tag{156}$$

Note that the argument that we give here is qualitative in nature, in the sense that we simply state that LTD should dominate LTP through the constant η, but we have not studied how to find that ratio. As this would depend on the exact parameters of the regular spike train –and thus would not be directly generalizable–, we will simply assume that the brain operates in a parameter regime in which spikes do not proliferate.

23.2 Late spike disappearance through synaptic noise

If latencies might be reduced, then two postsynaptic spikes that are triggered at distant points in time might become close as time progresses. We must then ask what happens to a pair of postsynaptic spikes that occur very close in time. In this section we show that in the absence of synaptic noise the two spikes can coexist, but random modifications of the presynaptic weights –induced, for instance, by other presynaptic inputs – can lead to the disappearance of the second postsynaptic spike.

There are many possible scenarios that we might consider when we have pairs of postsynaptic spikes in the same neuron: we must consider the time between the two spikes, the movements in time of both of them and the possibility of synaptic noise. The case when two postsynaptic spikes happen originally very close in time is extremely rare – because postsynaptic spikes are sparse. The case where the first postsynaptic spike also moves is not interesting, because the spike will move forward in time, increasing the distance between the two postsynaptic spikes and thus reducing the LTD effect on the second spike. Therefore we will consider the case where there is an early postsynaptic spike at some fixed time that will remain in place, and a second postsynaptic spike that will initially be triggered very far in time.

The intuition here is that there is a time interval for the second postsynaptic spike, in which the LTD of the first postsynaptic spike would lead to a decrease in

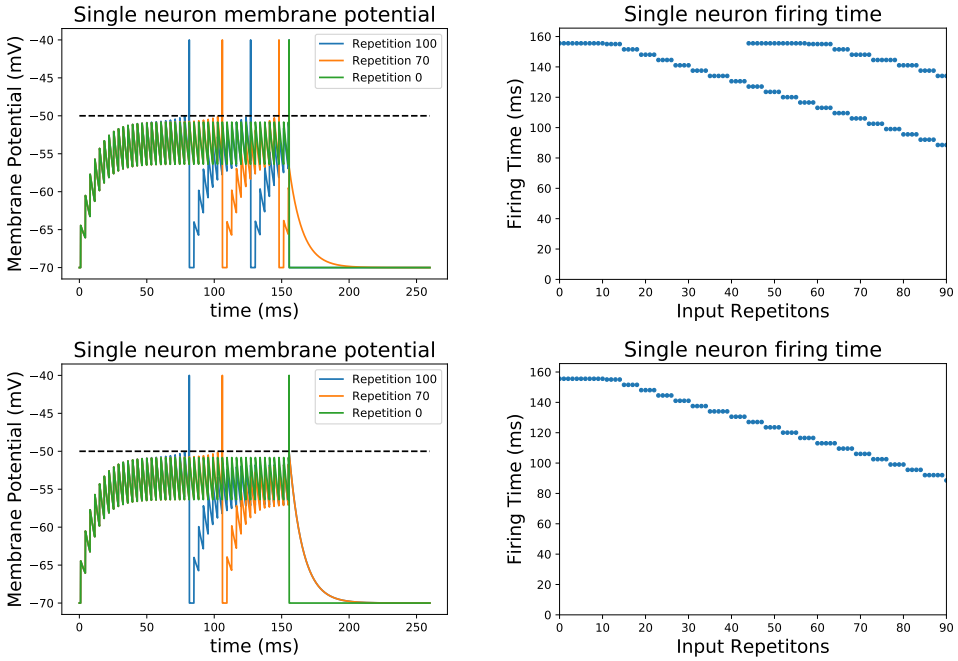


Figure 17: **Latency reduction and spike proliferation**: We plot the membrane potential (left) and firing times (right) of a postsynaptic neuron that receives a constant train of spikes with inter-spike interval of $3.5ms$ and strength $5.5mV$, from time $t = 0ms$ to $t = 150ms$, and we add an extra spike at $t = 150ms$ with potential $2mV$. The neuron generates a single postsynaptic spike at the original input presentation (Repetition 0). The upper plots reflect the case $\eta_+ = \eta_-$, while for the lower ones we picked $\frac{3}{2}\eta_+ = \eta_-$. After an initialization period, the postsynaptic spike moves forward in time at a constant rate. As this happens, a single presynaptic spike will get reinforced proportionally to the η_+ and dampened proportionally to η_-. If LTP is equal to LTD, after the postsynaptic spike happens much earlier than before, the membrane potential of the postsynaptic neuron will reach the threshold again. This second postsynaptic spike would move forward in time at the same speed as the strengths of the spikes are left unchanged by the compensation of LTD and LTP (upper plots). In the case where $\eta_+ < \eta_-$, the depression compensates the potentiation, so there is no second postsynaptic spike.

the membrane potential of the postsynaptic neuron at the time of the second postsynaptic spike, which could lead to the irreversible disappearance of the second postsynaptic spike or its recession. Outside of this time interval, the second postsynaptic spike will reduce its latency, approaching the early postsynaptic spike and the dangerous zone. In the remaining of this section we will show that this interval is never reached in a deterministic system but that the addition of noise can enforce this disappearance.

We start by showing that repeating always the same input spike train without noise cannot lead to the reduction of the number of postsynaptic spikes. Consider a long presynaptic spike train with presynaptic spikes arriving at $t_0, t_1, ...t_N$, which generates two postsynaptic spikes, one at time t_0, which is fixed and will appear at every presentation of the spike train, and another one that is originally triggered at t_N. For the second spike to disappear, it can either do so at t_N or first advance through the spike train – that means, being triggered at t_{N-1}, then at t_{N-2} and so on – and eventually die. For now, we assume that $t_N - t_0 \gg \tau_s$, so that initially the spike at time t_N evolves independently of the spike at time t_0, and it would not disappear at t_N. Consider now that the input has been repeated long enough so that the second postsynaptic spike is now triggered at t_i, and the effects of the STDP generated by the spike at t_0 are not negligible to the presynaptic weight t_{i-1}, which is associated to the presynaptic spike at t_{i-1}. If the postsynaptic spike is originally triggered at t_i, then it would move to t_{i-1} only if, after repeating the same input many times,

$$v(t_{i-1}) = \sum_{k=1}^{i-1} w_k e^{-\frac{t_k - t_{i-1}}{\tau_m}} \geq v_{th}. \tag{157}$$

After $v(t_{i-1})$ crosses the v_{th} threshold, the postsynaptic spike at t_i moves to t_{i-1}, and thus the time difference between every presynaptic spike at $t \leq t_{i-1}$ and the postsynaptic spike is reduced. This naturally implies that the synaptic weights w_k for all $k \leq i-1$ increase, thus the postsynaptic spike cannot disappear because the membrane potential at $v(t_{i-1})$ cannot decrease unless the postsynaptic spike moves to t_{i-2}.

This argument assumes that presynaptic spike trains are always repeated with fixed spike timings but with weights that are affected by LTP and LTD. This is generally not true, as there are many factors that can introduce stochasticity on the evolution of the weights, such as jitter, the stochastic nature of molecular dynamics on the synaptic cleft and on the neuron membrane.

If we now consider the stability of both postsynaptic spikes with respect to that noise, we easily realize that they are not equal: while the presynaptic spikes that generate the first postsynaptic spike are only subject to LTP and noise, the presynaptic spikes that generate the second spike – which happen necessarily be-

tween postsynaptic spikes – are subject to both LTP – from the late postsynaptic spike – and LTD –from the earlier postsynaptic spike – on top of the noise.

This difference implies that the noise can make a postsynaptic spike disappear or recede, either by directly weakening the associated presynaptic weights or strengthening them, so that the postsynaptic spike moves into a region where LTD dominates and it would be later erased or receded.

To explain this in the setting that we used before, consider a neuron with a postsynaptic spike at time t_i that would not move to t_{i-1} in the previous deterministic system. However, now the weights evolve by the combined effects of that spike, an earlier postsynaptic spike at time t_0 and some noise. The membrane potential at time t_i and after r repetitions of the input spike train follows

$$v(t_i) = \sum_{k=1}^{i} w_k e^{-\frac{t_k - t_i}{\tau_m}} + \xi_{t_i}, \tag{158}$$

where ξ_t is the contribution of the random evolution of the weights to $v(t)$.

If this postsynaptic spike train is repeated very often, the deterministic part of the weights goes to a fixed value, which is zero for $k > i$ and thus $v(t_k) \sim \xi_{t_k}$ for all $k > i$. Thus, under the assumption

$$\xi_{t_i} < v_{th} - \sum_{k=1}^{i} w_k e^{-\frac{t_k - t_i}{\tau_m}} \tag{159}$$

for a few repetitions of the input spike train, the second spike vanishes. Therefore among the subsequent input repetitions subjected to the ever present postsynaptic spike at t_0,the weights w_k will decrease for all values of k, hence it is possible that $v(t_i) < v_{th}$ thereafter. This will result in the irreversible disappearance of the postsynaptic spike at t_i or its delay. This is illustrated in Fig. 18.

23.3 Numerical verification for random input spike trains

The examples presented to illustrate the latency reduction and the disappearance of late postsynaptic spikes were simple, so we must now extend them to a more general case. To do so, we simulated spike trains where the times of the presynaptic spikes are random, including only excitatory or excitatory and inhibitory STDP, noise and the presence of an earlier postsynaptic spike. The results are presented in Table 1 and agree with our previous conclusions: a single postsynaptic spike tends to reduce its latency, if there are multiple postsynaptic spikes in a short time window the later one tends to disappear, and the presence of noise increases those effects. Note that we have not included jitter or probabilistic presynaptic spikes, choosing instead to have noise directly on the weight evolutions. As both

