UNIVERSITY DE SÃO PAULO
FACULTY OF PHILOSOPHY, SCIENCES AND LETTERS OF RIBEIRÃO PRETO
DEPARTMENT OF COMPUTING AND MATHEMATICS

LAERCIO DE OLIVEIRA JUNIOR

Clustered Echo State Networks for Signal Denoising and Frequency Filtering

Echo State Networks com Clusters na Remoção de Ruídos e Filtro de Frequências

RIBEIRÃO PRETO – SP

2020

LAERCIO DE OLIVEIRA JUNIOR

Clustered Echo State Networks for Signal Denoising and Frequency Filtering

Echo State Networks com Clusters na Remoção de Ruídos e Filtro de Frequências

Corrected Version

The original version is found at FFCLRP/USP.

Dissertation presented to Faculty of Philosophy, Sciences and Letters of Ribeirão Preto (FFCLRP) from the University de São Paulo (USP), as part of the requirements to hold the Master of Science degree.

Field of Study: Applied Computing.

Supervisor: Zhao Liang

Ribeirão Preto – SP

2020

Laercio de Oliveira Junior
Clustered Echo State Networks for Signal Denoising and Frequency Filtering. Ribeirão Preto – SP, 2020.
60p. : il.; 30 cm.

Dissertação apresentada à Faculdade de Filosofia, Ciências e Letras
de Ribeirão Preto da USP, como parte das exigências para
a obtenção do título de Mestre em Ciências,
Área: Computação Aplicada.

Supervisor: Zhao Liang

1. Echo State Networks. 2. Clustered Networks. 3. Complex networks.

Laercio de Oliveira Junior

Clustered Echo State Networks for Signal Denoising and Frequency Filtering

Modelo canônico de trabalho monográfico acadêmico em conformidade com as normas ABNT.

Trabalho aprovado. Ribeirão Preto – SP, _____ de _____ de 2020:

Supervisor:
Orientador

Professor
Convidado 1

Professor
Convidado 2

Professor
Convidado 3

Ribeirão Preto – SP

2020

Acknowledgements

First of all, I would like to thank Professor Zhao for giving me the opportunity to work with him, and for trusting this project on me. The professor always supported me all the time even though I was working in a full-time job outside the university. Many thanks to Florian Stelzer, from the Humboldt University, that helped me a lot to understand the ESNs, and helped me with the paper submissions. Another huge thanks to my wife Taís, who helped me with the thesis and the presentation, and to my mother Vilma for the moral support during this journey. I also would like to thank the University of São Paulo, for allowing me to work with so talented people.

Resumo

Esta dissetação tem como objetivo estudar um tipo de Rede Neural Artificial (RNA), conhecido como *Reservoir Computing*, mais especificamente as *Echo State Networks* (ESNs). ESNs são redes neurais recorrentes (RNNs), que fazem o mapeamento de entrada-saída através de projeções não-lineares de alta dimensão, chamada de *reservoir*. No modelo clássico da ESN, a matriz das conexões internas do reservatório é usualmente uma rede aleatória Erdös-Rényi. Estudos recentes investigaram o uso de redes com *clusters* dentro do reservatório de uma ESN, as Clustered ESNs (*CESNs*), sendo que essa nova rede do reservatório apresenta uma topologia com *clusters*. Ambos tipos de ESNs foram aplicadas ao problema de predição de séries temporais. Neste trabalho, são propostas uma ESN com redes Barabási-Albert em cada cluster (*Barabási-Albert CESN*), e uma *deep ESN* em que cada camada dessa rede contém uma rede com *clusters* (*Deep CESNs*). Além disso, foi proposto a aplicação de ESNs e suas extensões em dois novos problemas: o filtro de frequências e a remoção de ruídos de séries temporais. Uma comparação foi feita entre o modelo clássico da ESN e suas extensões. Experimentos númericos mostram que os modelos propostos de ESNs (*Barabási-Albert CESN* and *Deep CESNs*) superam o desempenho do modelo clássico da ESN, indicando que a organização dos reservatórios em *clusters* ou em camadas melhoram o desempenho da rede.

Palavras-chave: Redes Neurais Artificiais. *Echo State Networks*. Redes com *Clusters*. *Reservoir Computing*. Redes Complexas.

Abstract

This dissertation aims to study a type of Artificial Neural Networks (ANNs), known as *Reservoir Computing*, specifically, the Echo State Networks (ESNs). ESNs are Recurrent Neural Networks (RNNs), which make input-output mapping through a high dimensional nonlinear projection, called *reservoir*. In a classic ESN, the internal connection matrix of the reservoir usually is formed by an Erdös-Rényi random graph. Recent studies have also investigated Clustered ESNs (*CESNs*), which replaces the random network inside the reservoir by a clustered network. Both types of ESNs have been applied to time series prediction problems. In this work, an ESN with a clustered Barabási-Albert network (*Barabási-Albert CESN*), and a deep ESN with clustered reservoir layers (*Deep CESNs*) are designed. Moreover, we propose to apply ESNs in two new different tasks: the frequency filtering problem and the noise filtering problem of time series. We also compare the performance of the classical ESN and its various extensions in these two tasks. Numerical results show that the proposed ESNs (*Barabási-Albert CESN* and *Deep CESNs*) outperform the classical ESN, indicating that the organization of reservoirs in clustered or layered networks can improve the learning performance of ESNs.

Keywords: Artificial Neural Networks, Echo State Networks. Clustered Networks. Reservoir Computing. Complex Networks.

List of figures

List of tables

List of abbreviations and acronyms

ANN	Artificial Neural Network
CESN	Clustered Echo State Network
ECG	Electrocardiogram
ESN	Echo State Network
NRMSE	Normalized Root Mean Square Error
RNN	Recurrent Neural Network

Summary

1

Introduction

1.1 Context

The brain is considered the main organ responsible for information processing in humans and animals with a central nervous system. It can be seen as a signal processor that exploits massive parallelism across billions of processing elements called neurons. These nerve cells connect through chemical substances (neurotransmitters) forming neural synapses, which allow the continuity of the electrical and/or the chemical signal that travels through the cell. Synapses give the brain its capacity to store, process, recover, and, mainly, to generalize information (learning) (TRAPPENBERG, 2002).

Inspired by the structure of the human brain, Artificial Neural Networks (ANNs) are a simplistic computational approach that exploits the processing and memory mechanisms present in biological neural systems. Even though the ANN is far from the human brain comparing the processing power and the complexity, it is a parallel and distributed processing system composed of processing units (artificial neurons), which are regularized by certain non-linear mathematical functions, called activation function. Such units are arranged in one or more layers interconnected by a large number of connections (synapses). In most models of ANNs, connections are associated with weights, which are used to store the knowledge acquired (HAYKIN, 1998). Like the biological brain, ANNs are capable of learning through examples, generalizing the learned information, thereby becoming an important tool in the field of data mining and machine learning.

Some ANN models are more similar to the biological neural system in some features such as an associative memory (search for information based on part of it), and it contains characteristics of a dynamic system (SKARDA; FREEMAN, 1987), the recurrent neural networks, such as the Hopfield model (HAYKIN, 1998) and Reservoir Computing models (Echo State Network, Liquid-State Machines) (JAEGER, 2001; CHATZIS; DEMIRIS, 2011; HAZAN; MANEVIT, 2012; BOCCATO; ZUBEN, 2014; PONGHIRAN; SRINIVASAN;

ROY, 2019). However, this and other "classic" ANN models are still a long way off to offer the high learning and processing capabilities observed in the human brain, mainly because they have an oversimplified topological structure when compared to a biological neural network, which exhibits characteristics of a complex network (NEWMAN, 2004; NEWMAN, 2010; LATORA; NICOSIA; RUSSO, 2017). Complex networks is the field that studies graphs, such that the number of vertices and edges are large enough which cannot be studied by the classical graph theory (BOCCALETTI et al., 2006; ZAMORA-LÓPEZ; BRASSELET, 2019). For example, the Hopfield model has a complete network (fully connected network) while the Echo State Network (ESN) has a random network reservoir.

One of the important features is that the brain has specialized areas to deal with specific events (localized functional groups). Different activities are treated in different portions of the cerebral cortex, either low or high topological organization. In other words, neurons are organized into clusters in the brain (BERRY; TKACIK, 2020; CHEN et al., 2018; MARTENS et al., 2017). This is the main inspiration for the present study.

Since the human (animal) brain is a high-dimensional complex dynamical system, therefore, enormous effort has been paid to design the learning process of ANNs as nonlinear dynamical systems. One of the successful approaches is the reservoir computing (JAEGER, 2001), where the input-output mapping is learnt through a high-dimensional recurrent projection, called *reservoir*. One type of *reservoir* computing, called Echo State Network (ESN) triggered much attention. ESNs has been successfully applied to chaotic signal prediction (LU et al., 2017; PATHAK et al., 2017), short term stock price prediction (LIN; YANG; SONG, 2009), stock data mining (LIN; YANG; SONG, 2008), among other studies (LACY; SMITH; LONES, 2018; SCHUBERT; GROS, 2020; Zheng et al., 2020). The interesting point of the ESN is the integration of a dynamic system, machine learning, and complex network studies. In classic ESN, denoted *Random ESN* here, the structure of the *reservoir* is usually consisting of a large number of randomly connected nodes (neurons) forming an Erdös-Rényi network (ERDÖS; RENYI, 1961).

As stated before, complex networks are large scale graph with nontrivial connection patterns, which provides a powerful tool for modeling real complex systems by unifying spatial, topological, functional, and evolutionary relationship among the constituted elements (ALBERT; BARABÁSI, 2002; COHEN; HAVLIN, 2010; BREDE, 2012). One of the salient features of complex networks is the presence of communities representing the clustered structure. A community is a group of nodes that are densely connected internally, while the connections between nodes from different communities are relatively sparse. Such a clustered structure has long been found in the human (animal) brain. Data on both anatomical and functional connectome has shown the small-world structure with highly clustered modules at different scales (AKIKI; ABDALLAH, 2019; GLEISER; SPOORMAKER, 2010; HAGMANN et al., 2008). These communities are known to

represent subsystems of neurophysiological functions, e.g., the visual cortex. Moreover, several psychiatric disorders have shown selective disruption in particular brain communities (AKIKI; ABDALLAH, 2019).

Inspired by the neurophysiological findings, there is a growing interest in modeling the reservoir as a complex network with community structure and small-world properties, i.e., the average distance between nodes is short, such as in (KAWAI; PARK; ASADA, 2019) for example. In fact, various network topologies have been used inside the reservoir in past works, some of them have used a clustered network inside the reservoir, such as in (DENG; ZHANG, 2007) to approximate the model to the biological brain, in (LI et al., 2015) that creates a clustered network based on the data, and in (YU; MIAO; JIA, 2011) to reflect real learning mechanisms in the network.

Going towards the same direction, the *Deep ESN* proposed by (GALLICCHIO; MICHELI, 2016) gives a model to split the reservoir into smaller networks forming interconnected layers, analog to the different scales of brain structure. All these works show that a better performance can be achieved using a clustered network rather than a random network (DETTORI et al., 2020; CARMICHAEL; SYED; KUDITHIPUDI, 2019). Therefore, in this work, we will explore the flexibility of the reservoir by replacing the random network for a clustered network since previous works show that an improvement in the performance can be achieved with these non-random network topologies.

Moreover, the signal denoising problem, which consists of removing noise, perturbations and distortions from a signal, is widely studied in the literature. Some of the techniques are the wavelet-based techniques (JAISWAL; UPADHYAY; SOMKUWAR, 2014), and the nonlinear filtering (LEE; KASSAM, 1985), Wiener filter (CHEN et al., 2006; VASEGHI, 2001) for example. This task is particularly interesting because the output signal might be difficult to recover knowing only the input signal without any information about the type of noise.

1.2 Objective

This work aims the study of ANNs starting from the classical ESN model, and then restructuring the neurons' connections in the reservoir to create a topological structure of a complex network that contains communities. It is intended to investigate, mainly the relationship between the various topological structure of ESNs, and their performance in signal separation and noise reduction problems.

In this work, two new types of clustered network will be introduced to replace the random network in the reservoir, a scale-free clustered network, where each community is a Barabási-Albert scale-free network (BARABÁSI; ALBERT, 1999), and a random

clustered network, where each community is a random network (ERDÖS; RENYI, 1961). Past works have used different clustered networks in the reservoir instead of the random network, and they have achieved good results. The idea of our model is to create clustered networks based on complex networks models with controlled features, and check whether they can solve problems better than the classic ESN.

Here, these types of clustered ESNs are denoted by *Barabási-Albert CESN* and *Erdös-Rényi CESN*. It is well known that both random and scale-free networks possess small-world property. In this way, the reservoirs studied here contain both modular structures and small-world features. Furthermore, a Deep ESN with clustered layers, called *Deep CESN*, is also introduced in this work, where each layer of this deep network is a clustered network. With these proposed changes to the network topology, it is expected a performance improvement over the ESN, as it has occurred in previous work with other network variations.

Traditionally, ESNs have been designed for solving time series prediction problems. In this work, we propose to apply the ESN and its extensions, including the clustered ESN, deep ESN, and deep ESN with clustered layers, in a novel approach, which is filtering the noise from signals (VASEGHI, 2001). For this purpose, signals with various types of noises have been considered, and the performance of the classic and modified ESNs has been compared to the Wiener filters (CHEN et al., 2006; VASEGHI, 2001). Numerical results show that the clustered ESNs, particularly *Barabási-Albert CESN* and *Deep ESN*, present much superior performance than the classic *Random ESNs* and Wiener filters.

Our objective is to use the CESNs to solve the frequency filtering, and mainly the noise filtering task. Even though there are classical approaches that solve these tasks, the idea is to introduce the CESNs as a flexible approach. Another objective here is to propose new clustered network models to replace the random network in the reservoir. Worth mention this project only aims the study of artificial neural networks, specifically, the study of ESNs, where the reservoirs are organized in a way that is inspired by the modular feature of biological neural networks as studied in (LI et al., 2015; KAWAI; PARK; ASADA, 2019).

1.3 Document Organization

The following chapters of this document is organized as follows: Chapter 2 presents relevant concepts and models, specifically complex network models, Echo State Networks, and Wiener filter for noise reduction. Chapter 3 presents the proposed ESNs. Chapter 4 shows the obtained experimental results of various ESNs on signal separation and noise reduction problems. Finally, Chapter 5 presents conclusions and future works.

2

Literature Review

2.1 Complex Networks

As the name suggests, complex networks refers to a graph that consists of a set of vertexes and a set of edges, where the organization and the topology are not trivial. These graphs can be used to represent and study several organizations, such as the internet, social relations between individuals, biological neural networks, metabolic chains, food chains, among others (COSTA F. A. RODRIGUES; BOAS, 2006; NEWMAN; BARABÁSI; WATTS, 2006; NEWMAN, 2010; LATORA; NICOSIA; RUSSO, 2017).

The main property of such networks are (STROGATZ, 2001):

- Structural complexity – hard to visualize the network;
- The Evolution - constant changes in the network structure due to inclusion and removal of vertexes and connections;
- Diversity of connections – The connections between the vertexes can have multiple variations in their characteristics, such as capacity, the length, the width, and the direction;
- Complex dynamic – beyond the structure, what affects in large scale in the state of the network is its dynamic, which is the information flow, communication flaws, epidemic, synchronization, and correlation between vertexes, and others.

The way the edges are correlated defines the type of the network. There are three main categories of complex networks, which are:

- **Random networks**: It was the first type of complex network mathematically studied. Proposed in (ERDÖS; RENYI, 1961), this model considers that edges are

added randomly given a fixed number of vertexes N. The mean degree of each vertex is given by the equation:

$$\langle k \rangle = p(N-1) \tag{2.1}$$

where (p) is the probability of a given vertex to connect with any other vertex in the network, following a Poisson distribution. In this way, all the vertex have approximately the same number of connections, as well as the same chance to receive new connections. The Figure 1.a shows the structure of a possible random network.

