UNIVERSITY OF PADOVA

Department of Physics and Astronomy

Master's Degree Course in Physics

Final Master's Thesis

Closed-loop Neurostimulation via Reservoir Computing

Supervisor
Prof. Michele Allegra

Candidate: Luca Mazzocchetti
Student ID: 2058353

Academic Year 2023/2024

Abstract

The idea of "brain controllability" refers to the possibility of manipulating brain activity through suitable external stimuli, chiefly for therapeutic aims (restoring normal brain activity in patients). Traditional approaches to dynamical system control generally require a system identification step where an explicit model of the system dynamics is obtained - a difficult step in the case of brain dynamics, where a plenitude of competing dynamical models exist. It was recently proposed that the system identification step could be circumvented by enacting control through a neural network (reservoir computer) coupled to the system. In this thesis, we will simulate brain activity with existing dynamical models and test the ability of the neural-network-based controller to modulate activity. Results may be of direct applicability to real neurostimulation experiments.

Contents

List of Figures

Chapter 1

Introduction

The possibility to control a system's behavior by external inputs is not only fascinating, but of extreme importance in basically all fields of applied science. Examples range from feedback amplifiers in electronics to automotive systems in mechanical engineering, to autopilot and stability control in aircraft engineering, to control systems in chemical plants, to medical devices such as insulin pumps or pacemakers and so on.
With respect to the medical field, it is interesting to test the possibility of controlling the activity state of the brain through external inputs applied to its "nodes". A node is a region of the brain whose properties and activity are homogeneous with respect to a given criterion. Depending on the criterion, several "parcellations" of the brain into nodes are possible: an example is the Schaefer parcellation[16], which we will use in this work (see Section 3.1). Several devices and techniques can be used to modulate node activity, such as electrodes implanted in specific brain regions which can deliver electrical impulses, or transcranial magnetic stimulation where magnetic fields are used to induce electric currents in specific areas of the brain. These kinds of techniques have been studied also in relation to the treatment of mental health problems like depression and dementia and have received a lot of attention by clinicians[15], [2]. It should be pointed out, however, that only some brain regions can be directly stimulated, typically superficial regions of the cortex. For this reason, previous studies have investigated the possibility to control specific nodes indirectly, through remote perturbations acting on the accessible nodes[4]. In this work, we will investigate this possibility making use of a particular kind of neural network, namely an Echo State Network, to address the problem of remote stimulation of brain nodes. Making use of a neural network allows to overcome some of the difficulties of the traditional approach to controllability.
The traditional approach to the problem of control follows three steps. At first one has to define a model describing the dynamical system at hand, choosing the best parameters to fit the experimental data. This is usually done by applying external inputs to the so called system plant (i.e. to the parts of the system that can be stimulated from outside) and collecting the system outputs. Using metrics such as mean squared error, correlation

functions and others, the parameters of the model are iteratively adjusted to best match the collected data, a process called system identification[12]. The second step is usually the linearization of the model. In order to properly control the system by external inputs, one has to know how the system would evolve when perturbed, but this knowledge is basically inaccessible except for the simplest kind of dynamics. In particular, let's suppose that the variable $\mathbf{x}(t)$ describes the plant internal state, $\mathbf{y}(t)$ the observable outputs, $\mathbf{v}(t)$ the external inputs. Then the following equations will in general hold:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{v}) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x}) \end{aligned} \tag{1.1}$$

where a function $\mathbf{f}$ determines how the internal state evolves in terms of the state itself and the external input, and a function $\mathbf{g}$ determines what output a given internal state produces. Since it is usually impossible to solve these equations, one cannot know which input is needed to produce a desired output. For this reason one has to make a second step, i.e. linearizing the system, which involves reducing those equations to linear equations around a fixed point, obtaining a relation of the form:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{v} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{v} \end{aligned} \tag{1.2}$$

through an appropriate choice of matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. These are called state-space equations. The third step is to exploit the known fact that, in frequency domain, the response function for these equations, relating the outputs to the external inputs, can be expressed in terms of the entries of the matrices, thus solving the control problem.
The development of neural networks has opened the path to the possibility of studying controllability of systems without approximations of the dynamics. Studies on the usage of neural networks for identification and control of dynamical systems date back to the 1990s[14]. Neural networks can learn a system dynamics without prior knowledge of its model equations. The system identification step is replaced by a training phase where the neural network "learn" the relation between external inputs and plant outputs. In this way, after the training, the neural network is capable of producing the inputs needed to obtain a desired accessible output and to adjust them in real time in response to the changes of the plant (feedback or closed-loop control). A neural network can do this also for complex non-linear systems, without the need of linearization techiques.
We will make use of a model of the brain built to reproduce real fMRI data-sets to generate time series of the brain nodes when some external perturbations are applied to a

given subset of nodes. These signals will be used to train the neural network to identify the acting perturbations. In this way, the neural network should be able to predict the perturbations to apply to the chosen subset in order to obtain some desired signal for the other nodes. In particular we will procede as follows.

First, in Chapter 2 we will review the theoretical background required to define and understand the problem. In Section 2.1 and 2.2 the basics of Recurrent and Echo State Networks and their usage for control are introduced. In Section 2.3 we explain the equations of the dynamical model used to simulate brain activity. They are needed to generate the time series used to train the network.

In Chapter 3 the detailed implementation of the simulations and the network training is reported. In Section 3.1 specifics of the data on which the brain model was built are reported. In Section 3.2 we explain how all the parameters of the brain model were fixed. In Section 3.3 we specify the network architecture and working process.

In Chapter 4 we will report the main results of this work. We tested several control configurations, varying in the type and number of external inputs used during the network training, as explained in each Section of the chapter.

Chapter 5 is used to discuss the results, draw conclusions, and address possible future lines of research.

The coding programs we used in this work are documented in a GitHub repository. You can access the repository at the following link: https://github.com/LucaMazz/MasterThesis.

Chapter 2

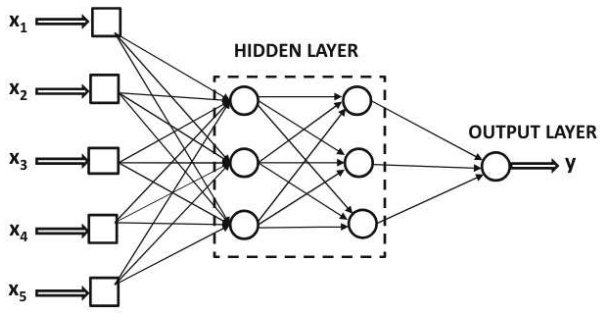
Theoretical Framework

2.1 Recurrent Neural Networks

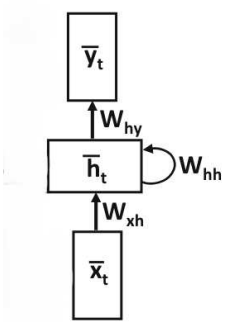
Recurrent neural networks (RNNs) are a type of artificial neural network designed to recognize patterns in sequences of data, such as time series, text and biological data, where there is a sequential dependency among the elements.

The most common neural networks, known as feedforward neural networks, consist of an input layer, some hidden layers, and an output layer. In this type of networks neurons in a layer are connected only to neurons in the subsequent layer, and each connection has an associated weight that determines the strength of the connection (see 2.1a).

Information flows in one direction, from the input layer through the hidden layers to the output layer, without any feedback loops or cycles. The network is thus unable to capture the temporal dependencies in the data.



(a) Feedforward network



(b) Recurrent network

Figure 2.1: Schematics of the two kinds of networks, FNNs on the left and RNNs on the right

Recurrent neural networks, on the other hand, have an architecture where the output from a layer is fed back into the network, allowing information to persist and creating a form of memory in the network. Supposing that a vector sequence $\bar{x}_t$ is fed to the recurrent

network as input, the network evolves a hidden state vector $\bar{h}_t$ through an update rule of the following kind:

$$\bar{h}_t = f(W_{hh}\bar{h}_{t-1} + W_{hx}\bar{x}_t + \bar{b}_h) \tag{2.1}$$

where W_{hx} is the weight matrix for the input, W_{hh} the weight matrix for the hidden state, b_h a bias term and f some kind of activation function, usually an hyperbolic tangent or a rectifier. Thus the state of the network at some time step $(t-1)$ has an impact on the state at time step t, enabling the emerging of some kind of recurrence (see 2.1b).
The network generates a vector sequence $\bar{y}_t$ as output:

$$\bar{y}_t = g(W_{yh}\bar{h}_t + \bar{b}_y) \tag{2.2}$$

where W_{yh} is the weight matrix for the output, b_y a bias term and g an appropriate function for the task to be performed (e.g. softmax). The matrices are optimized during a training phase through gradient descent algorithms such as back-propagation through time[11]. Adjusting dynamically all the weights in the network, this process is computationally intensive and potentially prone to issues like vanishing and exploding gradients.

2.2 Echo State Networks

Echo state networks represent a simplification of recurrent neural networks. They use random weights in the hidden-to-hidden layer and the input-to-hidden layer which are randomly initialized and remains fixed, and only the weights of the output layer are trained, usually using a simple linear regression. By keeping the hidden layer weights fixed and only training the output layer, ESNs require far fewer computational resources. The training process is also significantly faster, making ESNs suitable for applications where rapid training is needed.

It has been showed that ESNs work well when the dimensionality of the input is small. Therefore, the dimensionality of the hidden layer should be much larger than the dimensionality of the input. The large number of randomly connected neurons of an echo state network is often referred to as the reservoir (see Figure 2.2).

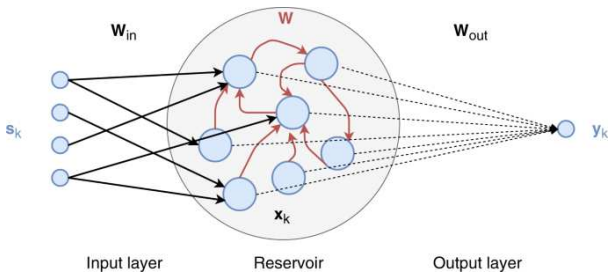


Figure 2.2: Echo state network

For the ESN to work correctly, one has to ensure the echo state property, which essentially means that, for sufficiently long time, the reservoir state should asymptotically depend only on the driving input signal (the state is an echo of the input), while the influence of initial conditions should progressively vanish with time. A number of precautions are to be held to ensure this property[1]:

- setting the spectral radius of the reservoir's weight matrix $\mathbf{W}$ (i.e. its largest absolute eigenvalue). It is the most critical parameter and should almost always be less than or equal to 1.
- using a sparsely connected weight matrix $\mathbf{W}$ (say 10% of connections). It is argued that sparse connectivity leads to a decoupling of the individual sub-networks, which increases the diversity of the features learned by the echo state network[10].
- scaling the input matrix $\mathbf{W_{in}}$. The matrix is scaled with different values to avoid that inputs in each time step damage the information carried in the hidden layer from the previous time step.
- choosing a regression technique which makes the output matrix $\mathbf{W_{out}}$ less sensitive to the states of the reservoir. An example is ridge regression, also know as Tikhonov regularization (see Section 3.3).

Among other things, echo state networks can be used to predict the evolution of a dynamical system: the input is a time series of the system, and the output a prediction of future values of the time series. In a recent work[9] it has been proved that, under appropriate mathematical assumptions, this setting induces a map from the phase space of the dynamical system to the reservoir space, which is almost surely an embedding. In particular, for stable dynamical systems, the reservoir exhibit dynamics that are topologically conjugate of the observed system. The work proved that an ESN can predict the next value of a sequence of scalar observations of a structurally stable dynamical system with arbitrary precision, by an appropriate choice of the output weights $\mathbf{W_{out}}$. Unfortunately, it does not prove what kind of learning algorithm would be able to find those weights, or how much training data is needed.

2.2.1 Controlling Dynamical Systems with ESNs

Control in dynamical systems refers to the use of external inputs to influence the behavior of a system over time. This is a fundamental problem in many fields, such as automotive systems, robotics, chemical control, medical devices and so on.
Controlling a system is a challenging task, since to drive it in the desired direction one should constantly evaluate the state of the system and adjust the external inputs

accordingly over time. This approach is known as closed-loop control, or feedback control, since it involves continuously monitoring the output of a system and adjusting the inputs based on this feedback to achieve the desired behavior. Since our goal was to learn the perturbations needed to force brain state transitions, our problem is exactly a closed-loop control problem.

The difficulties with the closed-loop approach is that the majority of the systems follow a dynamics which is a nonlinear function of the state variables and the external inputs, so that it is impossible to predict the effects of the control signal for every moment and every state of the system. This is why, the usual approach is to linearize the dynamics and then, having to do with a much more stable and predictable system, apply a predetermined set of inputs to the system without using feedback from the system's outputs to adjust those inputs (the so called open-loop control or feed-forward control).

Thanks to the cited ability of ESNs to exhibit a dynamics which is conjugate to the system dynamics, ESNs are well suited to be used in closed-loop control. During a training phase where random external perturbations are applied to the system, the ESN learns to reproduce those perturbations processing the system dynamical evolution. Then, the trained ESN will be able to process a desired dynamics and produce the perturbations needed to obtain that dynamics.

To better understand, let's suppose that a dynamical system evolves through an equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{v}) \tag{2.3}$$

where $\mathbf{f}$ is a function of the state $\mathbf{x}(t)$ of the system and of some external input $\mathbf{v}(t)$, which contributes to drive the system evolution. The state $\mathbf{x}(t+\delta)$ after a little time interval δ will depend on the values of $\mathbf{v}$ at each time in $[t, t+\delta]$. However, if $\mathbf{v}$ varies slowly during this time, one can assume that $\mathbf{x}(t+\delta)$ will approximately depend on just $\mathbf{x}$ and $\mathbf{v}$ at time t:

$$\mathbf{x}(t+\delta) \approx \mathbf{F}[\mathbf{x}(t), \mathbf{v}(t)] \tag{2.4}$$

for some function $\mathbf{F}$. In general, this function is not invertible since there may be multiple possible inputs $\mathbf{v}(t)$ that drive the system to a given future state and not all states may be reachable from the current state. However, restricting the domain to future states reachable from the current state and choosing a single value among possible inputs, one could in principle invert the relation and be able to evaluate the external input needed to evolve the system from $\mathbf{x}(t)$ to $\mathbf{x}(t+\delta)$:

$$\mathbf{v}(t) \approx \mathbf{F}^{-1}[\mathbf{x}(t), \mathbf{x}(t+\delta)] \tag{2.5}$$

What an Echo State Network can do is to learn the inverse function. Basically, during a training phase, one uses a time series $\mathbf{y}(t)$ of the system subjected to a known perturbation $\mathbf{v^{target}}(t)$ as input for the network. The network will start to synchronize its dynamics to that of the system, so that, for long enough time, its state $\mathbf{u}(t)$ will become a function of the system dynamics:

$$\mathbf{u}(t) \approx \mathbf{G}[\mathbf{y}(t), \mathbf{y}(t+\delta)] \tag{2.6}$$

During this training, the output weights matrix $\mathbf{W_{out}}$ is adjusted so that the known perturbation is reproduced by a linear combination of the network state, obtaining the following set of approximations:

$$\mathbf{F}^{-1}[\mathbf{y}(t), \mathbf{y}(t+\delta)] \approx \mathbf{v^{target}}(t) \approx \mathbf{W_{out}}\mathbf{u}(t) \approx \mathbf{W_{out}}\mathbf{G}[\mathbf{y}(t), \mathbf{y}(t+\delta)] \tag{2.7}$$

In this way, one ends up with a matrix $\mathbf{W_{out}}$ which can basically produce the external perturbation driving the system from a state $\mathbf{r}(t)$ to a desired state $\mathbf{r}(t+\delta)$ if applied on the network state obtained by feeding $[\mathbf{r}(t), \mathbf{r}(t+\delta)]$ as input.
It is important to remember that, in order for this approach to be effective, the time series should be long enough to ensure the network synchronizes to the system, and training must be done with a perturbation $\mathbf{v^{target}}(t)$ rich enough to ensure that the system is stimulated with many frequencies and explores as much of the phase space as possible[3].