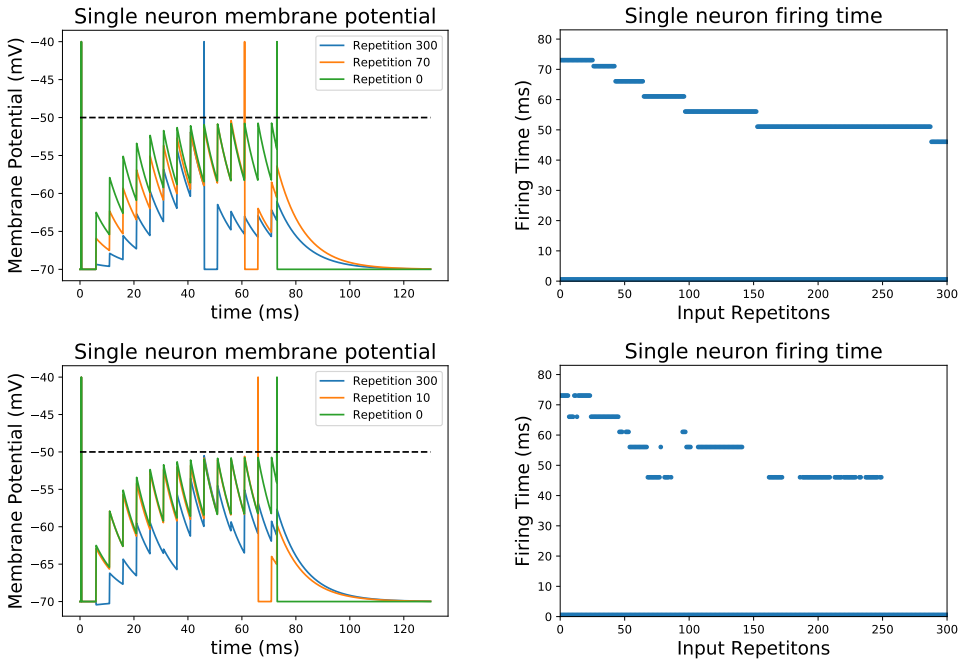


Figure 18: **Noise deletes late spike in a regular presynaptic spike train**: We plot the membrane potential (left) and firing times (right) of a postsynaptic neuron that receives a constant train of spikes with inter-spike interval of $5ms$ and strength $7.5mV$, from time $t = 0ms$ to $t = 150ms$, and we add an extra spike at $t = 80ms$ with potential $5mV$, with a forces postsynaptic spike at time $0.5ms$. Note also that, during its existence, the latency of the postsynaptic spike subject to noise decreases faster than its noiseless counterpart.

cases have been addressed before [GVT05] with similar conclusions, we shall not repeat them here.

So far we have only considered effects in small time scales, meaning that there were only a few spikes on a time interval of the order of $10ms$, and the postsynaptic spike train would evolve over a few repetitions, on the order of 20. This leads us to the conclusion that, with plausible assumptions on the parameters of our model, an individual postsynaptic neuron will fire a given postsynaptic spike earlier after many repetitions of the same presynaptic spike train and that if two postsynaptic spikes are close in time, then the later one could disappear.

24 Postsynaptic Spike Train

Now we will study the effects of the previously described phenomena, which act on small temporal scales and affect only one or two postsynaptic spikes, for a population of postsynaptic neurons, each one receiving many presynaptic spike trains happening over time scales much larger than τ_m or τ_s. Specifically, we will explore how phenomena explored before –the latency reduction and suppression or delaying of late postsynaptic spikes– change the postsynaptic spike distribution.

Before studying the previous effects, we must validate some of the assumptions that we made in the previous section. Specifically, we assume that all the input spikes came from different synapses, which allowed us to treat the weights of all presynaptic spike as independent. This is a valid assumption when we are considering a short time interval, as the sparsity of presynaptic firing and the existence of refractory periods implies that a single synapse would typically not fire more than once during a short presynaptic spike train. However, when there is a long presynaptic spike train, a presynaptic neuron might contribute to that spike train more than once, thus our assumption might be invalid and the phenomena described in the previous section might not appear. To ensure that the phenomena of latency reduction and late spike disappearance are still present in long spike trains we use a combinatorial argument and count the number of synapses that might evolve in a non-trivial fashion.

24.1 Postsynaptic spikes evolve independently

The problem with having multiple spikes per presynaptic neuron is that all of the presynaptic spikes coming from the same synapse have the same weight, and therefore when a postsynaptic spike is close to one of those presynaptic spikes, all of the presynaptic spikes that come from that synapse will undergo the same weight modifications. There are two scenarios when this would be a problem:

1. A single synapse undergoes STDP from two or more different spikes: If there are two postsynaptic spikes, affected by their respective presynaptic spikes, but some of those presynaptic spikes come from the same synapse, the resulting weight change from STDP would be a combination of the effects of both postsynaptic spikes. This is undesirable as the effects could be opposite: one postsynaptic spike could induce depression while the other potentiation, and thus the evolution of one of the presynaptic spikes would not evolve as our STDP rule predicts.

2. A new postsynaptic spike appears spontaneously from STDP: Typically, STDP applies only when there exists a postsynaptic spike. However, if some synapses are very strong due to STDP, and those synapses have spikes

that are close together, they could generate a new postsynaptic spike. This would automatically generate pairs of presynaptic spikes that are affected by two postsynaptic spikes simultaneously (thus we would be in the previous case). Furthermore, the spontaneous generation of new postsynaptic spikes is itself problematic.

Consider M presynaptic neurons which fire with a rate λ, and a postsynaptic neuron that receives them with a rate $\rho = M\lambda$ during a time interval of length T, generating s_{post} postsynaptic spikes. Furthermore, each one of those postsynaptic spikes imposes STDP that affects the presynaptic spikes that are close to it. For simplicity, we will assume that the noticeable effect on the presynaptic spikes is restricted to a time window of size $l\tau_S$ where l is a small integer number.

We start by studying case (1). If we have s_{post} postsynaptic spikes, then the effects of STDP are noticeable for

$$t_a = l\tau_S s_{post} \tag{160}$$

milliseconds in which all presynaptic spikes should come from different synapses. Given that the arrival times of each spike are uniform of the whole interval, the expected number of presynaptic neurons that fire in that interval more than once is given by

$$N \sum_{k=2}^{\infty} \frac{(\lambda t_a)^k e^{-\lambda t_a}}{k!} = N\left(1 - e^{-\lambda t_a} - \lambda t_a e^{-\lambda t_a}\right), \tag{161}$$

and by a Taylor expansion to order two,

$$\mathbb{E}\left[\#1\right] \approx N\left(1 - 1 + \lambda t_a - \frac{\lambda^2 t_a^2}{2} - \lambda t_a + \lambda^2 t_a^2\right) = N\frac{\lambda^2 t_a^2}{2}. \tag{162}$$

To get an intuition of the magnitude of these numbers, consider, for instance, an input spike train lasting $1s$ with presynaptic spike rate of $0.5Hz$ which generates two postsynaptic spikes and we pick the relevant time window to be twice τ_S, so $l = 2$ and $s_{post} = 3$. Then, the expected number of events of type (1) would be

$$\mathbb{E}\left[\#1\right] \approx \frac{M}{400}. \tag{163}$$

Furthermore, not all of those events would actually be problematic; if all of them are potentiating or depressing, then this does not change our analysis.

For case (2) we argue that in order to spontaneously generate new spikes, the synapses affected STDP must be very strong and excitatory, and a few of those strong excitatory synapses must coincide within a small time window of order τ_m.

The synapses that can be very strong are those in the t_a time, meaning that we expect

$$n_a = \rho t_a = \rho l \tau_S s_{post}, \tag{164}$$

independent synapses to be close to $w_{\max}$. Each one of those synapses can fire within the remaining $T - t_a$ time at a rate λ, so we would expect to have presynaptic rate of spikes affected by STDP of

$$\lambda_a = n_a \lambda. \tag{165}$$

Now we must compute the probability that enough of them coincide to generate a postsynaptic spike.

We denote this number by k and we will compute the number of spontaneous postsynaptic spikes that would appear for every k. We start by considering $k = 2$ of those presynaptic spikes (although for some choices of $w_{\max}$ we have to start at $k > 2$), and note that in order to have the postsynaptic spike, we must have

$$w_{\max} + w_{\max} e^{-\frac{\Delta t_{k=2}}{\tau_m}} + \sigma_v > v_{th} \tag{166}$$

where σ_v is a term that accounts for the presence of other spikes that could be driving the membrane potential higher, and $\Delta t_{k=2}$ is the time interval between the two spikes. By rearranging,

$$\Delta t_{k=2} < i_2 = \tau_m \ln\left(\vartheta - 1\right), \tag{167}$$

where $\vartheta = \frac{v_{th} - \sigma_v}{w_{\max}}$. Since the spikes follow a Poisson distribution, the probability of a time interval between spikes is given by an exponential distribution, so

$$\Pr\left[\#2 | k = 2\right] = \Pr\left[\Delta t_{k=2} < i_2\right] = 1 - e^{-\lambda_a i_2}, \tag{168}$$

and the number of those intervals tends to $\lambda_a T$ for large T, so

$$\mathbb{E}\left[\#2\right]_{k=2} = \lambda_a T \left(1 - e^{-\lambda_a i_2}\right), \tag{169}$$

For $k > 2$, the estimation can be done by applying the fact that two contiguous spikes are independent, and therefore the inter-spike intervals are also independent, so we can multiply their probabilities. Furthermore, we should not have any two spikes at a distance closer than i_2, so

$$\Pr\left[\#2 | k = 3\right] < \int_{i_2}^{\infty} \lambda_a e^{-\lambda_a x} \int_{i_2}^{\infty} \lambda_a e^{-\lambda_a y} \Theta\left[1 + e^{-\frac{x}{\tau_m}} + e^{-\frac{y}{\tau_m}} - \vartheta\right] dy dx, \tag{170}$$

where the inequality comes because we let the interval time go to infinity, while T is finite. We can ignore the $\Theta\left[1+e^{-\frac{x}{\tau_m}}+e^{-\frac{y}{\tau_m}}-\vartheta\right]$ term and we obtain

$$\Pr\left[\#2|k=3\right] < \left(1-e^{-\lambda_a i_2}\right)^2. \tag{171}$$

And here the number of pairs of contiguous time intervals is also lower than $T\lambda_a$, which gives us

$$\mathbb{E}\left[\#2\right]_{k=3} \leq T\lambda_a\left(1-e^{-\lambda_a i_2}\right)^2. \tag{172}$$

Naturally, the same upper bound can be computed for any k, so

$$\begin{aligned}\mathbb{E}\left[\#2\right]_{\forall k} &= \sum_{j=2}^{\infty}\mathbb{E}\left[\#2\right]_{k=j} \leq T\lambda_a\sum_{j=2}^{\infty}\left(1-e^{-\lambda_a i_2}\right)^{j-1} \\ &= T\lambda_a e^{\lambda_a i_2} = T\lambda_a\left(\vartheta-1\right)^{\lambda_a}.\end{aligned} \tag{173}$$

Which will be low as long as λ_a is low. If we have, for instance, $M=50$, $l=2$, $s_{post}=3$ and $\lambda=0.5\,Hz$, we obtain $\lambda_a = 6{\cdot}10^{-3}$. Then, if we take $\sigma_v = w_{\max}/2$, $\vartheta = 1.5$,

$$\mathbb{E}\left[\#2\right]_{\forall k} \approx \frac{3T}{1000}, \tag{174}$$

with T being in milliseconds, this means that for an input spike train lasting half a second, generating 3 postsynaptic spikes, there would be one expected spontaneous postsynaptic spike.