- **Scale-free networks**: The main feature of this type of network is the more connections a node has, the more likely this node is to receive new connections. Proposed by (BARABÁSI; ALBERT, 1999), the authors observed that such property is present in several real-world networks, such as the internet, metabolic networks, biological neural networks, article citation networks, and others. Therefore, the authors have shown the degree distribution of several of these networks follows a power law, which is given by the following equation:

$$P(k) \sim k^{-\lambda} \tag{2.2}$$

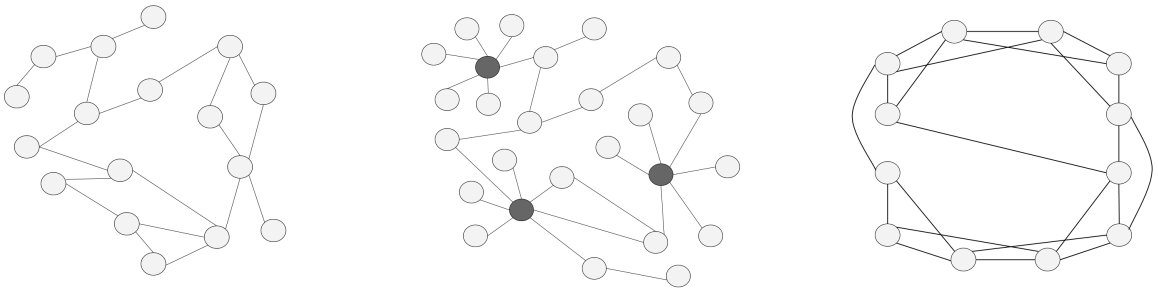
where k is the number of connections of a given vertex and λ is the scale exponent. This kind of network will have the majority of the vertexes with a low number of connections, and just a few of them with a high number of connections (these vertexes are known as *hubs*), as can be observed on the Figure 1.b.

- **Small-world networks**: In (STROGATZ, 1998), the authors proposed a model where the vertexes create connections with nearest vertexes with higher probability, as observed in social networks, for example. The majority of the vertexes create connections with other vertexes throughout the shortest path (the minimum number of edges between two nodes). The average length of the shortest path l between all the pairs of vertexes in a graph with undirected connections is given by the following equation:

$$l = \frac{1}{\frac{N(N+1)}{2}} \times \sum_{i \geq j} d(i,j) \tag{2.3}$$

where $d_{i,j}$ is the geodesic distance between the vertexes **i** and **j**. An example of such network can be seen on the Figure 1.c.

Note that complex networks can have a clustered topology, where similar nodes are connected and there are fewer connections between nodes of different clusters, as can be observed on the *World Wide Web*, in article citation networks, in traffic networks, social networks, and others (FORTUNATO, 2010).



(a) Random network (b) Scale-free network (c) Small-world network

Figure 1 – Example of the different types of complex networks.

The Figure 2 shows an example of a complex clustered network. The *clustering* (also known as community detection) is currently one of the biggest challenges in machine learning, due to the high complexity of traditional algorithms when handling large amounts of vertexes and edges. This fact makes this research area relevant and promising, as indicated by the works (FORTUNATO, 2010; NEWMAN, 2004; NEWMAN; GIRVAN, 2004; QUILES et al., 2008; SILVA; ZHAO, 2012; SILVA; ZHAO, 2016), and others.

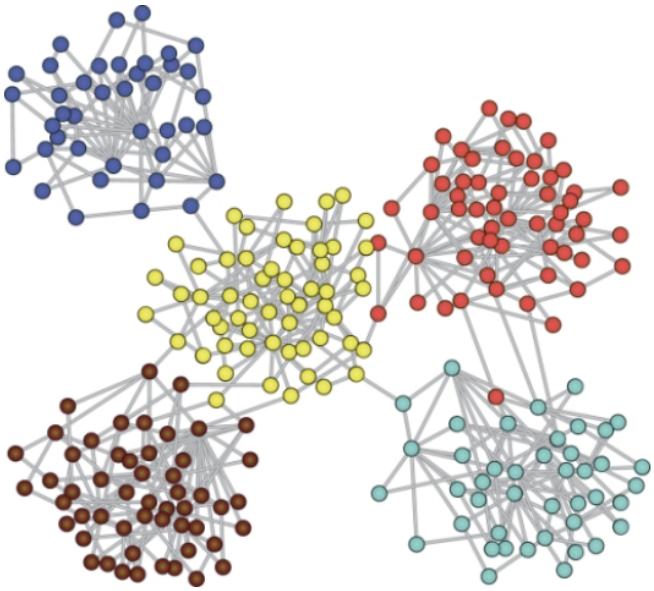


Figure 2 – Illustration of a clustered network, where the vertices with the same color belong to the same cluster.

Therefore, the aim is to investigate how does the ESN behaves with the neurons in the reservoir organized in clusters, where these clusters form a complex network, as it will be discussed in the next sections.

2.2 Echo State Networks

In this section, a quick review of various ESNs are presented, which is the main subject studied in this work. These are the classic ESN with a random network as reservoir (*Random ESN*) and the deep ESN with a random reservoir divided into several layers (*Deep ESN*).

Echo State Networks (ESNs) provide an architecture and a mechanism for recurrent neural networks for supervised learning. The main idea is to introduce a set of neurons between the input layer and the output layer, called a reservoir. Such a reservoir is composed of a network of neurons recurrently and randomly connected. Further details of this network will be given in the section bellow.

2.2.1 Random ESN

The ESN was proposed by (JAEGER, 2001) introducing an algorithm to build recurrent neural networks to recognize patterns. The main idea of the algorithm is to change only the weights of the output layer in the learning process. This is what makes the learning process of the ESN fast.

The general idea of the ESN is to provide an architecture of a recurrent neural network to perform supervised learning. There is an input layer ($\mathbf{u}(t)$), an output layer ($\mathbf{x}(t)$), and a hidden layer between them called reservoir ($\mathbf{r}(t)$). The reservoir is a dynamical system that forms a high-dimensional mapping to avoid the lineally non-separated problem. The interesting point is that, with the reservoir projection, the learning process only occurs at the output layer. Figure 3 illustrates a *Random ESN* with an input layer, a random network in the reservoir, and the output layer. The training stage of these networks is very fast compared to the training of other types of ANNs. The algorithm to build the classic ESN follows the steps:

- A random network is created to be used as a reservoir. Then, connections are created between all nodes in the input layer to all nodes inside the reservoir. And then, connections are created between all nodes in the reservoir to all nodes in the output layer. Initially, all connection weights are random real numbers between -1 and 1. In general the matrix is an sparse matrix to ensure the *echo* property of the network, which is controlled by a ESN parameter called *spectral radius* (ρ).

- Iterations are made in the states of the reservoir neurons without changing weights of reservoir connections. Each neuron is a dynamic system, and its state evolves throughout time.

- The training phase is done by changing the connections weight between the reservoir and the output layer, using the linear regression technique to minimize the error between the network output and the expected output (expected outputs are provided as part of the training data).

Going further in the algorithm, mathematically, the reservoir state $x(t)$ changes over time-based on the following equation:

$$\mathbf{x}(t+1) = (1-\alpha)\mathbf{x}(t) + \alpha f(A\mathbf{x}(t) + W^{in}\mathbf{u}(t) + \gamma\mathbf{1}), \tag{2.4}$$

where $0 < \alpha \leq 1$ is a leakage rate, $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix of the reservoir, $\mathbb{W}^{in} \in \mathbb{R}^{M_{\text{in}}\times N}$ contains input connections weights, $\mathbf{u}(t) \in \mathbb{R}^{M_{\text{in}}}$ is the input vector, and $\gamma\mathbf{1}$ is a bias vector, where $\gamma \in \mathbb{R}$ is the bias and $\mathbf{1} \in \mathbb{R}^N$ denotes a vector containing ones. Since all the input data used in this work only consists of one-dimensional data, then throughout this work we use $M_{\text{in}} = 1$.

The output of an ESN is given by the following equation:

$$\mathbf{s}(t) = W^{out}\mathbf{x}(t) + \mathbf{c} \tag{2.5}$$

The key fact in the ESNs is that only the connections with the output layer are trained. In this case, the aim is to find a good estimator ($\hat{\mathbf{s}}(\mathbf{t})$) by configuring W^{out} to fit the input data. The ESN training algorithm was already described in (LU et al., 2017). For the self-contained purpose, it is shortly described below.

Given an input signal U with the entries: $\mathbf{u}(1), \mathbf{u}(2), \ldots, \mathbf{u}(t_{max})$, the reservoir state will change according to the Equation 2.4. The reservoir has an initial state, $\mathbf{x}(0)$, and for each entry $\mathbf{u}(t)$, the reservoir will generate the state $\mathbf{x}(t)$ based on the input values. Linear regression can be used to find a good estimator ($\hat{\mathbf{s}}(t)$), which aims to approximate the expected output to the output given by the network. The first step to find this estimator is calculating the mean $\bar{\mathbf{x}}$ of the states in the reservoir, and the mean $\bar{\mathbf{s}}$ of the expected outputs, which are given by the equations:

$$\bar{x} = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \mathbf{x}(t), \tag{2.6}$$

$$\bar{s} = \frac{1}{t_{max}} \sum_{t=1}^{t_{max}} \mathbf{s}(t), \tag{2.7}$$

where t_{max} is the number of training steps.

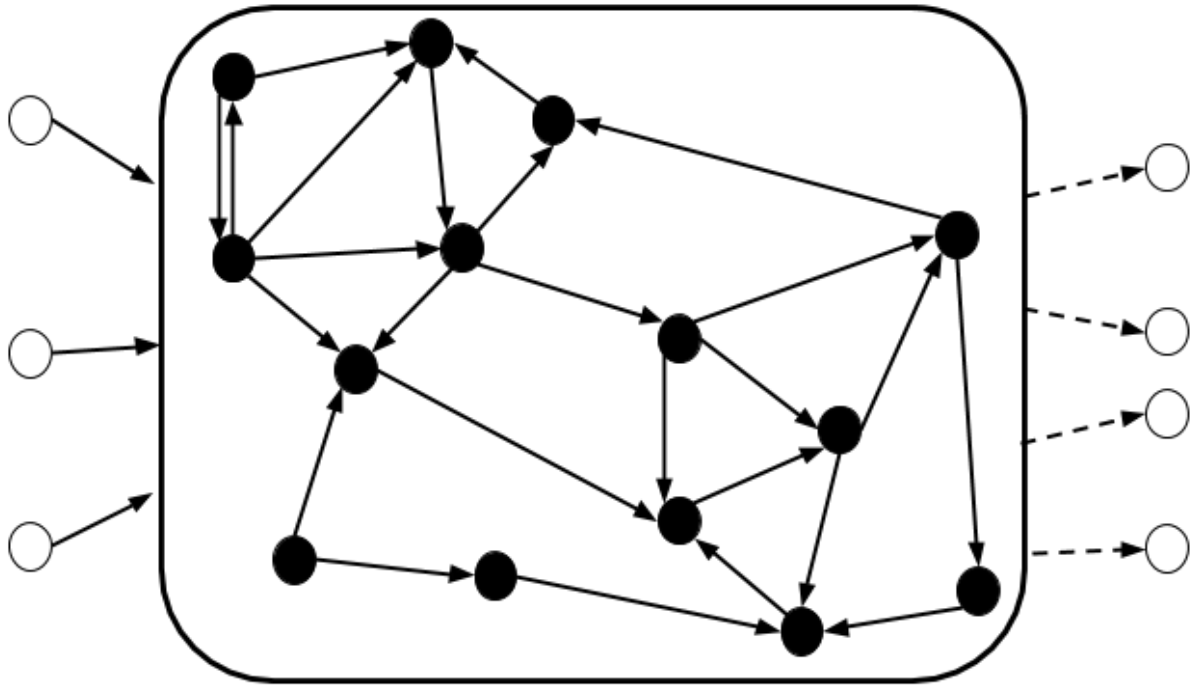


Figure 3 – This drawing illustrates a Random ESN, where the nodes colored in white on the left side are the input nodes, and the continuous arrows indicate a connection between an input node and all the nodes inside the reservoir. The nodes colored in black are the nodes inside the reservoir, and the continuous arrows represent the connections between the nodes. On the right side of the Figure, the nodes colored in white represent the output nodes, and the dashed lines represent the connections between all nodes inside the reservoir to an output node. All the connections inside the reservoir are drawn arbitrarily forming a random network.

Let $\delta\mathbf{X}$ be a matrix with t_{max} columns, where the t-th column is the vector $\mathbf{x}(t) - \bar{\mathbf{x}}$. Analogously, let $\delta\mathbf{S}$ be a matrix with t_{max} columns, where the t-th column is the vector $\mathbf{s}(t) - \bar{\mathbf{s}}$. That is

$$\delta X = \begin{bmatrix} x(0) - \bar{x} & \dots & x(t) - \bar{x} & \dots & x(t_{max}) - \bar{x}, \end{bmatrix} \tag{2.8}$$

$$\delta S = \begin{bmatrix} s(0) - \bar{s} & \dots & s(t) - \bar{s} & \dots & s(t_{max}) - \bar{s}. \end{bmatrix} \tag{2.9}$$

Therefore, the output matrix $\mathbf{W}^{out}$ can be found using the equation

$$W^{out} = \delta S \delta X^{\mathrm{T}} (\delta X \delta X^{\mathrm{T}} + \beta I)^{-1}, \tag{2.10}$$

where $\mathbf{I}$ is the identity matrix, and β is a regression parameter. The variable $\mathbf{c}$ can be found using the equation

$$c = -[W^{out}\bar{x} - \bar{s}]. \tag{2.11}$$

With this simple linear regression, the training part of the ESN is done. It is worth mentioning that the only weights that are trained, are the ones in the output layer. The weights in the input matrix and in the reservoir are drawn randomly in the beginning, and they remain the same for all computations.

2.2.2 Deep ESN

A deep neural network is a network that is split into layers, where the aim of each layer is to extract a meaningful amount of information from the input (DENG; YU, 2014). Using this definition the deep ESN can be constructed using this principle. The aim is to receive the data in the input and pass it through the network layer by layer until it reaches the output layer. The idea behind the Deep Echo Networks (*Deep ESNs*) as illustrated in Figure 4 is to split the network in the reservoir into smaller networks, such as each of these small networks are called layers (GALLICCHIO; MICHELI, 2016).

Let a *Deep ESN* with a total number of N nodes and L layers $(l_1, l_2, ..., l_L)$, where N is a multiple of L, each layer has N/L nodes. The algorithm to build the Deep ESN is the same as the one described above, the only difference is the part to build the network inside the reservoir. The algorithm to build the network is the following:

1. Given the network has N/L nodes and a mean degree D/L, build L random networks using the Erdös-Rényi model.

2. Then, create the connections between the layers. Firstly, create the connections between the input layer (W^{in}) and the first layer (l_1). Note that all nodes in the input layer will be connected to all nodes inside the first layer of the deep ESN. Secondly, create the connections between the layers, starting at the first layer (l_1), create the connections between the first layer and the second layer (l_2). Then, create the connections between the second layer, and the third, and so on until all the layers are connected sub sequentially (i.e., the layer l_i is connected only to the layer l_{i+1}.

