Universitat
de les Illes Balears

Constructive role of plasticity rules in reservoir computing

Guillermo Gabriel Barrios Morales

Master's Thesis

Master's degree in *Physics of Complex Systems*

at the

UNIVERSITAT DE LES ILLES BALEARS

Academic year 2018-2019

Date: 13/09/2019

UIB Master's Thesis Supervisors: Miguel C. Soriano and Claudio R. Mirasso

"Our treasure lies in the beehives of our knowledge. We are perpetually on our way thither, being by nature winged insects and honey gatherers of the mind. The only thing that lies close to our heart is the desire to bring something home to the hive."

— Friedrich Nietzsche

ABSTRACT

Over the last 15 years, Reservoir Computing (RC) has emerged as an appealing approach in Machine Learning, combining the high computational capabilities of Recurrent Neural Networks with a fast and easy training. By mapping the inputs into a high-dimensional space of non-linear neurons, this class of algorithms have shown their utility in a wide range of tasks from speech recognition to time series prediction. With their popularity on the rise, new works have pointed to the possibility of RC as an existing learning paradigm within the actual brain. Likewise, successful implementation of biologically based plasticity rules into RC artificial networks has boosted the performance of the original models. Within these nature-inspired approaches, most research lines focus on improving the performance achieved by previous works on prediction or classification tasks. In this thesis however, we will address the problem from a different perspective: instead on focusing on the results of the improved models, we will analyze the role of plasticity rules on the changes that lead to a better performance. To this end, we implement synaptic and non-synaptic plasticity rules in a standard Echo State Network , which is a paradigmatic example of an RC model. Testing on temporal series prediction tasks, we show evidence that improved performance in all plastic models may be linked to a decrease in spatio-temporal correlations in the reservoir, as well as a significant increase on individual neurons ability to separate similar inputs in their activity space. From the perspective of the reservoir dynamics, optimal performance is suggested to occur at the edge of instability. This is a hypothesis previously suggested in literature, but we hope to provide new insight on the matter through the study of different stages on the plastic learning. Finally, we show that it is possible to combine different forms of plasticity (namely synaptic and non-synaptic rules) to further improve the performance on prediction tasks, obtaining better results than those achieved with single-plasticity models.

Key words : Reservoir Computing, plasticity rules, Hebbian learning, intrinsic plasticity, temporal series prediction.

Contents

1 Introduction

1.1 On the shoulders of giants: a brief historical perspective.

From the first bird-inspired "flying machines" of Leonardo da Vinci to the latest advances in artificial photosynthesis, humankind has constantly sought to mimic nature in order to solve complex problems. It is therefore not surprising that the dawn of Machine Learning (ML) and Artificial Neural Networks (ANN) was also characterized by the idea of emulating the functionalities and characteristics of the human brain. Within his book *The Organization of Behavior*, Donald Hebb proposed in 1949 a neurophysiological model of neuron interaction that attempted to explain the way associative learning takes place [1]. Theorizing on the basis of synaptic plasticity, Hebb suggested that the simultaneous activation of cells would lead to the reinforcement of the synapses involved. Today known as Hebbian theory[1], its main postulate reads:

> *"When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."* [1]

Swiftly, Hebbian theory was taken by neurophysiologists and early brain modelers as the foundation upon which to build the first working artificial neural network. In 1950, a young Nat Rochester at the IBM research lab embarked in the project of modeling an artificial cell assembly following Hebb's rules [4]. However, he would soon be discouraged by an obvious flaw in Hebb's initial theory: as connection strength increases with the learning process, neural activity eventually spreads across the whole assembly, saturating the network. Further attempts were made to find the "additional rule" which could prevent the overgrowing of connections, but none of them proved successful, and the project was soon consigned to oblivion.

It wouldn't be until 1957 when Frank Rosenblatt, who had previously read *The Organization of Behavior* and sought to find a more "model-friendly" version of Hebb's assembly, came with a solution: the Perceptron, the first example of a Feed Forward Neural Network (FFNN) [5]. Dismissing the idea of an homogeneous mass of cells,

[1] As Hebb himself would recognized later, the idea of associative learning based on alterations in the strength of the connections between neurons was far from new [2]. Today, the merit of the theory is commonly conceded to Tanzi [3], although other authors claim that similar ideas had been already discussed by Alexander Bain even before [4].

Rosenblatt introduced three types of units within the network: *sensory units (S-units)*, to which impulses are transmitted when they are activated; *association units (A-units)*, integrating the signals from the S-units and transmitting it to the R-units; and *response units (R-units)*, which produce the output, becoming active if the total signal coming from the A-units exceeds a certain threshold. The original model of Rosenblatt even included inhibitory connections between the S and R-units, and the A-units, becoming a pioneer work in the implementation of feed-back connections within an ANN. The S, A and R sets in the original scheme, correspond today to what is usually known as input, hidden and output layer in a FFNN. Mathematically, the output of the perceptron is computed as:

$$f(x) = \begin{cases} 1 & if\, w \cdot x + b > 0 \\ 0 & otherwise \end{cases} \tag{1}$$

where $w \cdot x$ is the dot product of the input x with the weight vector w and b is a bias term that acts like a moving threshold. In modern FFNNs, the step function is usually substituted by a non-linearity $\varphi\,(w \cdot x + b)$ which receives the name of *activation function*. The basic scheme of an artificial neuron in the current notation of FFNNs together with its biological association is depicted in Fig.1.

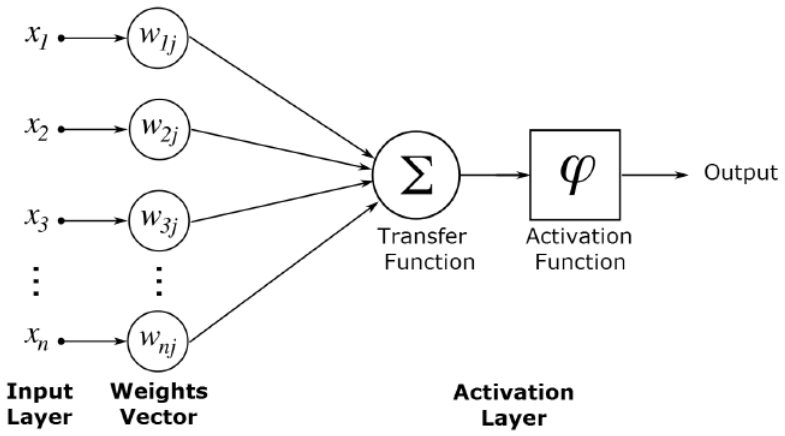


Figure 1: Architecture of a basic artificial neuron. Figure credit to Greezes [6].

Being computationally more applicable than the original ideas of Hebb, Rosenblatt paved the way that would progressively detach ML from its biological inspiration, as he himself would state in the following lines introducing the Perceptron:

> *"The perceptron is designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in*

> *the special, and frequently unknown, conditions which hold for particular biological organisms."* [5]

Despite the initial excitement in the promising new model, it was quickly proved that perceptrons could not be trained to recognize many classes of patterns. In fact, they were only capable of learning linearly separable patterns, becoming a linear classifier. This was shown in 1969 by Marvin Minsky and Seymour Papert in a famous book entitled Perceptrons, where they stated that it was impossible for this type of network to learn an XOR function [7]. In their book, the authors already foresaw the need for Multilayer Perceptrons (MLP) to tackle non-linear classification problems, but the lack of a suitable learning algorithm lead to the first of the AI winters [8], with neural network research stagnating for many years.

The solution to this problem came around 1974 through what today are widely known as Backpropagation algorithms. Backpropagation was derived by multiple researchers in the early 60's and implemented to run on computers as early as 1970 by Seppo Linnainmaa [9], but it was Paul Werbos —back then a PhD student at Harvard University— the first one to propose that it could be used to effectively train MLPs [10]. Understood as a supervised learning method in multilayer networks, backpropagation aims to adjust the internal weights in each layer so to minimize the error or loss function at the output, using a gradient-descent approach based on the chain rule to propagate the error from the outer to the inner layers. Interestingly enough, although backpropagation is usually criticized because of its complete lack of biological meaningfulness, Werbos himself originally found inspiration in the psychological theories of Freud, as he would later recount:

> *"In 1968, I proposed that we somehow imitate Freud's concept of a backwards flow of credit assignment, flowing back from neuron to neuron ... I explained the reverse calculations using a combination of intuition and examples and the ordinary chain rule, though it was also exactly a translation into mathematics of things that Freud had previously proposed in his theory of psychodynamics!"* [11]

The publication of Werbos' results and a later seminal paper in 1986 reformulating the algorithm in the way it is known today [12], would finally restore the faith among the ML enthusiasts. Backpropagation began to be widely used as training method in several new forms of neural networks and its success became evident in tasks as diverse

as speech recognition, natural language processing, medical image analysis or board game programs.

However, the achievements of the MLP + Backpropagation combination should not deviate our attention from one of its obvious drawbacks: it does not help to reproduce the functionality of a biological neural network. Signals in the human brain do not cross from one layer of neurons to the next following a feed-forward architecture, nor learning takes place in a way resembling the backpropagation scheme. Instead, any ANN that aims to recreate the biology behind the brain must include neurons that send feedback signals to each other. This is the idea behind a Recurrent Neural Network (RNN). From a theoretical point of view, RNNs are not only more realistic in what biology respects, they are also computationally more powerful than FFNNs. Whereas FFNNs are able to approximate a mathematical function, RNNs can approximate dynamical systems —i.e. functions with an added time component— so that the same input can result in a different output at different time steps. In theory, RNNs are capable of "remembering" input values for a time by preserving them in some form within the activations of nodes in the network [6].

In 1992, John Hopfield came up with the first successful implementation of an RNN: the Hopfield network [13]. Developed as a content-addressable (i.e. "associative") memory, it consisted on a fully connected net of binary units, usually implementing a Hebbian learning rule for its training. But the real boom of RNNs wouldn't emerge until the discovery of the Backpropagation Through Time algorithm (BPTT), derived by numerous researchers but popularized in 1990 by Paul Werbos [14]. Appearing as an adaptation of the Backpropagation algorithm of FFNNs for RNNs, the idea is to 'unroll' the recurrent neural network by treating each loop through the neural network as an input to another neural network [8]. With the new method at hand, RNNs seemed unstoppable and soon many people started to apply them to hand-writting and speech recognition tasks. Unfortunately, the joy did not last long. RNNs trained with BPTT algorithm weren't performing as well as expected, and soon a new crisis in the AI world took place. The reason however, did not relay on the RNNs themselves, but on the backpropagation method which suffered from today's widely known vanishing gradient problem [15]. In 1993 Yoshua Bengio, later one of the giants of Deep Learning, would summarize the general failure saying:

> *"Although recurrent networks can in many instances outperform static networks, they appear more difficult to train optimally. Our experiments tended to indicate that their parameters settle in a suboptimal solution which takes*

> *into account short term dependencies but not long term dependencies. [...] Since gradient based methods appear inadequate for this kind of problem we want to consider alternative optimization methods that give acceptable results even when the criterion function is not smooth."* [16]

It wouldn't be until the beginning of the 21st century when a new paradigm in RNNs design and training appeared. In 2000, a learning algorithm aiming to overcome the problems of the BPTT was proposed under the name of Atiya-Parlos Recurrent Learning (APRL) [17]. Among its results, it showed that the dominant changes appeared in the weights of the output layer, while the weights of the deeper layers converged slowly. This idea motivated two fundamentally new approaches to RNNs that would appear independently on the following years: the Echo State Network (ESN) of Herbert Jaeger [18] and the Liquid State Machine (LSM) of Wolfgang Maass [19]. They both constitute trailblazing models of what today is known as the Reservoir Computing (RC) paradigm [2]. The main idea behind the new approach was simple: if only the changes in the output layer weights are significant, then the treatment of the weights of the inner network can be completely separated from the treatment of the output layer weights [6]. Fig.2 shows an example of the typical architecture in a Reservoir Neural Network.

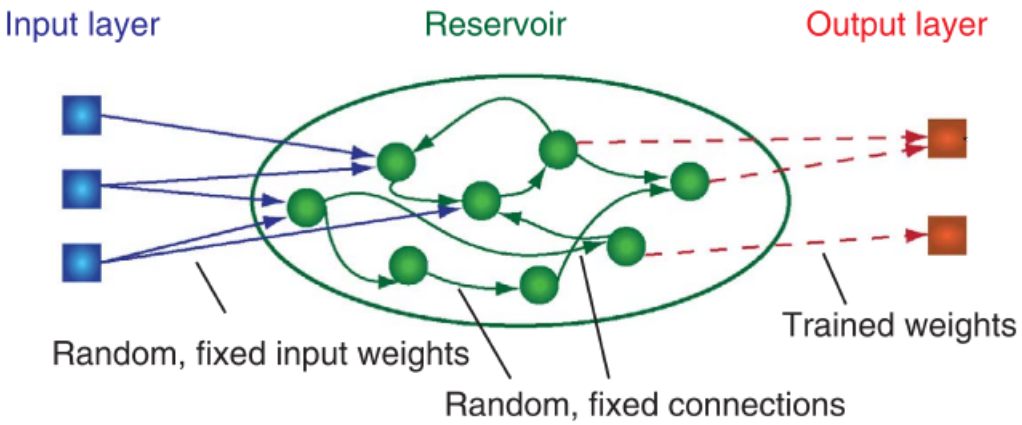


Figure 2: Architecture of a Reservoir Neural Network. Adapted from Appeltant et al [20].

As we will see later in more detail, original RC models were particularly fast and computationally inexpensive since training happened only at the output layer through adjustment of the readout weights. Although very flexible, this approach also leaves the

[2]However, it would not be until 2008 when these and other techniques were united for the first time under the term reservoir computing [6].

open question of how to choose the reservoir connectivity so to maximize performance. It is in this context that plasticity was rediscovered as an unsupervised, biologically inspired adaptation to train the reservoir. It appeared first as a type of Hebbian synaptic plasticity to modify the reservoir weights [21], but soon the ideas of nonsynaptic plasticity that inspired the first Intrinsic Plasticity (IP) rule [22] were also implemented in an Echo State Network [23]. After that, many different models of plasticity rules have been implemented in ESNs, LSMs and other forms of RC networks with promising results [24, 25, 26]. Today, the fact that biologically meaningful learning algorithms have a place in these models, together with recent discoveries suggesting that actual neural networks display RC properties [27, 28], make of Reservoir Computing a field of machine learning on the rise.

1.2 Motivation and objectives of the thesis.

At the light of the presented historical development of ML, the main motivation of this thesis is to recover the brain-like perspective of learning methods for ANNs without sacrificing the good performance achieved by less biologically plausible models. Aware of the vastness of neural network models and learning rules with biological grounds, we will narrow down our efforts to the implementation of neural plasticity rules on Echo State Networks. This choice is motivated by the simplicity of the architecture and supervised learning methods in these networks, which makes the influence of neural plasticity more tractable.

The questions that we aim to answer by the end of this thesis are fairly simple: does synaptic or functional plasticity improve the performance of a simple Echo State Network? And if so, what are the mechanisms by which neural plasticity boost this performance?

Although ESNs have been shown to do well in a wide number of tasks, ranging from speech recognition [29], noise modeling [30], or robot control [31], to stock data mining [32]; we will focus our attention in chaotic time series forecasting. This type of task has been approached on a great number of different temporal series [33, 34, 25, 35], and examples of ESNs implementing plasticity rules to solve these tasks can be easily found in the literature [25, 35, 21]. However, most of the previous works put the focus on outperforming the preceding models, seeking to reach the maximum predictive capability of the ESN, but pay little attention in the underlying mechanisms by which neural plasticity works.