The estimates from Eq. 163 and 174 give a relatively low number of coupled postsynaptic spikes or spontaneous spikes. We will therefore assume, from now on, that the effects described in Sec.23 are valid and happen in every postsynaptic spike of every neuron independently of the presence of other postsynaptic spikes.

24.2 Evolution of the postsynaptic spikes

We can now consider the first time that an input presynaptic spike train is presented. Every neuron starts at $v(0)=0$ and then its membrane potential will change depending on its inputs. As the input spike train consists of independent spikes with independent weights, the times of the first spike have a probability distribution $f_0^1(t)$ with support on $t>0$, which depends on the parameters of the input spike train. After spiking, every neuron resets its membrane potential to zero, and thus the distribution of inter-spike intervals $f_0^{ISI}(t)$ follows

$$f_0^{ISI}(t) = f_0^1(t-t_{ref}). \tag{175}$$

After the input has been repeated many times, the distribution of postsynaptic spikes changes to f_∞^1 and f_∞^{ISI} respectively. Specifically, the first spikes reduce

their latency on average and thus move closer to $t = 0$, while the inter-spike intervals increase, due to the depressive effect of postsynaptic spikes that repels or eliminates late postsynaptic spikes. Therefore,

$$\begin{aligned} F_{\infty}^{1}(t) &= \int_{0}^{t} f_{\infty}^{1}(x)dx \geq \int_{0}^{t} f_{0}^{1}(x)dx = F_{0}^{1}(t) \\ F_{\infty}^{ISI}(t) &= \int_{0}^{t} f_{\infty}^{ISI}(x)dx \leq \int_{0}^{t} f_{0}^{ISI}(x)dx = F_{0}^{ISI}(t) \end{aligned} \tag{176}$$

where $F_{\infty}^{1}, F_{\infty}^{ISI}, F_{0}^{ISI}$ and F_{0}^{F} are the cumulative probability distributions of the inter-spike intervals and first spikes respectively. This is illustrated in Fig. 19 showing that indeed the first spikes move forward through STDP and the later spikes are more separated, which is consistent with the results from previous sections.

It is worth noting that our results are only valid for the specific case where the plasticity rule potentiates the presynaptic spike to a neuron before its postsynaptic spikes and depresses those afterwards.

For the next section it will be convenient to look at the instantaneous firing rate, which is obtained by accumulating the times of all spikes.

$$s(t) = \lim_{\Delta t \to 0} \sum_{k=1}^{\infty} \frac{\Pr\left[t_k \in [t, t + \Delta t]\right]}{\Delta t} \tag{177}$$

where t_k is the time of the kth spike. Since the time of the kth spike is the sum of the inter-spike intervals of the first $k - 1$ spikes and the first spike, and the probability of a sum is given by the convolution of the probability distributions, we can rewrite the previous function as

$$s(t) = \left(f^{1} + f^{1} * f^{ISI} + f^{1} * f^{ISI} * f^{ISI} + ...\right)(t) = \left(f^{1} * \sum_{k=0}^{\infty} \left(f^{ISI}\right)^{*k}\right) \tag{178}$$

where $*$ is the convolution operator, ^{*k} is the convolution power. Note that f^{1} and f^{ISI} depend on how many times the input has been repeated. We will refer to the subindex 0 and ∞ to refer respectively to the cases where the presynaptic spike train is presented for the first time or when it has been presented many times.

The postsynaptic spike trains generated by neural populations are instantiate codes that transmit information about presynaptic spikes to other neurons. As STDP is a learning mechanism that modifies the postsynaptic spike train, we expect that it should improve this encoding. Each input stimulus triggers spikes in a certain neural population, and every neuron in that population has a certain performance associated to it, the two most common performance measures being energy consumption and resistance to noise [R^{+}96].

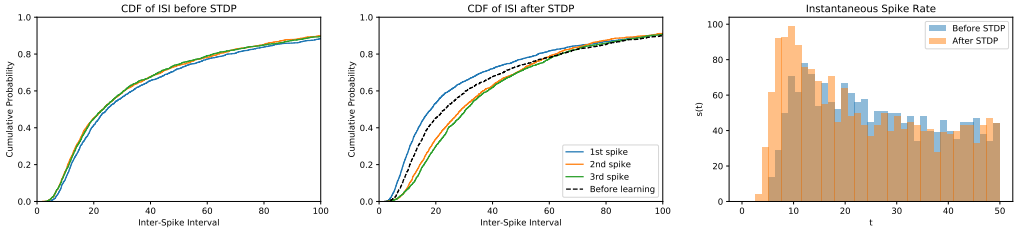


Figure 19: **Evolution of the spike train** : We plot cumulative probability distribution of the time of the first spike, the inter-spike interval when a presynaptic spike train is presented for the first time (left) and after many repetitions (center) and the Number of spikes per bins of $4ms$ on the first $50ms$ of a spike train (left). We simulate 2000 neurons each receiving a presynaptic spike train lasting $600ms$ with 200 presynaptic spikes, both inhibitory and excitatory, and whose arrival time is uniformly sampled. Every synapse evolves through STDP and being subject to both the fixed spike train with probability 0.33 or a random pair of pre- and post-synaptic spikes with $t_{post} - t_{pre} \in [-20ms, 20ms]$ with probability 0.66. We plot the time of the first spike (blue) and the inter-spike interval for second, third and fourth spikes, but subtracting the refractory period to have a pertinent comparison with the first spiking time. We can see that initially the first spike time is the same as the inter-spike interval for all the spikes, but after STDP is applied the average time of the first spike reduces, implying that the blue line moves to the left with respect to the time before learning (in the black dotted line) while the average inter-spike intervals increase, thus moving the curves to the right. This changes the distribution of spikes to have more of them concentrated in the begining of the spike train.

If we take the number of postsynaptic spikes generated by the neural population as a proxy for the metabolic costs of encoding a stimulus, then we would expect that number to decrease as the stimulus is presented more often, so that the encoding of common stimuli incurs less metabolic costs.

To evaluate how the number of spikes evolves, we consider the evolution of the first spike and inter-spike-interval cumulative probability distributions from Fig. 19. On one hand, the fact that the first spike moves forward implies that there will be more spikes concentrated on a small region at the beginning, so if we consider a very short time interval the concentration of spikes will increase. However, as we increase the length of the stimulus the average distance between spikes will start to depend mostly on the inter-spike interval, implying that the spike density will be lower. In more formal terms, the number of spikes is given

by the integral

$$S = \int_0^T s(t)dt = \int_0^T f^1(t)dt + \int_0^T f^1 * \sum_{k=1}^{\infty} \left(f^{ISI}\right)^{*k} dt, \tag{179}$$

which is dominated by the first term when T is small and by the second term when T is large. This can be quantified by the ratio in the decrease of spikes

$$\frac{S_\infty - S_0}{S_0}, \tag{180}$$

where S_0 is the number of spikes before STDP and S_∞ is the number afterwards. Naturally, there are many parameters that affect the change in the number of spikes, in particular the length of the stimuli and the input rate or how often the input is presented with respect to other stimuli, which are shown in Figure 20. In general, in short time intervals at most one spike would be present, thus the disappearance of second spikes induced by the depressive side of STDP does not play a role; at the same time, the spikes that would appear by the fluctuations in input weight, and which would simply disappear by the same process if STDP was not present, remain. Hence in that case the number of spikes increases, while for long spike intervals the number of spikes decreases.

It is worth noticing that the reduction in the number of spikes that we observe in Fig 20 does not correspond to the reduction in spike count that STDP induces in Poissonian spike trains. We tested this by checking how a Poissonian spike train with the same STDP parameters and the same weight distribution and input rate as in Fig. 19 changed, and we found that this leads to an increase of 10% in the number of postsynaptic spikes because excitatory presynaptic spikes tend to induce postsynaptic spikes, thus the excitatory weights systematically increase.

Besides the number of spikes, it is also interesting to note how the distribution of those spikes change. Specifically, as the first spikes move forward, the spike train will become more synchronous as the distribution of spiking times becomes sharper, as we can see in Fig. 19, where the postsynaptic spike train has a peak of spikes that grows after STDP is applied. We quantify this by counting the highest concentration of spikes in a small time window of size L with respect to the total number of spikes, which can be written as

$$\phi = \frac{\max_t \int_t^{t+L} s(t)dt}{S} \frac{T}{L}, \tag{181}$$

where T is the time interval for the full stimulus such that $S/T\frac{S}{T}$ is the average spike rate and $\frac{\max_t \int_t^{t+L} s(t)dt}{L}$ is the highest rate in a time window of length L. For a random spike train, the highest rate of spikes in a time window of length L would

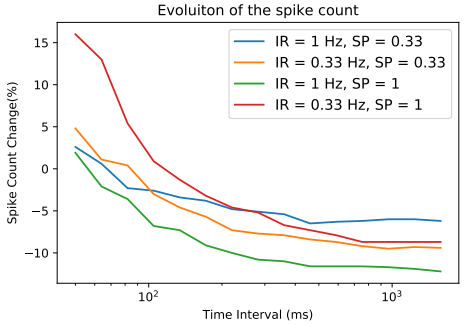


Figure 20: **Spike Count Evolution**: We simulated spike trains of various lengths for different parameters of the input rate (IR) and the probability that the stimulus is presented (SP), and when it is not we induce a random pair of pre-postsynaptic spikes in every synapse with a $t_{post} - t_{pre} \in [-20ms, 20ms]$. In either case we investigate the change in the number of spikes. As we can see, for long spike trains the inter-spike-intervals increase and thus the number of spikes decreases. For short spike trains, on the other hand, there is at most one spike that can fit, so the inter-spike intervals are irrelevant. Furthermore, the spikes in such short intervals are self-maintained when STDP is present: if a spike appears and disappears when the presynaptic weights evolve randomly, the presence of a postsynaptic spike will potentiate those weights, hence the spike will be maintained, implying that STDP increases the number of spikes in short time intervals.

be similar to the average firing rate, corresponding to a $\phi \approx 1$. However, if many spikes concentrate in a small time window, the spike trains are synchronized and we obtain a high value of ϕ. The results of simulations for various parameters are presented in Table 2, where the increase in ϕ can be easily seen.

25 The Emergence of Predictions

When a group of neurons encodes a stimulus we mean that those neurons fire when the stimulus is presented. However, the neurons themselves are not aware agents and do not know anything about that stimulus; they simply receive a spike train that is strong enough to trigger their spiking. From the point of view of an encoding neurons, there is no difference between the stimulus-induced presynaptic spike train and any other input spike train that always precedes the stimulus.