2.3 Dynamical Model of the Brain

We modeled the dynamics of the brain network as a set of Landau-Stuart oscillators close to a Hopf bifurcation. In fact, the resting-state spontaneous alpha activity of the brain bursts erratically between two distinct modes of activity. It has been shown[6], [7] that these bursts between low- and high-amplitude alpha oscillations are compatible with the particular type of dynamical instability described by an Hopf bifurcation.
We then made use of a whole-brain model made up of a set of nodes, each of which represents a distinct anatomical or functional region of the brain. The activity of these nodes is described by the normal form of a Hopf bifurcation, which exhibits a stable fixed point or a stable limit cycle, depending on a control parameter a. Such a bifurcation is described by the following equation of complex state variable z:

$$\frac{dz}{dt} = [a - \kappa|z|^2]z + i\omega z + \eta \tag{2.8}$$

In our model, we assume that the BOLD signal of each node is given by the real part x_j of a state variable z_j evolving through equation 2.8 with the addition of a coupling between the nodes. Specifically, the following set of coupled equations in cartesian coordinates

describes the whole-brain dynamics[4], [5]:

$$\begin{aligned}\frac{dx_j}{dt} &= [a_j - x_j^2 - y_j^2]x_j - \omega_j y_j + G\sum_{i=1}^{N} C_{ij}(x_i - x_j) + \beta\eta_j(t) \\ \frac{dy_j}{dt} &= [a_j - x_j^2 - y_j^2]y_j + \omega_j x_j + G\sum_{i=1}^{N} C_{ij}(y_i - y_j) + \beta\eta_j(t)\end{aligned} \tag{2.9}$$

where $j = 1, \ldots, N$ is the node index. In these equations:

- C_{ij} is a matrix representing the structural connectivity of the brain (SC), a quantitative measure of the physical connections between the different regions; it is used to couple the nodes through the simplest lower order linear coupling.
- G is a global scaling factor to adjust the weight of the couplings.
- η_j represents a Gaussian random noise (zero mean, unit standard deviation).
- a_j are the bifurcation parameters. If $a_j > a_c$, where a_c is a critical value (zero in absence of coupling), the system engages in a stable limit cycle with frequency $f_j = \omega_j/2\pi$, while if $a_j < a_c$ the local dynamics converge to a stable fixed point representing a low-activity noisy state.

In the methodology section, we will discuss how these parameters have been fixed.

Chapter 3

Methodology

3.1 Data and Preprocessing

We used the 96 unrelated subjects' subset from the Human Connectome Project (HCP)[18]. For the main analysis, we used the left-right (LR) phase-encoding runs from the first session resting state fMRI data. We later replicated the analysis for the left-right (LR) phase-encoding runs from the second session resting state fMRI data.
Recordings had $\approx$ 15 min duration with a TR of 0.72 sec. The full description of the imaging parameters and minimal preprocessing pipeline is given in Ref.[8]. In short, after correction for motion, gradient, and susceptibility distortions the fMRI data was aligned to an anatomical image. The aligned functional image was corrected for intensity bias, demeaned, and projected to a common surface space, which resulted in a cifti-file. All fMRI data were filtered between 0.1 and 0.01 Hz to retain the relevant frequency range for further analyses of the BOLD signal.
We obtain structural and functional matrices in different spatial scales using the Schaefer parcellation[16], which optimizes local gradient and global similarity measures of the fMRI signal in various spatial scales. We considered the spatial scale corresponding to 100 regions. In both fMRI datasets time series were extracted with the help Workbench Command provided by the HCP.

3.2 Setting the whole-brain model

Before using the equations 2.9 to produce the temporal sequences for the neural network, we needed to find the optimal values of the parameters in order to adjust the model to fit the empirical data.

Finding the frequencies. First of all, we found the power spectrum of the data. Some of the signals are visible in Figure 3.1. We fixed the frequencies f_j for the equations as the maximum of the power spectrum of each signal. As can be seen in Figure 3.2, most of

the frequency peaks are in the $[0.045, 0.055]$Hz range.

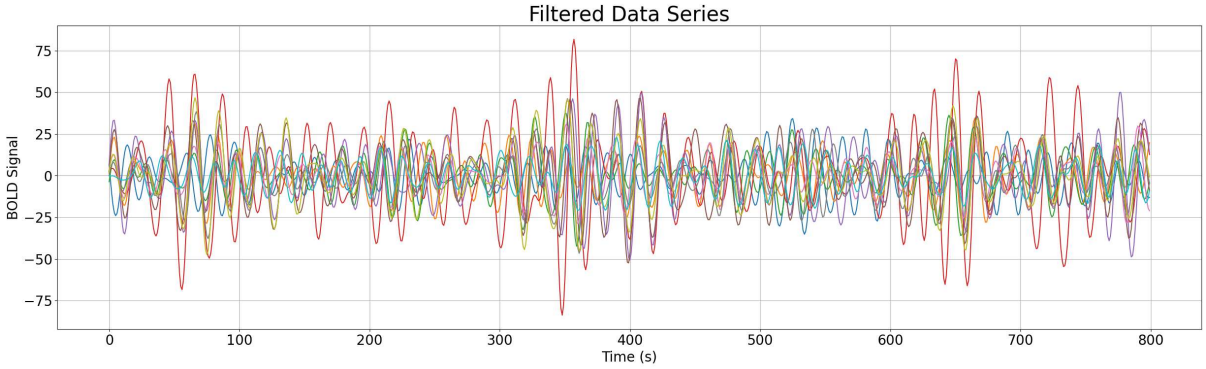
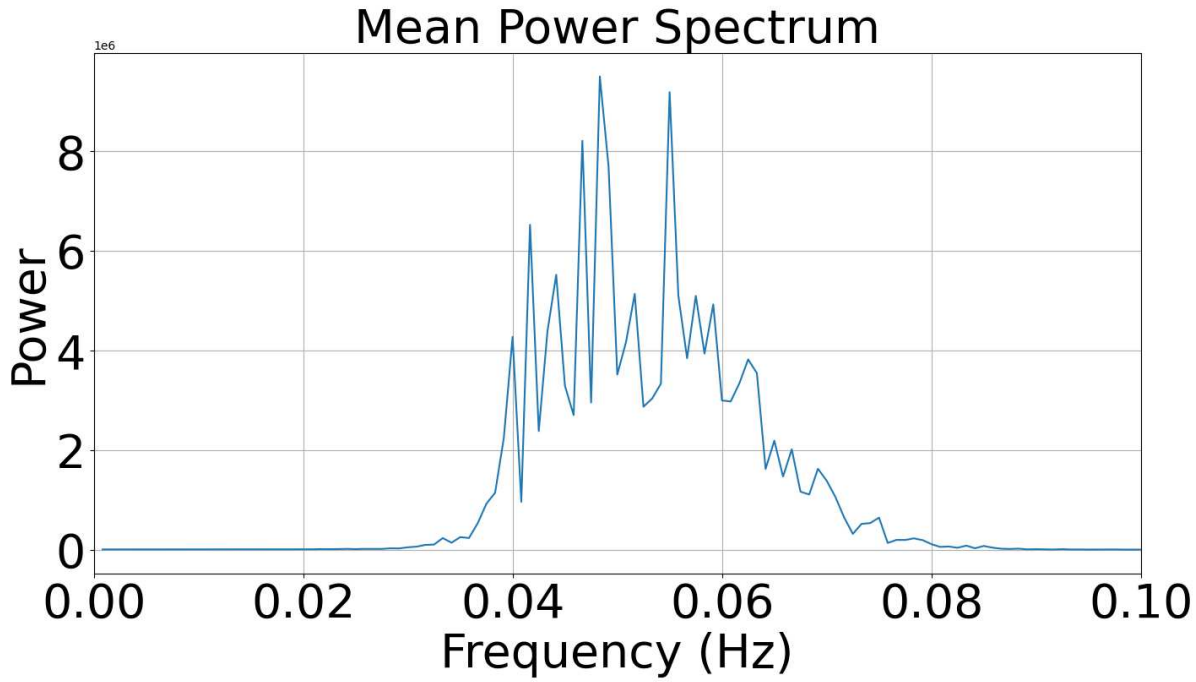


Figure 3.1: Filtered BOLD signals of 10 nodes chosen arbitrarily.

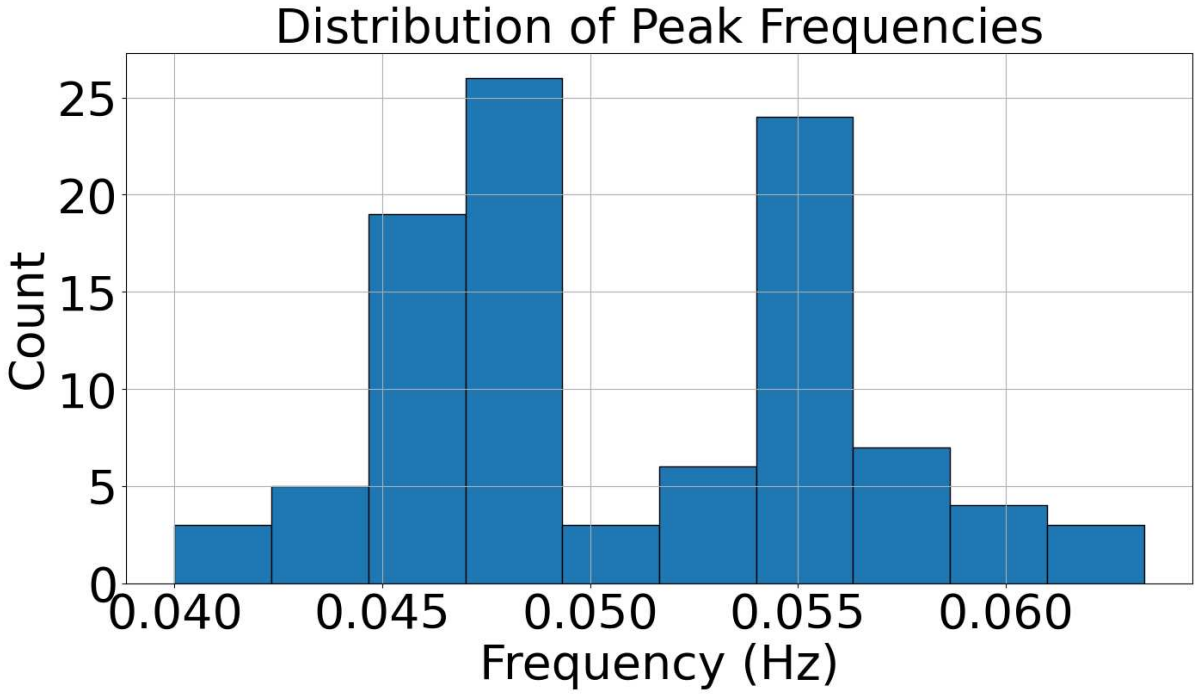
Fixing the bifurcation parameters. It has been shown[5], [6] that the resting state of the brain is a metastable state corresponding to a subcritical Hopf dynamics driven by noise. The noise plays a crucial role in the subcritical regime, avoiding that the signal of the oscillators dies over time. As in [4], we then fixed a noise scaling $\beta = 0.02$ and set the parameters a_j to zero, making sure we were in the desired subcritical regime.
This choice is not only supported by previous studies, but also validated by a high match between empirical and simulated time series, further increased by adjusting the connection weights.
Finding the connection weight scaling. Connection weights were fixed by maximizing the similarity between temporal synchronization properties of the original and simulated signal. A measure of synchronization between pairs of nodes can be obtained by Hilbert transforming the signals. This transformation yields an analytic signal from the original signal, made up of an amplitude part, representing the envelope (or instantaneous amplitude) of the original signal, and a phasor, whose phase represents the instantaneous phase of the original signal. After transforming the signals of all nodes, one can evaluate a matrix **FC**, called phase coherence matrix, as the time average of the following matrix:

$$\mathbf{dFC}_{ij}(t) = cos(\theta_i(t) - \theta_j(t)) \tag{3.1}$$

where $\theta_i(t)$ is the instantaneous phase of node i. The **FC** matrix reflects the "functional connectivity" of different brain regions, which is correlated but not equal to the structural connectivity. Functional connectivity expresses the temporal correlation between different brain regions, capturing the level of synchronization of those regions. In fact, non-vanishing functional connectivity can also occur between regions without direct structural connections, mediated by indirect pathways.
For different G values, we generated solutions of equation 2.9 and computed the **FC** matrix. Then we computed the Pearson correlation coefficient between the **FC** matrix of the model and of the empirical data. We fixed $G = 0.02$, as the value for which the



(a) Power spectrum mediated over all the signals.



(b) Histogram of the peak frequencies.

Figure 3.2: Frequency components of the signals.

correlation is maximum (see Figure 3.3).

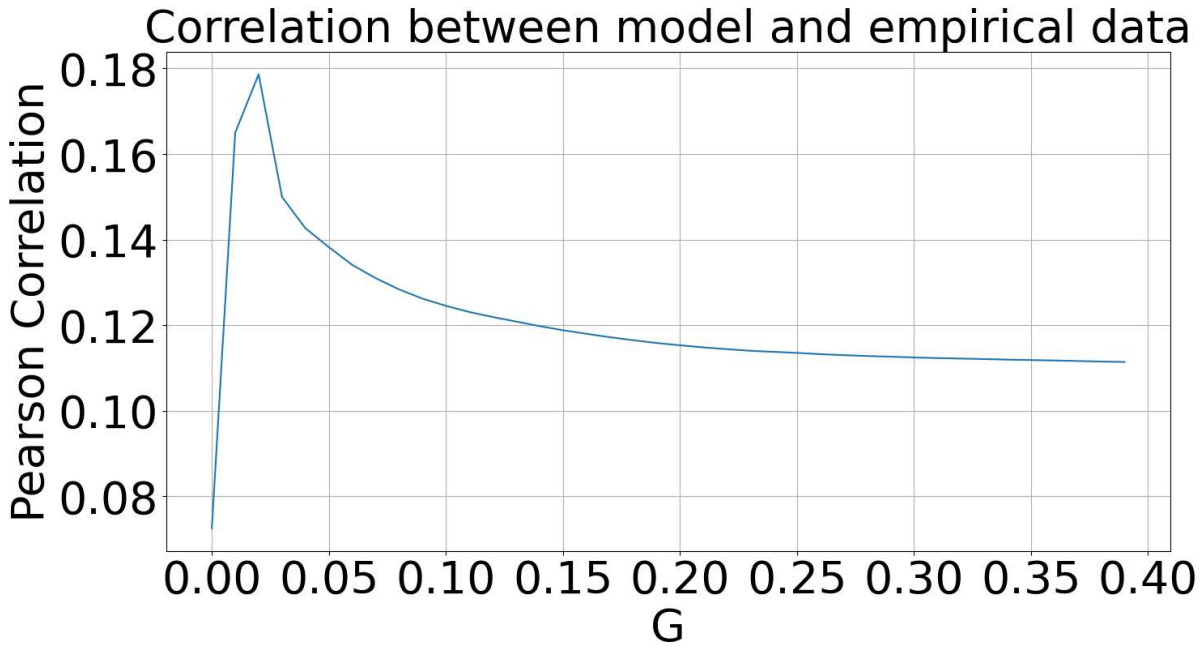


Figure 3.3: Correlation between model and empirical **FC** matrix at different G values.

Updating the connectivity matrix. The structural connectivity matrix C_{ij} (see left hand side of Figure 3.4) is obtained with dMRI, which is affected by heavy biases, including the fact that it cannot capture connectivity asymmetries and it often misses connections in the opposite hemisphere of the brain[4].