3. Then, connect the last layer (l_L) to all nodes in the output layer (W^{out}).

An example of a deep ESN with 3 layers can be observed in Figure 4. This network model was designed to split the network into layers (i.e., each of the small layers is connected to another layer sub sequentially), and not into clusters (i.e., each of the clusters

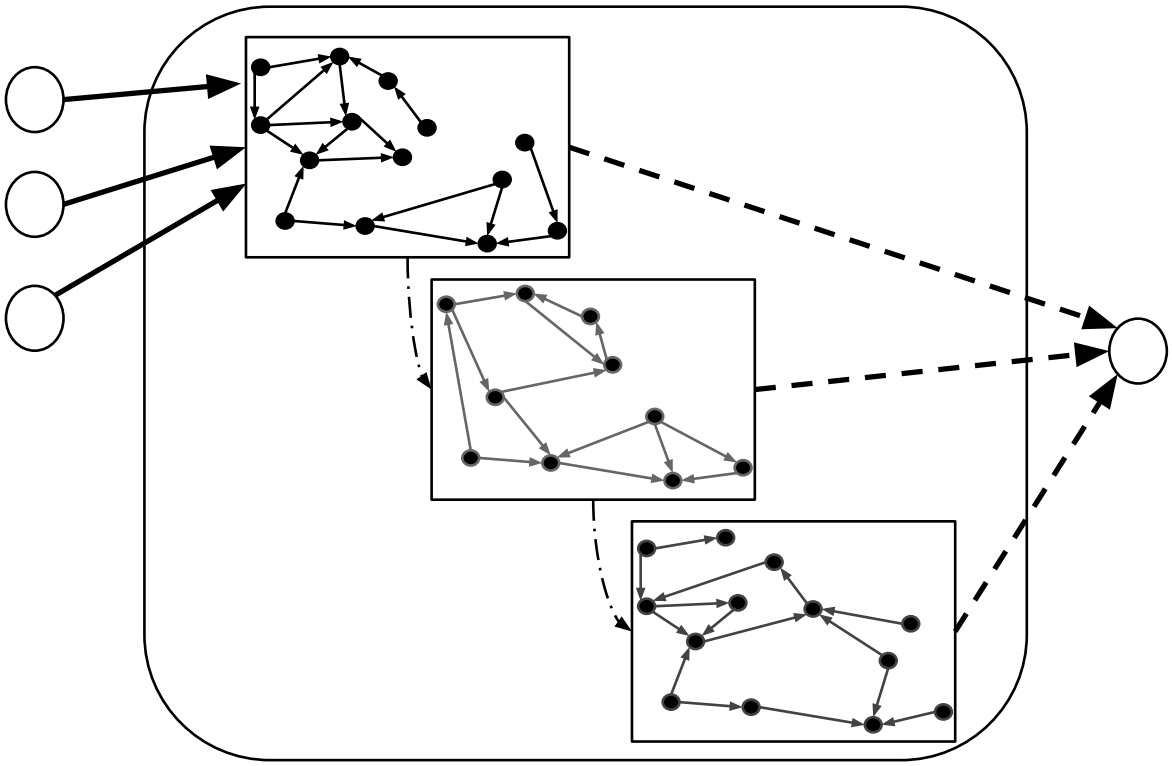


Figure 4 – Illustration of a deep ESN, with 3 nodes in the input layer, 3 layers inside the reservoir, and one node in the output layer.

can be connected to any other cluster). Note that the layer l_1 is not connected to the layer l_3.

While in the classical model the input layer nodes are connected to all nodes in the reservoir, in the *Deep ESNs*, they are only connected to the first layer (l_1). The topology of the reservoir is organized in such a way the nodes in the layer l_i are connected to the nodes in the same layer (l_i), and the nodes in the subsequent layer (l_{i+1}). All nodes in the reservoir are connected to the output layer. In the standard model for the *Deep ESN*, each layer is a random network.

2.3 Noise Reduction and Wiener filter

Wiener filter was designed to recover a desired signal d_k when it is corrupted by a noise signal v_k. It is considered that d_k and v_k are wide-sense stationary random processes, and the Wiener filter produces the minimum mean-square error (MMSE) for the desired signal (CORNELIS; MOONEN; WOUTERS, 2011). The minimum mean square error is given below.

$$J_{MSE}(w) = E\{e_k^2\} = E\{(d_k - y_k)^2\} = E\{(d_k - \mathbf{u_k^T w})^2\} \tag{2.12}$$

where d_k is the desired signal, y_k is the filtered (predicted) signal, $\mathbf{u_k}$ is the filter input, $\mathbf{w}$ is the filter to be defined, and $E\{x\}$ is the expected value of $\mathbf{x}$.

The MSE equation can be written as

$$J_{MSE}(w) = E\{d_k^2\} + \mathbf{w}E\{\mathbf{u_k}\mathbf{u_k}^T\}\mathbf{w^T} - 2\mathbf{w^T}E\{\mathbf{u_k}d_k\}\mathbf{w} \tag{2.13}$$

where $X_{uu} = E\{\mathbf{u_k}\mathbf{u_k}^T\}$ is the autocorrelation matrix and $X_{du} = E\{\mathbf{u_k}d_k\}$ is a cross correlation vector.

The minimum of the cost function can be obtained by setting the gradient equal to zero:

$$0 = \frac{\partial J_{MSE}(\mathbf{w})}{\partial \mathbf{w}} = 2X_{uu}\mathbf{w} - 2X_{du} \tag{2.14}$$

Therefore, the Wiener filter is

$$\mathbf{w} = X_{uu}^{-1}X_{du} \tag{2.15}$$

The Wiener filter aims to reduce/remove the noise from a signal. The good thing about this filter is the fact the method does not need to know anything beforehand.

3

Proposed Models

In this chapter, we propose three new ESNs, where two are clustered ESNs, such that each cluster in the reservoir is either a Barabási-Albert scale-free network (*Barabási-Albert CESN*) or a Erdös-Rényi network (*er-Albert CESN*). Another is the deep ESN with clustered layers (*Deep CESN*) where each layer is a clustered network rather than a random network. Still in this chapter, we present the materials and methods of the numerical study.

3.1 CESN model

In this work, two types of networks will be created using the algorithm described below, the Erdös-Rényi CESN and the Barabási-Albert CESN. Given a network with N nodes and mean degree equals D, to build the clustered networks, some parameters need to be defined first, which are:

- Number of clusters (C): Positive integer number, and divisible by N, indicates the number of clusters in the network. All the clusters contain the same number of nodes N/C. The parameter C need to be divisible by N for simplicity since all the nodes will have the same number of nodes in the experiments.
- P_{in}: Variable that indicates the percentage of connections between nodes of the same clusters. The value 0 for this parameter indicates that there are no connections between nodes of the same cluster, and 1 indicates a completely connected network. Each cluster contains a number of connections equals $N * D * P_{in}/C$.
- P_{out}: The same as stated for P_{in}, but it defines the percentage of connections between nodes of different clusters. Each network contains $N * D * P_{out}$ connections between nodes of different clusters.

In this work, $P_{in} + P_{out} = 1$ to preserve the mean degree of the network.

In summary, the variable C defines the total number of clusters in the network, and the P_{in} and P_{out} define whether the nodes inside the cluster are more connected or whether there are more connections between nodes of different clusters. Depending on the set of parameters used to create the clustered network, the resulting network might not be connected (i.e., it is not possible to go from a node in the input layer to a node in the output layer using the network edges). This issue can be solved with the following algorithm:

1. Using the *Depth-First Search* (DFS), it is possible to find all components of the network. As the network in the reservoir is a network with directed edges, the DFS starts in nodes with input degree equals to zero. If there are no such vertexes, the network is already connected.

2. Then, after finding all the components, all the starting nodes are kept in a list $(v_0, v_1, ..., v_{k-1})$. Then, a connection is made between the adjacent pairs that are in the list, and the connection receives a random weight (i.e., a connection is created between the nodes v_i and v_{i-1}, for $i \geq 1$).

These variables described here as well as the algorithm to make the network connected, will be used to build both CESNs, the Erdös-Rényi CESN, and the Barabási-Albert CESN. The process to build the networks is described bellow. Once the process is finished, the network is prepared to be used in the *reservoir*.

3.2 Erdös-Rényi CESN

More complex networks can be used in the reservoir, as the clustered networks for example, which is illustrated in Figure 5. In the *Erdös-Rényi CESN*, the reservoir is composed by a clustered network, where each cluster is a random network (Erdös-Rényi network). The idea is simple, create smaller random networks and then create connections between these smaller networks, which will form a clustered network. The variables number of clusters (C) and mixture level (P_{in}, P_{out}) defined before are used to build the networks, and control their properties. Given a network with N nodes and containing C clusters, each cluster will have a total of $M = N/C$ nodes, where N is divisible by C.

The mixture level is a combination of P_{in} and P_{out}, where $P_{in} + P_{out} = 1$, that defines how many connections will have between nodes inside the same cluster (P_{in}), and how many connections exists between nodes of different clusters (P_{out}). Given that the network contains N nodes, mean degree D, and C clusters, there will be $K = P_{in} * N * D/C$

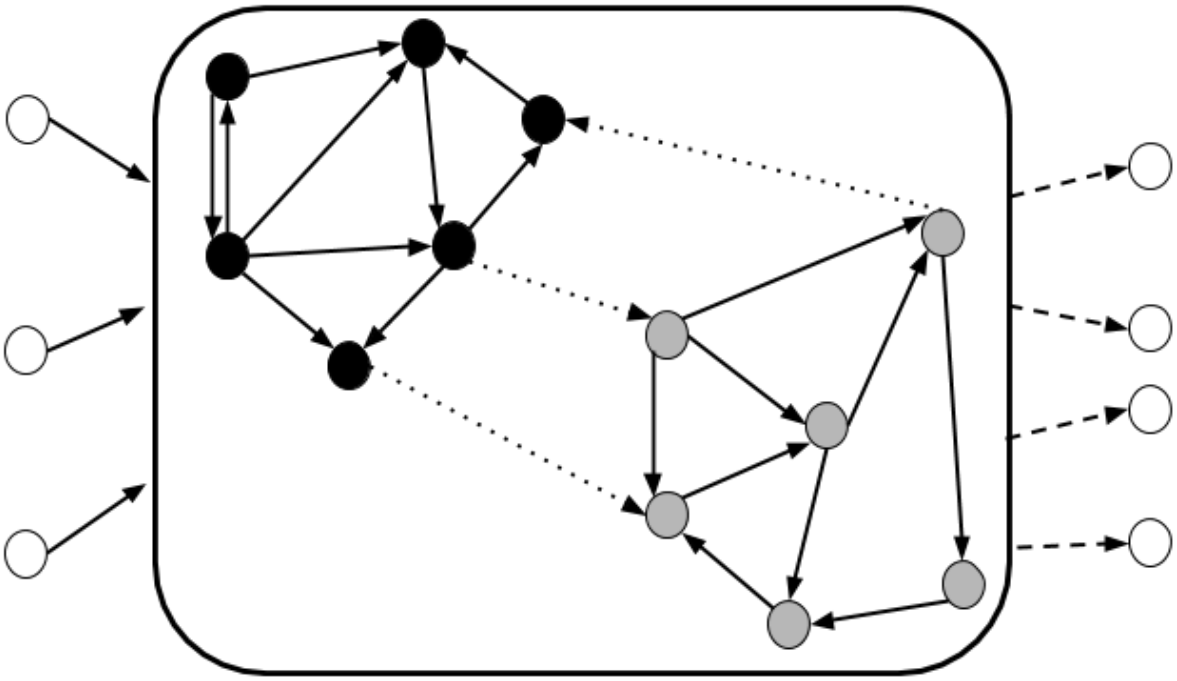


Figure 5 – The figure illustrates a *CESN*, which contains a clustered network in the reservoir, where the colors differentiate the nodes from different clusters. The continuous connections inside the reservoir denote connections between nodes of the same cluster, and dashed connections represent connections between nodes of different clusters.

connections inside each cluster, and a total of $M = P_{out} * N * D$ connections between nodes of different clusters.

1. Create a Erdös-Rényi random network with N/C nodes and $K = P_{in} * N * D/C$ edges.
2. Repeat the previous step C times.
3. Then, there are C random networks that are not connected to each other. The next step is to connect them.
4. Think as each of the C clusters as big nodes to create the connections between nodes of different clusters.
    a) Select two clusters randomly (c_i, c_j).
    b) Select one random node from the cluster c_i (u), and one random node from the cluster c_j (v), then create an edge between these two nodes.
    c) Repeat this process $M = P_{out} * N * D$ times.

In summary, the network construction process begins with M nodes, and then K pairs of nodes $(u_i,\ v_i)$ are chosen randomly with the same probability without repetition

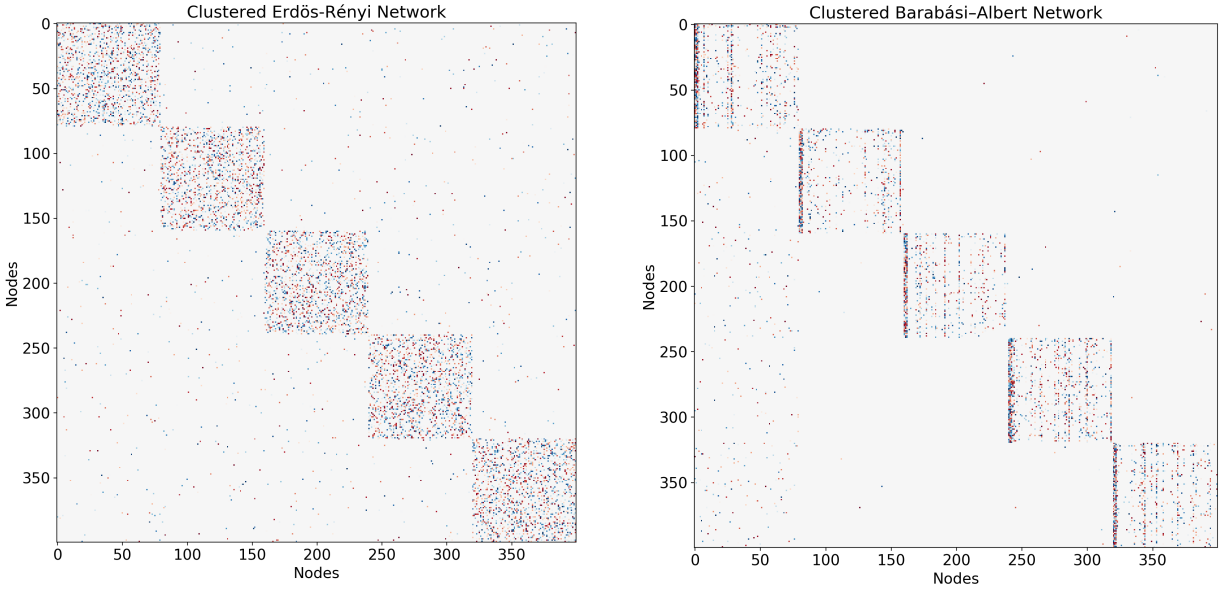
(ERDÖS; RENYI, 1961). Then, for each of these pairs, a connection is made between the nodes, choosing the connection weight randomly. This process is repeated C times, and then a total of C clusters are created. To create the connections between nodes of different clusters, M pairs of nodes (u_j, v_j) are chosen, such that u_j and v_j do not belong to the same cluster. After generating the pairs, a connection is made between (u_j, v_j) choosing a random weight for it. At the end of this process, a clustered network will be created, such that, each cluster is a random network, and the connections between nodes of different clusters are random networks.

3.3 Barabási-Albert CESN

In a *Barabási-Albert CESN*, the reservoir is composed of a clustered network, where each cluster is a scale-free network. The scale-free network model has been proposed by (BARABÁSI; ALBERT, 1999), and it receives this name due to the degree distribution of the nodes, which follows a power law.

In this model, the network begins with a small number of nodes n, where $n << N$. All n nodes are connected, i.e., there is a directed connection between all pairs of nodes. A total of $N - n$ nodes are added one by one in this network. When a new node is added to this network, a connection is created between the new node and a given node added previously in the network. Since the reservoir must be a directed graph, the algorithm proposed in (BOLLOBAS et al., 2003) was used to build the network.

1. Given that the network has N nodes, each cluster will have $N_C = N/C$ nodes, where $n << N_C$.
2. Create a directed complete network(i.e., all nodes are connected to each other) with n nodes. Then, add $N_C - n$ nodes with zero connections to the other nodes, and then add $N * D * P_{in}/C$ edges to this network using the scale-free model for directed graphs (Nodes with more connections have a higher probability to make new connections with other nodes).
3. Repeat the previous step C times.
4. At this point, we end up with a network that contains C scale-free networks that do not have connections with other clusters. Now, each cluster will be treated as a big node to create the connections between different clusters.
5. The complete network will start with only one big node, then, the other big nodes are added one by one.
    a) Let a network with C clusters, thus the network will have C nodes.



(a) Clustered Erdös-Rényi Network (b) Clustered Barabási–Albert Network

Figure 6 – The image shows how is the final adjacency matrix for either the clustered networks.

b) Start with a network with a single node, the first cluster in this case.

c) Add $C-1$ nodes($C-1$ clusters) to this network, and then create $N*D*P_{out}$ connections, and then choose the pair of big nodes to connect based on a scale-free distribution. Once the two clusters are chosen(c_i, c_j), get one random node from the c_i and a random node from c_j, and then make a connection between them.

d) Repeat this process until the total number of connections is created in the network.

In summary, the first step to build the scale-free clustered network is to generate C scale-free networks using the algorithm stated above. Each of these C scale-free networks has a mean degree equals K and starts with n initial nodes. Then, the next step is to connect these small networks between them, such that the connections between nodes of different clusters form a scale-free network. The same algorithm is applied to achieve this, where the C clusters are the nodes in this case.