Combining this observation with the results from previous sections showing that neurons will fire at the onset of a frequent input spike train, we can conclude that a neuron that ”encodes” a stimulus can start firing before the stimulus is presented if another stimulus appears before it. As an illustrative example, imagine

listening to a melody. Different parts of the melody trigger the activity of different groups of neurons in the same area of the brain. If the melody is repeated very often, the neurons $P1$ that react to an early part of the melody will systematically fire before the neurons $P2$ that react to a later part. As the melody is repeated, neurons in $P2$ will always fire after receiving spikes from neurons in $P1$ and thus the synapses from $P1$ to $P2$ will be reinforced. Eventually, the reinforced synapses might trigger spikes in $P2$ before the late part of the melody sounds. This can be extended to more populations encoding more stimuli, and thus the whole melody is encoded through simultaneous activity of all the neurons which originally encode only separate notes. This is illustrated and simulated in Fig. 21.

It is important to notice here that the predictions that we mention here are restricted to stimuli sequences that can be identified from the first input, meaning that we are not addressing the case of two sequences of stimuli which start activating the same neural population and then go on to activate different populations. If we have two possible stimuli sequences which start equally, STDP would force some neurons associated to both possible sequences fire at the onset of the stimuli, meaning that the system would learn that both sequences might follow. However, the differentiation of the two sequences can only be done when the two diverge, so the system must learn to maintain memory traces of the stimuli, a process that can also be implemented by STDP with lateral inhibition [KM13]

26 Discussion

In this chapter we analyzed and expanded previous computational findings on latency reduction [SMA00, GVT05]. Then we interpret them in communication terms: those mechanisms lead to encoding the more common inputs with less spikes while concentrating the remaining spikes in smaller time windows. This leads us to the conclusion that STDP can reduce the amount of spikes used to encode frequent stimuli, in line with the idea that metabolic efficiency is one of the guiding principles of the brain [HOCS10, Lau01]. The same phenomena also improves decoding performance of the neural code by concentrating encoding spikes on small time windows, corresponding to the notion that synchronization is a learned behavior used to improve communication between neuronal assemblies [Sin11, Fri05, VDM94] and remaining consistent with the latency code that has been found experimentally [ZSN$^+$11, GKR96]. Finally, we show that the latency reduction can explain how the nervous system learns to forecast even without any feedback.

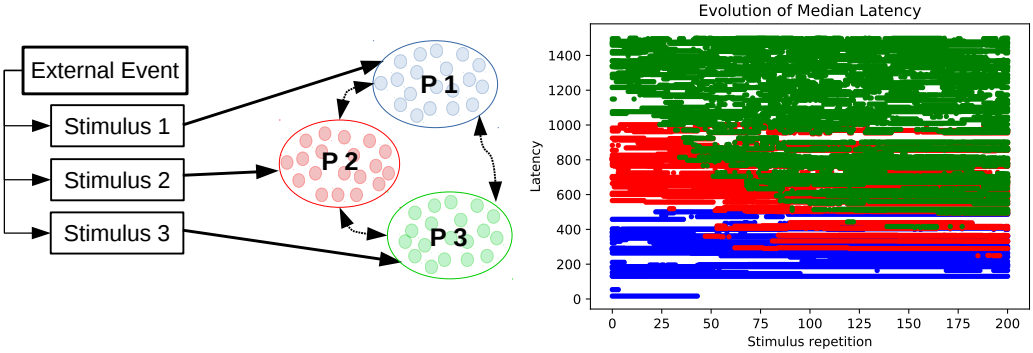


Figure 21: **Encoding Predictions**: Schema for the emergence of predictions (left) and firing latencies of neurons in encoding population (right): An external event creates three stimulus that trigger all the neurons in corresponding distinct neural populations $P1,\ P2,\ P3$ with the stimuli inducing spikes during the intervals $[0ms, 500ms]$ for $P1$, $[500ms, 1000ms]$ for $P2$ and $[1000ms, 1500ms]$ for $P3$ respectively. The three populations, with $N = 50$ neurons each, also have synapses between them with delays sampled from a uniform distribution between $d_{PiPj} \in [1ms, 5ms]$. Originally, almost all neurons in each population fire only after receiving inputs from their respective stimuli, but after the external event is repeated very often, the inter-population connections become strong enough to trigger some spikes before the stimulus is received.

STDP Type	Noise Var.	Spike at t=0	Spike Count Increase	Spike Count Decrease	Spike Latency Increase	Spike Latency Decrease	Average Latency Change
E & I	0	No	0.1 %	13.5 %	5.2 %	51.9 %	-2.7 ms
E & I	0.2	No	1 %	30.6 %	13.8 %	23.1 %	-1.5 ms
E	0	No	1.4 %	0.0 %	0.5 %	81.4 %	-3.1 ms
E	0.2	No	4.2 %	11.3 %	16.2 %	32.6 %	-0.34 ms
E & I	0	Yes	0.0 %	8.3 %	28.8 %	22.1 %	3.2 ms
E & I	0.2	Yes	0.6 %	21.5 %	33.1 %	13.1 %	2.8 ms
E	0	Yes	5.2 %	10.8 %	15.9 %	31.3 %	0.1 ms
E	0.2	Yes	2.8 %	12.9 %	27.2 %	18.7 %	3.3 ms

Table 1: **Effects of STDP on short random spike trains**: We explored the effects of STDP on the postsynaptic spike train of a neuron receiving 8 excitatory and 2 inhibitory presynaptic spikes arriving at uniformly sampled times on the interval $[0, 40ms]$ and the stimulus is repeated 100 times. The first three columns determine the set-up: the STDP Type indicates if STDP was active for excitatory presynaptic neurons only (E) or for inhibitory as well as excitatory (E & I) with the inhibitory STDP having the parameters to exactly compensate the excitatory one as presented in Section 18.2, the second column indicates the variance of the Gaussian noise added to every weight at every stimulus repetition, and the third column indicates whether we added a postsynaptic spike at the begining of the time window. The remaining columns explain the results: the fourth one indicates the percentage of spike trains in which new postsynaptic spikes appeared, the fifth one the percentage of spike trains in which a spike disappeared, the sixth one the percentage of spike trains in which a single postsynaptic spike (not counting the imposed one at $t = 0$) happened later after learning, the seventh one corresponds to the postsynaptic spike happening earlier, and the last one is the average latency change of the postsynaptic spikes (here we only accounted for the cases where there was a single postsynaptic spike at the beginning and at the end of the learning). We calculated the percentages and averages from 1000 randomly generated spike trains in which a single postsynaptic spike was triggered at the beginning of the training. The results clearly show that in all cases spike latencies tend to decrease when no spike is placed at $t = 0$, and increase otherwise. Naturally, adding noise or inhibitory plasticity increases the percentage of spikes that disappear. Similarly, adding the initial spike increases the number of disappeared spikes.

	SP = 0.33		SP = 1	
	IR = 0.33	IR = 1	IR = 0.33	IR = 1
L = 2	$2.1 \rightarrow 2.7$	$2.8 \rightarrow 3.9$	$2.3 \rightarrow 3.4$	$2.9 \rightarrow 4.9$
L = 5	$1.8 \rightarrow 2.4$	$2.4 \rightarrow 3.0$	$2.0 \rightarrow 3.0$	$2.5 \rightarrow 3.6$
L = 10	$1.7 \rightarrow 2.0$	$1.9 \rightarrow 2.1$	$1.9 \rightarrow 2.6$	$1.9 \rightarrow 2.3$

Table 2: **Synchrony Evolution**: We simulated spike trains with different values of the presynaptic input rate (IR) and the probability that the stimulus is presented compared to random pair of spikes per synapse (SP), and then measured the change in ϕ taking a time window of $100ms$ and using 1000 neurons. As we can see, the synchronization always increases.

Part IV

Conclusion

27 Summary and Contributions

This thesis started by formalizing a simple idea: that neurons in a reservoir should be similar to the output of the reservoir. This formalization allowed us to bring well-known notions from signal processing, improving standard Echo State Networks by making them resonate at specific frequencies.

However, this result goes beyond the practical applications of Echo State Networks. By studying which properties of a recurrent neural network make it well-suited for a particular problem, we are also addressing the converse question of how should a neural network be after it has been adapted to a specific task. Specifically, this suggests that biological neural networks could learn to process inputs by having their neurons improve the input representation by resonating at the appropriate frequencies.

Thus we have an hypothesis for biological neural systems. The first step in this case is to check whether the hypothesis is compatible with known rules of synaptic plasticity and neural dynamics. Nicely, this compatibility is straightforward: by using a statistical physics approach, we found that Spiking Time-Dependent Plasticity –one of the most well studied mechanisms of synaptic adaptation – does indeed generate the resonances that we required.

The self-consistency approach that we used, while indicative, is limited in that it works for periodic inputs and specific receptive fields. To make a more general study we shifted from a rate-based theory to one focused on the evolution of post-synaptic spikes, finding that the same phenomena could easily be interpreted in terms of neural coding, with increased signal to noise ratio and emergent predictions.

This thesis provides another example of how simple, well-known plasticity rules that are present at synaptic level lead to modifications that can be interpreted at a larger scale. Furthermore, the fact that the same mechanism improves the neural code and creates predictions might explain how the ability of the brain to make predictions and process time series –which is one of the core problems in cognitive science– could have emerged as a consequence of evolutionary pressures on metabolic cost and information transmission.

Naturally, our work is also interesting for researchers in machine learning, as it shows that synaptic plasticity rules, which are classically used to infer or reinforce correlations [DA01], can be used to find predictions and improve reservoir. Furthermore, the fact that the same mechanism gives rise to predictions and coding efficiency is another example of the intimate relationship between machine learning and coding [MMK03], thus it might be interesting for information theorists.

Finally, there are a few contributions of this thesis that are minor steps in the development of the general thread but are relevant as standalone results. The main

one is the hypotrochoid law of random matrices, which finds beautiful patterns of eigenvalues in the notably unintuitive setting of non-hermitian matrices. Similarly, the geometric bound on the training error is a simple construction that can be applied to a variety of problems concerning time series, as long as the appropriate basis for it is found. Similarly, on the neuroscience setting, the effect that late postsynaptic spikes disappear by being close to another postsynaptic spike has not been reported in the literature, but would easily explain the emergence of latency codes [ZSN+11].

28 Open Questions and Future Directions

The results exposed here also open new questions. The most immediate question is whether this unsupervised process is used in the nervous system to improve time series processing. An experimental study should identify the neurons that encode a periodic stimulus and follow the evolution of the spiking times of the involved neurons as the input is repeated and the animal learns the stimulus. A natural candidate for this would be the cerebellum because of its organization based on Granule and Purkinje cells, with the first being simple but abundant – thus easily interpretable as high-dimensional reservoir – while the latter are large and combine the inputs from the first – thus akin to a readout –, an interpretation that is consistent with Liquid State Machines [YT07], the biological version of Echo State Networks, implement periodic patterns of activity and seem to be important for proper movements [GS85] as well as other functions [MBC+12, Itō84], all the while having STDP rules [SLS+17].