In this thesis we will move away from the best-performance approach, focusing instead in understanding how unsupervised learning through plasticity rules affects the ESN architecture. To do so, we will experiment with different plastic rules found in literature, including examples of synaptic and nonsynaptic plasticity. Our aim is to see if general conclusions can be extracted about the effects of plasticity itself, or if on the contrary, results are mainly model-dependent. With this thesis, we target to provide new insights in the role of plasticity in artificial neurons, building up on the timeless challenge of understanding and emulating the greatest of all nature achievements: the human brain.

2 Theoretical framework

2.1 The ESN architecture.

For our purposes, we will consider temporal tasks where $t = 1, 2, ..., T$ denote discrete time steps and T is the total number of points in the training set. Using a supervised learning scheme, the goal of the ESN is to generate and output $y(t) \in \mathbb{R}^{N_y}$ (with N_y the number of output units), which not only matches as closely as possible the desired target $y^{target}(t) \in \mathbb{R}^{N_y}$, but can also generalize to unseen data. The error in the target reconstruction is usually defined in terms of the Mean Squared Error (MSE):

$$E(y, y^{target}) = \frac{1}{N_y} \sum_{i=1}^{N_y} \left(\frac{1}{T} \sum_{t=1}^{T} \left(y_i^{target}(t) - y_i(t) \right)^2 \right) \tag{2}$$

ESNs are characterized by a very simple structure with only three layers: an input layer, an internal layer or reservoir, and an output layer. Inputs $u(t) \in \mathbb{R}^{N_u}$ are fed into the reservoir through an input weight matrix $W^{in} \in \mathbb{R}^{N_x \times (1+N_u)}$, internal connections between neurons in the reservoir are defined by $W^{res} \in \mathbb{R}^{N_x \times N_x}$ —with N_x the number of neurons in the reservoir— and the states of neurons in the reservoir produce the final output after passing through a final output weight matrix $W^{out} \in \mathbb{R}^{N_y \times (1+N_u+N_x)}$ (see Fig.3). The typical update equation for the state $x(t)$ in the reservoir is given by:

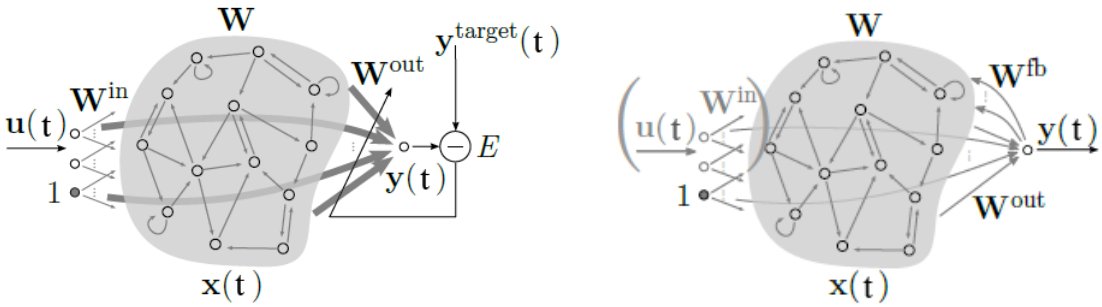
$$x(t+1) = tanh(W^{in}[1; u(t)] + W^{res} x(t-1)) \tag{3}$$

whereas the readout layer is computed as

$$y(t) = W^{out}[1; u(t); x(t)] \tag{4}$$

with ";" denoting vector concatenation. Additional modifications have been extensively applied in the literature, including the use of feedback connections W^{fb} from the output $y(t-1)$ back into the reservoir $x(t)$ [36, 37] or the implementation of leaky integrator neurons [38]. Here we will limit our scope to "simple" ESNs with states and outputs computed according to Eq.(3) and Eq.(4), and having two input units —one feeding a point of the series at each time and a second one acting as a bias— and one output neuron.

Finally, a particularly interesting characteristic of ESNs that we will exploit for time series prediction is the possibility of running them in generative mode, i.e. in a self–driven



(a) Basic architecture of an ESN.

(b) ESN running in generative mode.

Figure 3: Architecture of a basic Echo State Network during training and testing. Figures adapted from [18].

way in which the output at a particular time is fed as the next input $u(t+1) = y(t)$. Let it be noted how this way of using an ESN is in practice equivalent to an ESN with no input and feed-back connections carrying $y(t-1)$ back into the reservoir by means of a matrix W^{fb} (Fig.3b).

2.2 The role of hyper-parameters.

Although ESNs stand out by their simple architecture, training them to get the best performance can become quite a challenge because of their sensitivity to hyper-parameter values. For our purposes, we will take into account the influence of four hyper-parameters that have been shown to play a very important role in ESNs performance:

- ***Size of the reservoir:*** the number of neurons N_x in the reservoir plays a decisive role in the ability of the network to recall past inputs. As a general rule, the bigger the space of reservoir signals $x(t)$, the easier will be to find a combination of them through Eq.(4) that approximates $y^{target}(t)$ [39]. Important exceptions to this rule apply when there is risk of overfitting, for example if the number of training points is not big enough compared to the number of reservoir neurons, $T < 1 + N_u + N_x$. On top of that, it has been shown that the memory capacity of an ESN cannot exceed N_x, so the number of neurons in the reservoir should be at least equal to the estimate of independent real values the reservoir has to remember from the input to solve its task [40]. This being said, it is always important to bear in mind the trade-off between performance and computational

cost, as bigger reservoirs involve operations with bigger matrices, which can slow-down tremendously the code.

- ***Input scaling:*** for input weights W^{in} drawn from a uniformly random distribution, the input scaling is usually defined as the range a of the interval $[-a, a]$ from which the weights are sampled. Input scaling determines how nonlinear the reservoir responses are. For very linear tasks W^{in} should be small, letting units operate around the 0 point where their activation function $tanh$ is virtually linear. On the contrary, large W^{in} would lead to units with a more binary, switching behavior, as their values would easily saturate around -1 and 1. It is also common practice to scale the biases (first column of W^{in} following our notation) and the rest of input nodes separately, although here we will take the much simpler approach of defining a single scaling for all of them.

- ***Spectral radius***: it is defined as the maximal absolute eigenvalue of the reservoir matrix and, in basic terms, it scales the distribution of non-zero elements in W^{res}. Its importance relies on its relationship with the echo state property of the network, fundamental for the ESN approach to work. A reservoir is said to hold the echo state property when its state $x(t)$ is uniquely defined by the fading history of the input signal $u(t)$ [40]. In other words, for an input long enough, the reservoir state should not depend on the conditions existent before the input. The echo state property is usually violated when large values are present in the connection matrix leading to reservoirs hosting multiple fixed points, periodic, or even chaotic spontaneous attractor modes [39]. As a general rule, it is commonly accepted that a spectral radius $\rho(W^{res}) < 1$ ensures the echo state property in the majority of cases.[3]

- ***Sparseness:*** it is a common practice to define the reservoir matrices as sparse, this is, with most of the values equal to zero. Although the effect of the sparseness over performance is usually mild compared with the parameters already listed, it can have a strong impact in the computational cost of network state updates, and hence in the total computational time. For our ESNs, we set this parameter fixed to have 90% of null connections in the initial reservoir. As for the distribution of

[3]The opposite implication, namely that $\rho(W^{res}) > 1$ leads to reservoirs lacking the echo state property, has often proved wrong for nonzero inputs $u(t)$. This can be explained by the strong $u(t)$ pushing activations of the neurons away from 0, where the hyperbolic tangent has a unitary slope, to regions where this slope is smaller, thus reducing the gains of the neurons and the effective strength of feedback connections [39].

the non-zero values in W^{res}, we drew them randomly from a uniform distribution between -1 and 1, although previous studies showed that different distributions account for only slight differences in the final performance [41].

2.3 Training a simple ESN.

Echo State Networks are trained using a supervised learning algorithm whose goal is to update the readout weights W^{out} on the basis of the inputs processed by the reservoir. This has to be done in a way such that the output $y(t)$ matches as closely as possible the desired target $y^{target}(t)$, but there are two possibilities. Training can be performed either "online", meaning that at each step or input we modify the weights by comparing the target and actual outputs; or "offline", in which we present the complete training sequence and store the reservoir states at each step. In this latter case, adjustment of the readout weights is done in bulk during a final step, fitting them to produce the desired output.

The most simple case of online learning is known as delta rule, an stochastic gradient descent update originally devised to train the single-layer perceptron [42]:

$$\Delta W = \eta \left(y^{target}(t) - y(t) \right) x(t) \tag{5}$$

$$W^{out} = W^{out} + \Delta W \tag{6}$$

with η being the learning rate and $t = 1, 2, ..., T$ the time step along the training.

However, most often training of an ESN is done through an offline method, where the updates of the readout weights are performed a single time after all the inputs have passed through the reservoir. Again, the aim is to find optimal output weights that minimize the difference between the desired output signal and the actual output of the ESN in response to the inputs. For linear, feed-forward outputs, Eq.(4) can be rewritten in a matrix form as:

$$Y = W^{out} X \tag{7}$$

where $Y \in \mathbb{R}^{N_y \times T}$ contains all $y(t)$ and $X \in \mathbb{R}^{(1+N_u+N_x) \times T}$ are all concatenated vectors $[1; u(t); x(t)]$. Under this scheme, finding the optimal weights W^{out}_{opt} accounts for solving a typically overdetermined system (because $T \gg 1 + N_x + N_u$) of the form:

$$Y^{target} = W^{out}_{opt} X \tag{8}$$

A straightforward solution to Eq.(8) is:

$$W^{out} = Y^{target} X^{\dagger} \tag{9}$$

where $X^{\dagger}$ is the Moore-Penrose pseudo-inverse of X. Although simple in form, computation of the pseudo-inverse matrix can be expensive memory-wise for large reservoirs or training sets. A widely used alternative to find a solution of Eq.(8) is the Least Square Estimation method (LSE), which is a generalization of a linear regression method. Just like in a basic linear regression, its purpose is to find the optimal weights W^{out}_{opt} minimizing through the training the MSE defined as in Eq.(2). Replacing $y(t)$ for its expression in terms of the states of the reservoir as in Eq.(4), rewriting everything in matrix form and deriving to find the minimum error for a given readout matrix, we get:

$$\frac{\partial E}{\partial W^{out}} = \frac{\partial}{\partial W^{out}} \left(\left(Y^{target} - W^{out} X \right)^T \left(Y^{target} - W^{out} X \right) \right) = \\ = 2 \left(W_{out} X^T X - Y^{target} X^T \right) = 0$$

From the above expression, the LSE solution for the optimal output weight matrix can be easily extracted as:

$$W^{out}_{opt} = Y^{target} X^T \left(X X^T \right)^{-1} \tag{10}$$

Although this solution is already computationally less expensive, it can be easily extended to another form which is usually a more stable first choice when it comes to offline training. To understand how, let's look into one of the main problems when training ESNs to predict time series: over-fitting. When a network is overfitted, it learns very well —usually perfectly— the training set, but then cannot generalize to unseen data because it became very sensitive to deviations from the exact conditions in which training was carried out. Many times, this is associated to large output weights that exploit and amplify tiny differences among the dimensions of the reservoir states. This is a particular big problem in setups where the network receives its output as the next input, as it is the case for time series prediction in generative mode. To overcome this problem a regularization term can be added to the error in Eq.(2). Instead of just minimizing the MSE, we include a penalty for large weights:

$$E_{ridge} = \frac{1}{N_y} \sum_{i=1}^{N_y} \left(\frac{1}{T} \sum_{t=1}^{T} \left(y_i^{target}(t) - y_i(t) \right)^2 + \beta \left\| w_i^{out} \right\|^2 \right) \tag{11}$$

where w_i^{out} denotes the ith row of W^{out}, $\|\cdot\|$stands for the Euclidian norm and β is the

regularization coefficient. Eq.(11) represents a compromise between having small error during training and ending up with relatively small output weights. Replacing $y_i(t)$ for its expression in Eq.(3) with only one output unit, and deriving with respect to W^{out}, we arrive to:

$$W_{opt}^{out} = Y^{target} X^T \left(XX^T + \beta I \right)^{-1} \tag{12}$$

This is known as the ridge regression method or regression with Tikhhonov regularization. Notice that choosing $\beta = 0$ removes the regularization, turning ridge regression into the generalized linear regression of Eq.(10). Finally, another way of having regularization that was successfully implemented in Jaeger's original work [18] consisted on the addition of a scaled white noise to $x(t)$ in Eq.(3). Like in ridge regression, this method emphasizes the diagonal of $\left(XX^T\right)$, but it also propagates through W^{res}, modeling better the effects of noisy signals in the reservoir [39]. Recently, it has also been suggested that regularization of the reservoir weights W^{res} to reduce the recurrent connection strengths could help making the ESN more stable [43, 37].

To find the best training scheme for our time series prediction tasks we experimented with a simple ESN trained both online, by mean of a delta rule expression; and offline, with simple LSE method, noise and ridge regularization. In agreement with the results found in [25], offline methods showed to be faster and computationally less expensive, but also lead to more stable results. Instability in online learning can be probably attributed to an excessive increase of the reservoir matrix spectral value, a problem that is overcome particularly well when regularization is applied in the offline method. More or less the same performance was obtained during testing of the ESNs when training was carried with ridge regression or noise regularization. However, ridge regularization was preferred because its implementation involves only the last step of the training, when the states have already been computed. This means that if we want to perform a grid-search to look for the optimal regularization coefficient, the same states can be used to extract W_{opt}^{out} with the different coefficients. Besides, regularization with noise affects Eq.(3), so it would require computing the states over the whole training set for each noise scaling factor. Therefore, in all the results presented in this thesis we applied a supervised learning to offline-train the readout weights with the ridge regression formula of Eq.(12).

As a motivation for the next section, we would like to point out how all of the training methods so far focus on the optimization of the readout weights. This is one of the great advantages of RC, specially when offline learning methods are used, but it also leaves a lot of freedom within the ESN architecture. After all, reservoirs are initially constructed

with a set number of units but randomly assigned connection weights: how likely is that this initial choice of reservoir is really optimal for a given task? The next section will immerse us into a new type of learning developed over strongly established biological foundations, which aims to adapt either the connection weights or the reservoir neurons themselves according to the inputs arriving to the network.

2.4 Neural plasticity: biological and artificial implementations.

2.4.1 Plasticity in the brain.

The term plasticity has been used in brain science for well over a century to refer to the suspected changes in neural organization that may account for various forms of behavioral changes, either short-lasting or enduring. These changes include maturation, adaptation to a mutable environment, specific and unspecific kinds of learning, and compensatory adjustments in response to functional losses from aging or brain damage [2]. Mechanisms of plasticity in the brain can be grouped into two large categories: synaptic and nonsynaptic plasticity; both affecting the connecting neurons in the short and long run.

On the one hand, synaptic plasticity deals directly with the strength of the connection between two neurons, including the amount of neurotransmitter released from the presynaptic neuron, and the response generated in the postsynaptic neuron. To date, the most accepted and studied mechanism of this type is the spike time-dependent plasticity (STDP), in which connection strengths are adjusted based on the relative timing of a particular neuron's output and input action potentials. This includes the already mentioned Hebbian plasticity, where synapses are strengthened when a presynaptic spike occurs just before a postsynaptic spike; but also anti-Hebbian learning, where with the opposite timing or in the absence of a closely timed presynaptic spike, synapses are weakened. Biochemically, this type of plasticity appears in excitatory synapses through long-term potentiation (LTP) and long-term depression (LTD) mechanisms, leading to structural changes in the pre and post-synaptic cells [44, 45, 46]. However, other studies suggest that there could also be other non-Hebbian types of synaptic plasticity operating in the brain [47, 48]. Short-term plasticity, acting on a timescale of tens of milliseconds to a few minutes has been documented as well, with neural facilitation, synaptic augmentation, post-tetanic potentiation and synapse fatigue being the main mechanisms leading to transitory changes in the synapse efficacy [49].