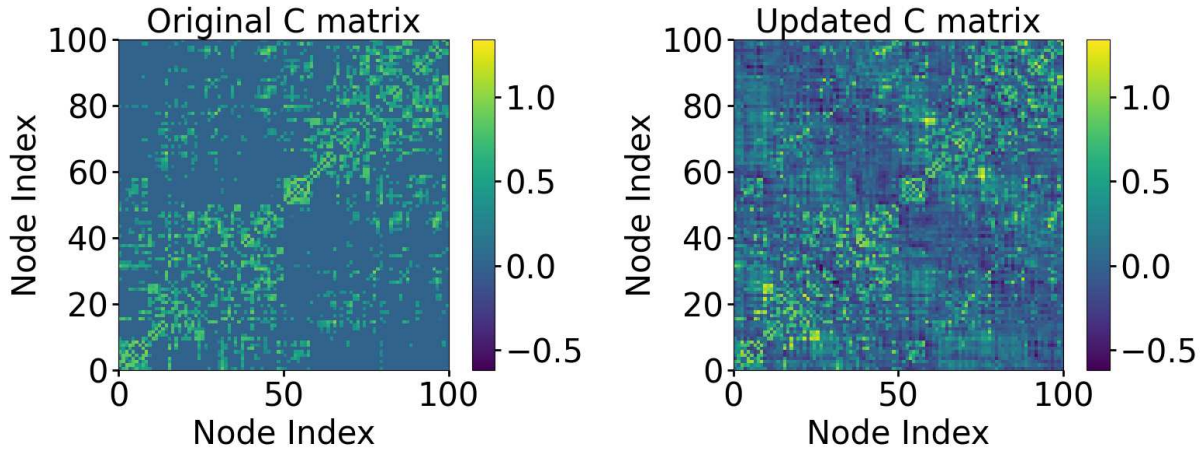
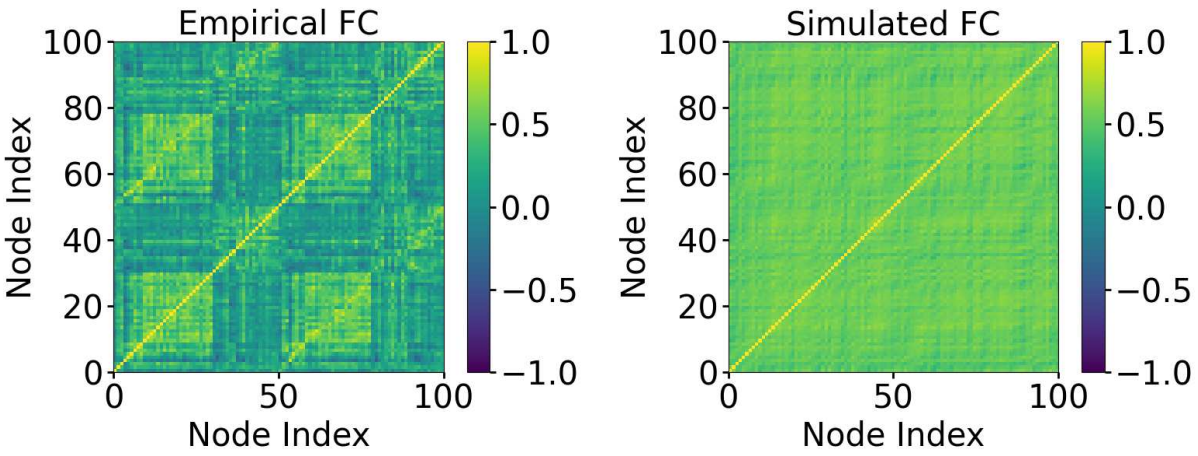


Figure 3.4: Comparison between connectivity matrices before and after the update.

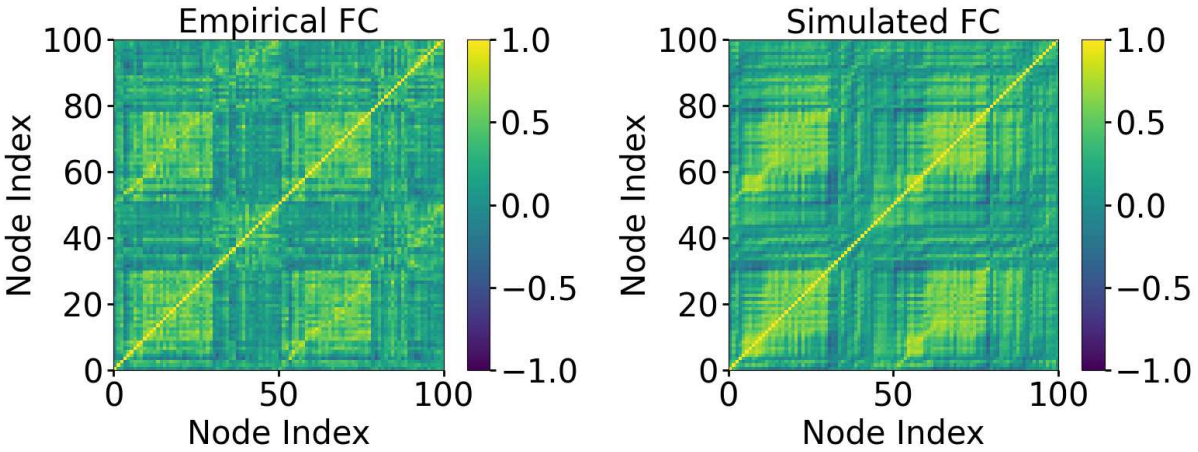
For this reason, we iteratively updated all connections in order to access the potential missing components of the anatomical connectivity, making use of the the following update rule:

$$C_{ij} = C_{ij} + \epsilon(FC_{ij}^{emp} - FC_{ij}^{sim}) \tag{3.2}$$

In practice, at each iteration we simulated the node signals, evaluated the **FC** matrix of the simulated series, updated C_{ij} through equation 3.2 with $\epsilon = 0.01$, and then used the updated matrix in the next iteration. We iterated the process until the variation was negligibly small.
It is worth noting that in this way we obtain simulated series whose **FC** matrix is strongly similar to that of the original data, as shown in Figure 3.5.



(a) Before the update.



(b) After the update.

Figure 3.5: Comparison between phase coherence matrices before and after the update procedure.

After all these steps, we could reach a really good agreement between the original empirical signals and the simulated ones, with a 0.63 correlation coefficient between the **FC** matrices of empirical and simulated data. As an example, in Fig. 3.6 it can be seen a comparison between the empirical signal and the model prediction for node 0 (the different amplitude values are just a matter of scaling).

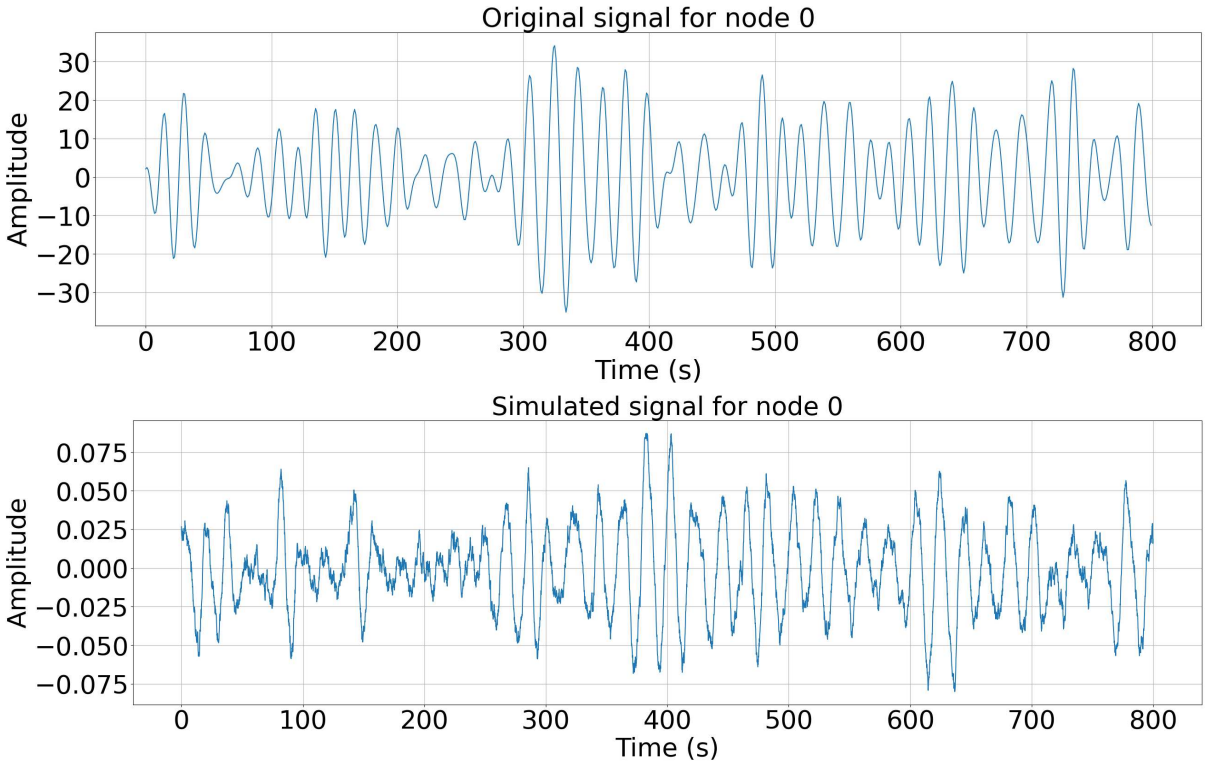


Figure 3.6: Comparison between original and simulated signal for node 0.

3.3 Building the Echo State Network

As pointed out in Chapter 2.2, an Echo State Network generates an output which is just a linear combination of the reservoir neuron activities. Specifically, given a N_{input} dimensional vector input $\mathbf{u}(t)$ at time steps t, and a $N_{network}$ dimensional vector $\mathbf{x}(t)$ representing the state of the reservoir, the network evolves his hidden state through the following equation:

$$\mathbf{x}(t) = (1-\alpha)\mathbf{x}(t-1) + \alpha \tanh(\mathbf{W_{in}}[1; \mathbf{u}(t-1)] + \mathbf{W}\mathbf{x}(t-1) \tag{3.3}$$

where α is a parameter called leaking rate (which we will discuss further), $\mathbf{W_{in}}$ the $N_{network} \times (1 + N_{input})$ input matrix, $\mathbf{W}$ the $N_{network} \times N_{network}$ matrix of internal connections and $[\cdot;\cdot]$ represents the vertical concatenation of vectors. The entry equals to one is used to stabilize the dynamics[13].
The output vector $\mathbf{y}(t)$ is then obtained through the relation[13]:

$$\mathbf{y}(t) = \mathbf{W_{out}}[1; \mathbf{u}(t); \mathbf{x}(t)] \tag{3.4}$$

where $\mathbf{W_{out}}$ is the $N_{output} \times (1 + N_{input} + N_{network})$ output matrix, which is fixed after a training phase where we make use of Ridge regression:

$$\mathbf{W_{out}} = \mathbf{y^{target}}\mathbf{x^T}(\mathbf{x}\mathbf{x^T} + \gamma\mathbf{1})^{-1} \tag{3.5}$$

Here, $\mathbf{y^{target}}(t)$ is a target output chosen to train the network and γ is known as Ridge parameter, a regularization coefficient preventing over-fitting and feedback instability. Notice that the output matrix is time independent and that the more time steps one has, the more efficient the training will be.

Following the guidelines in Section 2.2, we generated the entries of matrices $\mathbf{W_{in}}$ and $\mathbf{W}$ according to a normal distribution with zero mean and standard deviation to be adjusted manually, depending on the task to be performed, in order to optimize the training phase. $\mathbf{W}$ was always sparse and rescaled to ensure the echo state property. The leaking rate α and the Ridge parameter γ were also adjusted through repeated runs to optimize the training. The training phase was also made up of a validation part, where we checked if the goodness of the training was just due to over-fitting or it was actually due to the network dynamics having captured the features of the task.

In each section of the Results chapter, the values used for the parameters will be pointed out.

Chapter 4

Results

As pointed out in the Introduction, our main goal was to study the possibility of stimulating a node of choice by acting on a group of different nodes. The effect of an electro-stimulation applied through electrodes or magnetic fields to a node can be modeled in two ways:

- as an "additive" term which shifts the node signal by a certain amount, exciting or inhibiting it;
- as a variation of the bifurcation parameter a_j of the node, with the effect of increasing or reducing its excitability and hence raise or lower the oscillation amplitude.

We decided to test both kinds of stimulation model. The first kind, where some node signals can be directly shifted, can be described by the following equations:

$$\begin{aligned}\frac{dx_j}{dt} &= [a_j - x_j^2 - y_j^2]x_j - \omega_j y_j + G\sum_{i=1}^{N} C_{ij}(x_i - x_j) + p_j(t) + \beta\eta_j(t)\\ \frac{dy_j}{dt} &= [a_j - x_j^2 - y_j^2]y_j + \omega_j x_j + G\sum_{i=1}^{N} C_{ij}(y_i - y_j) + \beta\eta_j(t)\end{aligned} \tag{4.1}$$

where a term $p_j(t)$ is added to equations 2.9. We will refer to this stimulation model as "perturbation" case. This is the approach used by most of the previous works applying control theory to the brain, such as in [17].
The second kind, where the external stimuli can make the amplitude vary with time, can be described as follows:

$$\begin{aligned}\frac{dx_j}{dt} &= [a_j + m_j(t) - x_j^2 - y_j^2]x_j - \omega_j y_j + G\sum_{i=1}^{N} C_{ij}(x_i - x_j) + \beta\eta_j(t)\\ \frac{dy_j}{dt} &= [a_j + m_j(t) - x_j^2 - y_j^2]y_j + \omega_j x_j + G\sum_{i=1}^{N} C_{ij}(y_i - y_j) + \beta\eta_j(t)\end{aligned} \tag{4.2}$$

where $m_j(t)$ represents the variation of the amplitude of node j. We will refer to this stimulation model as "modulation" case. This is the approach recently followed in [4].
For both cases, we used a sequence of positive or negative square pulses as external inputs ($p_j(t)$ or $m_j(t)$), since this kind of pulses are the easiest to generate in practical applications.