But beyond the circuit studied, it is important to have the appropriate data and tools to verify if the temporal consistency of spike trains is indeed increased as the network learns. Since our only requirement is that neurons have coherent firing times associated to a periodic activity without any population-level synchrony, the learning framework proposed here should be studied using spikes rather than aggregate methods such as electroencephalography. On the data analysis side, we would need a tool that can check the temporal coherence of a long and repetitive spike train. For this purpose a research project in collaboration with Laura State [AS19], where spiking times are converted in phases of a complex number, allowing us to use linear algebra tools to discern temporal patterns. This tool in combination with multielectrode or calcium imaging recordings while an animal learns a periodic motor task should provide the necessary evidence of existence for the proposed learning theory.

Beyond the interest of this thesis for neuroscience, the use of frequency adaptation and filter theory can also be interesting for extensions of machine learning. Just as we could propose that biological neurons adapt their autocorrelations or

frequencies, it seems natural to propose that artificial networks do the same. Indeed it seems plausible that recurrent artificial neural networks keep only the parts of the frequency spectrum that are necessary for future computations, while disregarding the rest. This is in line with a recent theory [TZ15, SZT17] proposing that deep neural networks learn to compress their inputs by discarding irrelevant information for the output. Notice here that the frequency adaptation can be implemented by cycles, but also by feedforward structures where the paths from input to neurons generate the equivalent of a Finite Impulse Response filter; even if the frequency adaptation takes place, which filter structure is preferred would probably depend on the training algorithm and the number of neurons.

Similarly, if the frequency adaptation is indeed part of the learning mechanism of a recurrent neural network, then it might be worth integrating it into the training. A first possible extension would be to consider how recurrent neural networks are after being trained. Since most training methods work on the weights of each connection independently, there is large potential for improvement in designing algorithms which train structural features such as cycles directly.

Finally, another application of the techniques for Echo State Networks proposed here would be to adapt them to other, more complicated reservoirs. Indeed, the multitude of physical substrates that can implement a reservoir includes many systems that cannot be tuned or adapted through modern methods for training artificial neural networks, therefore offering an alternative path to adapt their dynamics could help improve their performance. A natural choice in this respect would be neuromorphic chips – silicon implementations of spiking neural networks – [Boa05] which are often used as reservoirs [ILBL$^+$13], and can also implement plasticity [LZZ$^+$14, VTP$^+$03], making them ideal candidates for implementing the learning rules to induce frequency adaptation.

Besides the field-specific questions addressed earlier, there is one larger question that underlies the thesis as a whole: if and which principles of time series processing from machine learning can be used to guide neuroscience and vice versa. Adapting the concepts presented here for both fields could help integrate well known concepts of signal processing into a general principled theory of learning in time series.

Code availability. The code used in this thesis is available for download through github under the following link:
`https://github.com/pvili`

1 Training Echo State Networks

The training of an ESN aims to find the output weights of each neuron state such that the output $y(t)$ can best approximate the target variable $\hat{y}(t)$ [Jae01b]. Here the output is a linear combination of the neurons' states $\mathbf{x}(t)$ and the input $u(t)$, i.e.,

$$y(t) = \mathbf{w}^{\text{out}} \begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix}. \tag{182}$$

This aim can be achieved by minimizing the squared training errors $\sum_{t=1}^{T}(\hat{y}(t) - y(t))^2$. Hence it becomes a classical linear regression problem: Given T vectors $\mathbf{x}(t)$ of dimension N and the target variable $\hat{y}(t)$, calculate the vector $\mathbf{w}^{\text{out}}$ that satisfies

$$\mathbf{w}^{\text{out}} = \arg\min_{\mathbf{w}^{\text{out}}} \sum_{t=0}^{T} \left(y(t) - \mathbf{w}^{\text{out}} \begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix} \right)^2.$$

We rewrite Eq. 182 as a matrix equation $Y = \mathbf{w}^{\text{out}} \cdot X$, where Y is the column vector containing all $y(t)$ for $t = 1, ..., T$ values and X a matrix where each row contains the values of the corresponding vector $\begin{pmatrix} \mathbf{x}(t) \\ u(t) \end{pmatrix}$. Thus, $\mathbf{w}^{\text{out}}$ can be solved through the Moore-Penrose pseudo-inverse $X^+ = X^* \left(XX^*\right)^{-1}$ where X^* is the Hermitian transpose (also known as conjugate transpose of X). In the case of real matrices, the Hermitian transpose is just the transpose. Thus we can write $X^+ = X^\top \left(XX^\top\right)^{-1}$, and

$$\mathbf{w}^{\text{out}} = Y \cdot X^+.$$

The calculation of X^+ is implemented with the command pinv in Matlab.

Depending on the task [Jae02], the output $y(t)$ can be fed back to the reservoir through $\mathbf{w}^{\text{ofb}}$. During the training phase, we do not have the actual $y(t)$, since $\mathbf{w}^{\text{ofb}}$ has not been trained. If the reservoir's output is expected to be close to the target variable, we can instead feed the target variable $\hat{y}(t+1)$. To have a valid comparison with the original benchmark problem [JH04], we use this process for the task of forecasting Mackey-Glass time series, and for the rest of tasks the values of $\mathbf{w}_{\text{ofb}}$ are set to be 0.

1.1 Selecting reservoir parameters

Besides the training of the readout, the reservoir has other parameters that must be optimized. Typically, those are the scaling of $\mathbf{w}^{\text{in}}$, $\mathbf{w}^{\text{ofb}}$ and $\mathbf{W}$. In this work, however, we take a very simple version of ESN where $\mathbf{w}^{\text{in}}$ and $\mathbf{w}^{\text{ofb}}$ are fixed and only the scaling of $\mathbf{W}$ is trained. This is often trained through the spectral radius $|\lambda_{\max}|$, which grows linearly with the weights of $\mathbf{W}$ – just as our metric $\langle|\lambda|\rangle$ does.

In the examples used in this work we trained this in the simplest possible way: by trying a range of scalings – from a spectral radius of 0.2 to 1 with intervals of 0.05. This is done with classical Erdős-Rényi reservoirs, and in the last experiments, we evaluated the $\langle|\lambda|\rangle$. By noticing that the Erdős-Rényi reservoirs have a uniform eigenvalue distribution within a circle in the complex plane centered at zero and with radius $|\lambda_{\max}|$, then we only need to integrate the radius over the uniform distribution and divide by the area,

$$\langle|\lambda|\rangle = \frac{1}{\pi|\lambda_{\max}|^2}\int_0^{|\lambda_{\max}|} r2\pi r dr = \frac{2}{3}\lambda_{\max}. \tag{183}$$

2 Network Generation Algorithms

In this section we described the methods of generating various model networks with different topological properties.

2.1 Scale-free networks

In scale-free (SF) networks the probability distribution of node degrees follows a power law, i.e., $P(k) \sim k^{-\gamma}$, and the exponent γ usually varies between 2 and 3.

In this paper we adopted the so-called static model [GKK01], to generate scale-free networks with tunable degree exponent γ and mean degree $\langle k\rangle$. Since we use directed networks, we have outgoing and incoming edges. For simplicity we use $\gamma_i n = \gamma_o ut$, meaning that for every node the expected outgoing and incoming degrees are the same. The static model can be described as follows:

Step-1: We start with N isolated nodes, labeled from 1 to N. Each node is assigned a weight $w_i \sim i^{-a}$, where $a = 1/(\gamma - 1)$.

Step-2: We independently pick up two nodes according to their assigned weights, and add a link between these two nodes if they have not been connected before. Self-links and double-links are forbidden.

Step-3: Repeat Step-2 until $M = \langle k\rangle N/2$ links have been added into the network.

2.2 Random regular networks

In a random regular (RR) network, all nodes have the same degree and the edges randomly connect node pairs. Random regular networks can be generated by rewiring, as described by the following algorithm, which follows the same logic as Wormald's algorithm [Wor84] but is designed for directed networks:

The algorithm takes the number of nodes N and the connectivity c as parameters.

Step-1: Create a list L_1 where, for each of the N nodes, there are $cN/2$ outgoing edges.

Step-2: Copy the previous list and make a random permutation, creating list L_2. The nth edge is the edge that goes from the node $L_1(n)$ to the node $L_2(n)$.

Step-3: If there are repeated edges, randomly swap the destinations, until there are no more repeated edges.

Step-4: If Step-3 was repeated M times, jump to Step-2.

The limited number of iterations given by step 4 was set to avoid infinite iterations. In this work we set $M = 500$.

For each node there are k outgoing links and the k links can go to k of the N nodes. Thus we have $\frac{N!}{(N-k)!}$ combinations of destinations that do not include a repetition against N^k possible combinations, giving us a probability of $p_r = 1 - \frac{N!}{N^k(N-k-1)!}$ of repeating an edge. Thus on the first iteration we will have $p_r cN^2$ repetitions, on the second one $p_r^2 cN^2$ and therefore the complexity is $O(\frac{1}{1-p_r} cN^2)$.

2.3 Erdős-Rényi networks

In an Erdős-Rényi (ER) random network, each pair of nodes are connected with probability p. We can use different probability distributions to assign link weights in the ER random networks, including binary, uniform, normal and power law distributions. Networks used in ESN typically have no preference for positive or negative weights, and we modified the link weight distributions to keep the mean as zero.

- For binary distribution, each link was assigned a weight $-a$ or a, with $a > 0$.
- For uniform distribution, the link weights were randomly drawn from the interval $[-a, a]$ where $a > 0$.
- For normal distribution, the link weights were drawn from a normal distribution with zero mean and standard deviation a.
- For the power law (PL) distribution we use the inverse transform sampling method to convert a uniform distribution in $[0, 1]$ into a power law [Deá90]. The power law distribution provides only positive numbers, thus we randomly inversed the sign of each link with probability 0.5.

2.4 Spectral radius and the variance of the weight distribution

In the main text we do not mention the variance of the probability distribution from which we draw the weights. Here we show that the variance of the link weight distribution is actually irrelevant.

The networks referred in the main text have an adjacency matrix $W = (W)_{ij}$ the weights are drawn from a normal distribution with zero mean and some variance v. As we consider the case where the dimension of the adjacency matrix is very high, the variance converges to its expected value. We recall the definition of the variance,

$$v = \text{Var}\left[W_{ij}\right] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}^2.$$

If we multiply the matrix by a scalar $a \geq 0$, when the variance becomes

$$\text{Var}\left[aW_{ij}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} a^2 W_{ij}^2 = a^2 v. \tag{184}$$

Note that the entries of aW are still drawn from a normal distribution, it is simply that the variance has changed. Consider now the spectral radius α of W,

$$\alpha = \max_{\mathbf{v}} \frac{\|W\mathbf{v}\|}{\|\mathbf{v}\|}$$

where $\|\cdot\|$ is the euclidean norm. If we scale the matrix by a,

$$\max_{\mathbf{v}} \frac{\|aW\mathbf{v}\|}{\|\mathbf{v}\|} = \max_{\mathbf{v}} a\frac{\|W\mathbf{v}\|}{\|\mathbf{v}\|} = a\alpha. \tag{185}$$

Putting together Eqs. 185 and 185, we see that both values can be set though α, meaning that there is a one-to-one correspondence between variance and spectral radius. Therefore, when we fix the spectral radius, we also fix the variance. Thus, the original variance of the distribution is irrelevant.