On the other hand, nonsynaptic plasticity involves modification of the intrinsic excitability of the neuron itself, operating through structural changes. These changes usually affect voltage-dependent membrane conductances in the axon, dendrites or soma of the neuron. Experimental evidence of this type of plasticity has been since long found in biological networks [50, 51], but now recent studies also suggest that these two forms of plasticity may act in a synergistic way, creating an intimate link between LTP/LTD and at least one form of potentiation/depression of intrinsic excitability [52]. Besides, studies in both vertebrate and invertebrate models suggest that intrinsic plasticity plays a fundamental role regulating other forms of synaptic plasticity, protecting against saturation or suppression of the circuit activity as a whole [53].

Without entering in too much detail about the biochemical mechanisms underlying plasticity, in the following section we will address the mathematical models that we will be using to implement the different rules in our Echo State Networks.

2.4.2 Plasticity in Echo State Networks.

One of the main characteristic features of an ESN, and of RC models in general, is the existence of a reservoir of neurons at the hidden layer whose states change with the given input. In the classic scheme, the output of the reservoir is adjusted through the readout weights so to match the target, but there is no direct training performed over the reservoir (which remains randomly connected). This initial simplification has led to all sort of studies trying to assess the way of achieving the optimal reservoir for a given task [21, 24, 23, 54, 25, 55]. Here, we focus on the optimization of the reservoir through unsupervised learning involving plasticity rules. These rules aim to modify either the strength of the synapses (synaptic plasticity) or the excitability of the reservoir units (nonsynaptic plasticity) based on the activities stimulated by the input, thus embedding within the reservoir some of the structural information carried by the input signal.

We evaluated numerically three plastic rules: two implementations of synaptic plasticity (anti-Oja and anti-Hebbian rule) and one of nonsynaptic (Intrinsic Plasticity rule). First, to develop a mathematical description of the Hebbian rule, we will build upon its two main ideas:

- If two neurons, located at the opposite sides of a synapse, are activated concurrently (synchronously), then the value of the synaptic weight will increase.

- If two neurons, located at the opposite sides of a synapse, are activated at different times (asynchronously), then the value of the synaptic weight will decrease.

If we denote by w_{kj} the synaptic weight (a number measuring the synaptic strength) of a Hebbian synapse connecting neurons k and j —where j triggers the activity of k— then according to Hebb the changes in the synaptic strength at time t will be a function of the pre and post-synaptic activities at that time:

$$\Delta w_{kj}(t) = F(x_j(t), y_k(t)) \tag{13}$$

where $x_j(t)$ represents the activity of the pre-synaptic neuron and $y_k(t)$ the activity of the post-synaptic one. Hebbian and anti-Hebbian plasticity rules are alike in what concerns the above expression, only changing the updating rule from $w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj}(t)$ for the Hebbian case to $w_{kj}(t+1) = w_{kj}(t) - \Delta w_{kj}(t)$ for the anti-Hebbian. Developing in Taylor series the expression in Eq.(13), we can rewrite these synaptic changes as a polynomial of the form:

$$\Delta w_{kj}(t) = a_0 + a_1 x_j(t) + a_2 y_k(t) + a_3 x_j(t) y_k(t) + a_4 y_k(t)^2 + ...$$

The simplest approach to ensure Hebbian-like behavior is to restrict the rule to the first second order term:

$$\Delta w_{kj}(t) = \eta x_j(t) y_k(t) \tag{14}$$

which expresses the change as a product of pre and post-synaptic activity, with a parameter η accounting for the learning rate. Sometimes referred to as the activity-product rule, this is the most pure version of the original Hebbian learning. It emphasizes the correlative nature of Hebbian synapse but also evidences its biggest flaw: as connection strength grows in accordance with Hebb's postulate, activity will eventually spread and increase uncontrollably throughout the network. To limit the growth of the synaptic weight, one of the possibilities is to introduce a nonlinear forgetting factor. This is the idea behind the generalized Hebbian rule:

$$\Delta w_{kj}(t) = \eta y_k(t) x_j(t) - \alpha y_k(t) w_{kj}(t) \tag{15}$$

With this formula, if $x_j(t) < \alpha w_{kj}(t)/\eta$ then the modified synaptic weight $w_{kj}(t+1)$ decreases proportionally to a post-synaptic activity $y_k(t)$. In turn, when $x_j(t) > \alpha w_{kj}(t)/\eta$ the synaptic weight $w_{kj}(t+1)$ increases proportionally to $y_k(t)$. Finally, a

further modification known as Oja's rule can be derived by modifying the forgetting term to be of second order in $y_k(t)$ [56]:

$$\Delta w_{kj}(t) = \eta y_k(t) \left(x_j(t) - y_k(t) w_{kj}(t) \right) \tag{16}$$

According to [21], the extra factor with respect to Eq.(15) helps providing results which are more stable in terms of variance.

We now consider the implementation of intrinsic plasticity, which requires a bit more of mathematical insight. While synaptic rules aim to change the connection weights, IP rules update the activation function of the individual neurons. These types of algorithms operate therefore in a local way, trying to maximize the amount of input information contained in the output of a neuron. Based on the idea that every single neuron tries to balance the conflicting requirements of maximizing its information transmission while obeying constraints on its energy expenditure, Jochen Triesch proposed a mathematical learning rule following three principles [22]:

(1) Information must be maximal, meaning that the output of the neuron should contain as much information of the input as possible. This is achieved by maximizing the entropy of the output firing rates.

(2) Output distributions are subject to constraints, which are given first of all by the limited output range of the neuron, but can also be related to the limited energy available.

(3) To adapt the neurons intrinsic parameters, a biological neuron is only able to adjust its internal excitability, and not the individual synapses.

In physical terms, the information maximization principle corresponds to a maximization of the entropy of the output distribution of each neuron. In combination with the second principle this leads to maximum entropy (ME) distributions with certain fixed moments. We can express the difference between the output and the desired ME distribution using the Kullback–Leibler divergence:

$$D_{KL}\left(\widetilde{p}, p\right) = \int \widetilde{p}(y) log \left(\frac{\widetilde{p}(y)}{p(y)} \right) dy \tag{17}$$

where $\widetilde{p}(y)$ is the actual probability density of the neuron's output activity and $p(y)$ the desired probability density function. In his original work Triesch derived the IP rule for Fermi activation functions and exponential desired distributions [22], but just a few

years later Schrauwen et al extended the rule to account for neurons with hyperbolic tangent functions [23]. Defining intrinsic parameters a (gain) and b (bias) for the non-linearity, they updated each neuron's state through the following expression:

$$y = f_{gen}(x) = tanh(ax + b) \tag{18}$$

In [23], they took the desired probability density function $p(y)$ to be a Gaussian distribution, as it is known to be the ME distribution for a given mean and standard deviation with support in $(-\infty, \infty)$. Minimization of the KL divergence with this constraint in the output activity leads to the following online learning rule with stochastic gradient descent

$$\Delta b = -\eta \left(-\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2} \left(2\sigma^2 + 1 - y^2 + \mu y \right) \right), \tag{19}$$

$$\Delta a = \frac{\eta}{a} + \Delta b x \tag{20}$$

where η is the learning rule and μ and σ the mean and standard deviation of the targeted distribution.

Finally, we will combine two of the above rules —the anti-Oja and IP algorithms— to assess the performance of an ESN when these two types of plasticity act synergistically. For this combination, there are three natural ways in which the training can be carried: applying both rules simultaneously to update the intrinsic parameters and connections weights after each input; modifying first the connections through the synaptic plasticity and then applying the IP rule; or the opposite way around, changing first the intrinsic excitability of the neurons and then the synapses strength among them. From all the alternatives, the application of the anti-Oja rule through the whole training set followed by the same training of the IP rule showed to retrieve the best performance. Therefore, in the results presented in Sec. 3 we will be combining the two plastic rules following the aforementioned order. Let it be noticed that computational models combining the effect of synaptic and non-synaptic plasticity have been already suggested in literature for simple model neurons [57], FFNNs [58] and RNNs [58, 59, 60]. However, we found that a simple combination of two "known" plastic rules could ease the tractability of the results, while allowing fairer comparisons against the other plasticity models.

2.5 Prediction of a chaotic time series.

Before we get into the challenging task of predicting the behavior in a chaotic dynamical system, let's first take a look in what the word "chaotic" actually means. Although there is no universally accepted definition of the word chaos, in his book *Nonlinear Dynamics and Chaos* Strogatz provides us with a good start:

> *"Chaos is aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions."* [61]

Thus, according to the author there are three main components that build up to create chaotic dynamics:

1. "Aperiodic long-term behavior": trajectories do not settle down into fixed points, periodic orbits or quasiperiodic orbits as $t \to \infty$.
2. "Deterministic": the system has no random or noisy input parameters.
3. "Sensitive dependence on initial conditions": nearby trajectories separate exponentially fast.

From the above three conditions, the first one is generally difficult to prove as it involves the behavior of the system at $t \to \infty$, while the second one can be easily assured given the equation governing the dynamics. However, it is the sensitive dependence on initial conditions that characterizes the most a chaotic series, being also strongly linked to predictability. Now understanding the concept is one thing, but how can we give a quantitative measure of it? Although since its discovery many techniques have been proposed to determine if a system is chaotic or not [62], we will discuss here what is probably the most common way of measuring chaos: the Lyapunov exponent. Suppose we take two nearby points within the phase space of our dynamical system as initial conditions, say $\mathbf{x}(0)$ and $\mathbf{x}(0) + \delta(0)$, where $\delta(0)$ accounts for a very small initial separation between the two points. Then according to the third property of chaos, if nearby trajectories separate exponentially fast this means that:

$$\|\delta(t)\| = \|\delta(0)\| \, e^{\lambda t} \tag{21}$$

The exponent λ at which this divergence takes place is what is commonly known as Lyapunov exponent, which fulfills $\lambda > 0$ for chaotic behavior. However, as pointed out in [61], two important remarks must be done:

- λ may depend slightly on the trajectory considered, so to obtain its real value we must average over many different initial points.
- For an n-dimensional system there are actually n different Lyapunov exponents. To understand this, we can think of an n-dimensional sphere of initial conditions which deforms in time into an n-dimensional ellipsoid as trajectories diverge. Each axis of the ellipsoid grows with a different λ_k, but for large t the diameter of the ellipsoid is basically governed by the largest positive λ_k, with $k = 1, ..., n$. In an abuse of notation, $\lambda_{max} = max(\lambda_k)$ is often referred as the Lyapunov exponent of the series.

Now that we have set the basis to understand the difficulties of predicting the evolution of a chaotic dynamical system, we present our task at hand. This will consist on the prediction or continuation of points in the Mackey-Glass series, generated[4] from the following delay differential equation:

$$\frac{dx}{dt} = \left[\frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{\beta}} - \gamma x(t)\right] \tag{22}$$

where τ represents the delay and the parameters were set to $\alpha = 0.2$, $\beta = 10$ and $\gamma = 0.1$, a common choice for this type of prediction tasks. Because this series presents a chaotic attractor when $\tau > 16.8$, we constructed two different sets: one with $\tau = 17$ (MG-17), often used as an example of mildly chaotic series; and a second series with wilder chaotic behavior, choosing $\tau = 30$ (MG-30). The maximum Lyapunov exponent for each series was also computed following the method presented in [63], resulting in a value $\lambda_{max} \approx 0.006$ for the MG-17 and $\lambda_{max} \approx 0.024$ for the MG-30. From this value we can calculate for each series the Lyapunov time, i.e. the inverse of the maximum Lyapunov exponent, resulting in $\Theta_{17}^{Max} \approx 170$ and $\Theta_{30}^{Max} \approx 40$ for the MG-17 and MG-30, respectively. This quantity is particularly interesting as it provides a rough estimate of the average predictability of the time series. Notice that each of the above times are given in normalized units, because we took 1 as the step size between points in the constructed series.

From the MG-30 we extracted at random $T = 6000$ consecutive points, and used them as training set, their values normalized to lay on the range [0,1]. For the mildly chaotic

[4]For this purpose we used *Matlab* dde23 delay differential equation solver, generating 10000 points with an initial washout period of 1000. The step size between points in the extracted series was set to 1, although it was initially computed with step size 0.1 and then sampled every 10 points. The absolute error tolerance was set to $1e-16$, as suggested in [18] for this kind of prediction task.

MG we found that 4000 training points suffice to obtain good results. The task assessed consisted on the continuation of the series from the last input of the training set, so the target series was defined as $y^{target} = [u_2, u_3, ..., u_{T+1}]$ for an input $u = [u_1, u_2, ..., u_T]$. We kept the internal states of the ESN at each stage of the training and only after passing all the set we applied Eq.(12) to derive the optimal output weights.

When implementing any of the plasticity rules we ran the corresponding online unsupervised learning using the same T points as in the training, passing over the whole training set a number of times (epochs) which varied according to the model. Finally, we collected the states of the reservoir after a last presentation of the training set and computed the new optimal readout weights as described above for the supervised learning method.

For testing performance, we initially provided the ESN with the last input of the training set, u_T, then ran the network in generative mode for a number F of steps. Predictions were finally re-scaled to the original series range of values before testing their accuracy. To quantify the error for this task, we made use of two different magnitudes:

- The root mean square error (RMSE) over the predicted continuation of the series:

$$RMSE = \sqrt{\sum_{t=0}^{F} \left(y(t) - y_{target}(t)\right)^2} \tag{23}$$

- The furthest point up to which the trained ESN is able to continue the series without significantly deviating from the original. The tolerance for significant deviation was taken to be $\varepsilon = 0.02$ for the MG-17 series and $\varepsilon = 0.03$ for the MG-30, which are approximately a 2% of the maximum value difference between two points in the MG-17 and MG-30 series, respectively.

2.6 Memory capacity task.

The concept of memory capacity (MC) is based on the network's ability to retrieve past information from the reservoir using the linear combinations of reservoir unit activations. To assess the ability of each model to restore previous inputs fed into the network, we computed the (short-term) MC as introduced by Jaeger in [40]:

$$MC = \sum_{k=1}^{\infty} MC_k = \sum_{k=1}^{\infty} \frac{cov^2(u(t-k), y_k(t))}{var(u(t)) \cdot var(y_k(t))} \tag{24}$$

where *cov* and *var* denote covariance and variance, respectively. In the above expression, $u(t-k)$ is the input presented k steps before the current input $u(t)$, and $y_k(t) = w_k^{out}x(t) = \tilde{u}(t-k)$ is its reconstruction at the output unit k, with a value $MC_k \sim 1$ meaning that the unit is able to accurately reconstruct the input fed to the network k steps ago[5]. Thus, the sum of all MC_k represents an estimation of the number of past inputs the ESN is able to recall. Although the sum runs to infinity in the original definition —accounting for the complete past of the network— in practice the data fed is finite and it will suffice with setting $k_{max} = L$, with L being the number of output units of the ESN. Each of the L output units is independently trained to approximate past inputs with a different value of k. A theoretical limit for the memory capacity was derived in [40] to be $MC_{max} \approx N - 1$, with N the number of reservoir neurons.

[5]Let it be noted that the expression of MC_k is nothing but the squared Pearson correlation coefficient for the real and reconstructed past input in unit k.

3 Results

In the following section we will be presenting the results obtained when applying ESNs over three different tasks, analyzing the difference in performance before and after implementation of the plastic rules. The tasks include the continuation of a mildly and strongly chaotic time series, MG-17 and MG-30, respectively; as well as a memory capacity task to evaluate the ability of the network to retrieve past inputs. Nevertheless, before tackling these problems we will begin by fine-tuning some of the hyper-parameters that we have seen can have a great impact on the ESN performance.