4.1 Perturbations

The first attempt was to study the "perturbation" case (equations 4.1).
We supposed that out of N total nodes, one has the possibility to directly stimulate only nodes $i_1, \ldots, i_n$ and that one wants to perturb node k, which isn't between the accessible nodes. During a training phase, we generated a solution to equations 4.1 where perturbations $p_{i_1}(t), \ldots, p_{i_n}(t)$ in the form of trains of positive and negative square pulses at random times were acting on the accessible nodes, while the remaining $i_{n+1}, \ldots, i_N$ nodes were unperturbed. It is clear that since the equations are coupled, the perturbations also affect the unperturbed nodes in an indirect way.
It is then possible to feed the signals of the unperturbed nodes to the neural network as input, and train it to produce the $p_{i_1}(t), \ldots, p_{i_n}(t)$ perturbations as target output. In this way, we should end up with a network which is able to receive solutions of equations 4.1 where only node k (chosen between nodes $i_{n+1}, \ldots, i_N$) is perturbed (which we call desired signal) and give back the perturbations one should apply to nodes $i_1, \ldots, i_n$ in order to obtain the desired signals for nodes $i_{n+1}, \ldots, i_N$ and in particular for node k.
We performed the analysis on a subset of the total nodes so that the network could work with a lesser complex system. We started working with 30 nodes. Between them, the number of nodes to be used as accessible nodes should be as little as possible, but sufficiently large to ensure that every other node could be sufficiently connected to them. For this reason we randomly selected a third of the 30 nodes, namely nodes [1, 2, 3, 11, 13, 14, 20, 22, 23, 24], as nodes to be perturbed. We generated the solution of equations 4.1 for an interval $[0, 30000]$s using 300000 time steps, with perturbations $p_1(t), p_2(t), \cdots, p_{24}(t)$ in the form of trains of positive and negative square pulses with amplitude uniformly distributed in range $[0.2, 0.8]$ and width 300s acting at random instants. The solution obtained (except an initial discarded transient) was used to train the network, using the unperturbed node signals as input $\mathbf{u}(t)$ and the acting perturbations as target output $\mathbf{y}^{\mathbf{target}}(t)$ (refer to Section 3.3 for details). Then we generated a different set of perturbations of the same type acting on the same nodes, but at different random instants. The solution obtained was used to validate the training. In fact, in order to be sure that a neural network actually learned to map the input to the output, and that a good result isn't just a consequence of over-fitting, one has to verify that the neural

network works well with different inputs.
As a quantitative measure of the goodness of the network outputs in training and validation phases we evaluated the normalized mean square error (NRMSE) between the target output (consisting in the perturbations on the selected nodes) and the output produced by the network. Specifically, if $\mathbf{y^{target}}(t)$ is the vector made up of all the target perturbations and $\mathbf{y}(t)$ the output of the network, the NRMSE is evaluated as the mean over time steps and over entries of the squared difference $\mathbf{y^{target}}(t) - \mathbf{y}(t)$, normalized by the maximum perturbation amplitude.
We used a network of 1000 neurons, fixing a ridge parameter $\gamma = 1$, $\mathbf{W_{in}}$ with entries normally distributed with unit variance and $\mathbf{W}$ with sparse entries (density 1%) normally distributed with unit variance. In order to optimize the ESN performance, we repeated training and validation phases for several values of the leaking rate in order to fix its parameters at the value which ensured the minimum NRMSE for validation. Notice that since the NRMSE is a mean over a lot of time instants (≈ 300000), small variations of NMRSE can correspond to drastic changes in the quality of the network output. We fixed $\alpha = 0.004$, having a 0.17 NRMSE during training and a 0.23 NRMSE during validation. An example of the outputs produced by the network during training and validation is visible in Figure 4.2a and 4.3a respectively. We should stress that, since the network is learning to produce these perturbations from the signals of the unperturbed nodes, which are more or less affected by the perturbed ones depending on the different strengths of the connections, some perturbation is reproduced better than others.
Having completed the training, we tested the performance of the network. We wanted to verify if the ESN could predict the perturbations one should apply to the accessible brain nodes in order to have an effect equivalent to the direct perturbation of an inaccessible target node. Thus, we simulated the activity of the brain subjected to a direct perturbation of the target node, and used the corresponding time series of the inaccessible nodes as input to the ESN. If working correctly, the ESN should predict the perturbations one needs to apply to the accessible nodes in order to obtain those time series.
As an example, consider the case where one wants to act on node 0 as the target node. We generated a solution of equations 4.1 with $p_0(t)$ as the only non-zero perturbation in the form of a square wave of amplitude 0.2 and frequency 0.001Hz. This solution represents a situation where node 0 is activated. With the signals of nodes [0, 4, 5, 6, 7, 8, 9, 10, 12, 15, 16, 17, 18, 19, 21, 25, 26, 27, 28, 29] as input, the ESN produced the perturbations to be applied to the rest of the nodes (the accessible ones) to obtain those signals. We then generated the solution of equations 4.1 when the $p_j(t)$ are the ones produced by the ESN and compared the time series of nodes [0, 4, 5, 6, 7, 8, 9, 10, 12, 15, 16, 17, 18, 19, 21, 25, 26, 27, 28, 29] to the desired ones, which were given as inputs to the network. If the ESN is able to correctly predict the external inputs to apply to the accessible nodes in order to activate node 0, the two solutions should be similar. We repeated the process many times,

trying to activate every single node one by one.
As an example, we report results obtained when trying to activate node 25. The network predicts that some nodes should be more perturbed, e.g. node 14 (see Figure 4.4a), some should be less perturbed, e.g. 23 (see Figure 4.5a), or not at all.
Notice that in this "perturbation" case we were unable to obtain the desired solution when applying the predicted perturbations. We reported the signals of node 25 and node 10 when trying to activate 25 (Figure 4.6a and 4.7a). Node 25 could only be slightly activated, despite the intense perturbations applied.
To keep track of the goodness of the activation procedure of the various nodes we report the matrix in Figure 4.8a. In position (i, j) of the matrix it is reported the NRMSE of the signal of node j when trying to activate node i. This NRMSE is evaluated as the root mean square difference over the time steps between the desired signal for node j and the signal produced by applying the ESN's predicted perturbations, normalized by the root mean square of the signal produced. Notice how the off-diagonal elements have small entries, meaning that the ESN is able to produce perturbations that correctly don't lead to an activation of some undesired node. The large diagonal elements, however, underline the failure in the activation of the desired node.

4.1.1 Training with Uniformly Randomized Perturbations

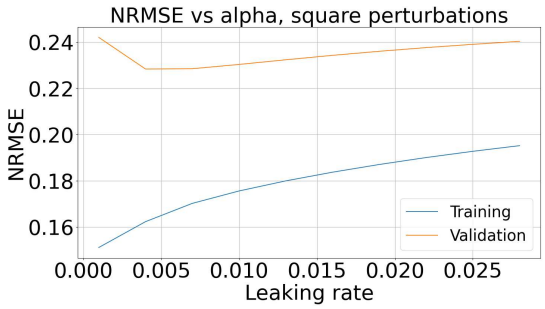
In order to improve the training phase it could be useful to make use of perturbations that could let the plant dynamics explore more of its phase-space. For this reason, following the approach described in [3], we generated sequences of random values uniformly distributed in $[-0.8, 0.8]$ for each time step. These perturbations are less realistic with current neurostimulation techniques. The sequences were filtered with a low-pass cutoff of the frequencies greater than $1/\delta$, where δ was a number much smaller than the characteristic time of variation of the external inputs (remember the discussion on the inversion of the plant dynamics in Section 2.2.1). Since we used square wave perturbations of period 1000s to generate active node series, we chose $\delta = 100$s. We are thus cutting off frequencies much greater than the characteristic frequency of the perturbation applied to the active node. To be consistent with the previous attempt, we chose the same set of accessible nodes and we repeated the same steps. The only different parameter for the ESN was the leaking rate, which was finely tuned as before. In particular, we fixed $\alpha = 0.007$ (see Figure 4.1b) and we obtained a NRMSE of 0.14 during training and 0.17 during validation. Panel (b) of Figures from 4.2 to 4.7 show results to be compared with the previous case. We observe a slight improvement in the training process, but the ESN still fails to correctly activate the desired node as visible in the NRMSE matrix for this attempt (Figure 4.8b).

4.2 Modulations

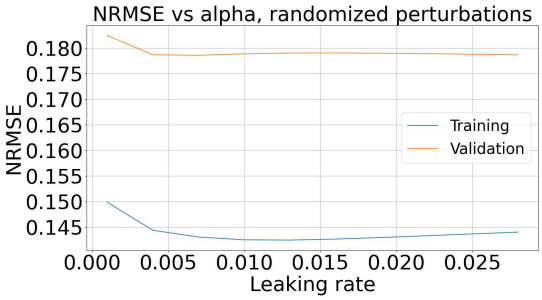
As explained above, the second way to model the external inputs acting on the system is in the form of modulations, i.e. variations of the bifurcation parameters a_j. We chose the same nodes as before as accessible ones. We generated the solution of equations 4.1 for an interval $[0, 30000]$s using 300000 time steps, with modulations $m_1(t), m_2(t), \cdots, m_{24}(t)$ in the form of trains of positive and negative square pulses with amplitude uniformly distributed in range $[0.2, 0.8]$ and width 300s acting at random instants. The solution obtained (except an initial discarded transient) was used to train the network, following the same training and validation steps as before. The leaking rate was adjusted to $\alpha = 0.01$ (see Figure 4.1c). We ended up with a 0.18 NRMSE during training and a 0.22 NRMSE during validation. Panel (c) of Figures from 4.2 to 4.7 show results analogous to the previous cases. Notice that the ESN has problems when learning the negative modulation pulses. This is probably due to the fact that the bifurcation parameters were chosen to stay in the sub-critical regime, which makes the system experience oscillation cycles driven by the noise. If the effect of an external input is to further reduce the amplitude of the oscillations, the effect felt by other nodes in the network is feeble. In this case there is little chance that the ESN can resolve the effect.
Despite the fact that the goodness of the training was comparable to the "perturbation" case, the activation of inaccessible nodes was a significantly better in this case, as can be seen observing the example of node 25 (Figure 4.6c) and the NRMSE matrix (Figure 4.8c).

4.2.1 Training with Uniformly Randomized Modulations

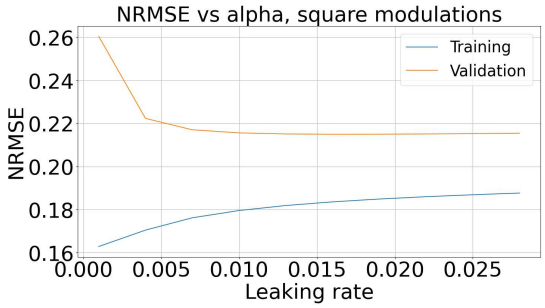
As for the "perturbation" case, we tried to use a randomized signal also for the "modulation" case. For consistency, we chose the same set of accessible nodes and we repeated the usual steps. The randomized modulations were generated like the randomized perturbations in Section 4.1.1. The leaking rate was fixed as $\alpha = 0.003$ (see Figure 4.1d) and we obtained a NRMSE of 0.20 during training and 0.22 during validation. Panel (d) of Figures from 4.2 to 4.7 show results to be compared with the previous cases. The results of the training are worse than the equivalent case in "perturbation". The reason is that, as pointed out earlier, the ESN has difficulties learning negative modulations.
Nonetheless, the results in the activation of remote nodes are comparable with those in Section 4.2 as can be seen in the example of node 25 (Figure 4.6d) and in the NRMSE matrix (Figure 4.8d).



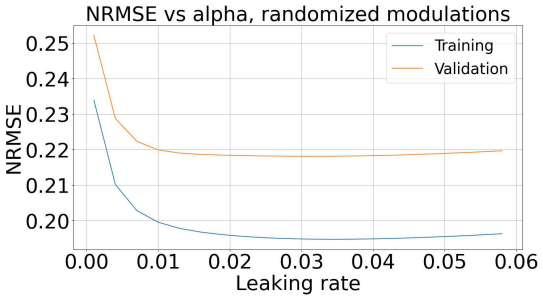
(a) NRMSE for training and validation as a function of the leaking rate using square pulse perturbations.



(b) NRMSE for training and validation as a function of the leaking rate using randomized perturbations.

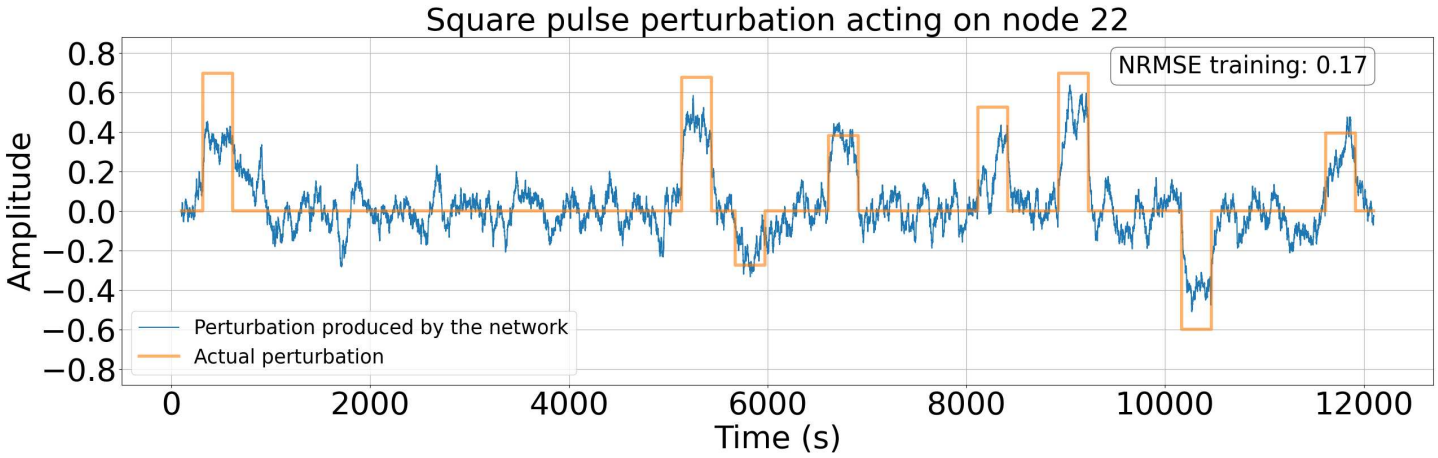


(c) NRMSE for training and validation as a function of the leaking rate using square pulse modulations.

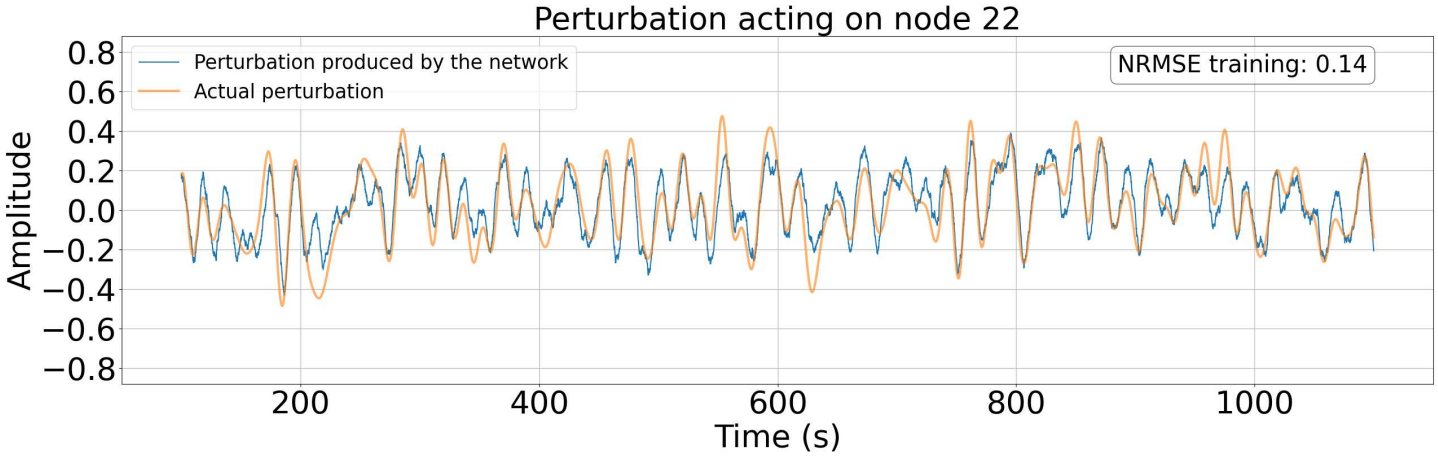


(d) NRMSE for training and validation as a function of the leaking rate using randomized modulations.