The statement is obviously only true as long as the variance does exist. When we draw numbers from a distribution whose variance diverges, then the spectral radius does not determine the variance, because it does not exist. We will thus still use α in those cases as we still need the stability of the reservoir.

3 Principal Component Analysis

The goal of the principal component analysis (PCA) is to find the directions explaining the variance in a cloud of points, for instance, the points as the values of a time series at different times. Naturally, this idea is only sensible when the cloud of points has correlations between the values at different dimensions.

Our goal is to find a vector $v_1\mathbb{R}^N$ such that var $(\langle v_1, \mathbf{z}(t)\rangle)$ is maximized, but as we are looking for a basis, we will set the constraint that $\langle v_1, v_1\rangle = 1$,

$$\frac{1}{T}\sum_{t=t_0}^{T+t_0}\langle v_1, \mathbf{z}(t)\rangle\langle v_1, \mathbf{z}(t)\rangle - \lambda_1^P(\langle v_1, v_1\rangle - 1) = v_1\left(\frac{1}{T}\sum_{t=t_0}^{T+t_0}\mathbf{z}^\top(t)\mathbf{z}(t)\right)v_1^\top - \lambda_1^P(\langle v_1, v_1\rangle - 1)$$
$$= v_1\mathbf{P}v_1^\top - \lambda_1^P(\langle v_1, v_1\rangle - 1). \tag{186}$$

Then by differentiating we obtain that

$$\frac{\partial v_1}{\partial (v_1)_i}v_1\mathbf{P}v_1^\top - \lambda_1^P(\langle v_1, v_1\rangle - 1) = 0 \Rightarrow \mathbf{P}v_1 = \lambda_1^P v_1 \tag{187}$$

which is an eigenvector equation. The choice of eigenvector is given by noticing that

$$\text{var}\,(\langle v_1, \mathbf{z}(t)\rangle) = v_1\mathbf{P}v_1^\top = \lambda_1^P v_1\lambda_1^P v_1^\top = \lambda_1^P, \tag{188}$$

which implies that we would pick the largest eigenvalue. The subsequent eigenvalue/eigenvector pairs are selected by being orthogonal to every previous eigenvector, this gives us a full orthonormal basis in which the variances var $(\langle v_i, \mathbf{z}(t)\rangle)$ are sorted.

4 Frobenius Norm and variances

The Frobenius norm of an $N \times N$ real matrix is defined as

$$\|A\|_F = \sqrt{\sum_{i=1}^{N} a_{ij}^2} = \sqrt{A^\top A}. \tag{189}$$

This norm is invariant under rotations, as

$$\|RA\|_F = \sqrt{A^\top R^\top RA} = \sqrt{A^\top R^\top RA} = \|A\|_F. \tag{190}$$

and therefore it is unchanged if our original matrix is symmetric and we take the corresponding diagonalized matrix

$$\|A\|_F = \|D_A\|_F = \sqrt{\sum_{i=1}^{N}\lambda_i^2(A)}. \tag{191}$$

If we take a correlation matrix, which is symmetric and positive semidefinite,

$$\|\mathbf{P}\|_F^2 = \sum_{i=1}^{N}\lambda_i^2(\mathbf{P}) \tag{192}$$

5 Fourier Transformation and Parseval's Theorem

For a vector $v \in \mathbb{R}^T$, its Fourier transform $\hat{v}$ is given by

$$\hat{v}\left[k\right] = \sum_{n=0}^{T-1} v(n) \exp\left[2\pi\left(\frac{n}{T}\right) ik\right], \tag{193}$$

where i is the imaginary square root of -1. The transformation can be seen as a change of basis where the new basis vectors are

$$u_k = \exp\left[2\pi i\left(\frac{0}{T}\right)\right], \exp\left[2\pi i\left(\frac{k}{T}\right)\right], \exp\left[2\pi i\left(\frac{2k}{T}\right)\right], ... \tag{194}$$

The basis is naturally orthogonal, as

$$\langle u_k, u_l \rangle = \sum_{n=0}^{T-1} \exp\left[2\pi i\left(\frac{n(k+l)}{T}\right)\right] = 0 \tag{195}$$

This basis can be normalized with the coefficient $\frac{1}{T}$.

The fact that the Fourier transform is an orthonormal basis implies that Euclidean distances are maintained when the transformation is applied. In equations,

$$\sum_{t=0}^{T-1} |v\left[t\right]|^2 = \frac{1}{T} \sum_{k=0}^{T-1} |\hat{v}[k]|^2 \tag{196}$$

which gives us Parseval's theorem.

6 Simulation of spiking neural networks

6.1 Continuous time approximation

The neural dynamics described by Eq. 112 can be simulated by a simple iterative algorithm in which time is discretized in small time bins and the membrane potential is updated. This gives us the discretized equation

$$v(t+1) = \left(1 - \frac{1}{\tau_m}\right) v(t) + u(t) \tag{197}$$

where $v(t)$ is already centered such that the resting potential is zero. This operation is repeated for every timestep, which must be chosen such that $\tau_m \gg 1$ in those units.

This solution is perfect when the input is a continuously evolving function of time. However, here we need as much computations as the number of time beans, which seems a waste or resources since in many of those steps no input will be given. Furthermore, if we have step size large enough to condensate multiple inputs, then the neurons will only receive the sum of the inputs, which might miss a spike in the case where an inhibitory spike arrives slightly after a excitatory one.

6.2 Event-Based Implementation

Thus for the case of inhibitory synapses and a purely spike-based input we will process spikes in an event-based manner. In that case, the membrane potential is only calculated when necessary, meaning at the time of arrival.

More specifically, when a spike with weight w is received at time t, the new membrane potential is calculated as

$$v(t) = v(t_{\text{last}})e^{-\frac{t-t_{\text{last}}}{\tau_m}} + w. \tag{198}$$

An important notion here is that spikes must be processed in strict temporal order. For the study of isolated neurons with predefined inputs, it suffices to put every input into a list, but when we deal with recurrent networks more care must be taken.

To make sure that we maintain the temporal order while remaining efficient, we must choose the appropriate data structure. The natural choice if we want to have heterogeneous delays in synapses is to use a heap.

Bibliography

[Ace18] Pau Vilimelis Aceituno. Eigenvalues of random graphs with cycles. *arXiv preprint arXiv:1804.04978*, 2018.

[AH18] Dario Amodei and Daniel Hernandez. Open ai, May 2018.

[AN00] Larry F Abbott and Sacha B Nelson. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3(11s):1178, 2000.

[ARS19] Pau Vilimelis Aceituno, Tim Rogers, and Henning Schomerus. Universal hypotrochoidic law for random matrices with cyclic correlations. *Physical Review E*, 100(1):010302, 2019.

[AS19] Pau Vilimelis Aceituno and Laura State. Training delays in spiking neural networks. In *International Conference on Artificial Neural Networks*, pages 713–717. Springer, 2019.

[BGMD18] Jacob LS Bellmund, Peter Gärdenfors, Edvard I Moser, and Christian F Doeller. Navigating cognition: Spatial codes for human thinking. *Science*, 362(6415):eaat6766, 2018.

[BHR14] Nicolas Brunel, Vincent Hakim, and Magnus JE Richardson. Single neuron dynamics and computation. *Current opinion in neurobiology*, 25:149–155, 2014.

[Bis95] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BKM10] Béla Bollobás, Robert Kozma, and Dezso Miklos. *Handbook of large-scale random networks*, volume 18. Springer Science & Business Media, 2010.

[Boa05] Kwabena Boahen. Neuromorphic microchips. *Scientific American*, 292(5):56–63, 2005.

[BOL+12] Joschka Boedecker, Oliver Obst, Joseph T Lizier, N Michael Mayer, and Minoru Asada. Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131(3):205–213, 2012.

[BP98] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[Bru00] Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8(3):183–208, 2000.

[BSVdS19] Daniel Brunner, Miguel C Soriano, and Guy Van der Sande. *Photonic Reservoir Computing: Optical Recurrent Neural Networks*. Walter de Gruyter GmbH & Co KG, 2019.

[BSVS08] Pieter Buteneers, Benjamin Schrauwen, David Verstraeten, and Dirk Stroobandt. Real-time epileptic seizure detection on intracranial rat data using reservoir computing. In *International Conference on Neural Information Processing*, pages 56–63. Springer, 2008.

[BY06] Michael Buehner and Peter Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

[BYY14] Paul Bourgade, Horng-Tzer Yau, and Jun Yin. Local circular law for random matrices. *Probability Theory and Related Fields*, 159(3-4):545–595, 2014.

[BZA01] Rune Brincker, Lingmi Zhang, and Palle Andersen. Modal identification of output-only systems using frequency domain decomposition. *Smart Materials and Structures*, 10(3):441, 2001.

[CE05] John Conklin and Chris Eliasmith. A controlled attractor network model of path integration in the rat. *Journal of computational neuroscience*, 18(2):183–203, 2005.

[CG89] TM Cameron and JH Griffin. An alternating frequency/time domain method for calculating the steady-state response of nonlinear dynamic systems. *Journal of Applied Mechanics*, 56(1):149–154, 1989.

[CLL12] Hongyan Cui, Xiang Liu, and Lixiang Li. The architecture of dynamic reservoir in the echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(3):033127, 2012.

[Cou10] Paulin Coulibaly. Reservoir computing approach to great lakes water level forecasting. *Journal of Hydrology*, 381(1):76–88, 2010.

[DA01] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*, volume 806. Cambridge, MA: MIT Press, 2001.

[Deá90] István Deák. Random number generators and simulation. *Mathematical methods of operation research*, 1990.

[Doy00a] Kenji Doya. Complementary roles of basal ganglia and cerebellum in learning and motor control. *Current opinion in neurobiology*, 10(6):732–739, 2000.

[Doy00b] Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

[DS12a] Gustavo Deco and Bernd Schuermann. *Information dynamics: foundations and applications*. Springer Science, 2012.

[DS12b] Ali Deihimi and Hemen Showkati. Application of echo state networks in short-term electric load forecasting. *Energy*, 39(1):327–340, 2012.