3.1 Hyper-parameter optimization.

One of the biggest drawbacks of Echo State Networks is their high sensitivity to hyper-parameters choice. Following the guidelines in [39], we focused on tuning the reservoir size N, the input scaling ε, the spectral radius ρ and the regularization parameter β; finding those values leading to the best performance for each type of ESN. Weights in the reservoir, input and output layers were initialized randomly according to a uniform distribution between -1 and 1. Sparseness in the reservoir matrix was set to 90%, meaning that only 10% of all connections held initially a non-zero value. In the case of plastic models, an extra tunable hyper-parameter η describing the learning rate in the update rules is included. When IP was implemented, we found that results where more stable when using $\mu = 0$ and $\sigma = 0.3$ as the mean and variance of the targeted distribution for the neuron states.

Thus, the optimal set of hyper-parameters was defined as the tuple $\{\rho, N, \varepsilon, \beta, \eta\}$ minimizing the RMSE along 500 predicted points of the MG-17 series (see Table 1). Averages were taken over 20 independent ESNs of the same type, each continuing a different part of the series. The errors were computed as the standard deviation of the resulting averages. Finally and for the sake of comparison between different models, we chose a common non-optimal, but generally well-performing set of hyper-parameters $\{\rho = 0.95, \varepsilon = 1, \beta = 10^{-7}, \eta = 10^{-6}\}$ for all of them, with $N = 300$ for the MG-17 series prediction and $N = 600$ for the MG-30. Finally we would like to highlight two important aspects of the grid-search performed for the hyper-parameter optimization:

1. Although bigger reservoirs with e.g. 600 neurons led to better prediction results, medium reservoirs of 300 neurons were generally preferred due to the large computational time involving operations with bigger weight matrices.

2. For the results showed in Table 1, the number of training epochs for each plastic model was fixed to $n_{ep} = \{5, 5, 18, (4+18)\}$ for the anti-Hebbian, anti-Oja, IP and (anti-Oja + IP rule). These values ensured that the resulting ESNs were stable for all the possible choices of reservoir size, but do not represent the optimal number of epochs, which will be later investigated for the selected set of hyper-parameters.

		N	ρ	ε	η	RMSE	FPP
Non-Plastic	Optimal	600	0.99	1.25	—	0.03 ± 0.03	276 ± 134
	Chosen	300	0.95	1.0	—	0.09 ± 0.06	143 ± 75
Anti-Hebbian	Optimal	600	0.95	1.0	10^{-6}	0.013 ± 0.009	426 ± 79
	Chosen	300	0.95	1.0	10^{-6}	0.03 ± 0.04	385 ± 113
Anti-Oja	Optimal	600	0.99	0.75	10^{-6}	0.010 ± 0.008	446 ± 81
	Chosen	300	0.95	1.0	10^{-6}	0.04 ± 0.05	298 ± 135
IP	Optimal	600	0.80	0.5	10^{-5}	0.012 ± 0.008	399 ± 71
	Chosen	300	0.95	1.0	10^{-6}	0.04 ± 0.03	271 ± 89
Anti-Oja + IP	Optimal	600	0.99	1.25	10^{-6}	0.009 ± 0.006	396 ± 115
	Chosen	300	0.95	1.0	10^{-6}	0.02 ± 0.02	342 ± 114

Table 1: Optimal and chosen hyper-parameter values for each model. The RMSE and Furthest Point Predicted (FPP) as described in Sec. 2.5. are given for comparison. The swept was performed within the following hyper-parameter space: $N = \{100, 300, 600\}$, $\rho = \{0.50, 0.75, 0.90, 0.95, 0.99\}$, $\varepsilon = \{0.50, 0.75, 1.00, 1.25\}$, $\eta = \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}$. For each of the possible combinations, read-out weights were fitted using $\beta = \{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ as regularization parameter, finding $\beta_{optimal} \sim 10^{-7}$ for all cases.

3.2 Performance in prediction task with MG-17.

We begin by comparing the performance of the simple ESN against its plastic versions when asked to continue, i.e. generate autonomously, a mildly chaotic Mackey-Glass series. Because our aim is to draw a performance comparison within a reservoir and training procedure with fixed parameters, we first have to find the optimal number of training epochs for each plasticity model, i.e. the number of times we should pass through the complete training set to obtain optimal results in the test. To this end, in

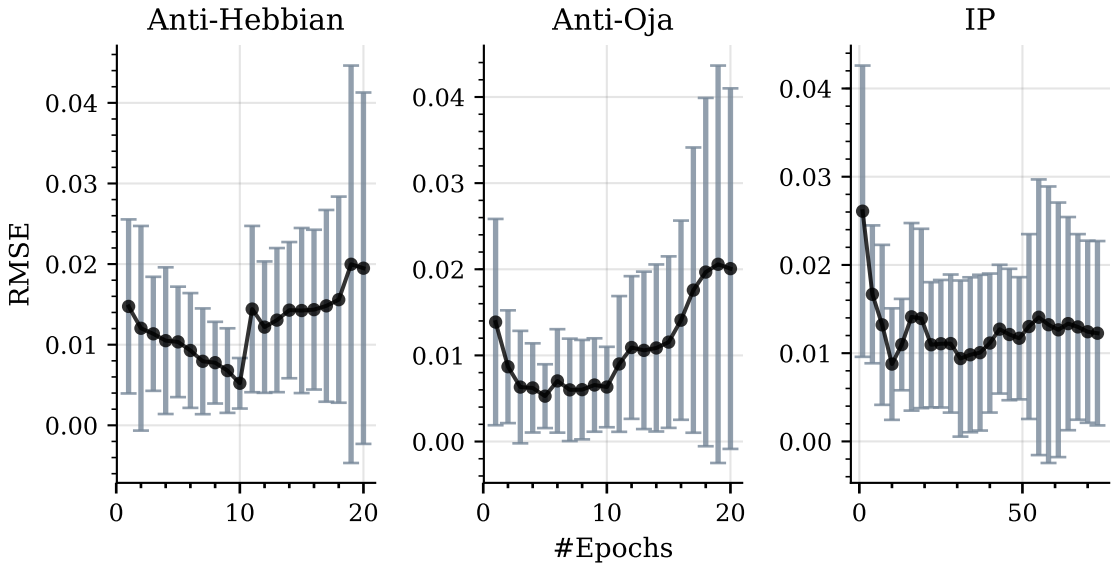


Figure 4: Evolution of the total RMSE over 300 points predictions of the MG-17 with the number of episodes during training of different plasticity rules. At each epoch, averages were computed over 20 independent realizations of a 300 neuron ESN.

Fig.4 we show for each plasticity rule model the RMSE accumulated over 300 points used as testing sequence after every epoch of the training, with each result averaged over 20 independent realizations of a 300 neuron ESN. Thus, we find $n_{ep} = \{10, 8, 18\}$ to be the optimal number of epochs for the anti-Hebbian, anti-Oja and IP rule models, respectively. Beyond this optimal number of epochs, we see that the RMSE tends to increase when synaptic plasticity was implemented, while remaining fairly stable for the IP rule.

Once the plasticity was optimally implemented, we trained the output weights of the different ESN models following the supervised learning described in Sec. 2.3. The resulting networks were tested to continue the original time series up to 50, 100, 200, 400 and 500 points following the last input of the training. As an example, we show the predicted and original chaotic time series along 600 steps in Fig.5. The corresponding results on the RMSE for each task and model are given in Fig.6. We see that not only the average error decreases when plasticity rules are implemented, but so does the uncertainty in the results, highlighting the ability of the plastic rules to make the network less dependent to the initial conditions. In Fig.7 we also estimated the furthest point that each type of ESN was able to reconstruct accurately from the sequence of points predicted. For example, in a 500 points continuation task, a furthest accurate predic-

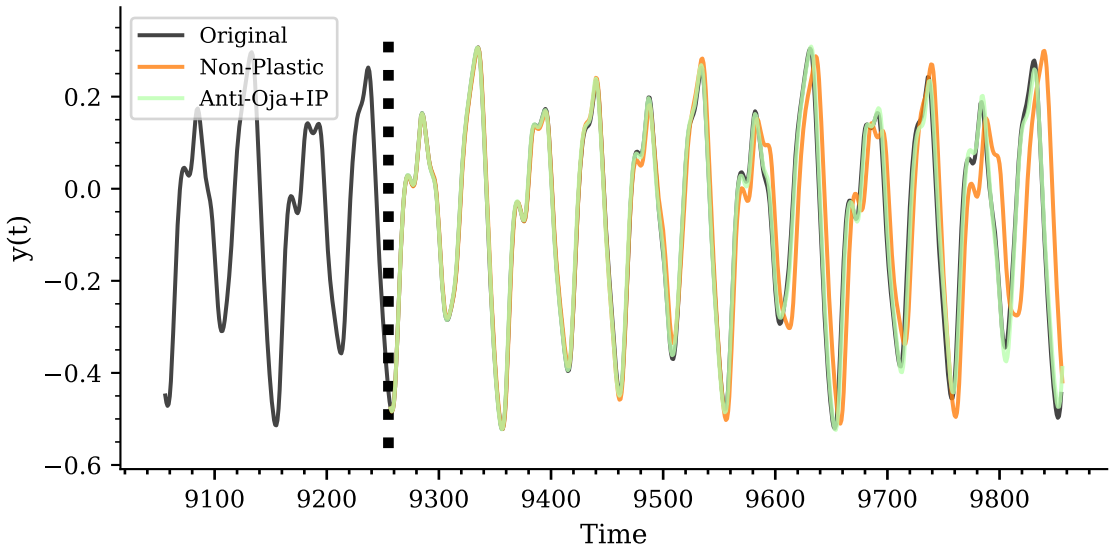


Figure 5: Visualization of the predicted series along 600 steps after the last training point, with the original series plotted in black. The results correspond to an ESN with no plasticity implemented (orange line) and the same network after application of anti-Oja + IP rules (green line). While the after-plasticity prediction overlays the original series in most of the points, in this example the non-plastic model diverges considerably after a few oscillations.

tion of 300 means that, on average, the generated 500 points series begins to diverge from the original one after 300 points. We can see in this plot how the combination of anti-Oja and IP rule outperforms the rest of models, predicting correctly up to three times the estimated Lyapunov time of the series.

Finally, we wanted to ensure that the constructed time series were equivalent to the chaotic Mackey-Glass, and not some sort of long-term periodic solution with similar looks. For this purpose we let the trained ESNs run in generative mode for 600 steps, using the output to reconstruct the chaotic attractor. The reconstructed attractors for the original time series and the ESN with different plasticity rules can be seen in Fig.8. While it is true that the resulting attractors do not reproduce exactly the original time series, we can dare to say that the chaotic behavior is fairly better emulated after implementation of plasticity rules. For ease of visualization we only plotted the reconstructed attractors corresponding to the anti-Oja and IP rules, but similar results are obtained with the anti-Hebbian rule and anti-Oja+IP combination.

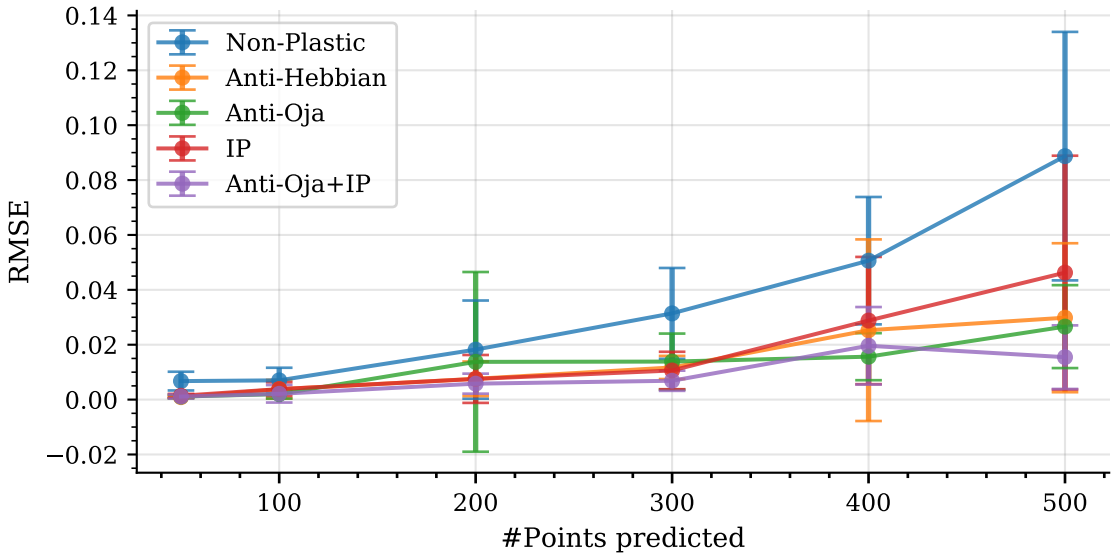


Figure 6: Evolution of the RMSE with the number of points predicted for each of the different implementations of an ESN. All networks were initialized with the same hyper-parameters and $N = 300$ neurons, the errors computed as the standard deviation over 20 independent realizations.

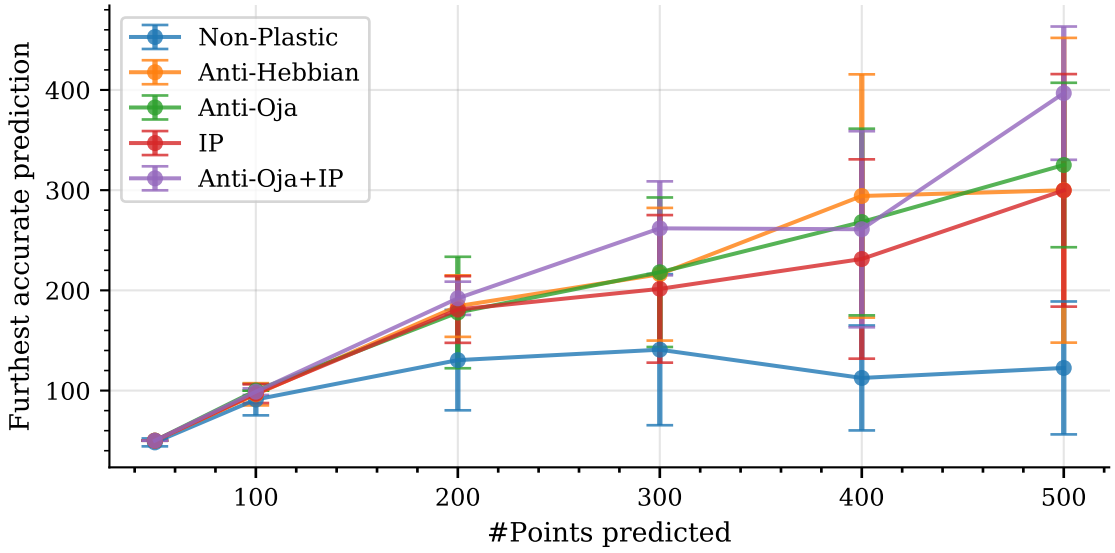


Figure 7: Evolution of the furthest accurate prediction for different number of total predicted points within each model of the ESN. All networks were initialized with the same hyper-parameters and $N = 300$ neurons, the errors computed as the standard deviation over 20 independent realizations. The tolerance to determine whether a prediction was good enough or not was set to 0.02, which represents a 2% of the total value range of the series.