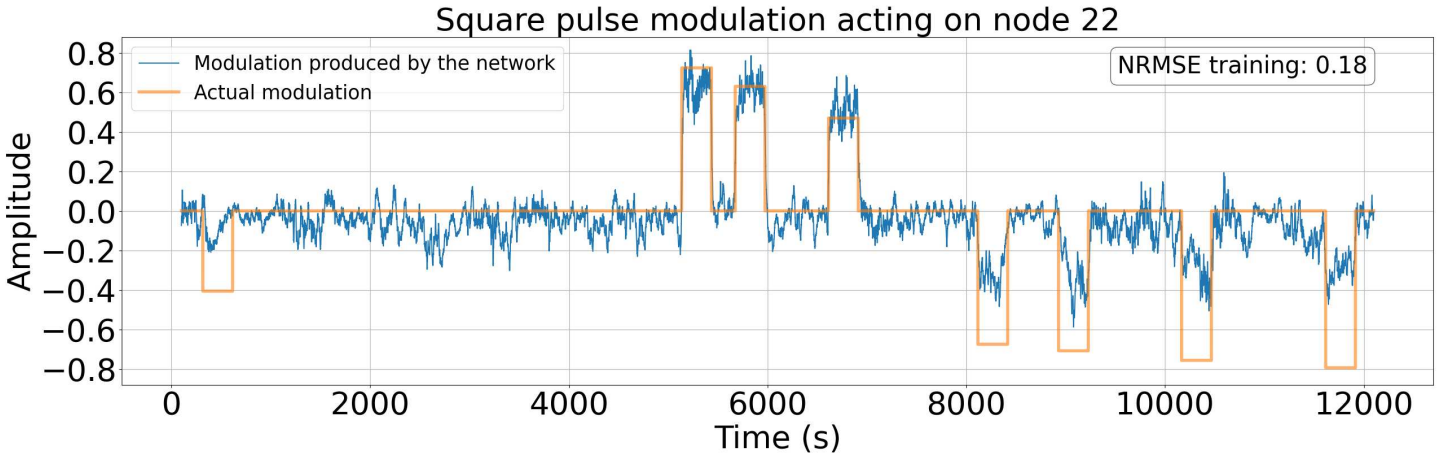
Figure 4.1: NRMSE as a function of the leaking rate for the four cases we studied. The best α was taken to minimize the validation NRMSE.



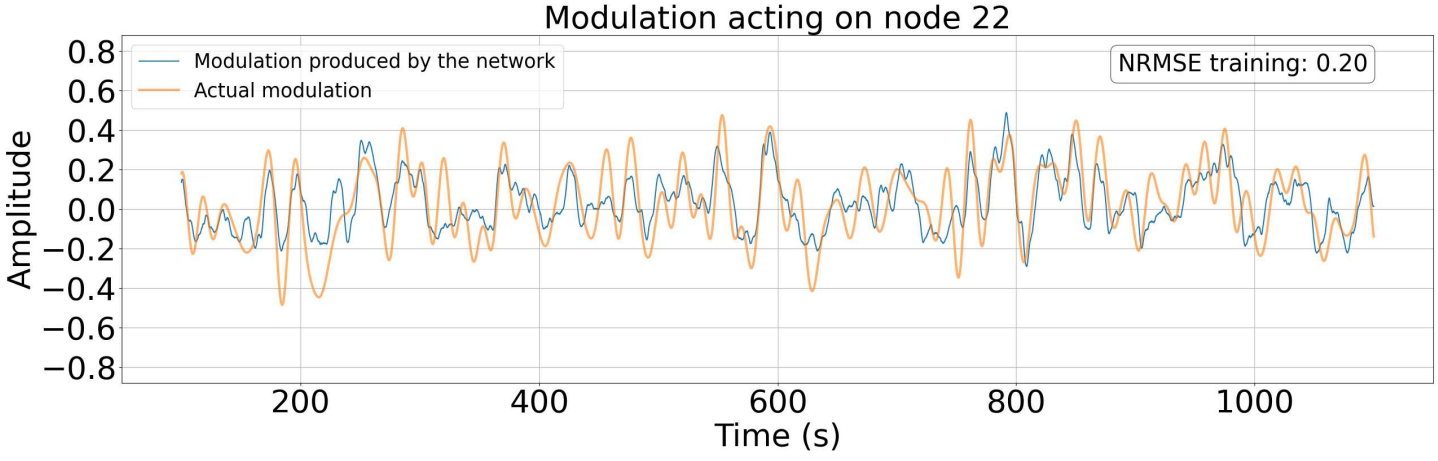
(a) An example of the ESN outputs when training with square pulse perturbations.



(b) An example of the ESN outputs when training with randomized perturbations.

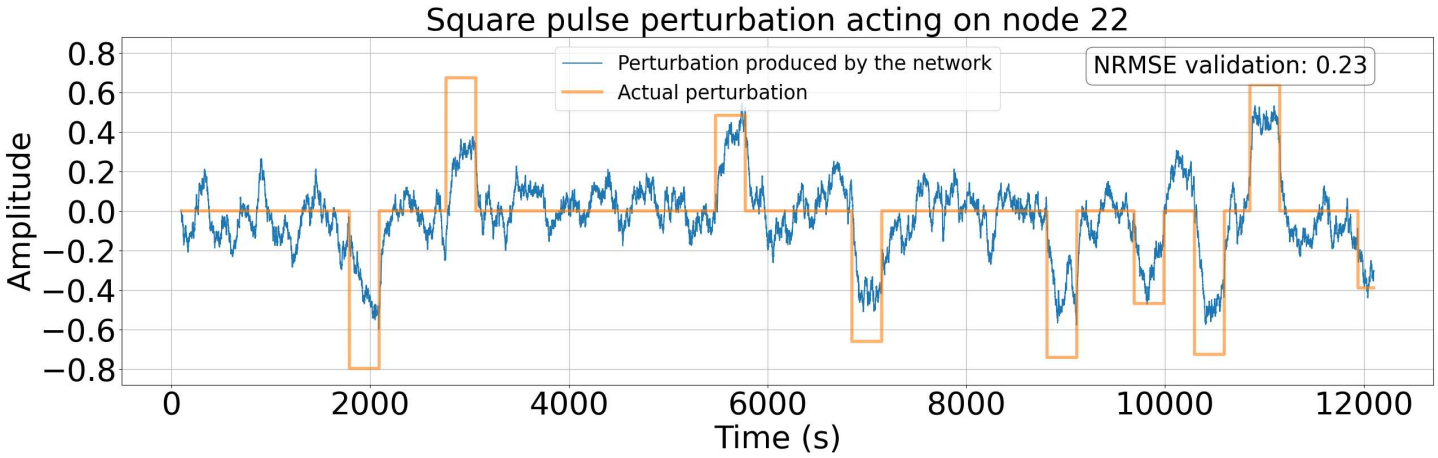


(c) An example of the ESN outputs when training with square pulse modulations.

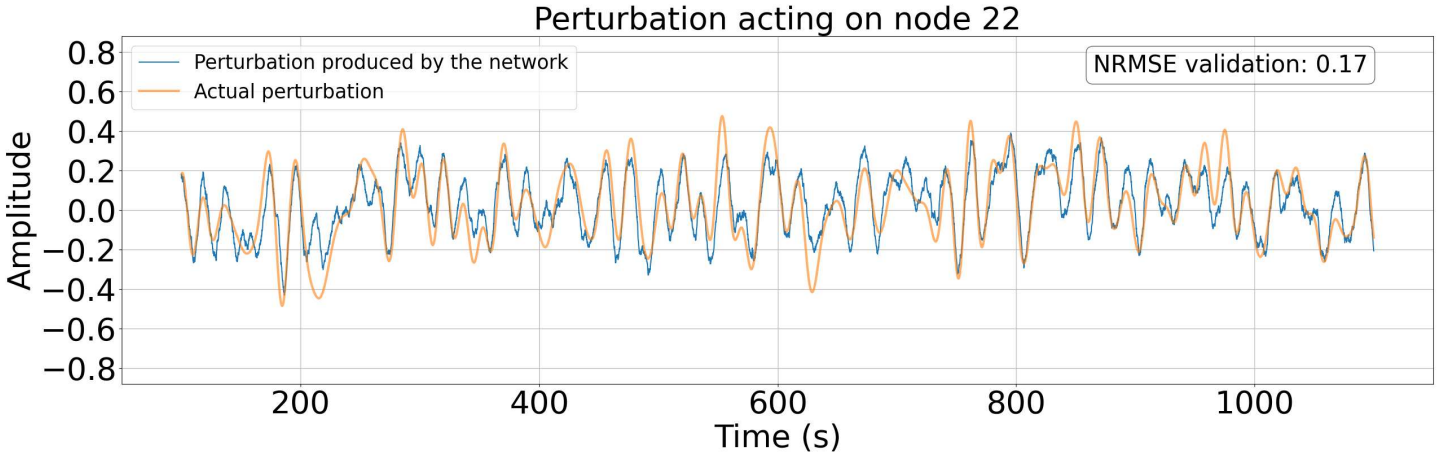


(d) An example of the ESN outputs when training with randomized modulations.

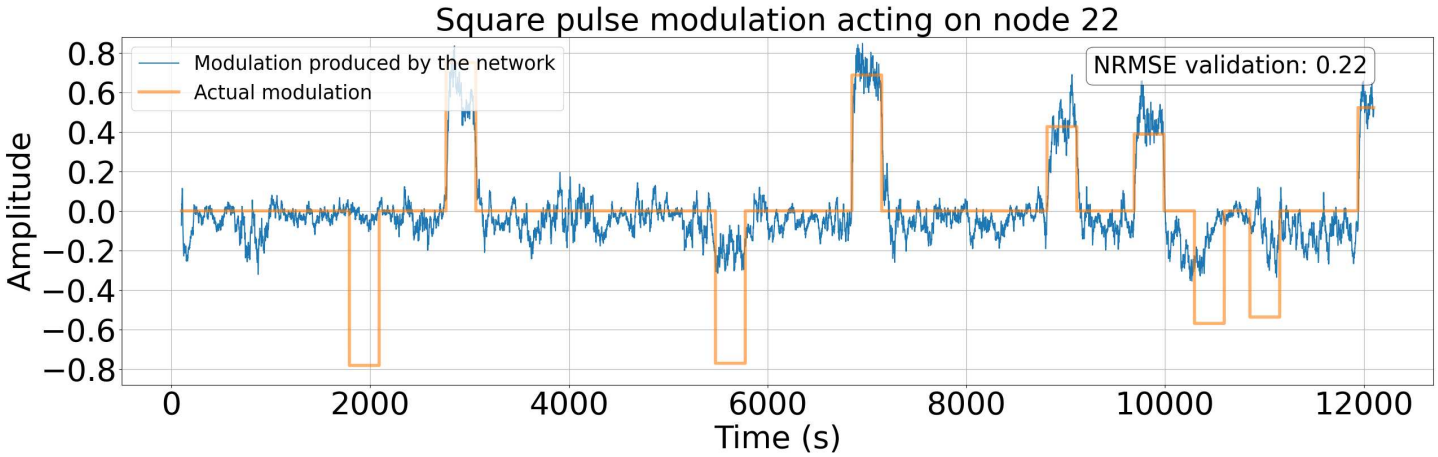
Figure 4.2: Training phase for the four cases. The ESN learns to reproduce the stimuli acting on the accessible nodes (here 22 for example). Notice how the negative modulations are reproduced worse.



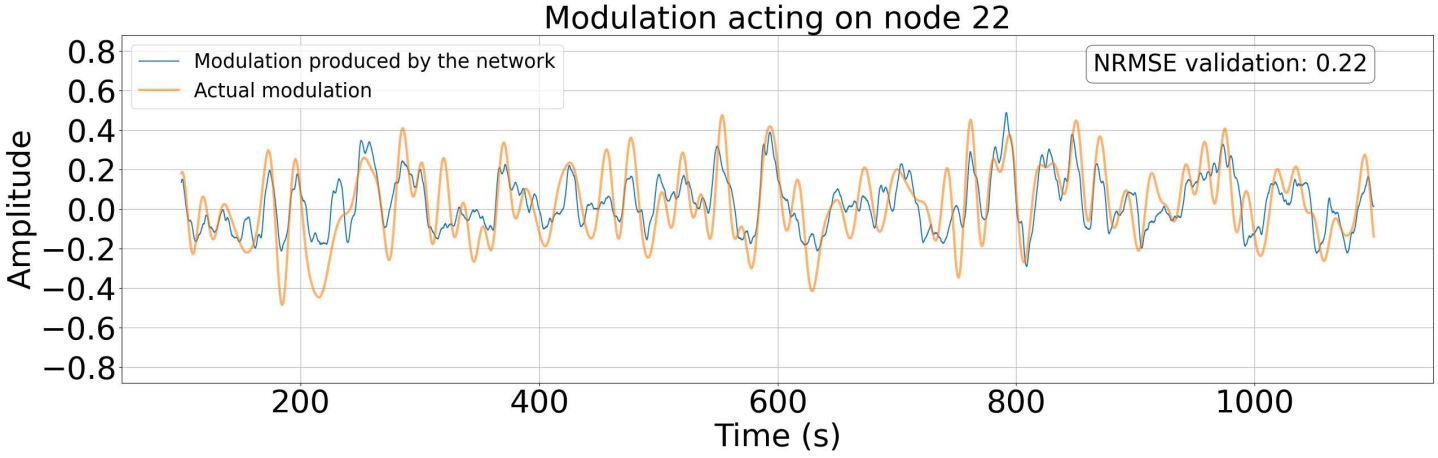
(a) An example of the ESN outputs when validating the training in the case of square pulse perturbations.



(b) An example of the ESN outputs when validating the training in the case of random perturbations.

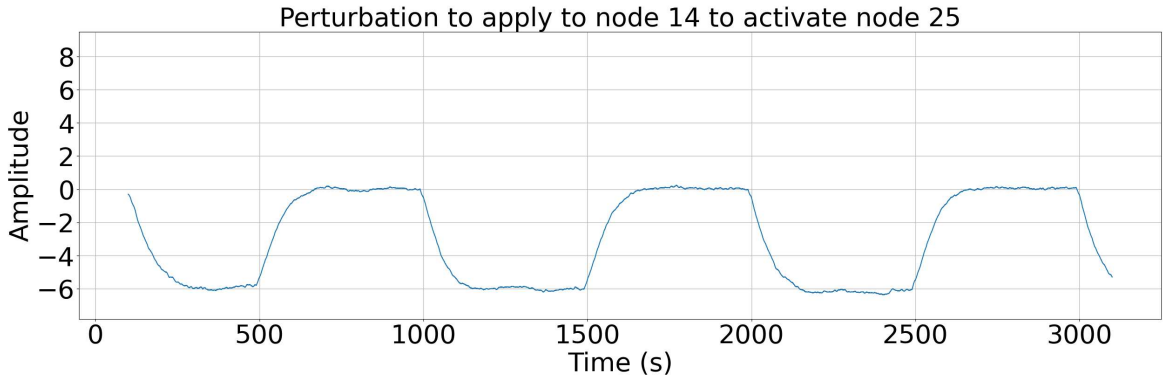


(c) An example of the ESN outputs when validating the training in the case of square pulse modulations.

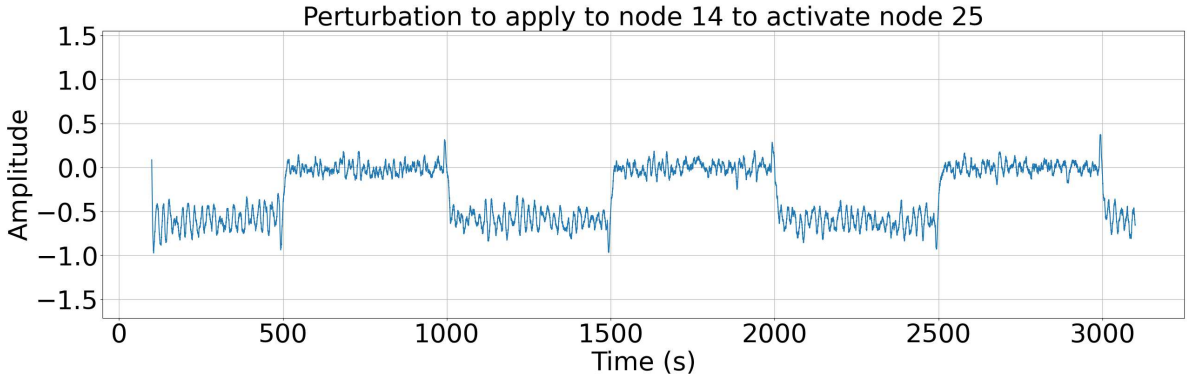


(d) An example of the ESN outputs when validating the training in the case of random modulations.