[DSG+13] Egidio DAngelo, Sergio Solinas, Jesus Garrido, Claudia Casellato, Alessandra Pedrocchi, Jonathan Mapelli, Daniela Gandolfi, and Francesca Prestori. Realistic modeling of neurons and networks: towards brain simulation. *Functional neurology*, 28(3):153, 2013.

[DVSM12] Joni Dambre, David Verstraeten, Benjamin Schrauwen, and Serge Massar. Information processing capacity of dynamical systems. *Scientific Reports*, 2, 2012.

[DZ06] Zhidong Deng and Yi Zhang. Complex systems modeling using scale-free highly-clustered echo state network. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3128–3135. IEEE, 2006.

[EA04] Chris Eliasmith and Charles H Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[EJL15] Felix Effenberger, Jürgen Jost, and Anna Levina. Self-organization in balanced state networks by stdp and homeostatic plasticity. *PLoS computational biology*, 11(9):e1004420, 2015.

[Ell13] Douglas F Elliott. *Handbook of digital signal processing: engineering applications*. Academic Press, 2013.

[FBG16] Igor Farkaš, Radomír Bosák, and Peter Gergel'. Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83:109–120, 2016.

[Fei12] Adina Roxana Feier. *Methods of proof in random matrix theory*. PhD thesis, Harvard University, 2012.

[FI16] Gergeľ P. Farkaš I., Bosák R. Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83,:109–120, 2016.

[FL11] Aida A Ferreira and Teresa B Ludermir. Comparing evolutionary methods for reservoir computing pre-training. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 283–290. IEEE, 2011.

[Fri05] Pascal Fries. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends in cognitive sciences*, 9(10):474–480, 2005.

[GBFK19] Joshua I Glaser, Ari S Benjamin, Roozbeh Farhoodi, and Konrad P Kording. The roles of supervised machine learning in systems neuroscience. *Progress in neurobiology*, 2019.

[Ger18] Wulfram Gerstner. Private Communication during Bernstein Network Booth, 2018.

[GHS08] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105(48):18970–18975, 2008.

[Gir86] VL Girko. Elliptic law. *Theory of Probability & Its Applications*, 30(4):677–690, 1986.

[GKK01] K-I Goh, Byungnam Kahng, and Doochul Kim. Universal behavior of load distribution in scale-free networks. *Physical Review Letters*, 87(27):278701, 2001.

[GKNP14] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[GKR96] Timothy J Gawne, TROELS W Kjaer, and BARRY J Richmond. Latency: another potential code for feature binding in striate cortex. *Journal of neurophysiology*, 76(2):1356–1360, 1996.

[GKvHW96] Wulfram Gerstner, Richard Kempter, J Leo van Hemmen, and Hermann Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383(6595):76, 1996.

[GMP17] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Deep reservoir computing: a critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

[GMP18] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. Design of deep echo state networks. *Neural Networks*, 108:33–47, 2018.

[GS85] C. Ghez and Fahn S. The cerebellum. *Principles of neural science*, 1985.

[GTJ12] Manjunath Gandhi, Peter Tiňo, and Herbert Jaeger. Theory of input driven dynamical systems. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 25–27, 2012.

[GvG15] Umut Güçlü and Marcel AJ van Gerven. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27):10005–10014, 2015.

[GVT05] Rudy Guyonneau, Rufin VanRullen, and Simon J Thorpe. Neurons tune to the earliest spikes through stdp. *Neural Computation*, 17(4):859–879, 2005.

[GY05] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

[HA16] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:23, 2016.

[HAW89] U Hübner, NB Abraham, and CO Weiss. Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared nh 3 laser. *Physical Review A*, 40(11):6354, 1989.

[HB10] Nacereddine Hammami and Mouldi Bedda. Improved tree model for arabic speech recognition. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 521–526. IEEE, 2010.

[Heb05] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[HGMJ06] Andreas VM Herz, Tim Gollisch, Christian K Machens, and Dieter Jaeger. Modeling single-neuron dynamics and computations: a balance of detail and abstraction. *Science*, 314(5796):80–85, 2006.

[HH53] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physicology*, 1953.

[HKAW89] U Huebner, W Klische, NB Abraham, and CO Weiss. On problems encountered with dimension calculations. In *Measures of Complexity and Chaos*, pages 133–136. Springer, 1989.

[HOCS10] Andrea Hasenstaub, Stephani Otte, Edward Callaway, and Terrence J Sejnowski. Metabolic cost as a unifying principle governing neuronal biophysics. *Proceedings of the National Academy of Sciences*, 107(27):12329–12334, 2010.

[Hop82] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSP99] Geoffrey E Hinton, Terrence Joseph Sejnowski, and Tomaso A Poggio. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[HW62] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[IGB19] Bernd Illing, Wulfram Gerstner, and Johanni Brea. Biologically plausible deep learning–but how far can we go with shallow networks? *Neural Networks*, 2019.

[ILBL+13] Giacomo Indiveri, Bernabé Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38):384010, 2013.

[Itō84] Masao Itō. *The cerebellum and neural control*. Raven press, 1984.

[Izh04] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[Jae01a] Herbert Jaeger. *Short term memory in echo state networks*. GMD-Forschungszentrum Informationstechnik, 2001.

[Jae01b] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

[Jae02] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002.

[Jae07] Herbert Jaeger. Discovering multiscale dynamical features with hierarchical echo state networks. *Jacobs University Bremen, Technical Reports*, 2007.

[JBS08] Fei Jiang, Hugues Berry, and Marc Schoenauer. Supervised and evolutionary learning of echo state networks. In *Parallel Problem Solving from Nature–PPSN X*, pages 215–224. Springer, 2008.

[JH04] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[JLPS07a] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[JLPS07b] Herbert Jaeger, Mantas Lukosevicius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[KF17] Ingmar Kanitscheider and Ila Fiete. Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. In *Advances in Neural Information Processing Systems*, pages 4529–4538, 2017.

[KGGM16] Saeed Reza Kheradpisheh, Masoud Ghodrati, Mohammad Ganjtabesh, and Timothée Masquelier. Deep networks can resemble human feed-forward vision in invariant object recognition. *Scientific reports*, 6:32672, 2016.

[KGH01] Richard Kempter, Wulfram Gerstner, and J Leo van Hemmen. Intrinsic stabilization of output rates by spike-based hebbian learning. *Neural computation*, 13(12):2709–2741, 2001.

[KH00] Werner M Kistler and J Leo van Hemmen. Modeling synaptic plasticity in conjunction with the timing of pre-and postsynaptic action potentials. *Neural Computation*, 12(2):385–405, 2000.

[Khi34] Alexander Khintchine. Korrelationstheorie der stationären stochastischen prozesse. *Mathematische Annalen*, 109(1):604–615, 1934.

[KM13] Stefan Klampfl and Wolfgang Maass. Emergence of dynamic memory traces in cortical microcircuit models through stdp. *Journal of Neuroscience*, 33(28):11515–11529, 2013.

[KRK14] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.

[Lap07a] L Lapique. Recherches quantitatives sur lexcitation des nerfs traitee comme une polarization. *J Physiol Pathol Gen*, 9:620–635, 1907.

[Lap07b] L. Lapique. Recherches quantitatives sur lexcitation electrique des nerfs traite comme une polarization. *Journal de Physiologie Pathologie Genetique*, 1907.

[Lau01] Simon B Laughlin. Energy as a constraint on the coding and processing of sensory information. *Current opinion in neurobiology*, 11(4):475–480, 2001.

[LBB+98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Li14] Zhaoping Li. *Understanding vision: theory, models, and data*. Oxford University Press, USA, 2014.

[Lic13] M. Lichman. UCI machine learning repository, 2013.

[Lie04] Benjamin Liebald. *Exploration of effects of different network topologies on the ESN signal crosscorrelation matrix spectrum*. PhD thesis, University Bremen, 2004.

[Lit74] William A Little. The existence of persistent states in the brain. In *From High-Temperature Superconductivity to Microminiature Refrigeration*, pages 145–164. Springer, 1974.

[LJ09] Mantas LukoševičIus and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[LMBA06] Francisco López-Muñoz, Jesús Boya, and Cecilio Alamo. Neuron theory, the cornerstone of neuroscience, on the centenary of the nobel prize award to santiago ramón y cajal. *Brain research bulletin*, 70(4-6):391–405, 2006.

[LPT09] Andreea Lazar, Gordon Pipa, and Jochen Triesch. Sorn: a self-organizing recurrent neural network. *Frontiers in computational neuroscience*, 3:23, 2009.

[LSB+12] Laurent Larger, Miguel C Soriano, Daniel Brunner, Lennert Appeltant, Jose M Gutiérrez, Luis Pesquera, Claudio R. Mirasso, and Ingo Fischer. Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Optics express*, 20(3):3241–3249, 2012.

[LUTG17] 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.

[LWY+16] Chang-Shing Lee, Mei-Hui Wang, Shi-Jim Yen, Ting-Han Wei, I-Chen Wu, Ping-Chiang Chou, Chun-Hsun Chou, Ming-Wan Wang, and Tai-Hsiung Yan. Human vs. computer go: Review and prospect [discussion forum]. *IEEE Computational intelligence magazine*, 11(3):67–72, 2016.

[LYS09] Xiaowei Lin, Zehong Yang, and Yixu Song. Short-term stock price prediction based on echo state networks. *Expert systems with applications*, 36(3):7313–7317, 2009.

[LZZ+14] Yi Li, Yingpeng Zhong, Jinjian Zhang, Lei Xu, Qing Wang, Huajun Sun, Hao Tong, Xiaoming Cheng, and Xiangshui Miao. Activity-dependent synaptic plasticity of a chalcogenide electronic synapse for neuromorphic systems. *Scientific reports*, 4:4906, 2014.

[Maa11] Wolfgang Maass. Liquid state machines: motivation, theory, and applications. In *Computability in context: computation and logic in the real world*, pages 275–296. World Scientific, 2011.

[Mar06] Henry Markram. The blue brain project. *Nature Reviews Neuroscience*, 7(2):153, 2006.

[MBC+12] Mario Manto, James M Bower, Adriana Bastos Conforto, José M Delgado-García, Suzete Nascimento Farias Da Guarda, Marcus Gerwig, Christophe Habas, Nobuhiro Hagura, Richard B Ivry, Peter Mariën, et al. Consensus paper: roles of the cerebellum in motor controlthe diversity of ideas on cerebellar involvement in movement. *The Cerebellum*, 11(2):457–487, 2012.

[Mea89] Carver Mead. Analog vlsi and neutral systems. *NASA STI/Recon Technical Report A*, 90, 1989.

[Mer76] Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 116:374–388, 1976.

[MMK03] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[MMRS06] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12, 2006.