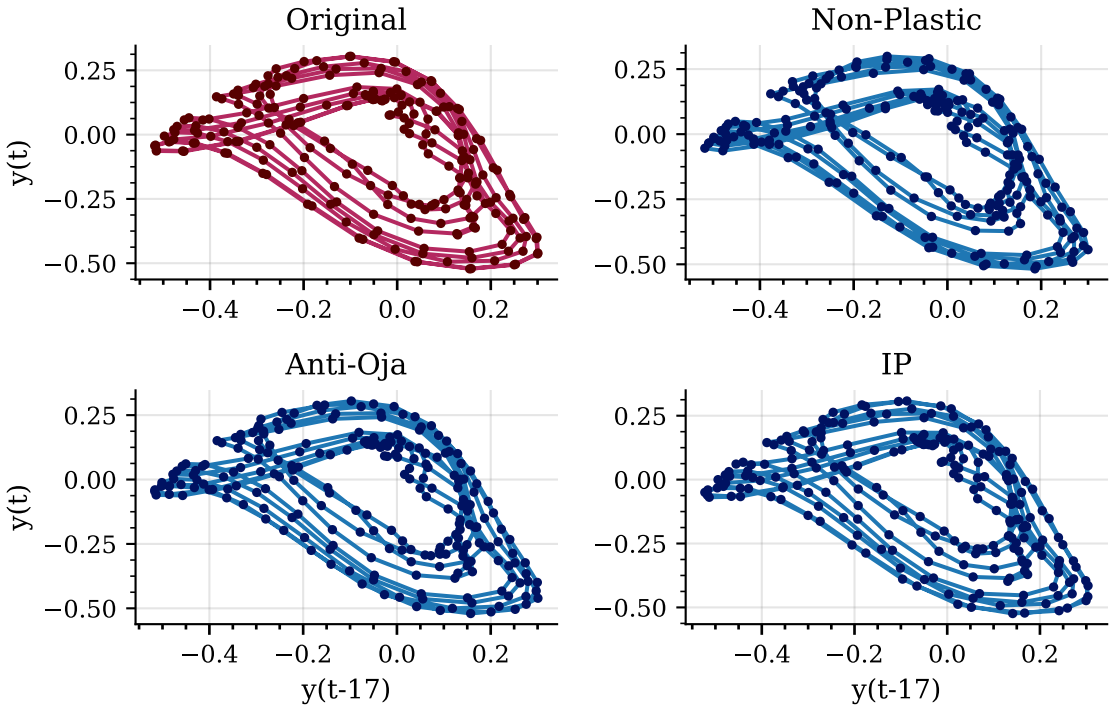


Figure 8: Reconstruction of 600 points of the Mackey-Glass attractor by a "simple" ESN, an ESN with Oja's rule implemented and one with IP plasticity. On the upper left corner the original attractor for the predicted points is plotted for comparison.

3.3 Performance in prediction task with MG-30.

We apply now the same procedure to test the effects of plasticity in a series which is regarded as highly chaotic. For the same hyper-parameters but now a reservoir size of 600 neurons, the optimal number of epochs was found to be {4,2,75} for the anti-Hebbian, anti-Oja and IP, respectively. Because the reconstructed series diverge in some cases very fast from the original ones, specially without plasticity (see Fig.9), we limit our results to a maximum window prediction of 200 points for the test. Fig.10 and Fig.11 show the RMSE and furthest accurate prediction for each model, as defined for the MG-17 before. We can see how in this case results vary greatly even within the same plasticity implementation, but a common trend is the huge improvement in performance (up to two orders of magnitude in the RMSE) when plasticity is implemented.

As with the mildly-chaotic MG, we reconstruct in Fig.12 the attractor for the 200 points predicted. The differences among the simple ESN and the plastic ones become now more evident, with the non-plastic ESN not being able to reproduce at all the complex dynamics of the original series.

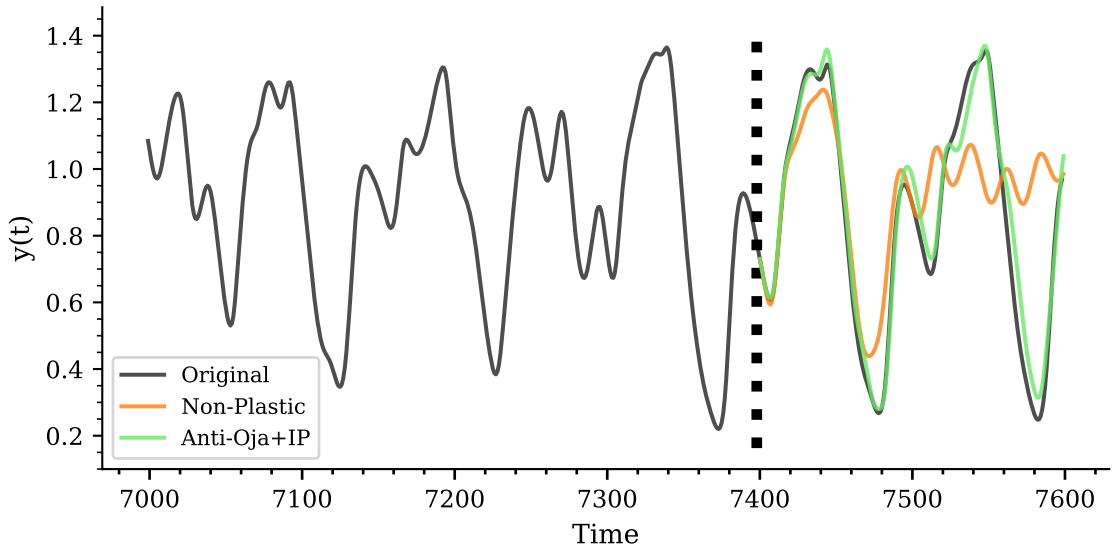


Figure 9: Visualization of the predicted series along 200 steps after the last training point, with the original series plotted in light black. The results correspond to an ESN with no plasticity implemented (orange line) and the same network after application of anti-Oja + IP rules (green line). In this example the non-plastic model diverges considerably after a few steps.

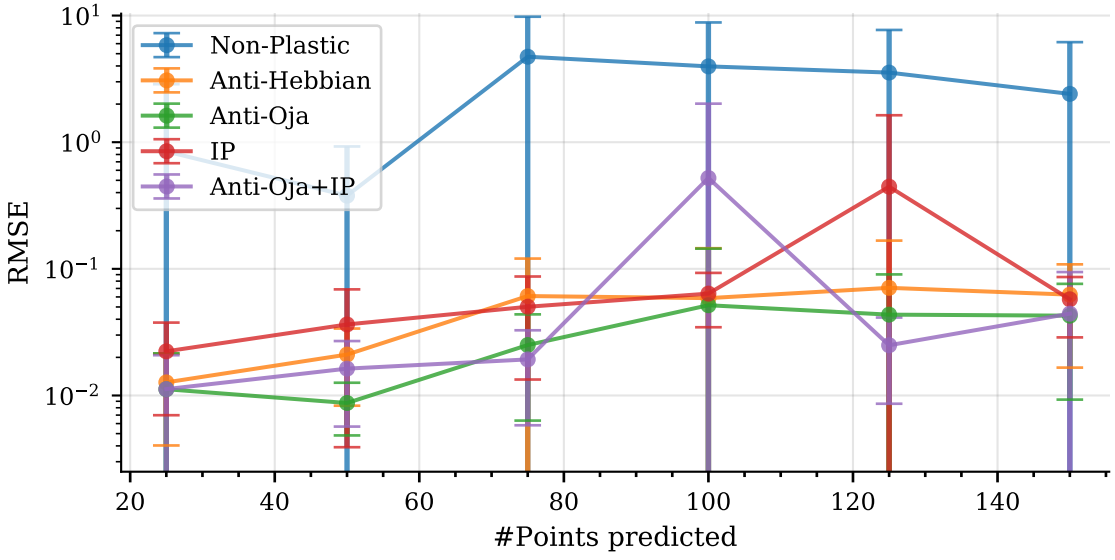


Figure 10: Evolution of the RMSE with the number of points predicted for each of the different implementations of an ESN. All networks were initialized with the same hyper-parameters and $N = 600$ neurons, the errors computed as the standard deviation of 20 independent realizations.

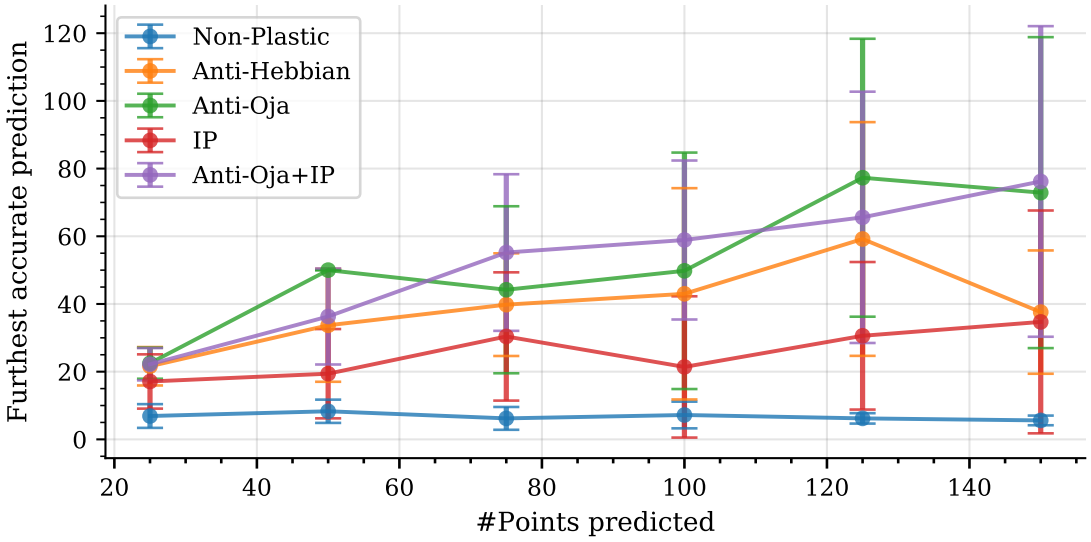


Figure 11: Evolution of the furthest accurate prediction for different number of total predicted points within each model of the ESN. All networks were initialized with the same hyper-parameters and $N = 600$ neurons, the errors computed as the standard deviation of 20 independent realizations.

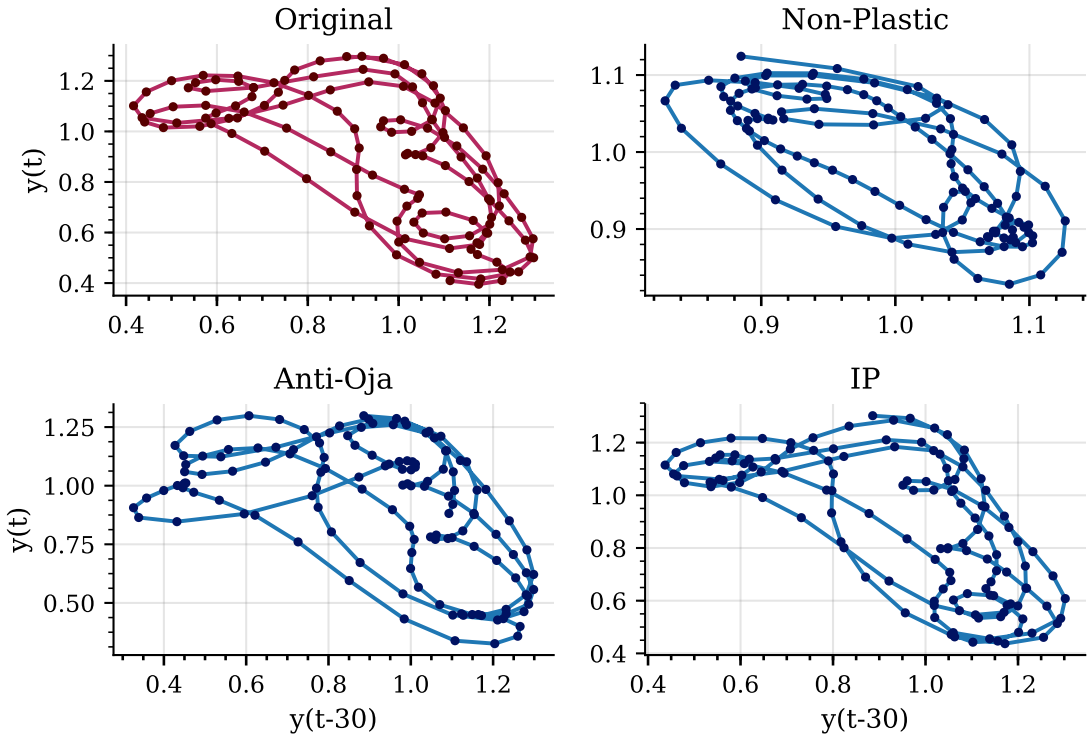


Figure 12: Reconstruction of 200 points of the Mackey-Glass attractor by the simple ESN, an ESN with Oja's rule implemented and one with IP plasticity.

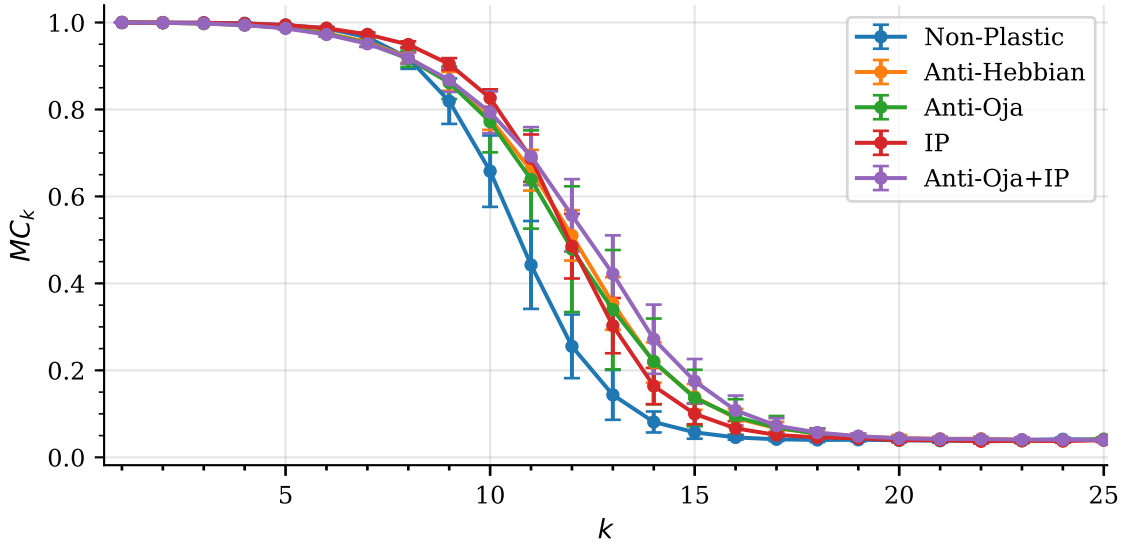


Figure 13: Forgetting curves for the different models of ESNs studied. Averages of the values were taken over 20 independent realizations of each model, the error expressed as the standard deviation of the results.

3.4 Performance in memory capacity task.

To draw a comparison for the influence of the plasticity on the memory capacity with the results in [64], we constructed single input node ESNs with $L = 300$ output nodes and $N = 150$ reservoir neurons. For this task, we fed as input a uniformly random time series of 4000 steps, drawn from the interval [-1,1]. Fig.13 shows the forgetting curve for an ESN before and after implementation of the different plasticity rules, representing the values of MC_k with the number of time steps k to the current input. We see how information about points recently given as input is somehow imprinted in the reservoir states, so the network can retrieve them easily, whereas information about older inputs begin to fade considerably after a 10-steps span. Again, we notice how models with implemented plasticity outperform the original ESN, with memory fading faster in the latter case. In Table 2 we gather the MC computed for each type of plasticity, before and after its implementation, thus giving a numerical value to this tendency observed in the plot. It is worth noting that our results agree with the average values observed in [64], where the ESNs were driven to criticality to prove that best performance is achieved at the edge of chaos.

	Non-Plastic	Anti-Hebbian	Anti-Oja	IP	Anti-Oja + IP
MC	21.6 ± 0.4	22.8 ± 0.3	22.7 ± 0.7	22.8 ± 0.4	23.1 ± 0.5

Table 2: Memory Capacity for an ESN with 150 nodes and 300 output neurons with and without implementation of the different plastic rules.