A total of $N*D*P_{out}$ connections are created between the nodes of different clusters. The network starts with c clusters, where $c < C$. Then $C-c$ clusters are added one by one. When a new cluster (c_i) is added to the network, a connection is created between a node of this cluster$(u \in c_i)$, and a node from a previously added cluster $(v \in c_j)$. In Figure 6, it can be observed the difference between the two clustered CESNs, and how it looks like the final configuration of the network.

In the end, the degree distribution of the networks inside the cluster, and the degree distribution of the clusters when connected to other clusters is a scale-free distribution. This network will have a very different topology as can be observed in the Figure 6.

3.4 Deep CESN

In this work, an extension will be made for the deep ESN, such that each layer is a clustered network. The model consists of keeping the layer structure of a *Deep ESN* network, and use a clustered network in the layers rather than a random network. This type of network will be denoted as *Deep CESN*. The idea behind this is to have a network with both layered and clustered properties.

The parameters for the *CESN* will be the same as stated earlier, and the new parameter for this network is the number of layers (L). The networks in the layers can be an Erdös-Rényi CESN (see section 3.2 for more details) or a Barabási-Albert CESN (see section 3.3 for more details). The algorithm to build this type of network is the same as the deep ESN, and the only change in the algorithm is when creating the networks for each layer.

The algorithm will work as follow, L clustered networks $(l_1, l_2, ..., l_L)$ are created, where each network has the same size N/L. Then, N/L connections are created between the layer i and the layer $i + 1$ if $i < L$.

3.5 Materials and Methods

The section below describes the tasks which the ESN and its extensions are submitted to solve. There are three different tasks, the estimation of an unobserved variable of the chaotic system, specifically, the Rössler system, the frequency filtering task, and the noise filtering task.

3.5.1 Observer Problem

ESNs can be applied to observer problems, i.e., the estimation of an unobserved variable of a dynamical system based on measurements of an observed variable. One example system used by the authors is the chaotic Rössler system (LU et al., 2017), which is defined by the following differential equations:

$$\begin{aligned} \frac{dx}{dt} &= -y - z, \\ \frac{dy}{dt} &= x + ay, \\ \frac{dz}{dt} &= b + z(x - c). \end{aligned} \tag{3.1}$$

In this work, we will apply the CESNs to the observer problems using the Rössler system. In the numerical studies described in the next section, the x-component of the Rössler system is used as the observed variable, i.e., as the input signal for the CESN. The y-component is used as the target signal which should be approximated by the CESN's output.

To obtain the trajectory of the Rössler system, the system is solved using the 4-th order Runge-Kutta method. The constants a, b and c, are set to 0.5, 2.0 and 4.0, respectively. The initial conditions for the training, as well as for the test, are chosen randomly from a uniform distribution on $[0, 1]^3$.

3.5.2 Frequency Filtering

As a second task with comparison purposes, we use a sum of multiple sine signals with different frequencies and phase shifts and variable amplitudes determined by random envelope functions as the CESN's input signal. One of these sine signals is the target signal, i.e., the CESNs' tasks are to let a certain frequency pass and filter out the others.

A disclaimer is that there are methods in the literature that excels to solve this problem as the Fourier Transform. In this task, the aim is only to compare performance between the ESNs and the CESNs.

Frequency filtering is an interesting task to evaluate the ability of the ESNs to extract features from the data. Moreover, the task can become difficult depending on the parameters to generate the data. In the simulations, the mixed input signal S, is a sum of the signals $\mathbf{s}(1); \mathbf{s}(2); ...; \mathbf{s}(k)$. The aim is to identify these signal components, which generate $\mathbf{S}$. In our experiments, sinusoidal signals multiplied by a function called, an envelope is used. Some parameters are required to generate the data.

The algorithm to generate the input signal is described below:

- Let *steps* be the total number of steps, ΔT a time interval and k the number of frequencies generated, where $freq_i \in \mathbb{R}$, k sine functions are generated. Each sin function has the following form:

$$s_i = sen_i(freq_i \times (x + \lambda)), \tag{3.2}$$

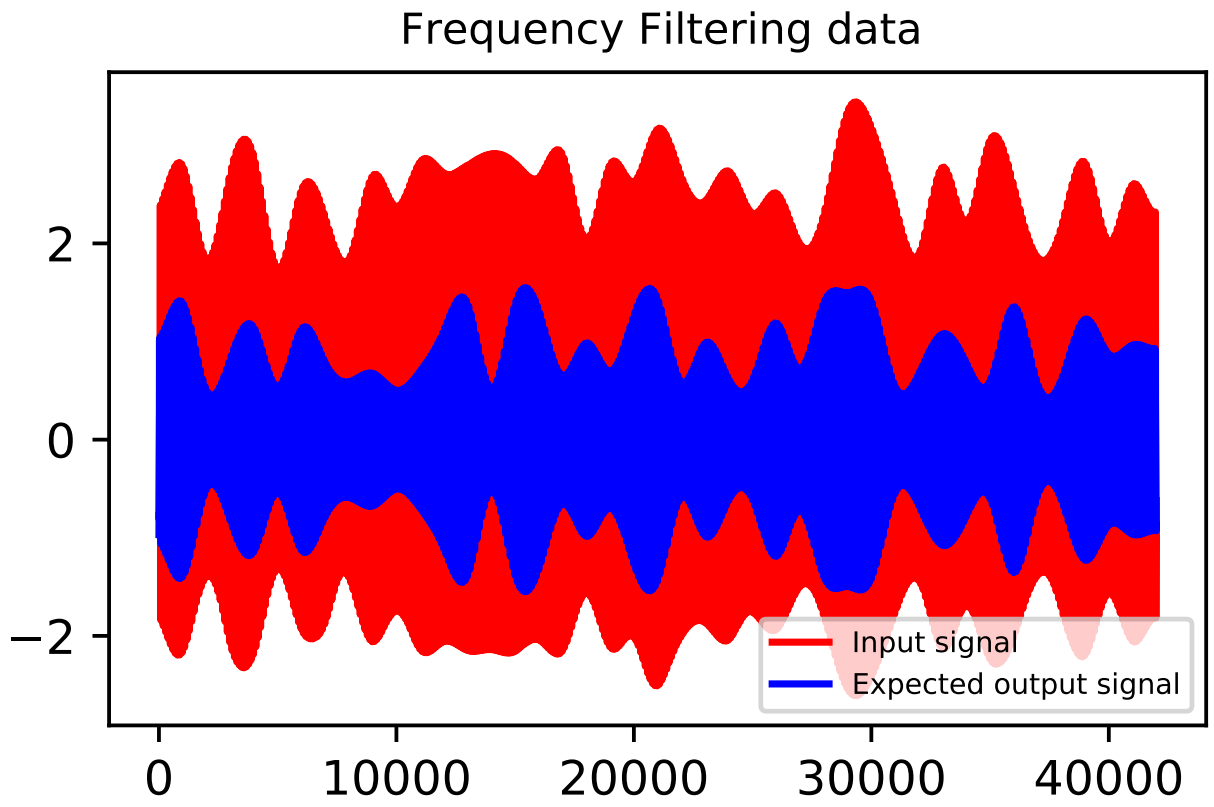


Figure 7 – The red signal is the mixed signal of 3 component signals with the frequencies (1; 2; 3). The blue color is the signal with frequency 1, which is the desired signal as an output of the neural network.

where $\mathbf{x}$ is between [0; steps], and λ is an offset value chosen randomly in the range [0; 2 π].

- Data points are generated randomly on the y-axis in the interval $[y_0; y_1]$. On the x-axis, the data points are equally spaced between $[0; steps]$, where the difference between two points is ΔT, i.e., $x_{i+1}x_i = \Delta T$. Subsequently, a cubic interpolation is made between these points, and the resulting function is called envelope function. in the end, there are k envelope functions, $ef_1, ef_2, ..., ef_k$.

- After that, each sine function ($\mathbf{s_i}$) is multiplied by an envelope function ($\mathbf{ef_i}$). And finally, we get $\mathbf{S}$, which is the sum of these multiplications, i.e.,

$$S = s_1 \times ef_1 + s_2 \times ef_2 + ... + s_k \times ef_k. \tag{3.3}$$

Figure 7 shows the shape of the data generated by the algorithm detailed above. ESNs aims to extract a single component from the signal's sum. In this case, the input layer and the output layer of the neural network contain one neuron. Worth mentioning that identifying more than one signal is similar to this process. The only modification needed is in the output layer.

After submitting the training phase to ESN with the data previously generated as shown in Figure 7, the network is subjected to a test stage to evaluate the performance of methods.

3.5.3 Denoising task datasets

The denoising problem can be described as follow: A signal with noise is given as input and the aim is to give as an answer the filtered signal. For the artificial types of noise, a signal will be generated as described in the section 3.5.2, then the noise will be added to the raw signal using the methods below. For the ECG dataset, which is a real-world signal, the dataset was collected by (GOLDBERGER et al., 2000; A.P., 2005).

We study the performance of the ESN methods described above using three different tasks. For the first two tasks, we add Gaussian noise and impulse noise to a randomly generated wave signal consisting of multiple sin signals with random phases and frequencies and fluctuating amplitudes determined by randomly generated envelope functions. Then we apply the ESNs to denoise the signal and reconstruct the original wave signal. As a third task, we use the ECG dataset, which contains a set of electrocardiograms records of different persons (GOLDBERGER et al., 2000; A.P., 2005).

3.5.3.1 Gaussian Noise

The Wiener filter is a well-known method, which was made to remove specifically this type of noise. Let $\mathbf{x}$ be a signal without any type of noise, the idea is to add noise to this signal following a Gaussian distribution, which is given by the following equation:

$$p_G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.4}$$

where the μ is equals to 0 and the σ is chosen arbitrarily.

3.5.3.2 Impulse Noise

Given a signal $\mathbf{x}$, a noise level δ, and n_{dp} noisy data points chosen randomly from $\mathbf{x}$, each of these data points will be updated according to the equation:

$$x(i) = x(i) + r, \tag{3.5}$$

where r is a value between [-δ, δ]. The main idea of the impulse noise is to add different perturbations along the signal, and try to remove them. This kind of task is a bit different

from the previous one (Gaussian noise) since the noise is not spread onto the signal according to a probability distribution.

3.5.3.3 Electrocardiogram dataset

An electrocardiogram is a signal which measures the electrical activity of the heart. The ECG dataset used in this work consists of a set of recordings collected from 90 different persons. Each of these recordings is divided into two different signals, a raw signal, which contains noise, and a filtered signal, where the noise was removed.

For the numerical studies, due to computational and hardware limitations, we split the dataset into a training set consisting of records from 50 persons and a test set consisting of records from 20 persons. The dataset contains multiple records for each person, but we restrict ourselves to use only one record per person. To obtain one training signal and one test signal for the ESN, we merge the individual records in the training and test phases.

3.5.3.4 Timesteps delay in dataset

In order to make use of non-causal relationships between input and target data, we introduce the delay parameter d. Then, the input signal will be shifted by d time steps relative to the target signal. With this strategy, the ESN methods can take into account d input data points ahead of the target signal.

4

Experimental Results

In this section, each of the experiments is divided into sections, and each of them describes the results obtained. The main goal is to compare the performance between the ESNs and its extensions. To better evaluate the performance of the methods, a numerical analysis will be made to find the best set of parameters for the networks that yields the best performance, and then compare these results. To measure the performance, the Normalized Root Mean Square Error (NRMSE) will be used. In the charts, the average NRMSE will be plotted with the standard deviation.

4.1 The Observer Problem

While evaluating a particular variable, some default values will be set to the other parameters as stated in the Table 1. The parameters in this table are the same as used in (LU et al., 2017). In this task, for each set of parameters, the ESNs and the CESNs were submitted to **100** executions to generate the results, and then the overall performance will be the average of the executions.

The first task is to give as input to the ESNs, the x component of the Rössler

Table 1 – This Table lists the ESN and CESN variable's default values.

Variable	**Default value**
Nodes (N)	512
Mean Degree (D)	20
Activation function (f)	tanh(x)
ΔT	0.01
α	0.22
Bias (γ)	0.1
Learning rate (β)	2×10^{-7}
Spectral Radius (ρ)	1

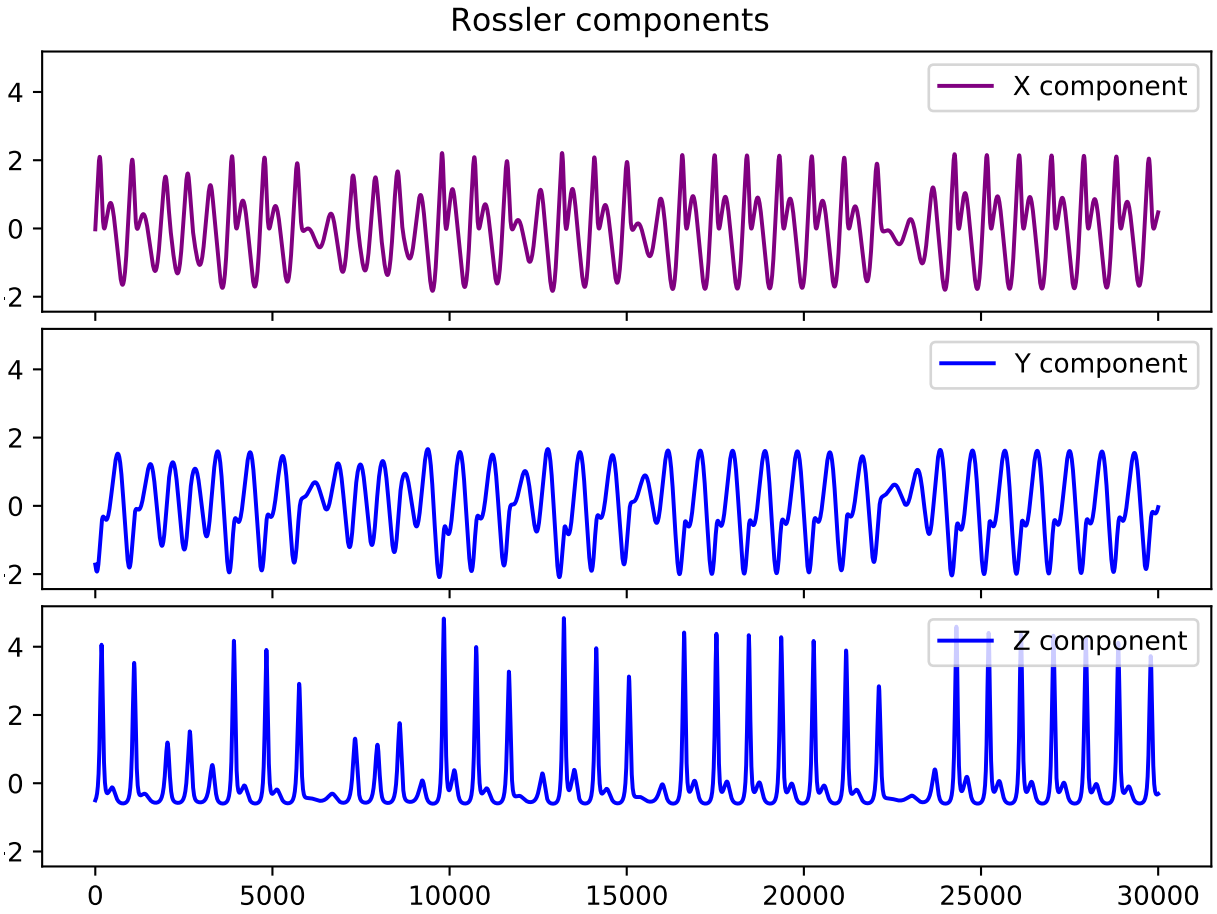


Figure 8 – The X component, colored in purple, is the input of the network, and the Y and Z components, colored in blue, are the expected outputs of the network.

System, and get as output the components y and z, as the Figure 8 shows. The ESNs can solve this problem with high precision, as shown in (LU et al., 2017), but the goal here is to compare the performance of the ESNs with a random network in the reservoir and the CESNs.

In the observer problem with Rössler System, the *train steps* and *test steps* were set to $20k$ and $10k$ respectively, since the convergence is fast in this case. To find the optimal number of clusters, the mixture level was fixed ($P_{in} = 0.75$, therefore $p_{out} = 0.25$). Moreover, for the Barabási clustered network, the number of *initial vertices* in each cluster was set to 1 for simplicity. Note that the Figures 9 and 10 shows only the results for the component y. The results for the component z were omitted since the results are quite similar for both components.