[MNM02] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MQW00] Mayank R Mehta, Michael C Quirk, and Matthew A Wilson. Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron*, 25(3):707–715, 2000.

[MS69] Minsky Marvin and Papert Seymour. Perceptrons, 1969.

[NS12] Michael J Newton and Leslie S Smith. A neurally inspired musical instrument classification system based upon the sound onset. *The Journal of the Acoustical Society of America*, 131(6):4785–4798, 2012.

[NVS16] Robert A Nawrocki, Richard M Voyles, and Sean E Shaheen. A mini review of neuromorphic architectures and implementations. *IEEE Transactions on Electron Devices*, 63(10):3819–3829, 2016.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[Ola96] Mikel Olazaran. A sociological study of the official history of the perceptrons controversy. *Social Studies of Science*, 26(3):611–659, 1996.

[OXP07] Mustafa C Ozturk, Dongming Xu, and José C Príncipe. Analysis and design of echo state networks. *Neural computation*, 19(1):111–138, 2007.

[PAGH03] Paul G Plöger, Adriana Arghir, Tobias Günther, and Ramin Hosseiny. Echo state networks for mobile robot modeling and control. In *Robot Soccer World Cup*, pages 157–168. Springer, 2003.

[Par06] Marc-Antoine Parseval. Mémoire sur les séries et sur lintégration complète dune équation aux différences partielles linéaires du second ordre, à coefficients constants. *Mémoires présentés par divers savants, Academie des Sciences, Paris,(1)*, 1:638–648, 1806.

[PHG+18] 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.

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning*, 28:1310–1318, 2013.

[PSvRN13] Hugh Pastoll, Lukas Solanka, Mark CW van Rossum, and Matthew F Nolan. Feedback inhibition enables theta-nested gamma oscillations and grid firing fields. *Neuron*, 77(1):141–154, 2013.

[R+96] Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. prentice hall PTR New Jersey, 1996.

[RHHD56] Nathaniel Rochester, J Holland, L Haibt, and W Duda. Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on information Theory*, 2(3):80–93, 1956.

[RIA17] Nathaniel Rodriguez, Eduardo Izquierdo, and Yong-Yeol Ahn. Optimal modularity and memory capacity of neural networks. *arXiv preprint arXiv:1706.06511*, 2017.

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[RT12] Ali Rodan and Peter Tiňo. Simple deterministically constructed cycle reservoirs with regular jumps. *Neural computation*, 24(7):1822–1852, 2012.

[Sch93] Eric L Schwartz. *Computational neuroscience*. Mit Press, 1993.

[SCKS15] David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7):1025, 2015.

[SG02] Harvey A Swadlow and Alexander G Gusev. Receptive-field construction in cortical inhibitory interneurons. *Nature neuroscience*, 5(5):403, 2002.

[SG10] J. Sjstrm and W. Gerstner. Spike-timing dependent plasticity. *Scholarpedia*, 5(2):1362, 2010. revision #184913.

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[Sin11] Wolf Singer. Dynamic formation of functional networks by synchronization. *Neuron*, 69(2):191–193, 2011.

[SL15] Daniel Stimson and Marsh Love. Nih launches the human connectome project to unravel the brain's connections, Oct 2015.

[SLS+17] Martina Sgritta, Francesca Locatelli, Teresa Soda, Francesca Prestori, and Egidio Ugo D'Angelo. Hebbian spike-timing dependent plasticity at the cerebellar input stage. *Journal of Neuroscience*, 37(11):2809–2823, 2017.

[SMA00] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919, 2000.

[Spo10] Olaf Sporns. *Networks of the Brain*. MIT press, 2010.

[SS90] Samir S Soliman and Mandyam D Srinath. Continuous and discrete signals and systems. *Englewood Cliffs, NJ, Prentice Hall, 1990, 523 p.*, 1, 1990.

[Str93] Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[SV99] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[SW12] Harvey A Swadlow and Stephen G Waxman. Axonal conduction delays. *Scholarpedia*, 7(6):1451, 2012.

[SWL12] Tobias Strauss, Welf Wustlich, and Roger Labahn. Design strategies for weight matrices of echo state networks. *Neural computation*, 24(12):3246–3276, 2012.

[SWV+08] Benjamin Schrauwen, Marion Wardermann, David Verstraeten, Jochen J Steil, and Dirk Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, 2008.

[SZT17] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[TBCC07] Matthew H Tong, Adam D Bickett, Eric M Christiansen, and Garrison W Cottrell. Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424–432, 2007.

[Tho90] Simon J Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel processing in neural systems*, pages 91–94, 1990.

[TN04] Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews neuroscience*, 5(2):97, 2004.

[TRA+17] Jacob Torrejon, Mathieu Riou, Flavio Abreu Araujo, Sumito Tsunegi, Guru Khalsa, Damien Querlioz, Paolo Bortolotti, Vincent Cros, Kay Yakushiji, Akio Fukushima, et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature*, 547(7664):428, 2017.

[TYH+19] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: a review. *Neural Networks*, 2019.

[TZ15] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[VDM94] Christoph Von Der Malsburg. The correlation theory of brain function. In *Models of neural networks*, pages 95–119. Springer, 1994.

[VdSBS17] Guy Van der Sande, Daniel Brunner, and Miguel C Soriano. Advances in photonic reservoir computing. *Nanophotonics*, 6(3):561–576, 2017.

[Ver09] David Verstraeten. *Reservoir computing: computation with dynamical systems*. PhD thesis, Ghent University, 2009.

[VFD+13] Tim P Vogels, Robert C Froemke, Nicolas Doyon, Matthieu Gilson, Julie S Haas, Robert Liu, Arianna Maffei, Paul Miller, Corette Wierenga, Melanie A Woodin, et al. Inhibitory synaptic plasticity: spike timing-dependence and putative network function. *Frontiers in neural circuits*, 7:119, 2013.

[VLS+10] Thierry Verplancke, S Looy, Kristof Steurbaut, Dominique Benoit, F Turck, G Moor, and Johan Decruyenaere. A novel time series analysis approach for prediction of dialysis in critically ill patients using echo-state networks. *BMC Medical Informatics and Decision Making*, 10(1):1, 2010.

[VRBT00] Mark CW Van Rossum, Guo Qiang Bi, and Gina G Turrigiano. Stable hebbian learning from spike timing-dependent plasticity. *Journal of neuroscience*, 20(23):8812–8821, 2000.

[VTP+03] R Jacob Vogelstein, Francesco Tenore, Ralf Philipp, Miriam S Adlerstein, David H Goldberg, and Gert Cauwenberghs. Spike timing-dependent plasticity in the address domain. In *Advances in Neural Information Processing Systems*, pages 1171–1178, 2003.

[Wei94] William Wu-Shyong Wei. *Time series analysis*. Addison-Wesley publ Reading, 1994.

[Wie30] Norbert Wiener. Generalized harmonic analysis. *Acta mathematica*, 55(1):117–258, 1930.

[Wie48] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. Technology Press, 1948.

[WLS04] Olivia L White, Daniel D Lee, and Haim Sompolinsky. Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92(14):148102, 2004.

[Wor84] Nicholas C Wormald. Generating random regular graphs. *Journal of algorithms*, 5(2):247–280, 1984.

[Wor99] Nicholas C Wormald. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.

[XT07] Fei Xu and Joshua B Tenenbaum. Word learning as bayesian inference. *Psychological review*, 114(2):245, 2007.

[YD16] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016.

[YJK12] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.

[YT07] Tadashi Yamazaki and Shigeru Tanaka. The cerebellum as a liquid state machine. *Neural Networks*, 20(3):290–297, 2007.

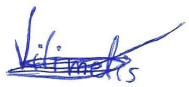
[ZSN+11] Oran Zohar, Trevor M Shackleton, Israel Nelken, Alan R Palmer, and Maoz Shamir. First spike latency code for interaural phase difference discrimination in the guinea pig inferior colliculus. *Journal of Neuroscience*, 31(25):9192–9204, 2011.

Erklärungen

- Ich erkenne die Promotionsordnung der Fakultät für Mathematik und Informatik der Universität Leipzig.

- Die eingereichte Arbeit wurde nicht in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde zum Zwecke einer Promotion oder eines anderen Prüfungsverfahrens vorgelegt.

- Es haben keine früheren erfolglosen Promotionsversuche stattgefunden.



.....
(Pau Vilimelis Aceituno)

Curriculum Vitae

PERSONAL INFORMATION Pau Vilimelis Aceituno

Max Planck Institute for Mathematics in the Sciences, Inselstraβe 22, 04103 Leipzig Germany

☎ +34 647515639 Georg-Schumann Straβe 73, 04155 Leipzig, Germany

✉ Institutional: aceituno@mis.mpg.de, Personal: pau.vilimelis.aceituno@gmail.com

LinkedIn Profile Google Scholar Profile

Date of birth 1st February 1989 | Nationality Spanish

EXPERIENCE

10/2016–12/2019 — **Ph.D. student – Max Planck Institute for Mathematics in the Sciences**

Max Planck Institute for Mathematics In the Sciences (Leipzig, Germany), Advisor: Jürgen Jost

Find optimization principles behind synaptic plasticity that explain how network structures change and that can be interpreted in terms of learning and information processing.

06/2019–12/2019 — **0-year student – Max Planck School of Cognition**

Max Planck School of Cognition

Mentor the first wave of Ph.D. students, select applicants, and build the course handbook and e-lectures.

11/2015–08/2016 — **Visiting Scholar – Harvard Medical School**

Channing Division for Network Science, Harvard Med. School, (Boston, USA), Supervisor: Yang-Yu Liu

Investigated the effects of network structure in learning for artificial recurrent neural networks.

01/2014–11/2014 — **Software Developer – Amadeus IT Group**

Pricing Division Amadeus IT Group, (Sophia Antipolis, France)

Back-end software development. Launched a project on fraud detection through data mining

03/2013–08/2013 — **Intern on Space Robotics – Airbus Defence & Space**

Group for Space Robotics and Exploration, Airbus Defence & Space, (Bremen, Germany)

Design and implementation of an algorithm for pose estimation applied to the recovery of dead satellites.

EDUCATION

09/2007–09/2013 — **Master's in Telecommunications Engineering – INSA Lyon**

Institut National des Sciences Appliquées de Lyon (INSA Lyon), France

Core Subjects: Algorithms, Networks, Signal Processing and Coding Theory.

09/2011–02/2013 — **Master Thesis & Research Assistant – KIT**

Karlsruhe Institut für Technologie, (Karlsruhe, Germany) Advisor: Muhammad Shafique

Thesis: Software and hardware error modeling to ensure software resilience under unreliable software.

02/2011–06/2011 — **Capstone Project and Academic Exchange – Yonsei University**

Yonsei University, South Korea

Project: Reducing Wi-Fi Interference through Power Control.