3.5 Understanding the effects of plasticity.

How can we then explain the observed improvement in the performance of the ESN after the implementation of plasticity rules? And more important, can we say something about why one rule may perform better than other? To analyze these questions, we first cast our attention into the dependence of the performance with the number of training epochs. For the synaptic plasticity rules we saw in Fig.4 that the error decreases as the training proceeds up to around 10 epochs, and then the RMSE begins to increase again. If we now plot in Fig.14 the spectral radius of the reservoir matrix after each sweep of the training data, we see that the source of instability beyond 10 epochs could be related to the increase of the spectral radius over 1. This has been often regarded as a source of instability due to the lost of the "echo state" property [18, 65, 39], although later studies proved that the latter can be maintained over unitary spectral radius depending on the input [66, 67]. Here we see that the optimal number of epochs coincides with the transition area between a low spectral radius ($\rho < 1$) and a high one ($\rho > 1$), seeming to suggest that optimal performance can be found at the edge of instability. This appears to agree with the results presented in [64], where it was suggested that information processing in ESNs is maximized at the edge of chaos.

If we look now at the performance evolution of the IP rule in Fig.4 we see a different trend, with the error decreasing up to a certain number of epochs and then remaining fairly unchanged with further sweeps over the training data. Since the reservoir matrix is not affected by the nonsynaptic plasticity, there is no use in looking for changes in its spectral radius as we did before. Instead, we can define an effective reservoir as done it in [23]:

$$\widetilde{W}^{res} = diag(\mathbf{a})W^{res} \tag{25}$$

This allows us to study the variations in the effective spectral radius with each new value of the intrinsic gain, epoch after epoch. Nonetheless, we see in Fig.14 that the effective spectral radius for each realization of the ESN with IP undergoes very different trajectories, usually decreasing and then raising again, but ranging over too broad of a span to see any clear correlation with the optimal number of epochs. Therefore, we

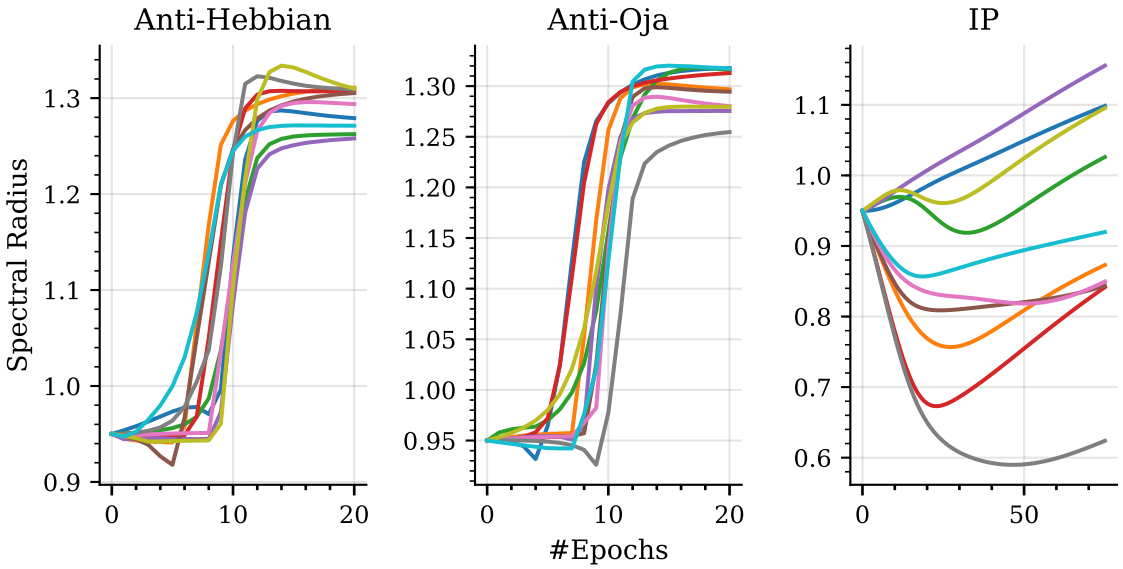


Figure 14: Evolution of the spectral radius with the number of plasticity training episodes. Each of the lines represent one realization of an ESN with 300 neurons, trained over 4000 points of the $\tau = 17$ MG series. After each epoch, the ESN was tested in predicting the following 300 points of the series and the overall RMSE was computed. The spectral radius for the IP case is taken of the effective reservoir, defined as in Eq.(25).

conclude that the improvement observed for the IP rule is not mediated by changes in the effective spectral radius, but through a different mechanism which we will further investigate.

Now, what is plasticity exactly doing over the reservoir states? Looking closer at Eq.(15), we can understand every discrete update of the reservoir matrix in anti-Hebbian learning as a change in the opposite direction of an "error" gradient:

$$\frac{\partial E'}{\partial W} \propto x(n-1)x(n)$$

This is, we tend to adjust the weights in W^{res} so to reduce correlations among states at consecutive times. To see this effect, we compute the Pearson correlation coefficient between activity of unit i at time t and that of unit k at time $t+1$ as:

$$corr(x_i(t), x_k(t+1)) = \frac{\sum_{t=1}^{T-1}(x_i(t)-\overline{x}_i)(x_k(t+1)-\overline{x}_k)}{\sqrt{\sum_{t=1}^{T-1}(x_i(t)-\overline{x}_i)^2}\sqrt{\sum_{t=1}^{T-1}(x_k(t+1)-\overline{x}_k)^2}} \tag{26}$$

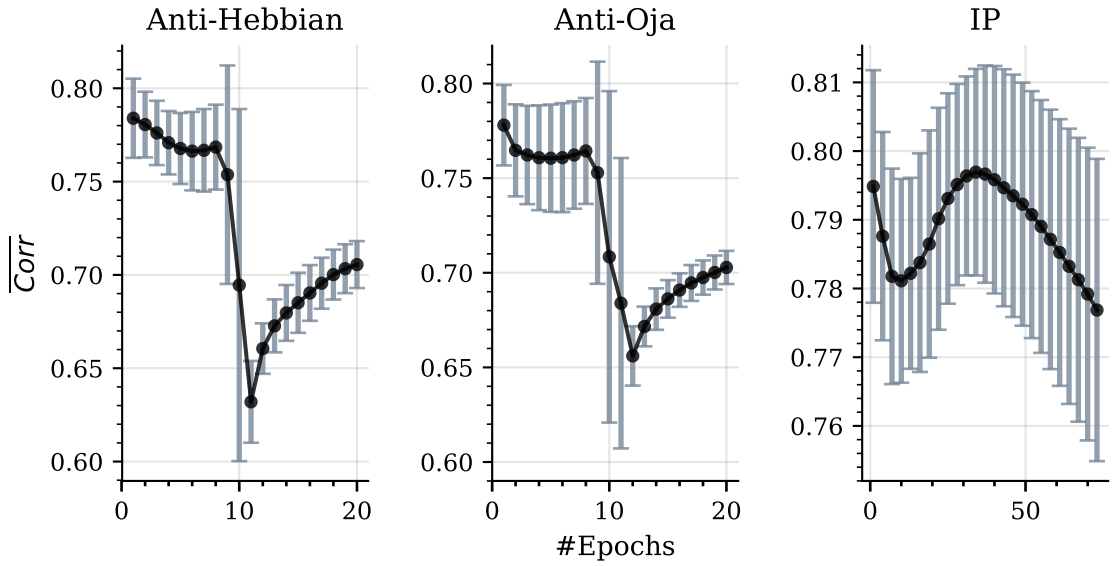


Figure 15: Average correlation between states at time t and $t+1$ for different plastic models. Computation at each episode of the plasticity training was carried out according to Eq.(27), taking $N = 300$ for the size of the reservoir and $M = 20$ for the number of realizations. Error is given as the standard deviation in the resulting value of $\overline{Corr}$.

After each epoch of the plasticity training, the mean absolute correlation is given as

$$\overline{Corr} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{k=1}^{N}|corr(x_i(t), x_k(t+1))| \tag{27}$$

where N denotes the size of the reservoir and M the number of independent realizations over which we average the result. In Fig.15 we plot the evolution of the above quantity with the number of training episodes in each plasticity implementation. We observe that in general terms, correlation between the state of one neuron at time t and the rest of the reservoir at time $t+1$ diminishes as we advance in the training during the first epochs. At a certain number of repetitions —which for the two synaptic rules coincide with the sudden increase in spectral radius leading to overall instability— the correlation drops as in a phase change, then raising again for the rest of the training. Interestingly, up and downs are also observed within the plot for the IP rule, although there is no direct implication of state decorrelation in the mathematical description of the rule.

To further evaluate the effects of plasticity over the units of the reservoir, we now analyze the distribution of the states before and after implementation of each rule. We

begin by performing the classical supervised training over a simple ESN of 300 neuron with 4000 points of the MG-17 series. We keep the values of the 300 states at each input u_t of the training sequence —resulting in a 4000x300 matrix— then plot the normalized histogram of the matrix values, i.e. a sort of spatio-temporal distribution of the states. Alternatively, this same ESN can be subject to an unsupervised online training of the corresponding plasticity rule, applying in each case the optimal number of epochs found before. The final version is fitted again to the same 4000 points to compute the new readout weights, and with the states of the reservoir at each time step we extract the new spatio-temporal distribution under the same input signal. The distribution of states for the unchanged ESN and for the ESN optimized with the different plasticity rules is shown in Fig.16. As we could expect, the distribution of the states after implementation of IP rule approach that of a Gaussian centered in zero and with variance similar to the one of the targeted distribution. Remarkably, the application of synaptic plasticity also shapes the initial distribution to a unimodal one peaking around the center of the non-linearity, with tails on its extremes. In all the cases this has an immediate consequence: neurons are now operating mainly on the linear regime of the hyperbolic tangent where slight differences in the input can get further amplified, thus potentially increasing the computational capability of the network at the risk of losing non-linearity properties.

So far we have focused on understanding the effects of plasticity at the network level, discussing its influence in the distribution of the reservoir states and its spectral radius. But what happens at the neuron level? How does plasticity affects the way an individual unit "sees" and "reacts" to the input? To shed some light on this problem, we define the "feed" $\widetilde{u}_n(t)$ of a neuron n at time t as the sum of the input and bias once filtered through the input mask:

$$\widetilde{u}_n(t) = W^{in}_{0,n} + W^{in}_{1,n} \cdot u_n(t) \tag{28}$$

Plotting now the activity of a particular reservoir neuron against its feed for each time t of the training series, we get an overall picture of the response of the neuron to the input. In Fig.17 we use this representation to study the response of 4 different neurons to the same input before (blue dots) and after (red dots) the implementation of the anti-Oja plasticity rule. On the right side, we zoomed in on one of these neurons and plotted also 800 points of the total feed $\widetilde{u}_n(t)$ that arrives to it. We can see very clearly that the plasticity has the effect of widening the activity range of the neurons, specially in those areas —as highlighted in blue— in which the same point may lead to very different continuations of the series depending on its past. Although maybe not very impressive at first sight, this effect has a clear biological implication which comes to

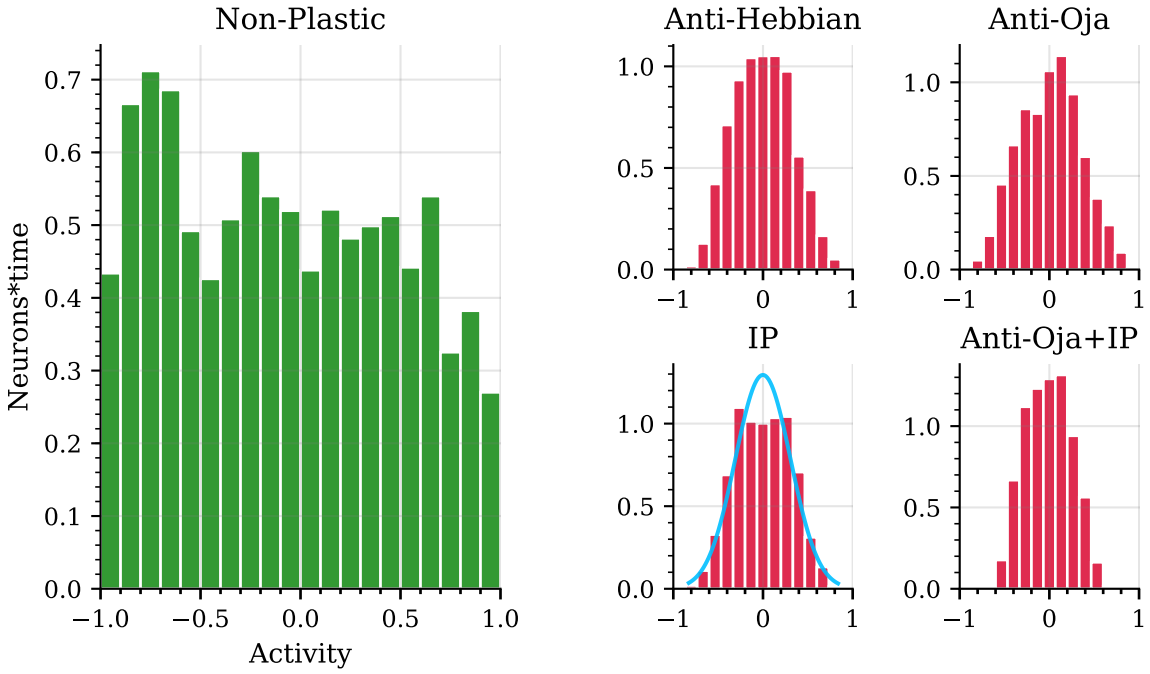


Figure 16: Distribution of states after training for the simple ESNs and the different plastic models. For the ESN with IP implemented, the Gaussian distribution fitting the histogram is also plotted, with resulting mean $\mu = -0.008$ and standard deviation $\sigma = 0.31$.

light by rewriting first the neuron update rule in terms of the feed and previous state:

$$x_j(t+1) = tanh\left(\widetilde{u}_j(t) + x_k(t)W^{res}_{kj}\right) \tag{29}$$

According to the above equation, if a neuron is mainly influenced by the input at each time t, then $x_n(t+1) \approx tanh(\widetilde{u}_n(t))$ and the corresponding scatter plot follows closely that of the activation function. This is what we see in the non-plastic case, where units states gather mostly around the curve of the hyperbolic tangent. On the contrary, plots $x_n(t+1)$ vs $\widetilde{u}_n(t)$ deviating from the non-linearity curve (as it happens after plasticity implementation) imply a greater influence of the past states and the interaction among units. For the anti-Oja and anti-Hebbian rules this is mediated by changes in the synaptic weights, whereas in the nonsynaptic plasticity, modifications take place at the activation function of individual neurons. The fact that mechanisms apparently so disparate may ultimately have similar effects at the neuron and network level, motivates the idea of synergistic learning with synaptic and nonsynaptic plasticity, which has been extensively backed up also at the biological level [52, 53].