The very first step is to find the optimal value for the number of clusters, which is different for each CESN as Figure 9 shows. Note that the Erdös-Rényi CESN model with only one cluster is not the same as the random ESN in the number of connections if, and only if $P_{in} < 1$, which in this case is different because $P_{in} = 0.75$. After finding the number of clusters that yield the best performance, new simulations are performed fixing

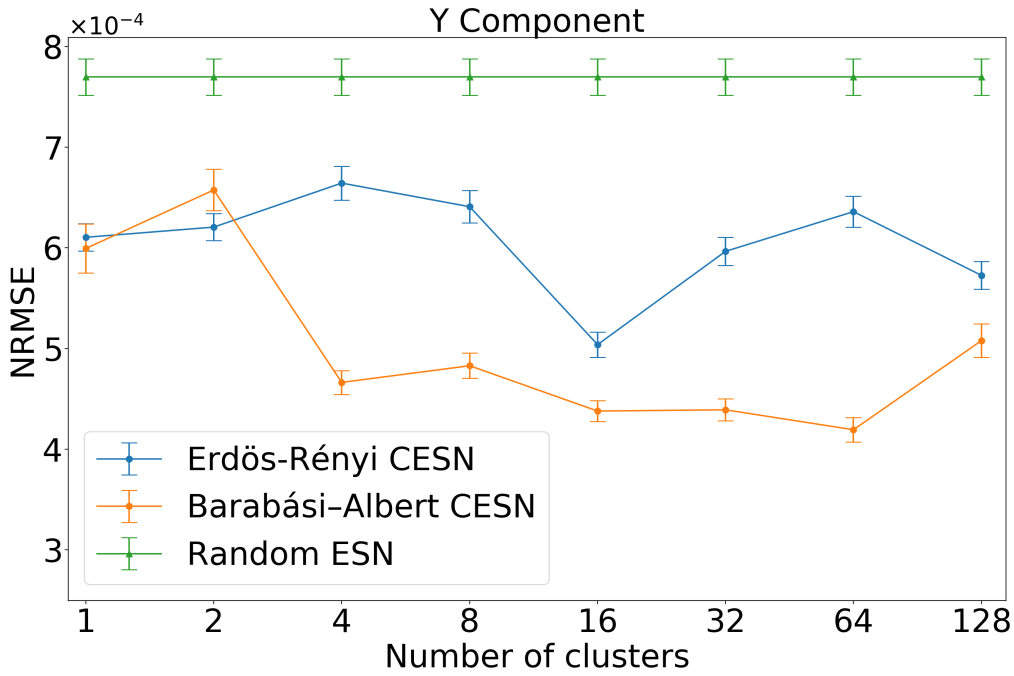


Figure 9 – The performance of the CESNs over different values for the number of clusters.

the number of clusters (C) and varying the mixture level parameter (P_{in} and P_{out}).

As the Figure 10 shows, the CESNs can achieve better results compared to the ESN. Even though this is an easy task for the ESNs in general, with the correct set of parameters, the CESNs can achieve better results compared to the ESN. Worth mentioning that the CESN with Barabási-Albert clusters has a slight advantage over the other two other methods.

Although the NRMSE given by all the methods are small, in a qualitative observation, as the Figure 11 shows, all the methods tested here have good results in this task, and they can predict the trajectory with a small error rate (NRMSE less than 10^{-3}). The plots for the z-component were omitted since they are similar for the y-component.

Despite the fact the NRMSE is small for all methods, these results shed a light on how CESNs can improve the performance compared to the ESN. Moreover, these results demonstrate the implementation of the CESNs is correct, as it can solve the problem with similar and even better results compared to the ones found in (LU et al., 2017) with the classical ESN model. In the section bellow, more complex tasks will be used to test this hypothesis.

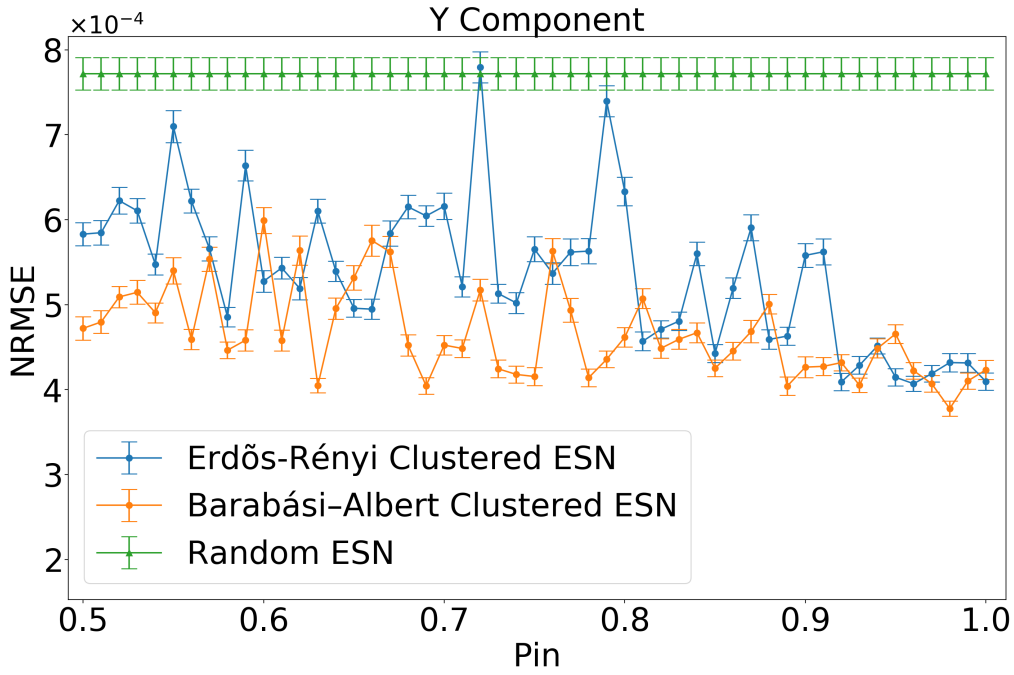


Figure 10 – The performance of the CESNs over different values for the P_{in} parameter.

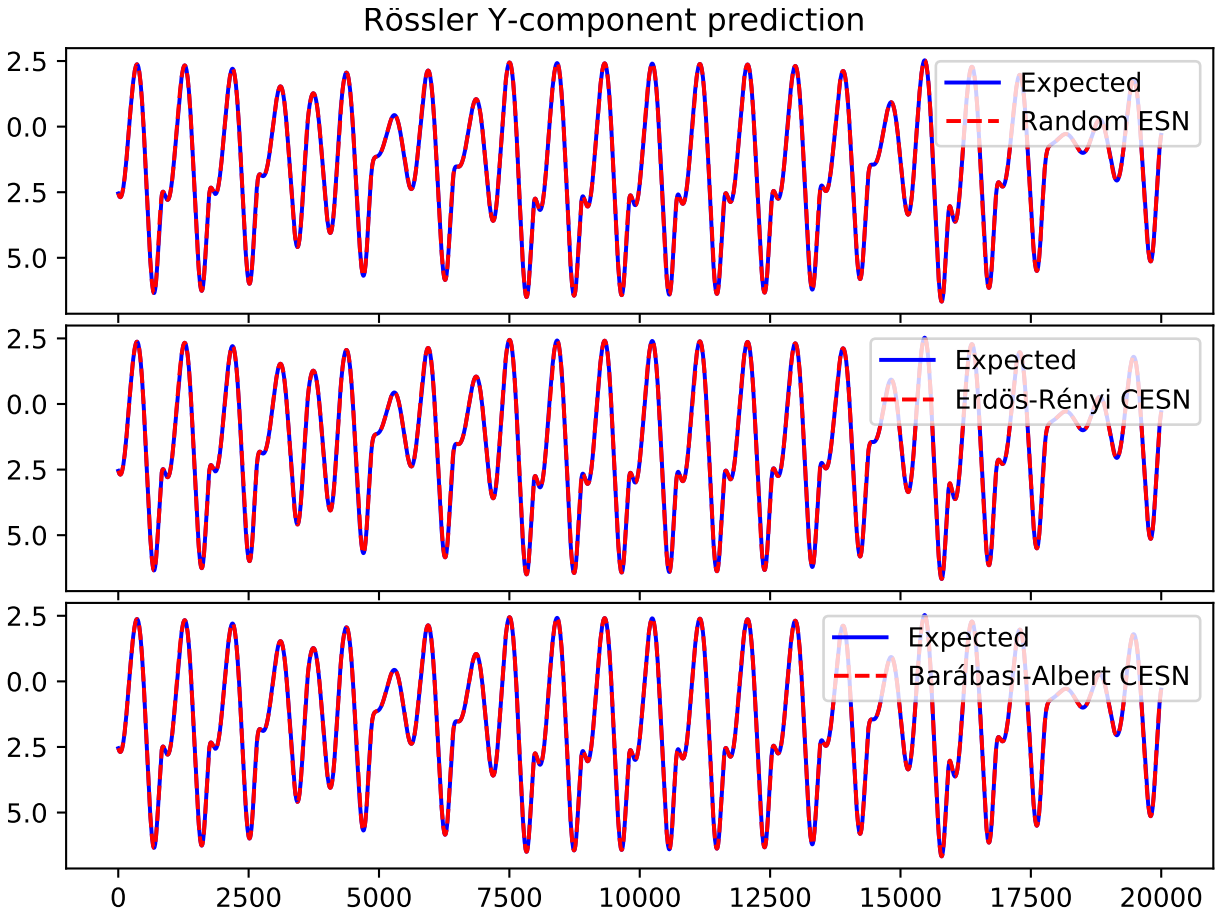


Figure 11 – The blue line is the expected output, and the dashed line in red is the output given by the networks.

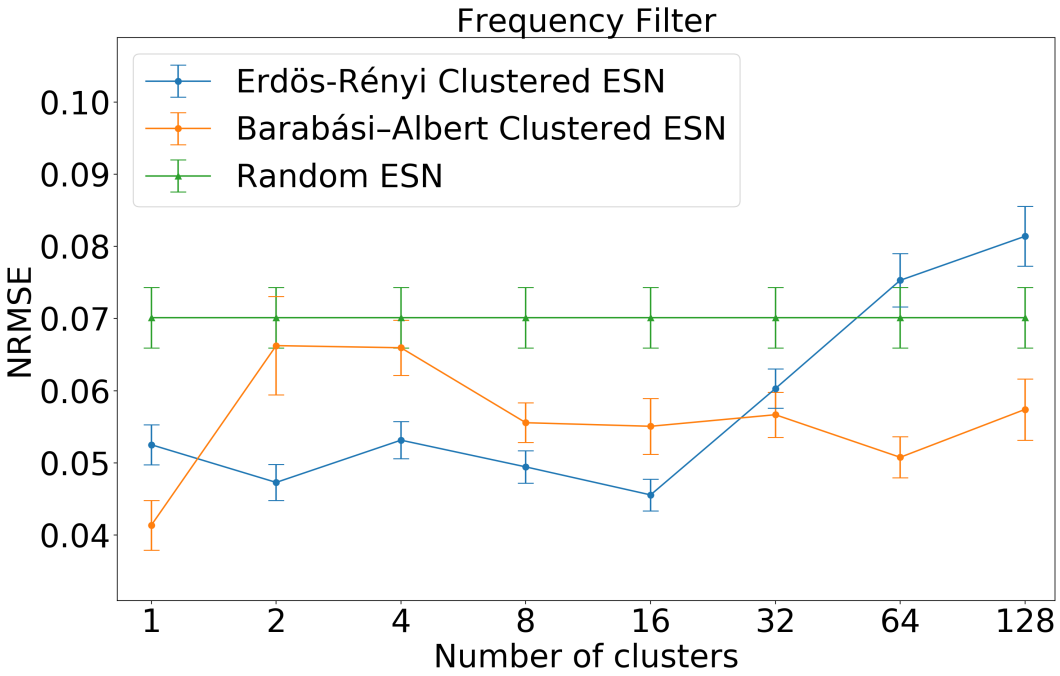


Figure 12 – The performance of the CESNs as the number of clusters changes.

4.2 Frequency Filtering Task

The second task to be tested is the frequency filtering. At first, a signal that is a sum of two other signals with frequencies equals 1 and 2 will be used. The input data and the training data will be generated as explained in section 3.5.2. The idea is the same as in the previous experiment, which is to find the set of parameters that gives better performance.

Comparing the results in Figure 12, the CESNs also perform better than the random networks in the reservoir, by only changing the number of clusters. Interesting to note that as the number of clusters increases, the network adjacency matrix will become less sparse and more centered around the diagonal, and that can explain why the performance get worst when the number of clusters is too high. As we can observe in the figures, the number of clusters indeed has an impact on the CESNs performance.

The Figure 13 shows the performance of the clustered networks as the mixture level changes. The CESNs improves the performance as the P_{in} increases, which means that the networks with a high number of connections inside the clusters, and sparse connections between nodes of different clusters results in a better performance. Using the optimal values for the parameters C and P_{in}, the CESNs can achieve an expressive gain in performance over the ESNs. Using CESNs one can achieve a (**30** to **40**) percent smaller NRMSE compared to the random ESN. While the parameters C and the P_{in} are important in the overall performance, the number of initial nodes in the Barabási-Albert

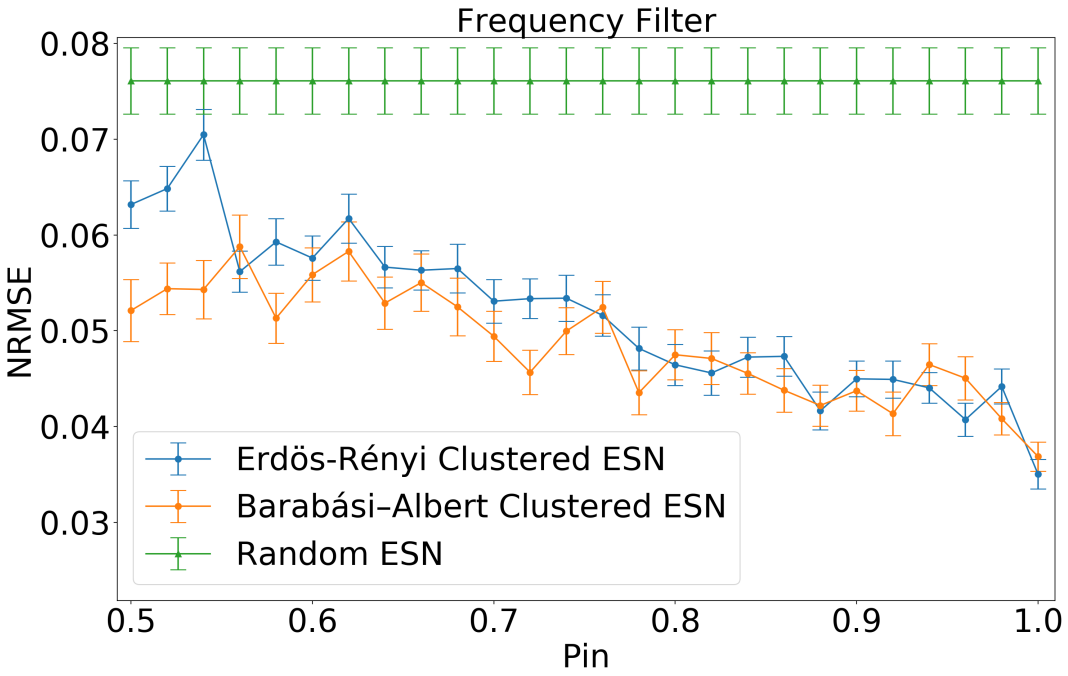


Figure 13 – The performance of the CESNs as the parameter P_{in} changes.

networks does not decrease the NRMSE, as the Figure 14 shows.

These results shows that when the network is split into clusters that can indicate a substantial gain in performance over the random networks. The idea of having a network with different number of clusters is due to the fact the input signal is a linear combination of other signals each with different frequencies. This might be due to the fact that different clusters can "capture" different types of frequencies present in the input signal, but this needs to be further investigated.