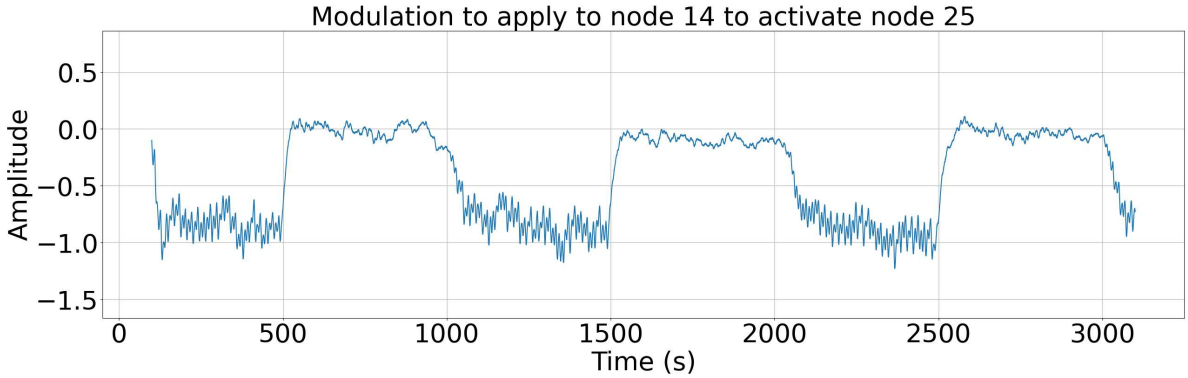
Figure 4.3: Validation phase for the four cases. After the training, the ESN is able to map different inputs in the correct outputs (in this case the external stimuli).



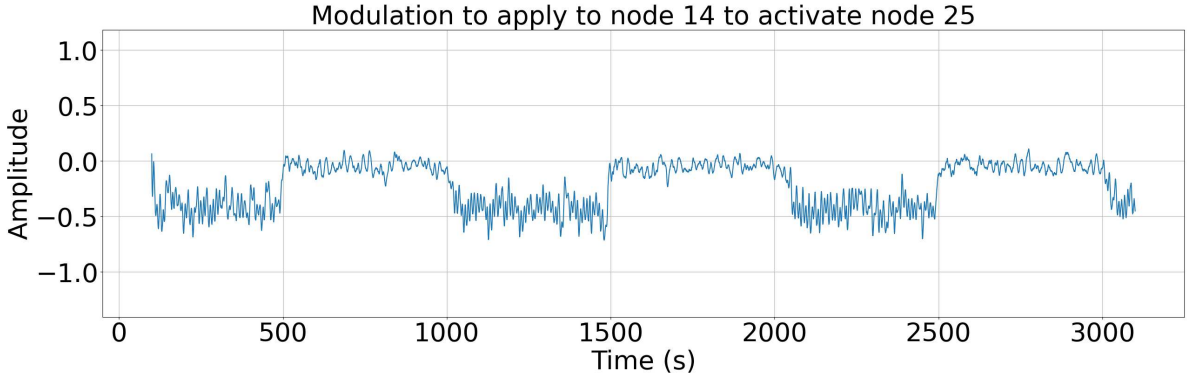
(a) Perturbation to apply to node 14 to activate node 25 as predicted by the ESN trained with square pulse perturbations.



(b) Perturbation to apply to node 14 to activate node 25 as predicted by the ESN trained with random perturbations.

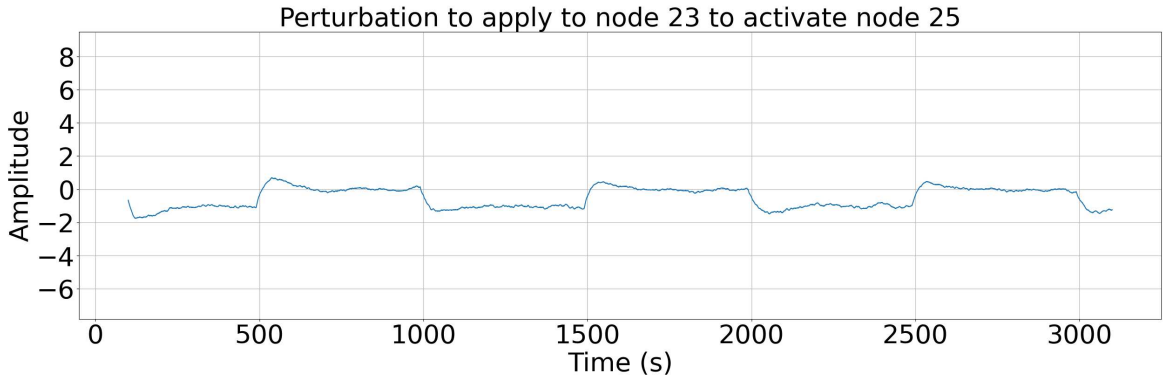


(c) Perturbation to apply to node 14 to activate node 25 as predicted by the ESN trained with square pulse modulations.

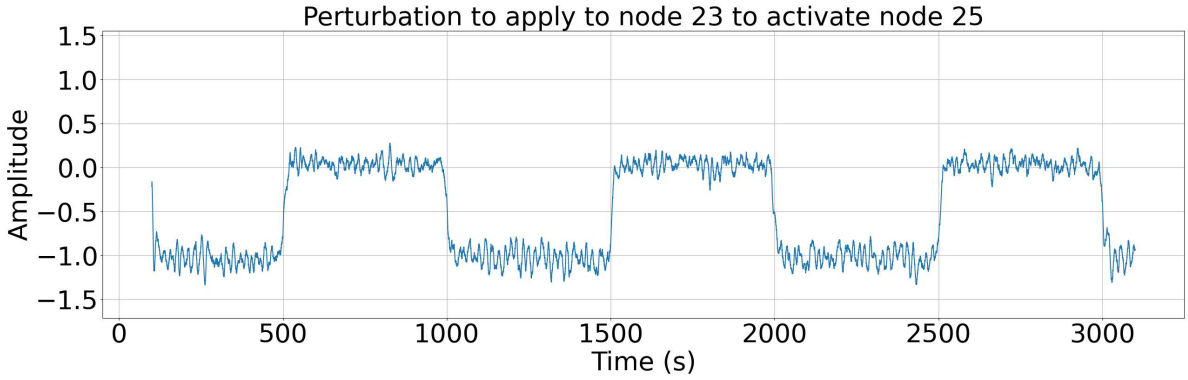


(d) Perturbation to apply to node 14 to activate node 25 as predicted by the ESN trained with random modulations.

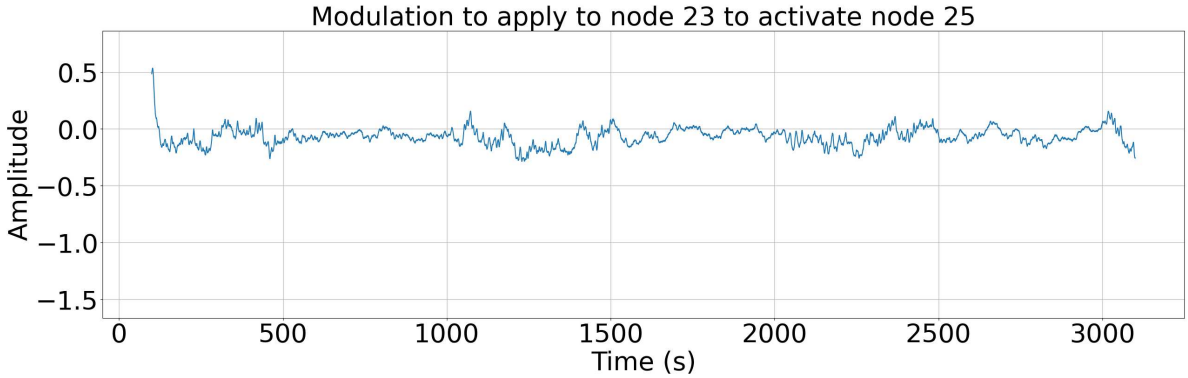
Figure 4.4: An example for the four cases of the external inputs predicted by the ESN in order to activate node 25.



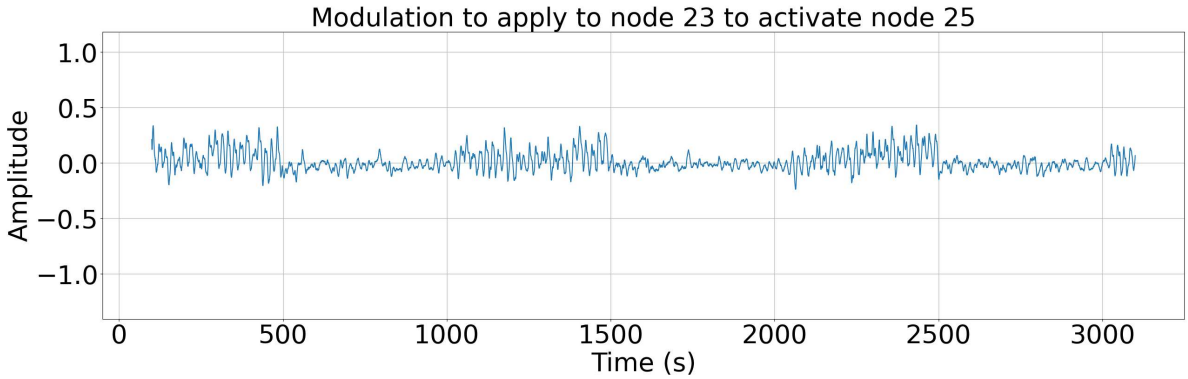
(a) Perturbation to apply to node 23 to activate node 25 as predicted by the ESN trained with random modulations.



(b) Perturbation to apply to node 23 to activate node 25 as predicted by the ESN trained with random perturbations.

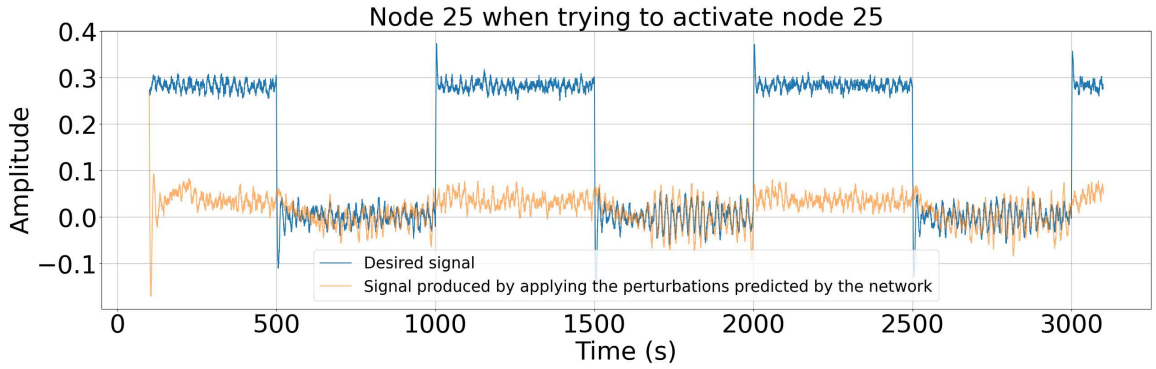


(c) Perturbation to apply to node 23 to activate node 25 as predicted by the ESN trained with square pulse modulations.

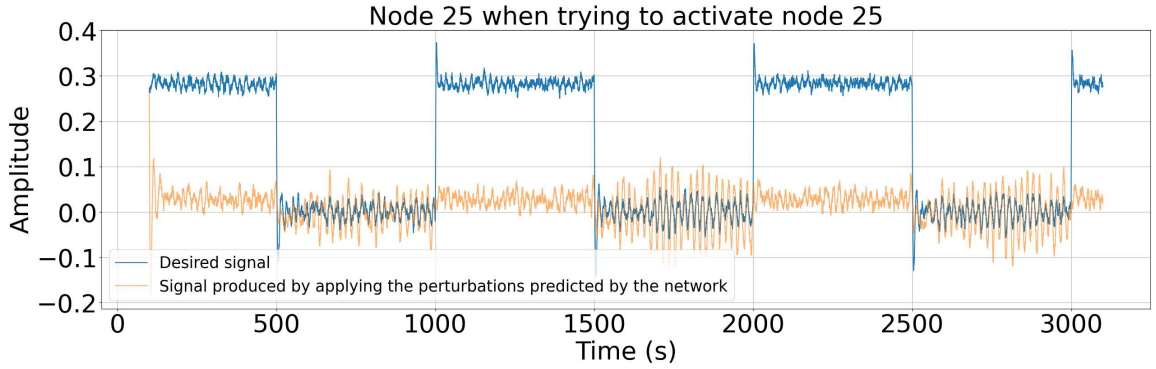


(d) Perturbation to apply to node 23 to activate node 25 as predicted by the ESN trained with random modulations.

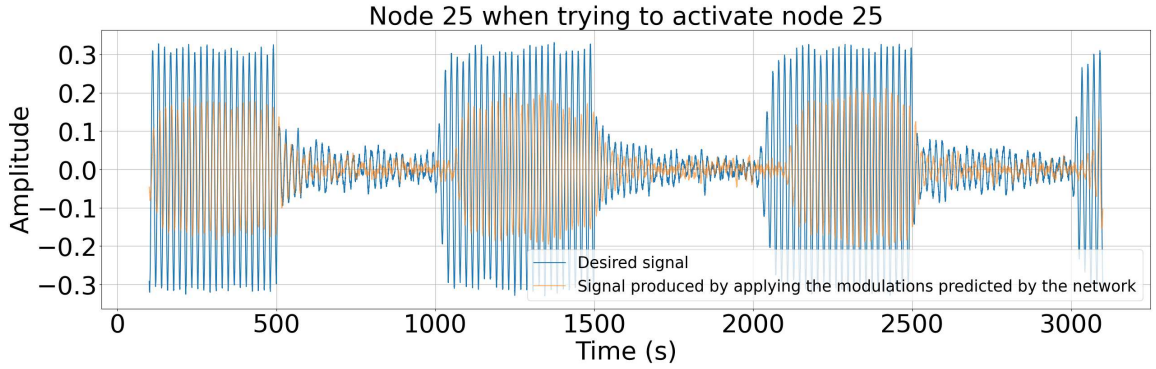
Figure 4.5: Another example for the four cases of the external inputs predicted by the ESN in order to activate node 25.



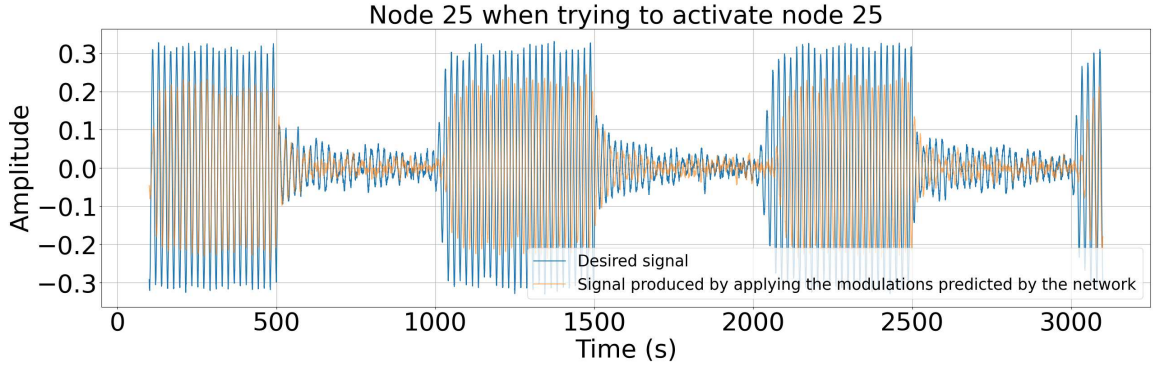
(a) Comparison between the target node signal and the one obtained through the ESN predictions when using square pulse perturbations.