The same type of representation can now be applied to analyze the effect of intrinsic

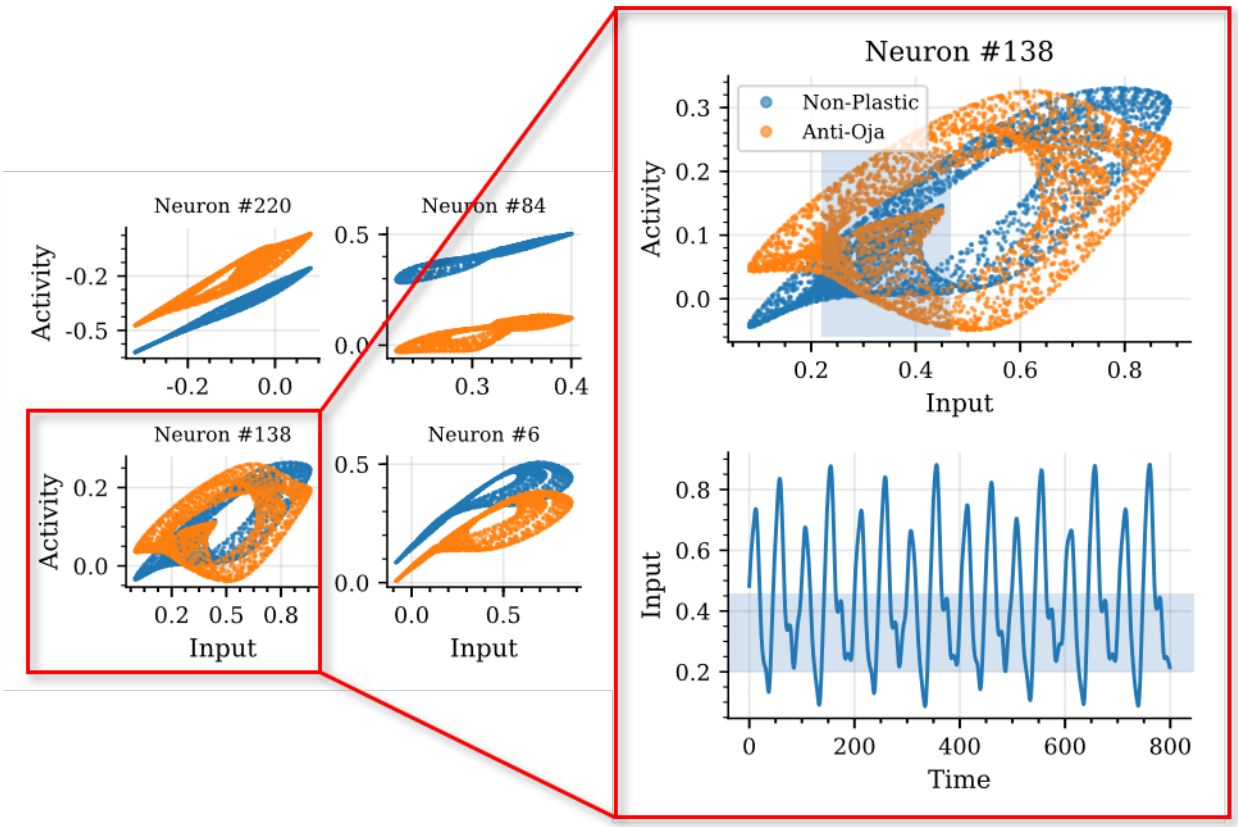


Figure 17: Activity vs filtered input for 4 different neurons in a non-plastic ESN (blue) and in the same network after training it with anti-Oja rule for 8 epochs (orange). On the right side we zoom in on one of the neurons, plotting also the evolution of the feed over a section of the training. We highlight in blue the range of inputs for which the activity broadens more notably with respect to the non-plastic case, coinciding with one of the most variable parts of the input.

plasticity alone[6]. For comparative purposes, we used the same initial ESN as starting point, as well as the same fragment of the MG series for training, therefore having identical feeds $\widetilde{u}_n(t)$ as the ones shown in Fig.17. We see in Fig.18 that the widening effect is similar, but with a particularity worth to highlight: individual neurons activation functions are modified so to make them fire around the center of the non-linearity (marked by the green dashed line). This comes as no surprise since we chose $\mu = 0$ as the mean of our targeted distribution in the IP rule, but it is interesting to note how the Gaussian spatio-temporal distribution of states in Fig.16 appears then to be the result of summing individual Gaussian reservoir units.

For the sake of completion, we also present in Fig.19 the results for the training with

[6]Anti-Hebbian rule leads to a very similar picture as the one shown above, with neuron states distribution stretching slightly less compared to the anti-Oja case.

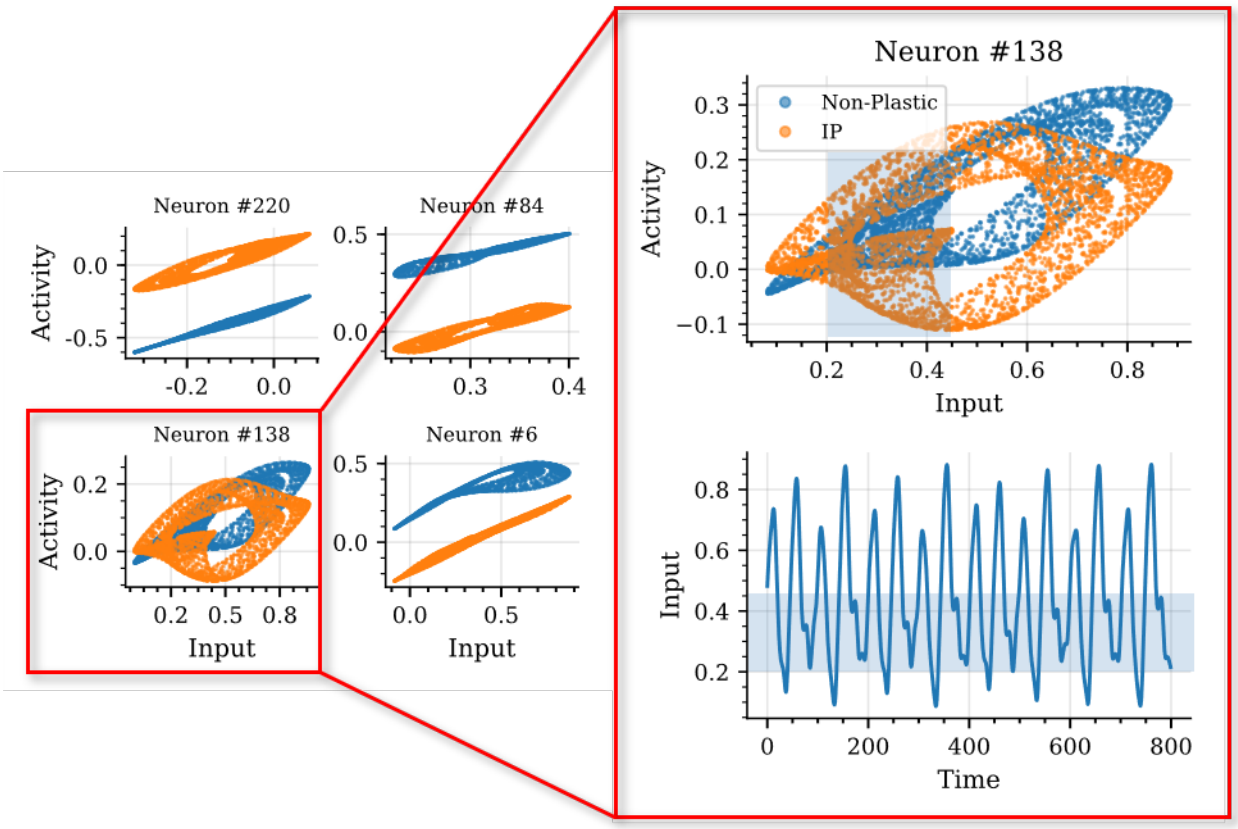


Figure 18: Activity vs filtered input for 4 different neurons before and after training with IP rule for 18 epochs. On the right side we zoom in on one of the neurons, plotting also the evolution of the feed over a section of the training. We highlight in blue the range of inputs for which the activity broadens more notably with respect to the non-plastic case, coinciding with one of the most variable parts of the input.

anti-Oja + IP rule. Although the improvement in performance through the combination of this two rules does not stand out very clearly in this picture, we can appreciate that the resulting states are both, broadened and zero-centered with respect to the non-plastic version, boosting the computational capabilities of the network.

Now that we have seen the usefulness of the the activity vs feed plots, let's try to use it to understand what happens exactly when we "overtrain" synaptic plasticity. We saw in Fig.4 that once a certain number of epochs were exceeded, the prediction error for the synaptic models began to increase, and that this co-occurred with a sharp raise in the spectral radius of the weight matrix (Fig.10). Is this transition from a stable dynamics to an unstable one somehow reflected in the neurons activity? Choosing the same initial ESN and training set as in Fig.17, we now apply the plasticity rule for a total of 12 epochs, a number which we saw in Fig.14 can drive the spectral radius

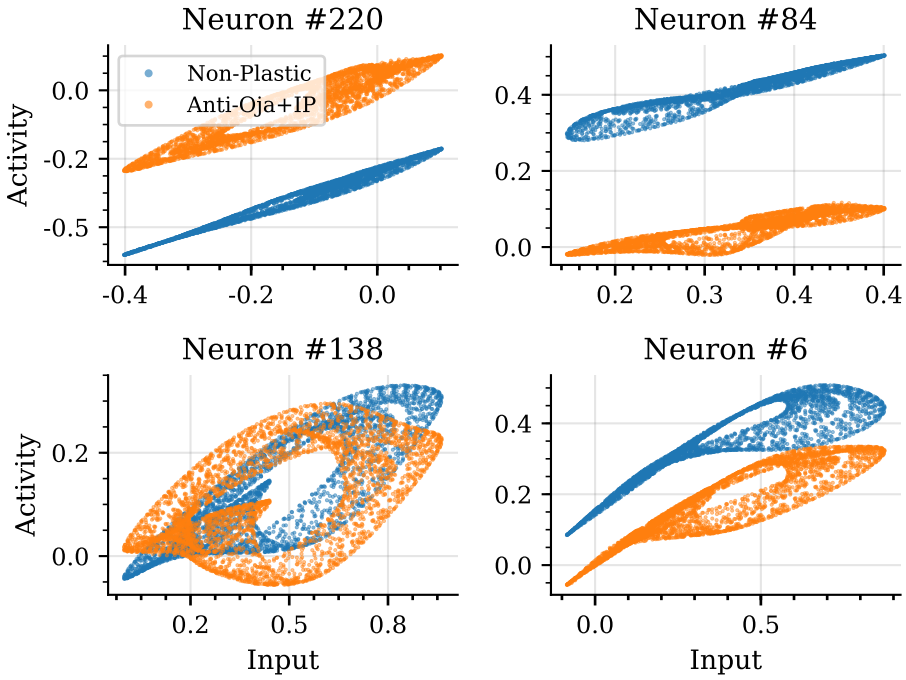


Figure 19: Activity vs filtered input for 4 different neurons before and after training with 6 epochs of anti-Oja and 18epochs of IP rule.

considerably over 1. Plotting the resulting activity vs input in Fig.20, we can observe for three out of four neurons that now the state values seem to be distributed around two different parts of the phase space. But why is this a problem? It could seem even desirable to have such diversity of states within a single neuron. After all, should it not increase the computational capacity of the reservoir?

The point lays, as we see in Fig.21, on the way the neuron states jump from one part of the phase space to the other with every step. In the stable case, with 8 epochs of training, the evolution from one state to the next one is smooth as we move along the input sequence. In contrast, it is clear that for 12 epochs the neuron state evolves frantically from one side of the phase space to the opposite, which we suggest is the reason behind the unstable results that comes with an excessive training.

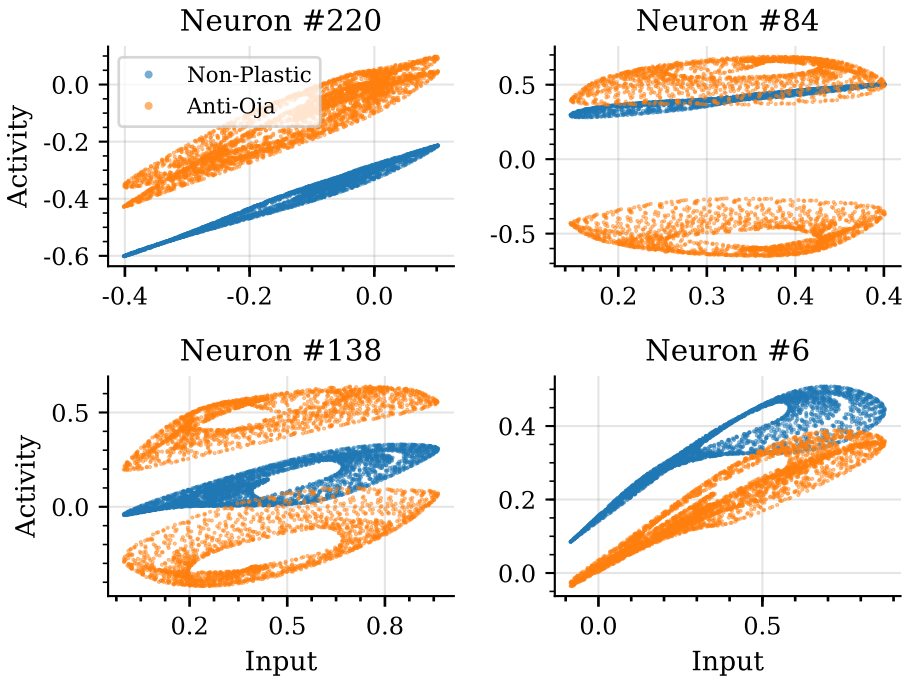


Figure 20: Activity vs filtered input for 4 different neurons before and after training with anti-Oja rule for 12 epochs.

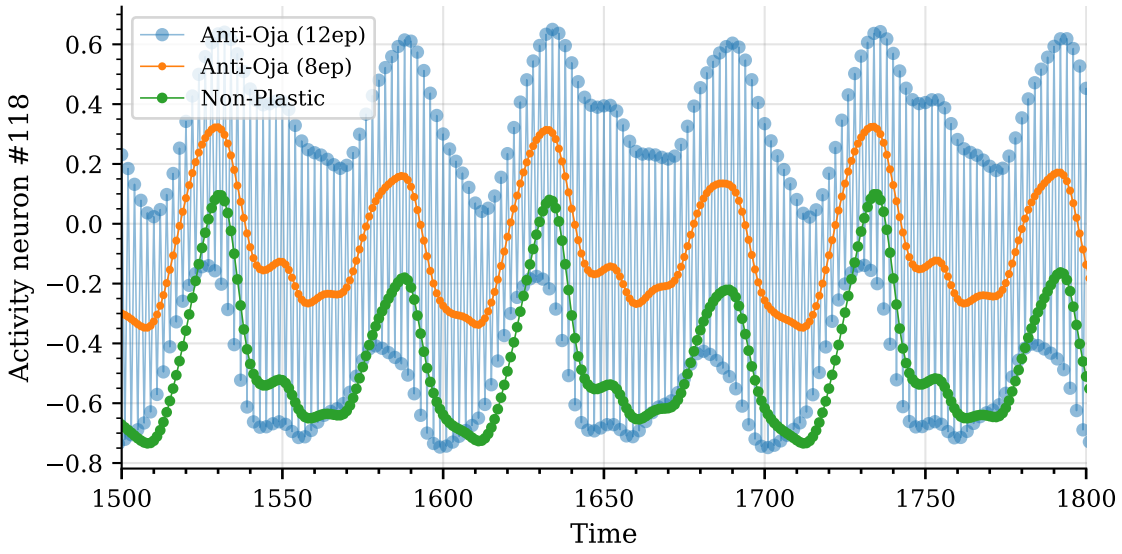


Figure 21: Evolution of the state of one particular neuron in time for a simple ESN of 300 reservoir neurons. Plots are presented for the activity before implementation of plasticity and after 8 and 12 epochs of anti-Oja rule training. For clarity, we only plot a span corresponding to 300 points of the training set.