So far, the ESNs have been submitted to only a signal with two frequencies. To make the task harder, a test was made to check whether the number of frequencies reflects in the performance of the ESNs. The ESNs will be tested with signals with more than two frequencies. That is, the input will be the sum of k signals, where the frequency of the signal s_i is i, for $i = 1, ..., k$. For each signal, the parameters of the CESNs will be optimized, but tested in a smaller set of values for the parameters $C \in [1, 2, 4, 8, 16, 32, 64, 128]$ and $P_{in} \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

The performance of the methods seem to follow the same trend as stated in Figure 15, oscillating in some cases. Since the set of parameters tested is smaller in this case, there might be some room for improvement. Note that the three methods follow the same trend, and there is an advantage for the CESNs over the ESN in all the cases. Moreover, the performance of the CESNs is more stable with smaller error bars than the ESN.

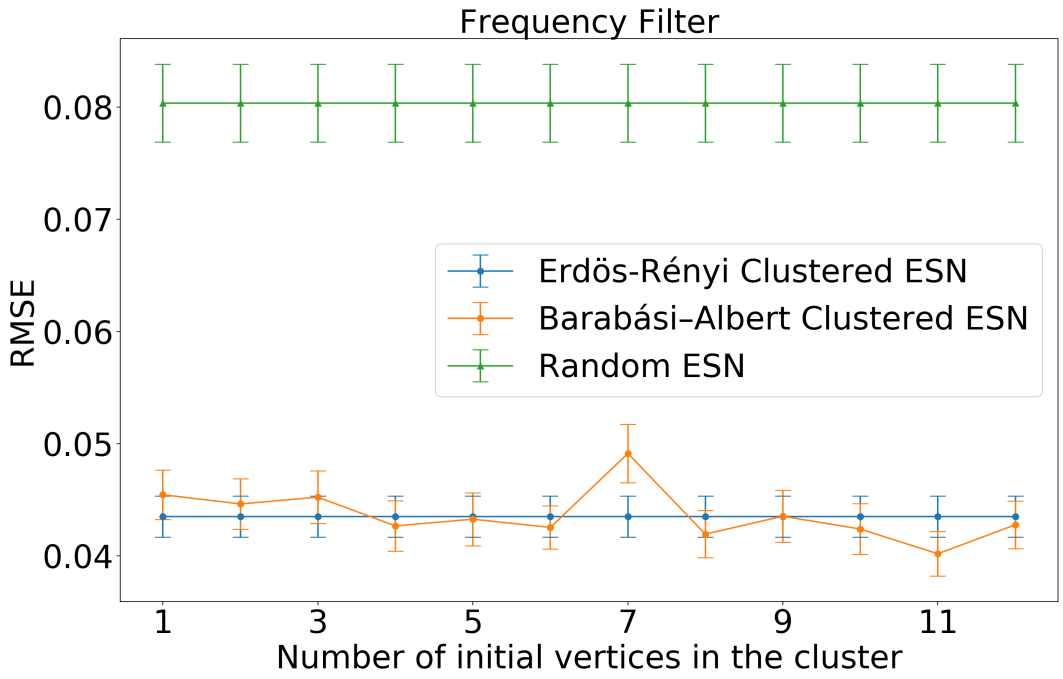


Figure 14 – The NRMSE over different values for the initial nodes parameter in the *Barabási-Albert CESN*, which does affect the performance.

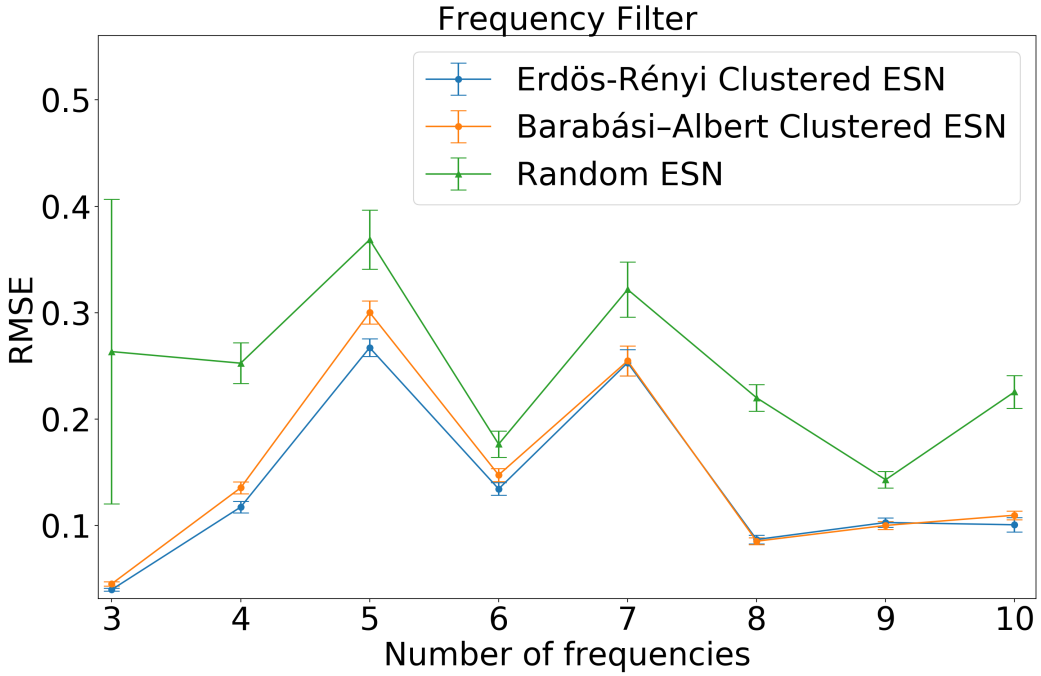


Figure 15 – The performance(NRMSE) of the CESNs as the parameter number of frequencies changes.

4.3 Denoising task

In this task, the aim is to find a solution for filtering the noise from a given signal using ESNs and its extensions. Moreover, three different types of noise will be added to the signals, the impulse noise, the Gaussian noise, and real-world noise in an ECG dataset. Going towards the same direction as the results observed in the previous section, as the ESNs can distinct different frequencies, it also may detect the noise from the signals, remove it, and give as output the filtered signal. These experiments will try to prove this hypothesis.

The general idea is to feed the model with a noisy signal and get the filtered signal as output. To compare the results, the Wiener filter method will be used as a baseline. To find the best parameters for the Wiener filter, a grid search is performed. The parameters to be optimized in the Wiener filter are the *window size* and the *noise-power*. For the ESNs, some default parameters are fixed as shown in Table 2, while others, specifically, α, number of clusters (C), cluster mixing level (P_{in}), number of layers (L) are left as free parameters to be studied. Note that some values in Table 2 are not the same from the ones in Table 1. All the results presented here are averaged over **8** executions due to an increase in the complexity of the experiments.

Table 2 – This table lists the ESN and *CESN* parameters default values in the denoising task.

Parameter	**Default value**
Nodes (N)	1050
Average Degree (D)	20
Learning rate (β)	2×10^{-7}
Training steps	5×10^{4}
Test steps	3×10^{4}

The idea is to find the best parameters in the artificial datasets, the impulse noise and the Gaussian noise, and then try to use the set of parameters in a real-world application, which is to filter the noise from ECG signals in this case.

Firstly, the best parameter value of α and the delay parameter (d) are determined. After that, for the CESNs, the parameters number of clusters (C) and the mixture level (P_{in}, P_{out}) are optimized.

4.3.1 Impulse noise reduction

An illustrative example is shown in Figure 16, which presents a noisy signal and the filtered signals by Wiener filter and by the ESNs methods. In this specific case, the desired signal

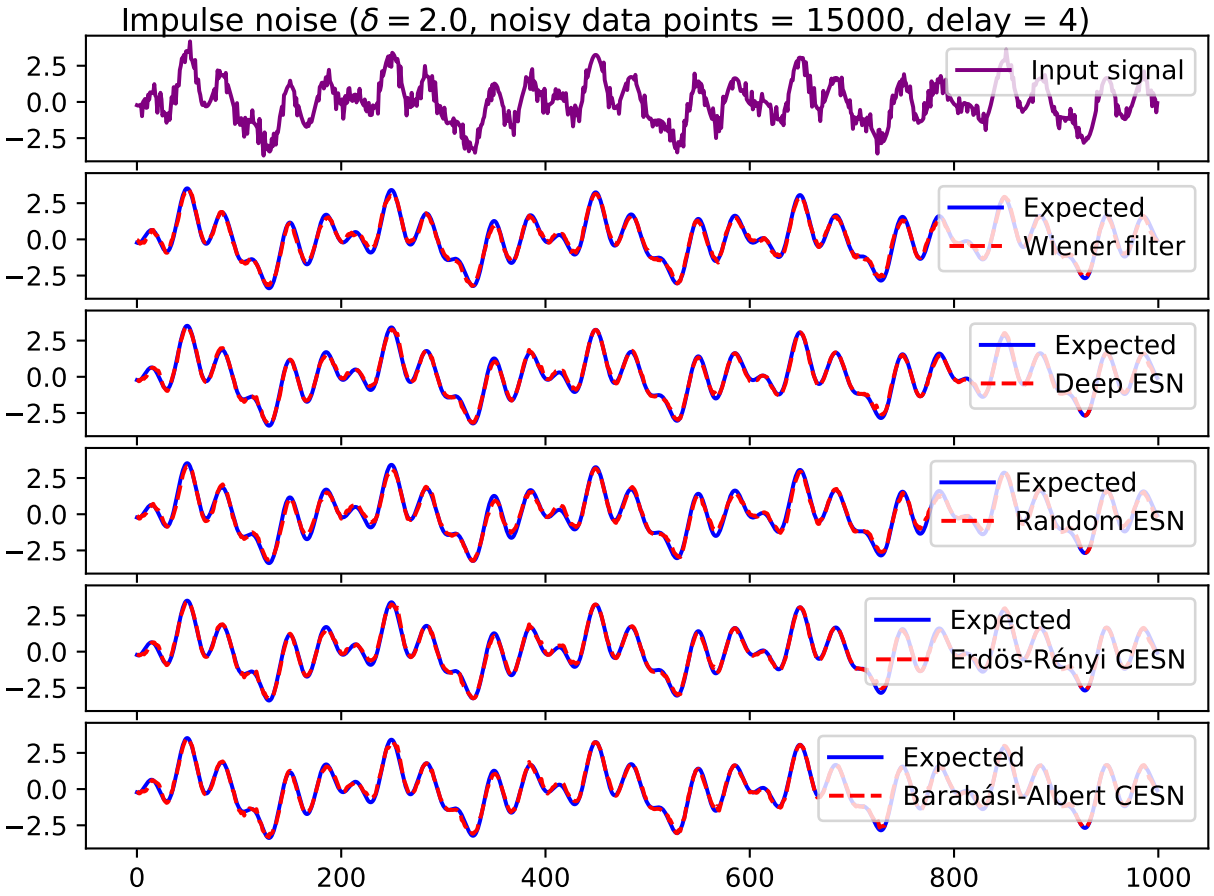


Figure 16 – A specific noisy signal and the filtered signals by Wiener filter and various ESNs.

without noise, marked as "Expected" in the Figure is known, because the noise is added to the original signal. Qualitatively, all the methods generate reasonable results.

Before presenting the performance of various ESNs on impulse noise reduction, the effect of the parameter α is analyzed, which is a common parameter that controls the memory in the reservoir, and it appears in all ESNs versions. The variable α is directly affected by how many steps are delayed in the input signal. To find the best combination of these two parameters, a grid search is performed using only the *Deep ESN* method as a baseline. The results are presented in Figure 17. The heat-map shows as the delay increases, the performance of the *Deep ESN* improves, and for larger values of α, the performance gets better. This is reasonable because if the delay increases, the window of the ESN gets larger, therefore, less memory is required to remember past states.

Now, the performance of ESNs varying α parameter can be investigated by fixing the time-steps delayed in the input signal. The Figure 18 shows that the impulse noise reduction's NRMSE of the *Barabási-Albert CESN*, the *Erdös-Rényi CESN*, and the *Deep ESN* are larger when α is small, while those get much better results than Wiener filter when $\alpha \geq 0.6$ and a stable performance is observed when $\alpha \geq 0.7$. However, the classic

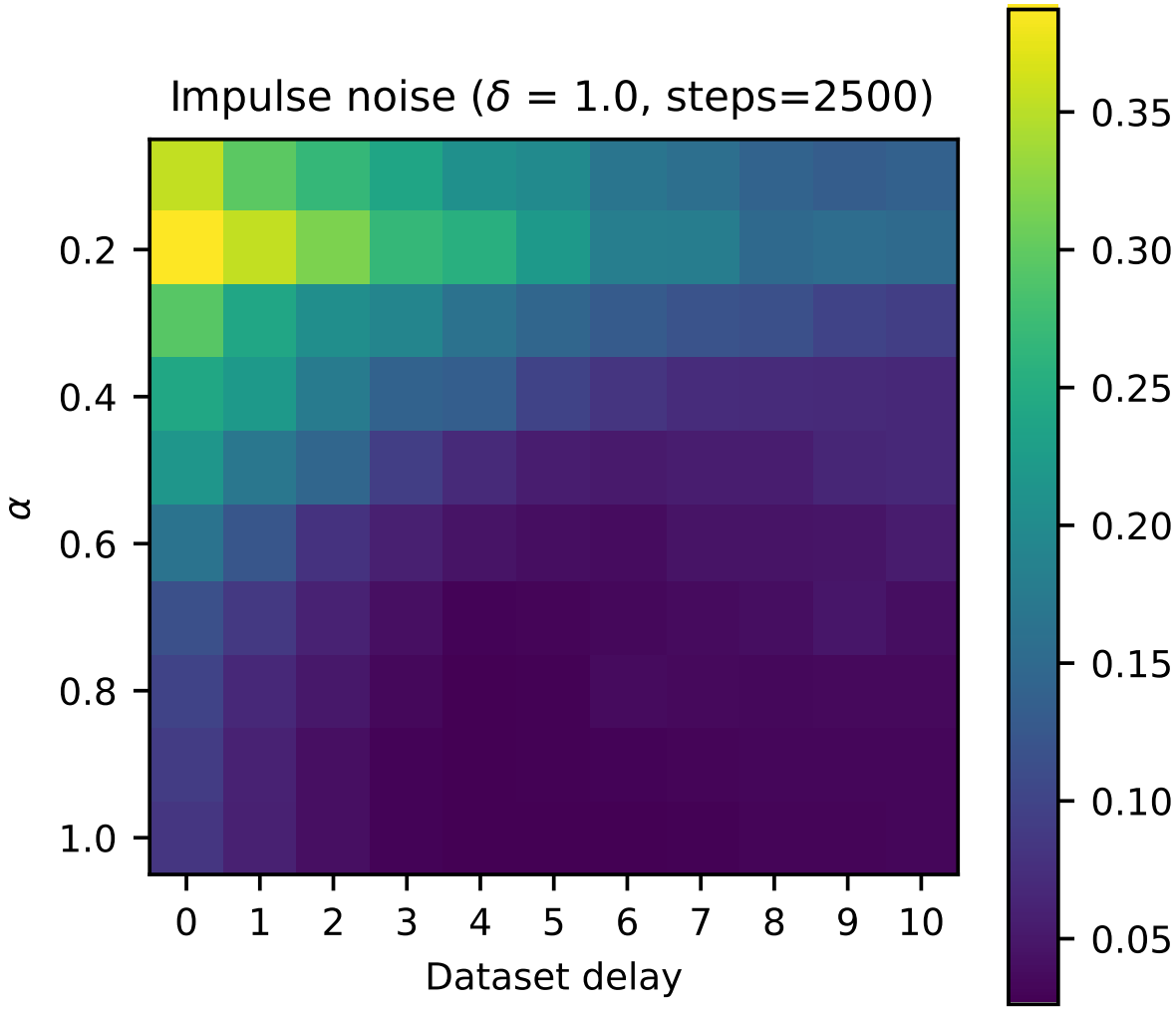


Figure 17 – Heat-map of the performance(NRMSE) over different values for α and the timesteps delay d parameter.

random ESN presents much worse performance than Wiener filter for small values of α until $\alpha = 0,95$. When $\alpha > 0.95$, *random ESN* has similar performance to Wiener filter. Therefore, for the next experiments, the delay will be fixed in 4 timesteps and the $\alpha = 0.95$.

Now checking the effect by varying the number of clusters in the reservoir. In this case, it is convenient to study only *CESNs*. Figure 19 shows that, for both *Barabási-Albert CESN* and *Erdös-Rényi CESN*, good performance can be achieved when the reservoir has a large number of communities. From the same Figure, note that *Barabási-Albert CESN* is much stable than *Erdös-Rényi CESN*.