(b) Comparison between the target node signal and the one obtained through the ESN predictions when using randomized perturbations.

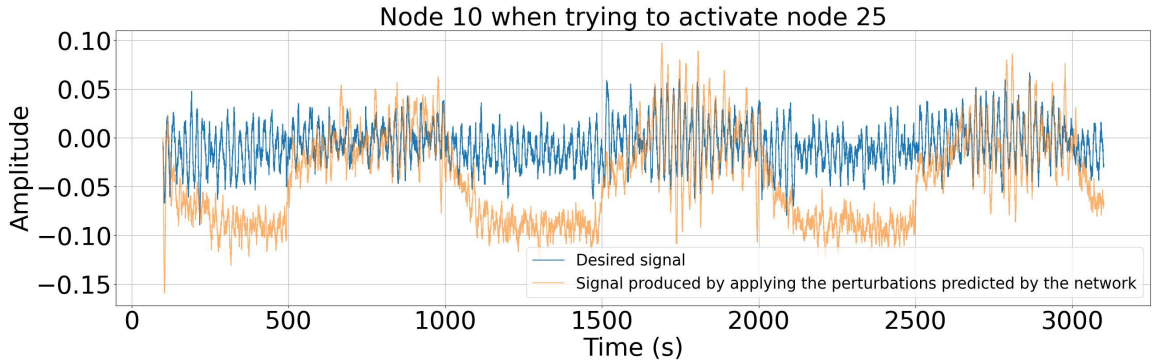


(c) Comparison between the target node signal and the one obtained through the ESN predictions when using square pulse modulations.

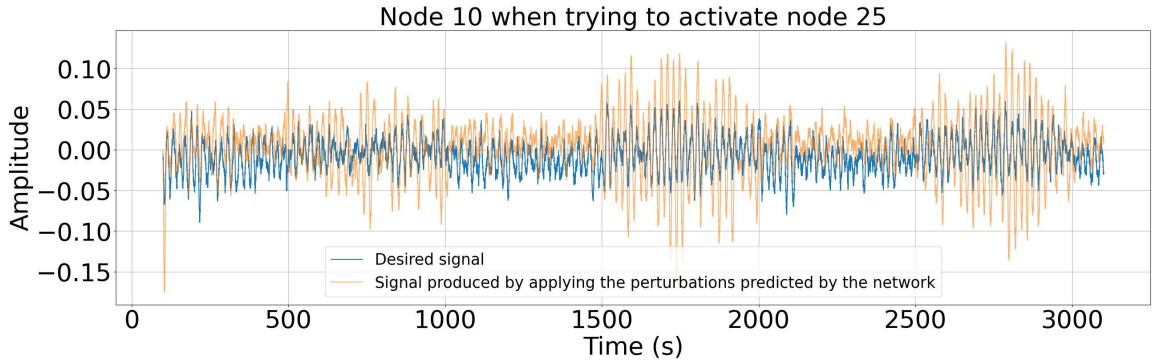


(d) Comparison between the target node signal and the one obtained through the ESN predictions when using randomized modulations.

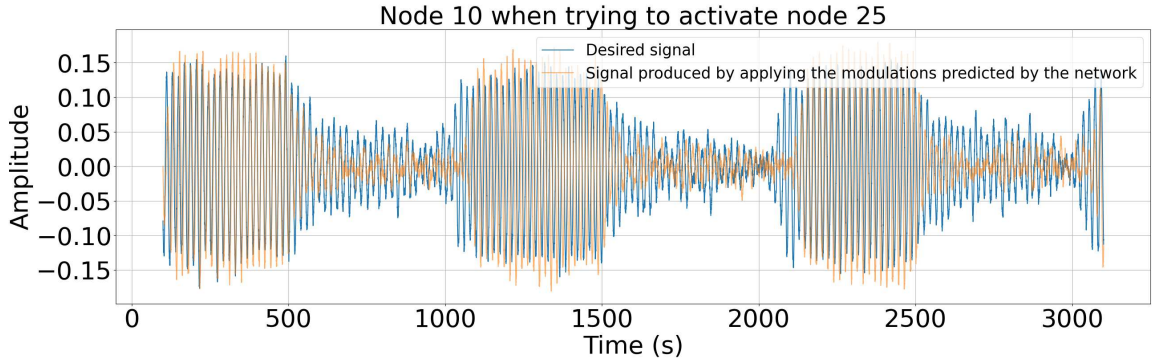
Figure 4.6: Activation of node 25 for the different training methods. Notice the two different types of activation tested, a shift of the signal or a modulation of the oscillation amplitude.



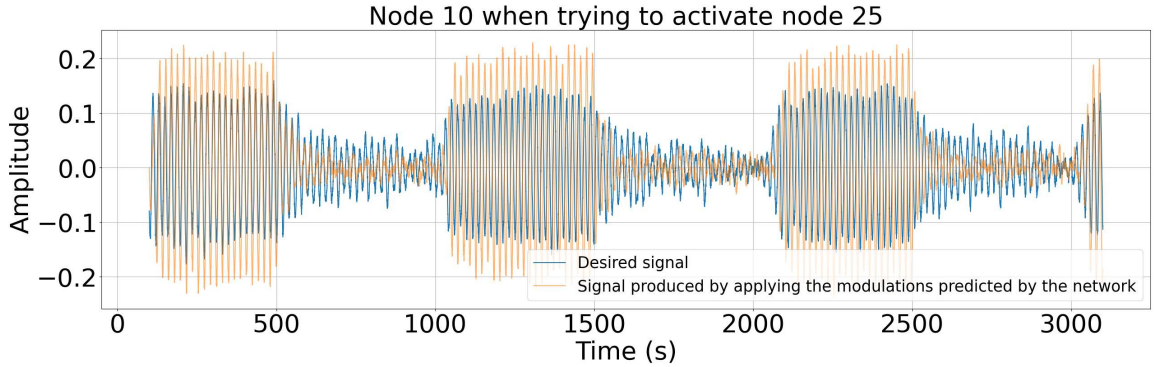
(a) Comparison between a non-target node signal and the one obtained through the ESN predictions when using square pulse perturbations.



(b) Comparison between a non-target node signal and the one obtained through the ESN predictions when using randomized perturbations.

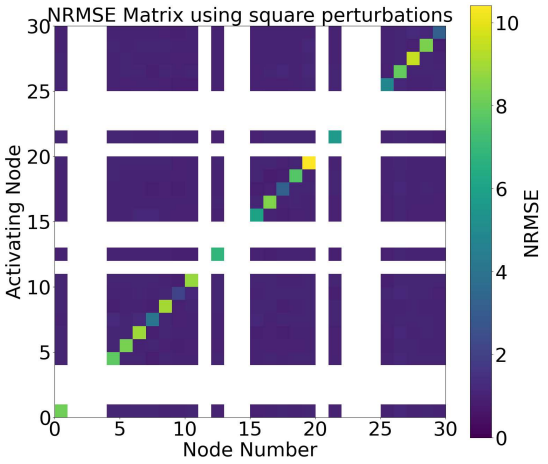


(c) Comparison between a non-target node signal and the one obtained through the ESN predictions when using square pulse modulations.

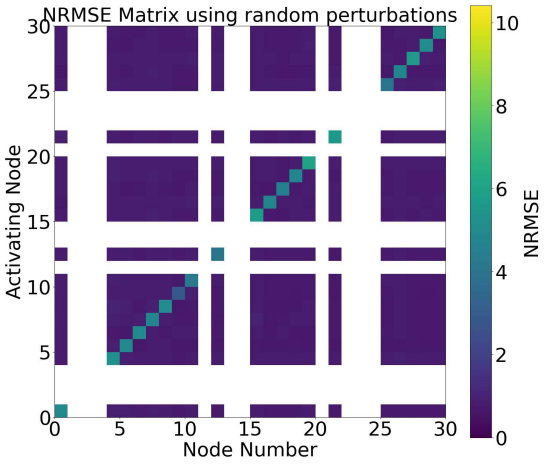


(d) Comparison between a non-target node signal and the one obtained through the ESN predictions when using randomized modulations.

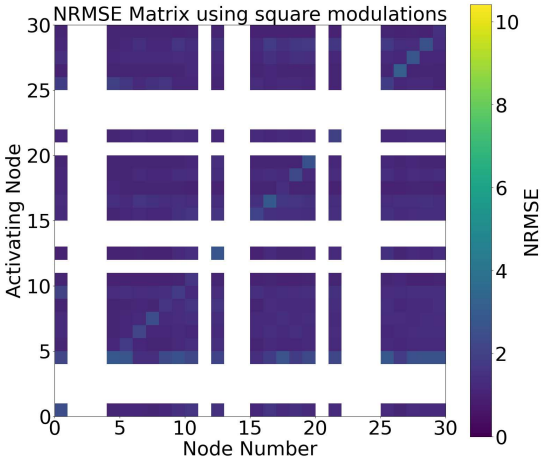
Figure 4.7: Comparing the effectiveness of the ESN predictions for the different training methods in reproducing non-target node signals.



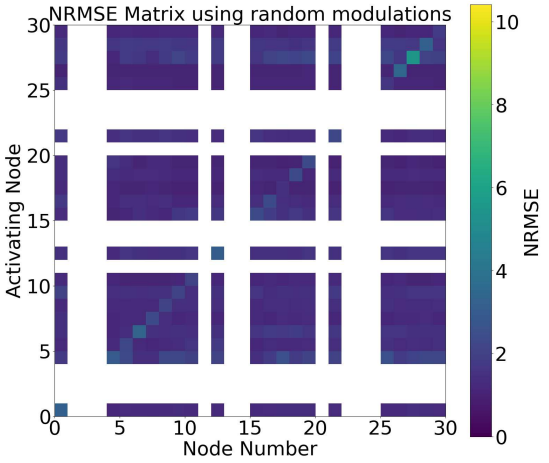
(a) Case of square pulse perturbations.



(b) Case of randomized perturbations.



(c) Case of square pulse modulations.

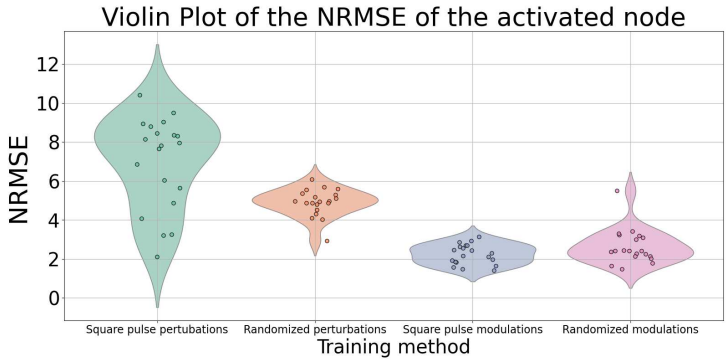


(d) Case of randomized modulations.

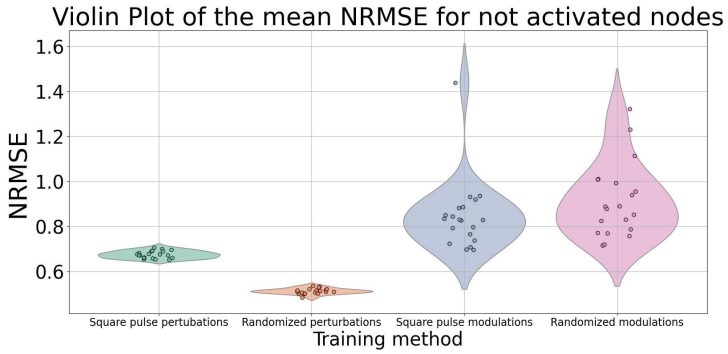
Figure 4.8: Comparison between the NRMSE matrices for the studied training methods.

4.3 Using all the nodes

The previous attempts were made by using 30 of the 100 total nodes given by the parcellation, so that the neural network could perform better. The studied cases allowed us to identify the best model to train the ESN. Figure 4.10a shows the distribution of the diagonal elements of the NRMSE matrix for the four explored training methods, which represent how well was a chosen node activated. It is clear that the two "modulation" training methods led to definitely better results. Figure 4.10b shows, for each row of the NRMSE matrix, the distribution of the mean of the row's entries (except the diagonal element). Remember that each row of the matrix refers to the attempt to activate a chosen node. Thus, the mean value of the row's entries represents a measure of how well the signals of the nodes non-activated nodes were produced. The image shows that the "perturbation" methods for training allowed to better avoid the unwanted activation of the non-target nodes. However, since the objective was the activation of a target node, we opted to use the square pulse modulations as the best training method for the attempt with 100 nodes.



(a) Distribution of the NRMSE of the target node signal for the studied training methods.



(b) Distribution of the mean NRMSE of the non-target node signals for the studied training methods.

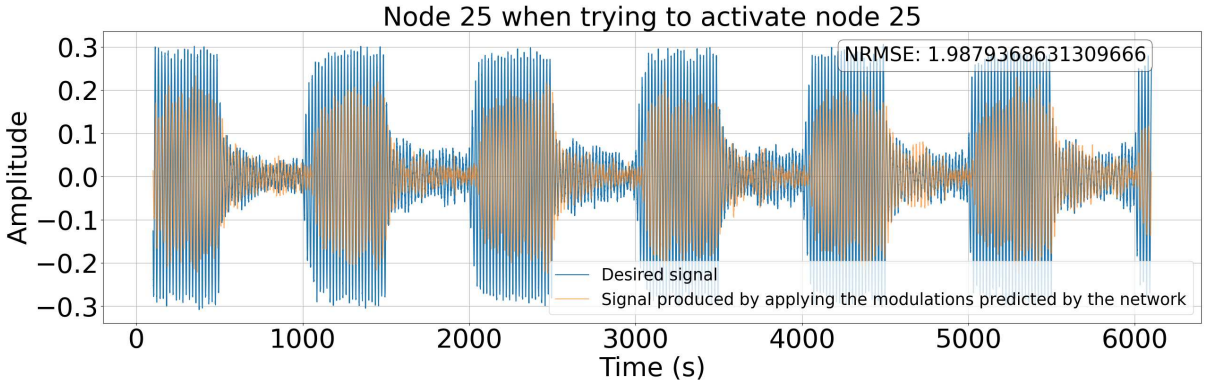
Figure 4.9: Comparison between the different training methods.

We randomly selected 30 out of the total 100 nodes as accessible nodes to be perturbed and followed the procedure outlined in Section 4.2. The ESN parameters were set in the

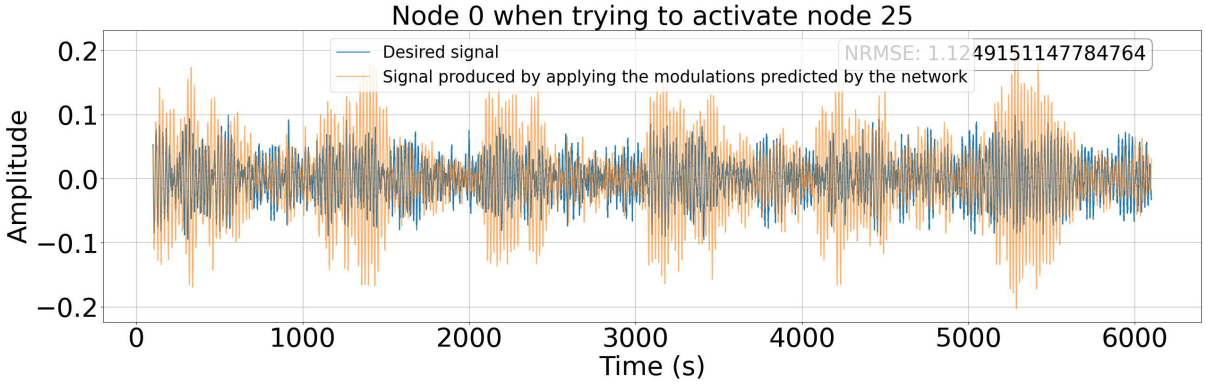
same way. As before, we report Figures 4.11a and 4.11b as examples of the training and validations sessions. Figures 4.11d and 4.11 show the usual comparison between the signal obtained through the ESN predictions and the actual signal for two nodes, the one to activate and a random different node.