4 Conclusions

We have shown that numerical implementation of plasticity rules can improve the performance of ESNs on chaotic time series prediction tasks. For this purpose, we implemented two synaptic rules based on Hebbian learning, namely anti-Hebbian and anti-Oja; and one non-synaptic or IP rule. Based on the synergistic action of synaptic and non-synaptic plasticity mechanisms in the brain , we also proposed the combination of anti-Oja and IP algorithms operating sequentially as an alternative form of plastic rule. We showed that not only the accumulated RMSE over the predicted points is reduced after implementation of plasticity, but also plastic ESNs can make further predictions without diverging from the original MG and show an improved memory capacity. When comparing the different types of plasticity, we observed that in general synaptic plasticity rules outperform the IP, but it is the combination of both, anti-Oja and IP, which leads to the best results. Average performance of models with anti-Hebbian and anti-Oja rules alone reached very similar values, but the latter should be preferred due to its reduced variance. Finally, we showed how the aid of any of the plastic adaptations becomes more evident as the predictability of the series decreases, reducing the error in the continuation of the MG-30 series up to 2 orders of magnitude.

At the network level, we investigated different quantities that were suspected to be modified by the plasticity rules: the spectral radius of the reservoir and the absolute correlation between states at different times. For the synaptic rules, we saw in Fig.14 and Fig.15 how the sudden increase in the spectral radius co-occur with a sharp drop on the state correlations at consecutive times. Being the optimal number of epochs just before this transition, we suggest that anti-Hebbian and anti-Oja rules operate at its best when they drive the network towards the edge of instability. This would be achieved by decreasing the correlations in the input-driven activities of the reservoir, with the risk of an eventual loss of the "echo state property" due to the associated increase in the spectral radius of W^{res}. These observations are not that clear when looking at these same quantities for the IP rule. We saw that the defined effective spectral radius of the reservoir may go over or below 1 depending on the realization, not giving room to speculations on its effect over the performance. Likewise, the absolute correlation between states at consecutive times present a local minimum at the number of epochs in which the rule achieves its optimal performance, but further work needs to be done to investigate the causes of these changes in the states correlations.

From the spatio-temporal distributions of the states presented in Fig.16 we can conclude

that all plasticity rules agree on the benefit of having unimodal distributions centered around zero. This would imply that optimal performance is achieved —for this type of task— when most neurons operate at the linear part of the hyperbolic tangent, where small differences in the input can be maximized. Similar results observed both in vivo, on large monopolar cells of the fly [68], and in artificial single neuron models [69], suggest that these modification on the spatio-temporal distribution of the states are helping to achieve optimal encoding of the inputs. In particular, single neuron models applying the Bell and Sejnowski's algorithm showed that input-output mutual information is maximized when a neuron uses all of its possible response levels equally, applying the steep parts of the activation function to respond to the high density parts of the input probability density function [69]. The above conclusions are specially true for the results observed in the IP model, as the Gaussian distribution achieved has been proved to be the maximum entropy distribution for our hyperbolic tangent non-linearity, thus increasing the information that can be encoded within the reservoir.

Perhaps the most interesting results emerge from the comparative analysis of single reservoir neurons before and after implementation of plasticity. At this level we see how increasing computational capability is achieved by expanding the neuron activity on the phase space. This is particularly true for the anti-Hebbian and anti-Oja rules, but it also stands for many neurons with IP implemented. The same approach helped us to understand why too much plasticity could lead to overall instability of the ESNs: after we pass a critical number of training episodes, the original attractor for the activities of each single neuron split in two, and the neuron begins to jump from one attractor to the opposite with every new input.

It happens often in research that one immerses himself deeply into a problem looking for its solution, just to come back with a partially good answer and a few new problems. This thesis is no exception and we are aware that the findings exposed also raise a lot of interesting new questions. At the performance level it is clear that implementation of plasticity helps in prediction tasks of temporal series because long-term memory of the inputs increase at the edge of stability [64, 70]. However, the same plastic models would need to be tested on a different type of task before asserting that they can always improve the results of simpler ESNs. On top of that, some of the results showed here would require a further analysis to be completely understood. For example, what is the reason behind the local minimum and maximum showed by the defined $\overline{Corr}$ during IP training in Eq.(27)? Why do activity attractors in individual neurons split in two when we overtrain synaptic plasticity, as in Fig.20? Are individual neurons actually

operating at the edge of chaos?

But perhaps the most striking open question arrives when comparing the right-hand sides of Fig.17 and Fig.18. Looking at the modified activity of the neuron after implementation of anti-Oja and IP rules, we see that the similarities are remarkable, and this same picture can be found in many other neurons of the reservoir. How is it possible that these two rules which are fundamentally different —one modifying the connections weights, the other the intrinsic parameters of the activation function— lead to phase space attractors that are so similar? We cannot conclude without further analysis if these similarities are a reflection of common features among the rules, a result of the training process, or maybe something fundamentally different. It could be that for this type of task (with a given fixed input, hyper-parameters and initial conditions of the ESN) there exists a universally optimal phase space at the neuron level which can be reached through different methods. Whatever the answer might be, we are convinced that further studies in this direction will take us a little bit closer to the recipe for creating optimal reservoirs, hoping to reach on the way a deeper understanding of the plasticity mechanisms in the brain.

References

[1] D. O. Hebb, *The Organization of Behavior. A Neuropsychological Theory.* Wiley, 1949.

[2] G. Berlucchi and H. A. Buchtel, "Neuronal Plasticity: Historical Roots and Evolution of Meaning," *Experimental Brain Research*, vol. 192, no. 3, 2008.

[3] E. Tanzi, "I fatti e le induzioni dell'odierna istologia del sistema nervoso," *Rivista Sperimentale di Freniatria e Medicina Legale*, vol. 19, 1893.

[4] P. Milner, "A Brief History of the Hebbian Learning Rule.," *Canadian Psychology*, vol. 44, no. 1, 2003.

[5] F. Rosenblatt and Cornell Aeronautical Laboratory, *The Perceptron : a Theory of Statistical Separability in Cognitive Systems (Project PARA).* Cornell Aeronautical Laboratory, 1958.

[6] F. Grezes, *Reservoir Computing. Dissertation Submitted to the Graduate Faculty in Computer Science, The City University of New York.* PhD thesis, 2014.

[7] M. Minsky and S. Papert, *Perceptrons: an Introduction to Computational Geometry.* Cambridge, Mass.-London, 1969.

[8] A. Kurenkov, "A 'Brief' History of Neural Nets and Deep Learning," 2015.

[9] S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.* PhD thesis, 1970.

[10] P. Werbos, "New Tools for Prediction and Analysis in the Behavioral Sciences.," *PhD thesis, Harvard University, Cambridge, MA.*, 1974.

[11] P. Werbos, "Backwards Differentiation in AD and Neural Nets: Past Links and New Opportunities," in *Automatic Differentiation: Applications, Theory, and Implementations*, Springer, 2006.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, 1986.

[13] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities.," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, 1982.

[14] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, 1990.

[15] S. Hochreiter, Y. Bengio, and J. Schmidhuber, "Gradient Flow in Recurrent Nets: the Difficulty of Learning Long Term Dependencies," in *A Field Guide to Dynamical Recurrent Networks*, Wiley-IEEE Press, 2001.

[16] Y. Bengio, "A Connectionist Approach to Speech Recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 07, no. 04, 1993.

[17] A. Atiya and A. Parlos, "New Results on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, 2000.

[18] H. Jaeger, "The "Echo State" Approach to Analysing and Training Recurrent Neural Networks," *GMD-Report 148, German National Research Institute for Computer Science*, 2001.

[19] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing without Stable States: a New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11, 2002.

[20] L. Appeltant, M. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature Communications*, vol. 2, no. 1, 2011.

[21] Š. Babinec and J. Pospíchal, "Improving the Prediction Accuracy of Echo State Neural Networks by Anti-Oja's Learning," *Lecture Notes in Computer Science*, vol. vol 4668, 2007.

[22] J. Triesch, *A Gradient Rule for the Plasticity of a Neuron's Intrinsic Excitability.* Artificial Neural Networks: Biological Inspirations ICANN 2005, 2005.

[23] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt, "Improving Reservoirs Using Intrinsic Plasticity," *Neurocomputing*, vol. 71, no. 7-9, 2008.

[24] J. J. Steil, "Online Reservoir Adaptation by Intrinsic Plasticity for Backpropagation–Decorrelation and Echo State Learning," *Neural Networks*, vol. 20, no. 3, 2007.

[25] M.-H. Yusoff, J. Chrol-Cannon, and Y. Jin, "Modeling Neural Plasticity in Echo State Networks for Classification and Regression," *Information Sciences*, vol. 364-365, 2016.

[26] X. Wang, Y. Jin, and K. Hao, "Echo State Networks Regulated by Local Intrinsic Plasticity Rules for Regression," *Neurocomputing*, vol. 351, 2019.

[27] H. Ju, M. R. Dranias, G. Banumurthy, and A. M. J. VanDongen, "Spatiotemporal Memory Is an Intrinsic Property of Networks of Dissociated Cortical Neurons," *Journal of Neuroscience*, vol. 35, no. 9, 2015.

[28] P. Enel, E. Procyk, R. Quilodran, and P. F. Dominey, "Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex," *PLOS Computational Biology*, vol. 12, no. 6, 2016.

[29] M. D. Skowronski and J. G. Harris, "Automatic Speech Recognition Using a Predictive Echo State Network Classifier," *Neural Networks*, vol. 20, no. 3, 2007.

[30] H. Jaeger, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, no. 5667, 2004.

[31] J. Hertzberg, H. Jaeger, and F. Schönherr, "Learning to Ground Fact Symbols in Behavior-Based Robots," *ECAI'02 Proceedings of the 15th European Conference on Artificial Intelligence*, 2002.

[32] X. Lin, Z. Yang, and Y. Song, "The Application of Echo State Network in Stock Data Mining," *Advances in Knowledge Discovery and Data Mining*, 2008.

[33] Š. Babinec and J. Pospíchal, "Gating Echo State Neural Networks for Time Series Forecasting," *Advances in Neuro-Information Processing. Lecture Notes in Computer Science*, vol. 5506, 2009.

[34] X. Lin, Z. Yang, and Y. Song, "Short-term Stock Price Prediction Based on Echo State Networks," *Expert Systems with Applications*, vol. 36, no. 3, 2009.

[35] H. Wang, Y. Bai, C. Li, Z. Guo, and J. Zhang, "Time Series Prediction Model of Grey Wolf Optimized Echo State Network," *Data Science Journal*, vol. 18, 2019.

[36] M. Lukoševičius, "Echo State Networks with Trained Feedbacks," *Technical Report, Jacobs University Bremen*, vol. 4, no. 4, 2007.

[37] R. F. Reinhart and J. J. Steil, "Reservoir Regularization Stabilizes Learning of Echo State Networks with Output Feedback," *ESANN 2011 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[38] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and Applications of Echo State Networks with Leaky-Integrator Neurons," *Neural Networks*, vol. 20, no. 3, 2007.

[39] M. Lukoševičius, "A Practical Guide to Applying Echo State Networks," *Lecture Notes in Computer Science*, 2012.

[40] H. Jaeger, "Short Term Memory in Echo State Networks," *GMD-Report 152, German National Research Center for Information Techonology*, 2001.

[41] Q. Wu, E. Fokoue, and D. Kudithipudi, "On the Statistical Challenges of Echo State Networks and Some Potential Remedies," tech. rep., 2018.

[42] B. Widrow, "Generalization and Information Storage in Network of Adaline 'Neurons'," 1962.

[43] R. F. Reinhart and J. J. Steil, "A Constrained Regularization Approach for Input-Driven Recurrent Neural Networks," *Differential Equations and Dynamical Systems*, vol. 19, no. 1-2, 2010.

[44] L. F. Abbott and S. B. Nelson, "Synaptic Plasticity: Taming the Beast," *Nature Neuroscience*, vol. 3, no. S11, 2000.

[45] I. Song and R. L. Huganir, "Regulation of AMPA Receptors during Synaptic Plasticity," *Trends in Neurosciences*, vol. 25, no. 11, 2002.

[46] K. Gerrow and A. Triller, "Synaptic Stability and Plasticity in a Floating World," *Current Opinion in Neurobiology*, vol. 20, no. 5, 2010.

[47] A. Alonso, M. de Curtis, and R. Llinas, "Postsynaptic Hebbian and non-Hebbian long-term potentiation of synaptic efficacy in the entorhinal cortex in slices and in the isolated adult guinea pig brain.," *Proceedings of the National Academy of Sciences*, vol. 87, no. 23, 1990.

[48] H. K. Kato, A. M. Watabe, and T. Manabe, "Non-Hebbian Synaptic Plasticity Induced by Repetitive Postsynaptic Action Potentials," *Journal of Neuroscience*, vol. 29, no. 36, 2009.

[49] R. S. Zucker and W. G. Regehr, "Short-Term Synaptic Plasticity," *Annual Review of Physiology*, vol. 64, no. 1, 2002.

[50] W. Zhang and D. J. Linden, "The other side of the engram: experience-driven changes in neuronal intrinsic excitability," *Nature Reviews Neuroscience*, vol. 4, no. 11, 2003.

[51] A. Frick, J. Magee, and D. Johnston, "LTP is accompanied by an enhanced local excitability of pyramidal neuron dendrites," *Nature Neuroscience*, vol. 7, no. 2, 2004.

[52] E. Hanse, "Associating Synaptic and Intrinsic Plasticity," *The Journal of Physiology*, vol. 586, no. 3, 2008.

[53] R. Mozzachiodi and J. H. Byrne, "More than Synaptic Plasticity: Role of Nonsynaptic Plasticity in Learning and Memory," *Trends in Neurosciences*, vol. 33, no. 1, 2010.

[54] A. Rodan and P. Tino, "Minimum Complexity Echo State Network," *IEEE Transactions on Neural Networks*, vol. 22, no. 1, 2011.

[55] S. Zhong, X. Xie, L. Lin, and F. Wang, "Genetic Algorithm Optimized Double-Reservoir Echo State Network for Multi-Regime Time Series Prediction," *Neurocomputing*, vol. 238, 2017.

[56] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, 1982.

[57] J. Triesch, "Synergies between intrinsic and synaptic plasticity mechanisms," *Neural Computation*, vol. 19, pp. 885–909, Apr. 2007.

[58] Y. Li and C. Li, "Synergies between intrinsic and synaptic plasticity based on information theoretic learning," *PloS One*, vol. 8, no. 5, p. e62894, 2013.

[59] M. K. Janowitz and M. C. W. van Rossum, "Excitability changes that complement Hebbian learning," *Network (Bristol, England)*, vol. 17, pp. 31–41, Mar. 2006.

[60] W. Aswolinskiy and G. Pipa, "RM-SORN: a reward-modulated self-organizing recurrent neural network," *Frontiers in Computational Neuroscience*, vol. 9, Mar. 2015.

[61] S. H. Strogatz, *Nonlinear Dynamics and Chaos : with Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, 2000.

[62] N. B. Abraham, A. M. Albano, A. Passamante, and P. Rapp, eds., *Measures of Complexity and Chaos*. Nato Science Series B:, Springer US, 1989.

[63] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets," *Physica D: Nonlinear Phenomena*, vol. 65, no. 1-2, 1993.

[64] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, "Information Processing in Echo State Networks at the Edge of Chaos," *Theory in Biosciences*, vol. 131, no. 3, 2011.

[65] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, 2009.

[66] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Networks*, vol. 35, 2012.

[67] G. Manjunath and H. Jaeger, "Echo State Property Linked to an Input: Exploring a Fundamental Characteristic of Recurrent Neural Networks," *Neural Computation*, vol. 25, no. 3, 2013.

[68] S. Laughlin, "A simple coding procedure enhances a neuron's information capacity," *Zeitschrift Fur Naturforschung. Section C, Biosciences*, vol. 36, pp. 910–912, Oct. 1981.

[69] A. Bell and T. Sejnowski, "An Information-Maximization Approach to Blind Separation and Blind Deconvolution," *Neural computation*, vol. 7, pp. 1129–59, Dec. 1995.

[70] P. Barancok and I. Farkas, "Memory Capacity of Input-Driven Echo State Networks at the Edge of Chaos," in *ICANN*, 2014.