The choice of the mixture level parameter in the CESNs does not impact the performance of the *Barabási-Albert CESN*, but it has a huge impact on the Erdös-Rényi CESN as it can be observed on Figure 20. As the P_{in} becomes closer to 1, the NRMSE decreases. Interesting behavior of the CESN methods that repeats in this task.

For the *Deep ESN* and the *Deep CESNs*, their performance is evaluated by varying

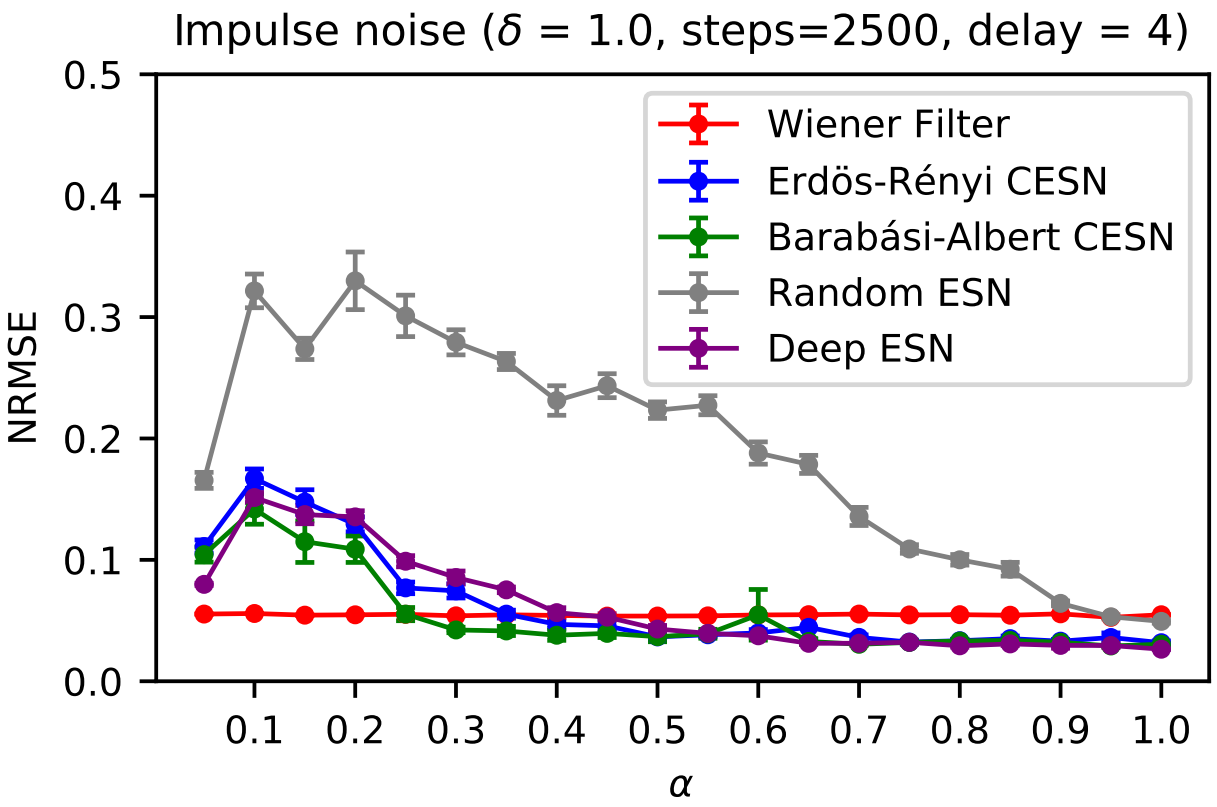


Figure 18 – NRMSE (Normalized Rooted Mean Squared Error) as a function of parameter α.

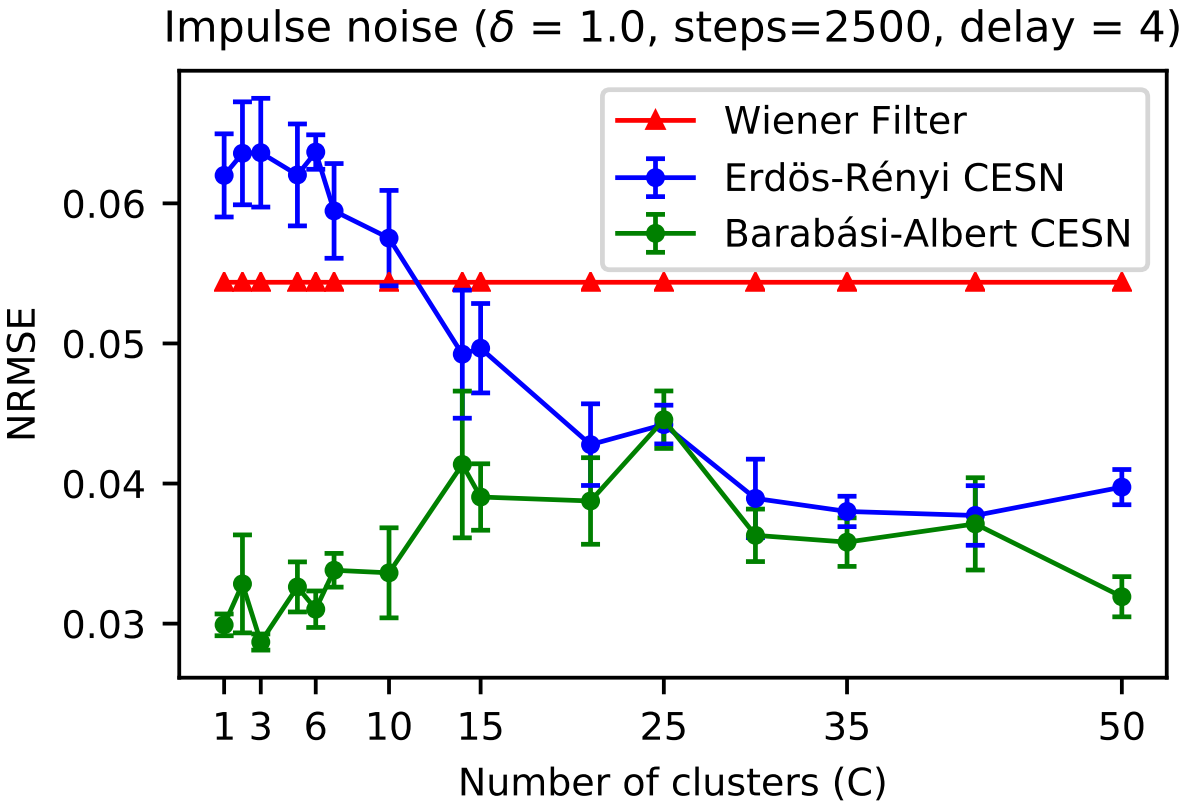


Figure 19 – NRMSE over different number of clusters, where the number of clusters tested is the following set: $C = 1, 2, 3, 5, 6, 7, 10, 14, 15, 21, 25, 30, 35, 42, 50$.

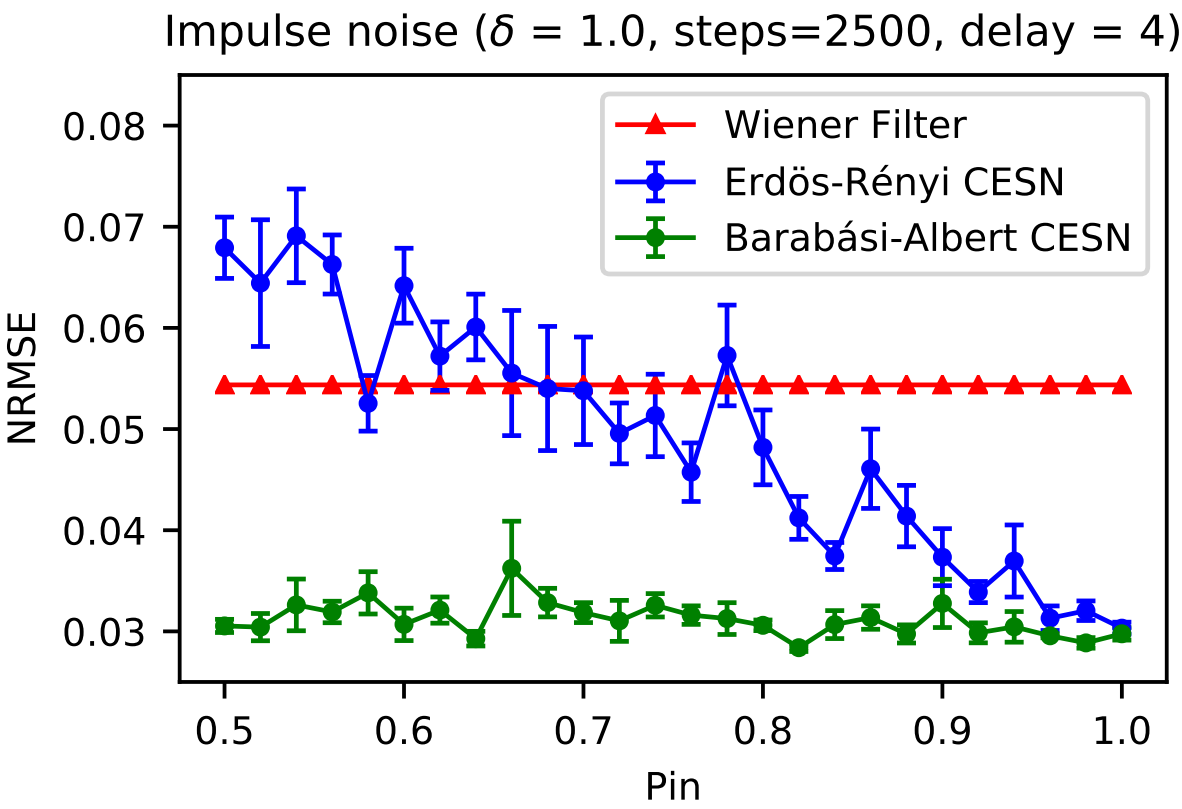


Figure 20 – NRMSE over different values for the P_{in} in Impulse noise reduction.

the number of layers(L). Basically, all the Deep ESN and Deep CESN under study present stable and much better performance than the Wiener filter as Figure 21 shows. Still in this Figure, note that the performance of Deep CESNs is qualitatively similar to Deep ESN when $L > 2$.

The next step is to observe the performance of ESNs and the Wiener filter with different noise levels. Figure 22 shows that all the methods, except the classic *Random ESN*, get much better results than the Wiener filter as the noise level increases. At the same time, *Random ESN* and Wiener filter present similar performance for all noise levels studied here.

Note that the parameters used in this experiment were the ones found when $\delta = 1.0$, therefore it might have some room for improvement, since in this case, the parameters are not optimized for each noise level.

The results in this section show that ESNs with clustered or layered reservoirs can improve the performance on the impulse noise reduction task. Moreover, the ESN and its extensions are capable to detect the noise in the signal, even though the pattern of the noise is not well defined, and it produces excellent results.

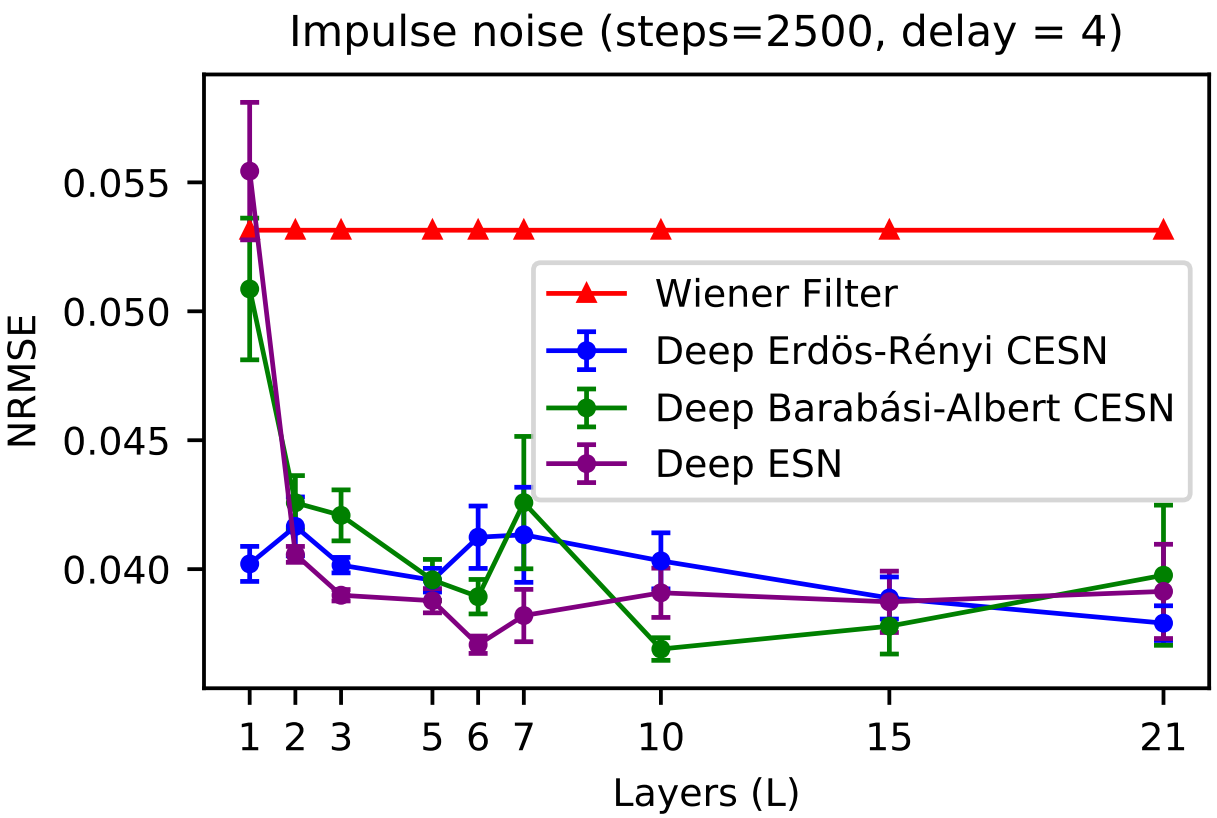


Figure 21 – NRMSE of the deep ESN and deep CESN as a function of the number of layers.

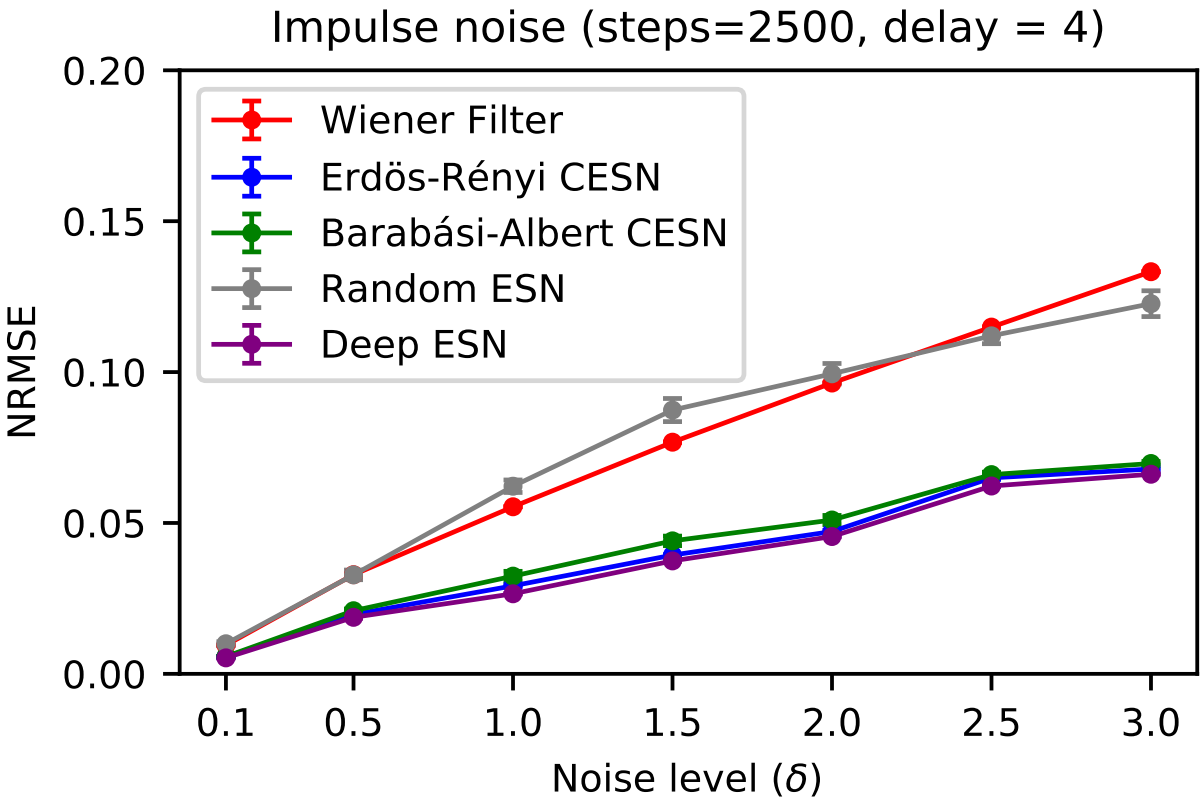


Figure 22 – NRMSE over different values for the noise level δ.