As can be seen, the training phase has a much smaller NRMSE than the validation, indicating a bit of overfitting. Since the ESN now processes more data, the Ridge regression leads to a better output, but this improvement doesn't carry over to the validation. In any case, the results were compatible with the attempt with 30 nodes, with a partial activation of the target.

We also tried to verify if a careful choice of the accessible nodes could improve the results. In particular, if the goal was to activate node 25 for example, we selected 30 nodes that had the strongest connections with it. Two results for this example are in Figure 4.10: node 25, which was the target, and node 0, to be compared with the corresponding plots in Figure 4.11. As can be seen from the NRMSE value, there is an improvement in the activation. This was true for all the nodes we tried to activate. In any case, even in this ideal situation, the process is still not satisfactory.



(a) Node 25 activation through modulations acting on nodes selected to be strongly connected with it.

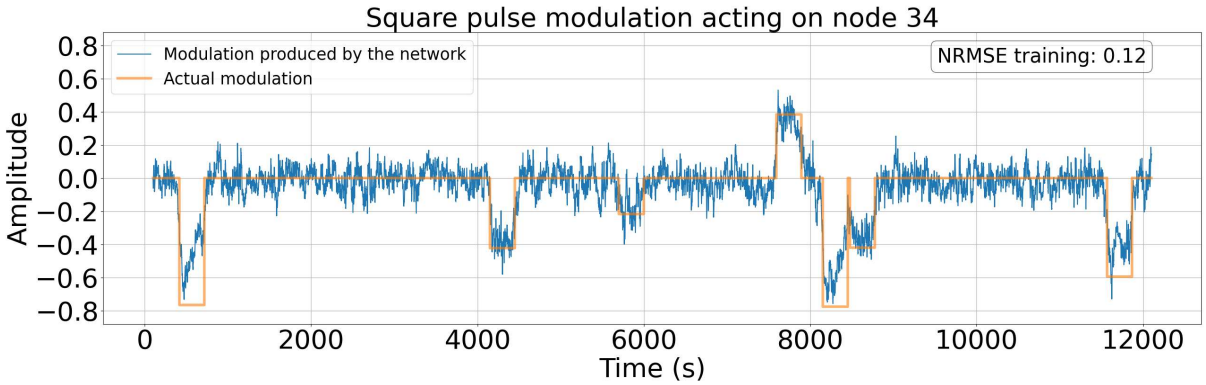


(b) Node 0 when activating 25 with selected accessible nodes.

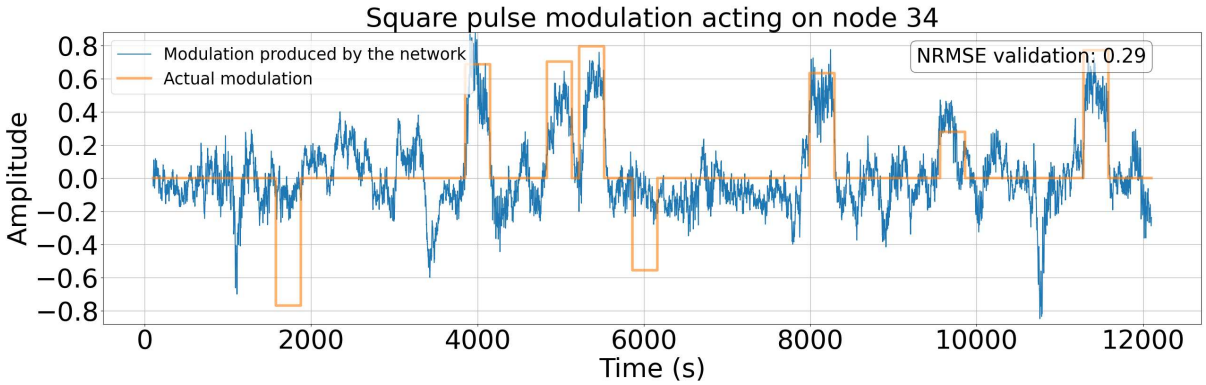
Figure 4.10: Plots for the 100 nodes case, with selected accessible nodes.

4.4 Using target node signal as unique output

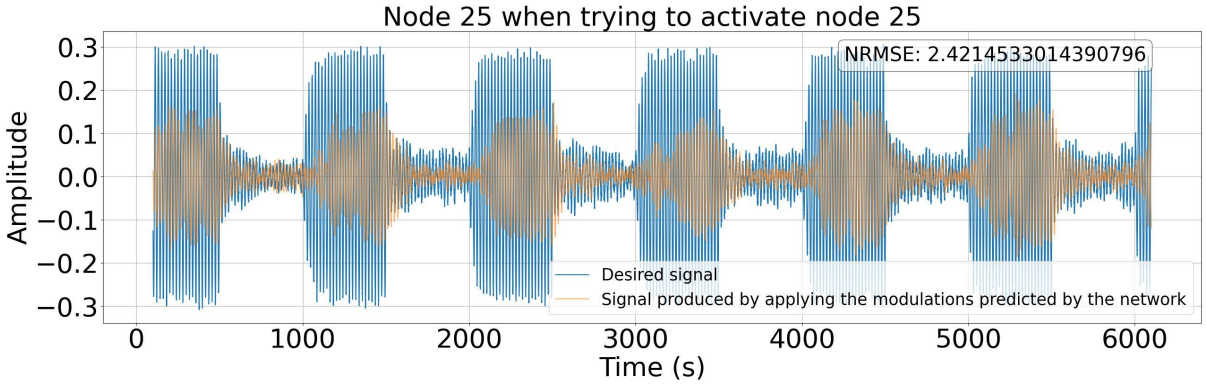
Notice that the procedure we followed makes use of the signals of all the inaccessible nodes as input for the neural network. This choice seems natural if one thinks that a) the neural network would thus be able to process more information in order to reconstruct the external stimulations during training and b) the neural network would be trained to activate the target node while trying to not activate the others.
We wanted to test if, ignoring the requirement b), we could obtain a better performance of the ESN, which could try to activate a target node without caring for the others. Unfortunately, we couldn't obtain significant results, since, despite the many attempts, we couldn't obtain a proper training and validation phase. The only ESN input was, in fact, the signal of the target node, and the ESN processed that signal to reconstruct all the perturbations or modulations acting on the group of accessible nodes, but the results weren't satisfactory, even when selecting as accessible nodes the most connected to the target.



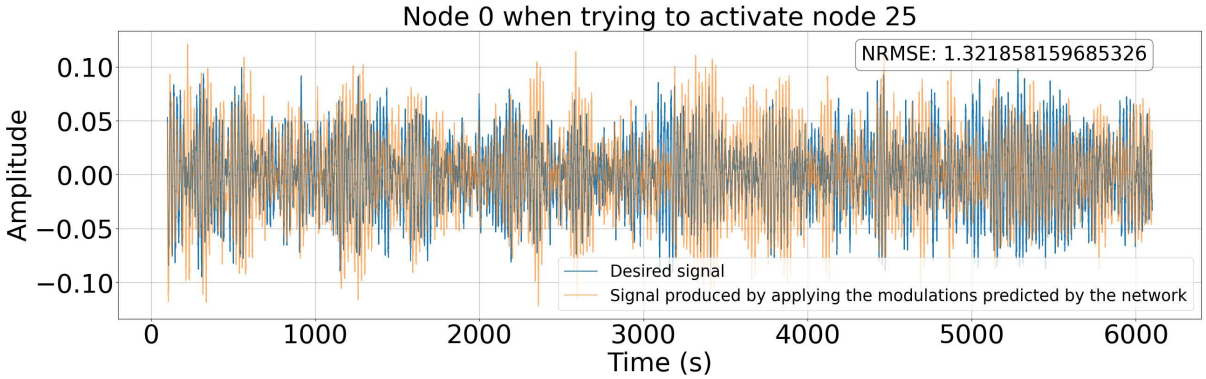
(a) A plot of the training session with 100 nodes.



(b) A plot of the validation session with 100 nodes.



(c) Comparison between the target node signal and the one obtained through the ESN predictions.



(d) Comparison between a non-target node signal and the one obtained through the ESN predictions.

Figure 4.11: Plots for the 100 nodes case, explored by using the square pulse modulations method.

Chapter 5

Conclusions

The present work focused on the attempt to using an Echo State Network to theoretically address the problem of controlling brain dynamics, in particular achieving desired activity changes in specific regions by acting indirectly on distant brain regions. This theoretical problem is of immediate applicability to a clinically relevant goal, that of indirect stimulation of deep-brain regions: medical studies have shown that direct deep-brain stimulation can be beneficial in the treatment of some neurological conditions, like Parkinson's disease[17], but also that the implantation of a neurostimulator is a surgical operation and as such involves risks and complications. For this reason, testing the possibility to indirectly stimulate deep-brain regions through non-invasive stimulation of cortical regions is of great importance.
As we pointed out through this work, an Echo State Network is exactly the kind of tool that can be used to learn an unknown dynamical system and predict the external inputs needed to cause desired variations in that dynamics. Thus, endowed with a whole-brain model to emulate brain dynamics, we proceeded to study if an adequately trained ESN could predict the stimulations to apply to some brain-nodes (assumed to be directly accessible to the stimulations) in order to activate a target node (assumed to be inaccessible). We investigated two types of external inputs, one in the form of perturbations directly exciting or inhibiting the activity of the node signals (probably the kind of stimulation that can be more easily produced in practical applications via electrodes), one in the form of modulations varying the oscillation amplitude of the node signals (which corresponds, physiologically, to increasing/decreasing the excitability of brain regions).
We can summarize the main results of the investigation as follows:

- in general, variations of the oscillation amplitude of one node cause significant variations of the oscillation amplitude of other nodes, unlike the action of a shifting perturbation (see the blue signals in Figure 4.7). This means that, in practical applications, one should be careful when increasing the excitability of a region and be aware that the excitation would propagate to other nodes;

- all the different training methods were quite succesful, with the only exception of the negative modulations. This fact leaves opened the possibility to explore more complex perturbations, allowing more sophisticated training sessions;

- the particular type of stimulation one adds to the brain-dynamics affects the ESN performance. In particular, the best procedure to follow in order to activate a target node with the help of a ESN is by using modulations of the amplitude and by training the neural network with square pulse modulations at random instants (Figure 4.10a;

- this procedure leads to a grater-than-expected activation of the other nodes (Figure 4.10b). This problem could be related to the difficulties (underlined above) the ESN has when learning negative modulations of the amplitude (see the various validation plots reported for the square pulse modulation case, which clearly show how the ESN learns to perfectly reproduce the positive pulses, but poorly the negative ones). We hypothesize that this is due to the particular dynamics on which the ESN operates, that of a set of Landau-Stuart oscillators in the sub-critical region of a Hopf bifurcation. In this regime, when even a moderately negative modulation is applied, the oscillation can get completely damped rather than continuing with reduced amplitude, which makes it difficult for the ESN to resolve weakly and strongly negative modulations.

- the performance of the ESN declines with the increase in the number of nodes. Nonetheless the results for 100 nodes are comparable with those for 30 nodes. In any case, we showed that, when using larger numbers of nodes, the results can be improved by a careful selection of the accessible nodes.

In the end, the approach we followed had quite limited success. The network effectively learns the external inputs perturbing the brain-dynamics during the training session, but is then unable to produce the best suited stimulations to activate a target node.
We suggest to address this problem in further studies, using larger reservoirs where possible and analyzing in detail which brain nodes are more easily controllable. It could also be useful trying many different types of perturbations. Notice however that, despite the different approaches we followed, each methods led the ESN to predict square wave stimulations to be applied to the accesible nodes (see Figures 4.4), i.e. external inputs of the form of the actual stimulation with which the target node was excited. This could indicate an inherent problem in the approach, that could not be solved just by extending the space of perturbations. It is also recommended to address the problem thorugh the classical approach to control, linearizing the oscillator model, in order to have the chance to compare the results.
Finally, other models of brain-dynamics could be studied with the procedure we followed,

comparing the otucomes, in order to exclude the possibility of a problem with the equations simulating brain signals.

Bibliography

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[2] Alim Louis Benabid, Stephan Chabardes, John Mitrofanis, and Pierre Pollak. Deep brain stimulation of the subthalamic nucleus for the treatment of parkinson's disease. *The Lancet Neurology*, 8(1):67–81, 2009.

[3] Daniel Canaday, Andrew Pomerance, and Daniel J Gauthier. Model-free control of dynamical systems with deep reservoir computing. *Journal of Physics: Complexity*, 2(3):035025, 09 2021.

[4] Gustavo Deco, Josephine Cruzat, Joana Cabral, Enzo Tagliazucchi, Helmut Laufs, Nikos K. Logothetis, and Morten L. Kringelbach. Awakening: Predicting external stimulation to force transitions between different brain states. *Proceedings of the National Academy of Sciences of the United States of America*, 116(36):18088–18097, 09 2019. Epub 2019 Aug 19.

[5] Gustavo Deco, Morten L. Kringelbach, Viktor K. Jirsa, and Petra Ritter. The dynamics of resting fluctuations in the brain: metastability and its dynamical cortical core. *Scientific Reports*, 7(1):3095, 2017.

[6] Frank Freyer, James A. Roberts, Robert Becker, Peter A. Robinson, Petra Ritter, and Michael Breakspear. Biophysical mechanisms of multistability in resting-state cortical rhythms. *Journal of Neuroscience*, 31(17):6353–6361, 2011.

[7] Frank Freyer, James A. Roberts, Petra Ritter, and Michael Breakspear. A canonical model of multistability and scale-invariance in biological systems. *PLOS Computational Biology*, 8(8):1–15, 08 2012.

[8] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. The minimal preprocessing pipelines for the human connectome project. *Neuroimage*, 80:105–124, 2013.

[9] Allen Hart, James Hook, and Jonathan Dawes. Embedding and approximation theorems for echo state networks. *Neural Networks*, 128:234–247, August 2020.

[10] Herbert Jaeger. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148, 01 2001.

[11] Timothy P Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current Opinion in Neurobiology*, 55:82–89, 2019. Machine Learning, Big Data, and Neuroscience.

[12] Lennart Ljung and Torkel Glad. Modeling of dynamic systems. 1994.

[13] M Lukovsevivcius. A practical guide to applying echo state networks. *Neural Networks: Tricks of the Trade, Reloaded*, 01 2012.

[14] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[15] Simone Rossi, Mark Hallett, Paolo M. Rossini, and Alvaro Pascual-Leone. Safety, ethical considerations, and application guidelines for the use of transcranial magnetic stimulation in clinical practice and research. *Clinical Neurophysiology*, 120(12):2008–2039, 2009.

[16] Alexander Schaefer, Ru Kong, Evan M Gordon, Timothy O Laumann, Xi-Nian Zuo, Avram J Holmes, Simon B Eickhoff, and BT Thomas Yeo. Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri. *Cerebral cortex*, 28(9):3095–3114, 2018.

[17] Evelyn Tang and Danielle S Bassett. Colloquium: Control of dynamics in brain networks. *Reviews of modern physics*, 90(3):031003, 2018.

[18] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.