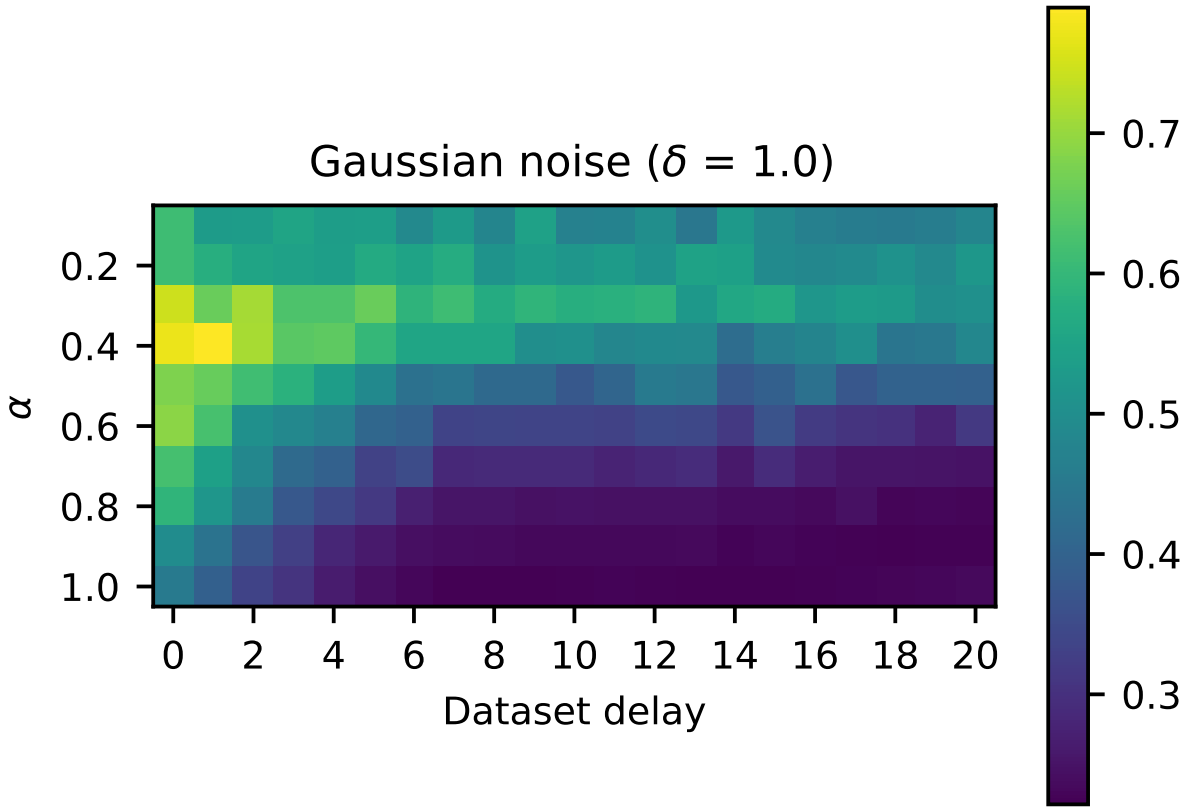


Figure 23 – Heat-map of the performance(NRMSE) over different values for α and the time steps delay d parameter.

4.3.2 Gaussian noise reduction

Another task to test is the Gaussian noise filtering. This task is particularly interesting because the Wiener filter is specially designed to solve this type of problem. Moreover, apart from filtering impulse noise, filtering Gaussian noise from a signal is harder for the ESNs since it contains more noisy data points.

Since it is a different task, another parameter optimization is required in this case. To find such parameters, we can use the same approach used in the section above. For the Gaussian noise, the required number of time-steps delayed in the dataset to achieve the best performance is greater than in the previous task as the Figure 23 shows. It is a little bit difficult to see by only looking at the heatmap, but the optimal value for the delay is 16, which is **4** times the value used for the Impulse noise task. This is due to the large amount of noisy data points.

Fixing the time-steps delay, the performance of ESNs varying α parameter can be studied. The Figure 24 shows that the Gaussian noise reduction errors of all the ESNs are larger when α is small, while *Barabási-Albert CESN*, *Erdös-Rényi CESN*, and *Deep ESN* get slightly better results than Wiener filter when $\alpha \geq 0.7$. The classic *Random ESN* presents slightly better performance than Wiener filter only when $\alpha > 0.9$.

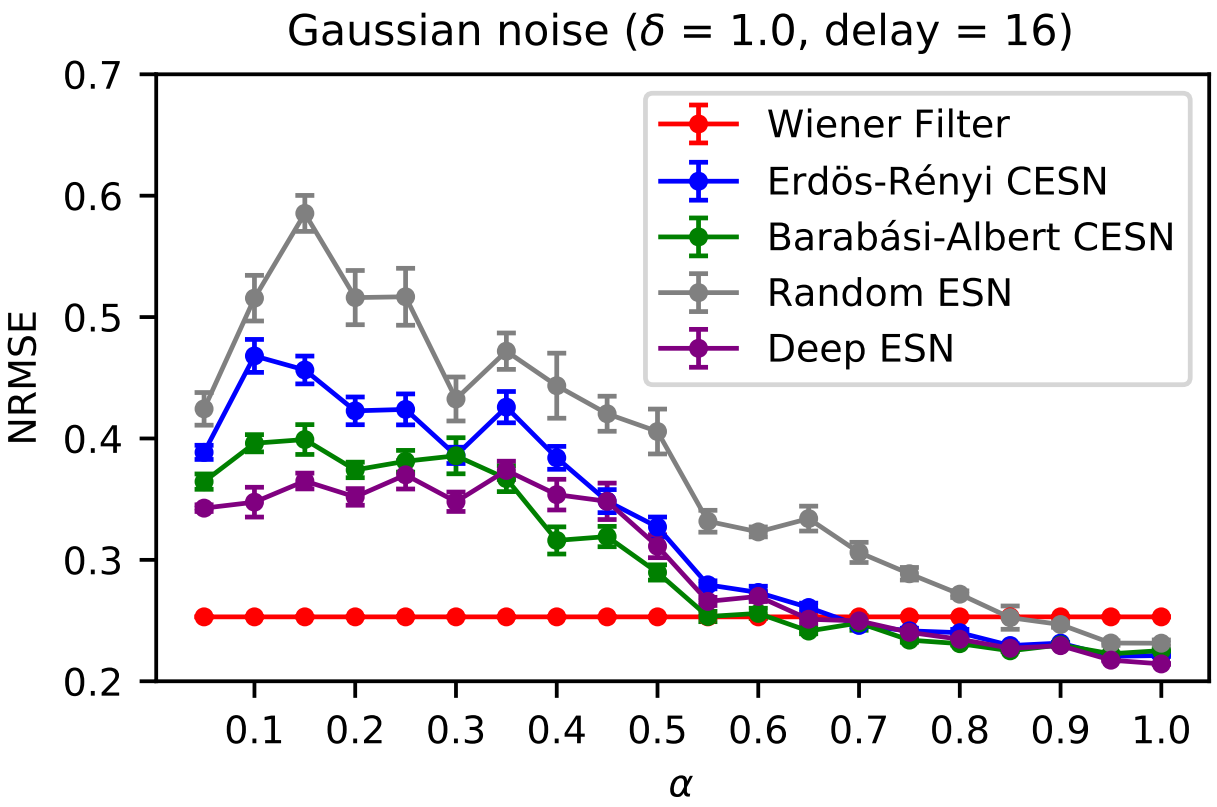


Figure 24 – NRMSE (Normalized Rooted Mean Squared Error) as a function of parameter α.

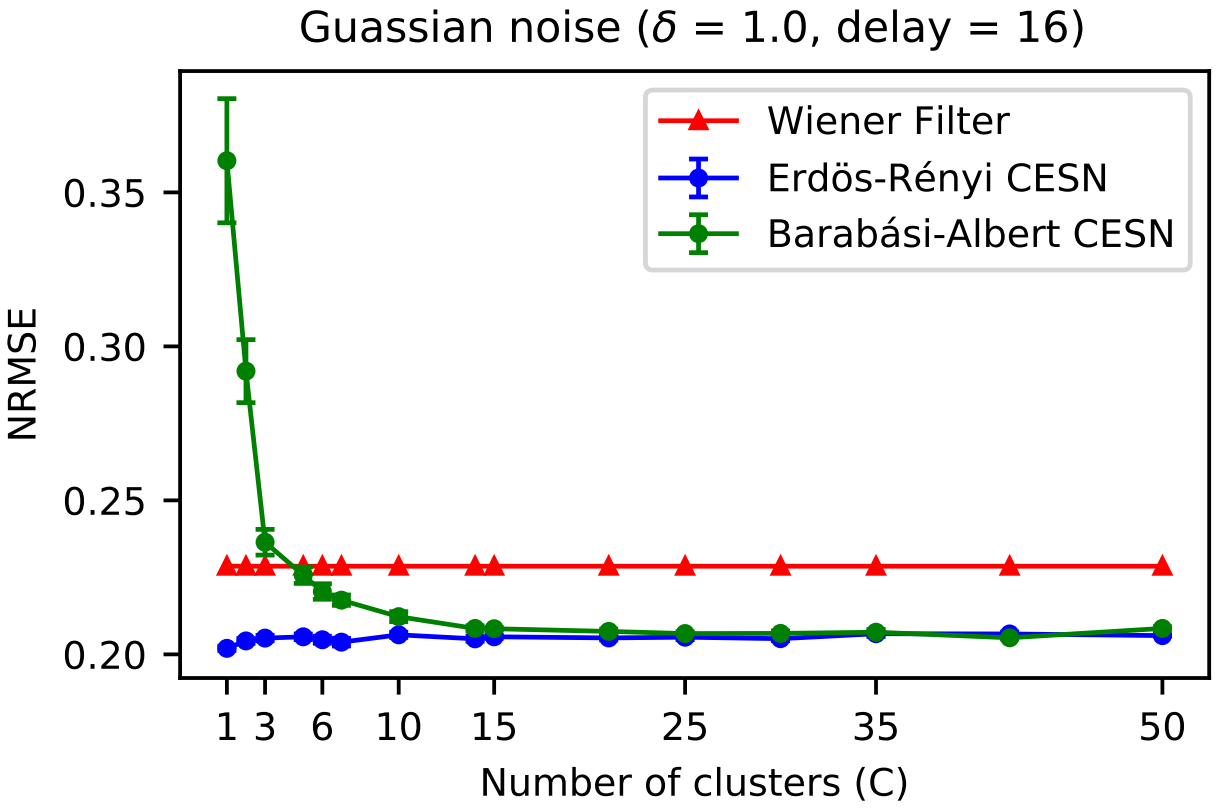


Figure 25 – NRMSE over number of clusters in Gaussian noise reduction.

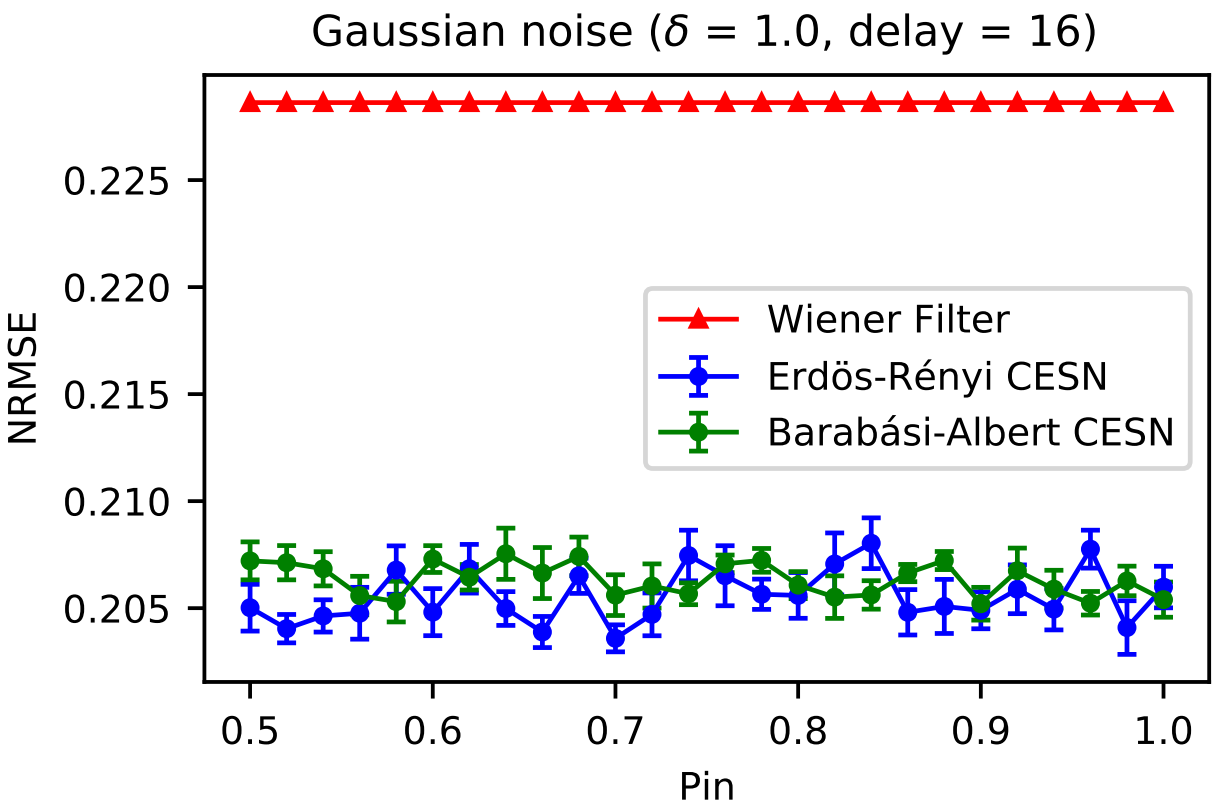


Figure 26 – NRMSE over different values for the P_{in} in Gaussian noise reduction.

The performance of the *Erdös-Rényi CESN* and the *Barabási-Albert CESN* can be checked by varying the number of clusters (C). Note in Figure 25 the number of clusters does not affect the performance for the *Erdös-Rényi CESN*, but it does for the *Barabási-Albert CESN* when the number of clusters is small. For both methods, as the number of clusters increases, the performance stabilizes, which is slightly better than the Wiener filter. The mixture level parameter for the CESNs when removing Gaussian noise from a signal does not impact the performance as the Figure 26 shows, therefore, this parameter can be chosen arbitrarily.

Then, a comparison between *Deep ESN*, *Deep Erdös-Rényi CESN*, and *Barabási-Albert CESN* is made on the Gaussian noise reduction task by varying the number of layers. Note in Figure 27 that all the Deep ESNs get slightly better results than Wiener filter and qualitative similar performance among themselves.

An analysis can be made on how different Gaussian noise levels affect the performance of the ESNs. Figure 28 shows that all the ESNs methods have a qualitative similar performance in this task, including the Wiener filter. Even though the difference between the methods is relatively small in this case, still worth to use the ESN extension because it offers a more flexible approach, and the NRMSE is smaller than the Wiener filter for $\delta \leq 2.5$ in the experiments. Moreover, optimizations can be made by fixing the noise level (δ), as it can be observed in the Figures 25 and 27.

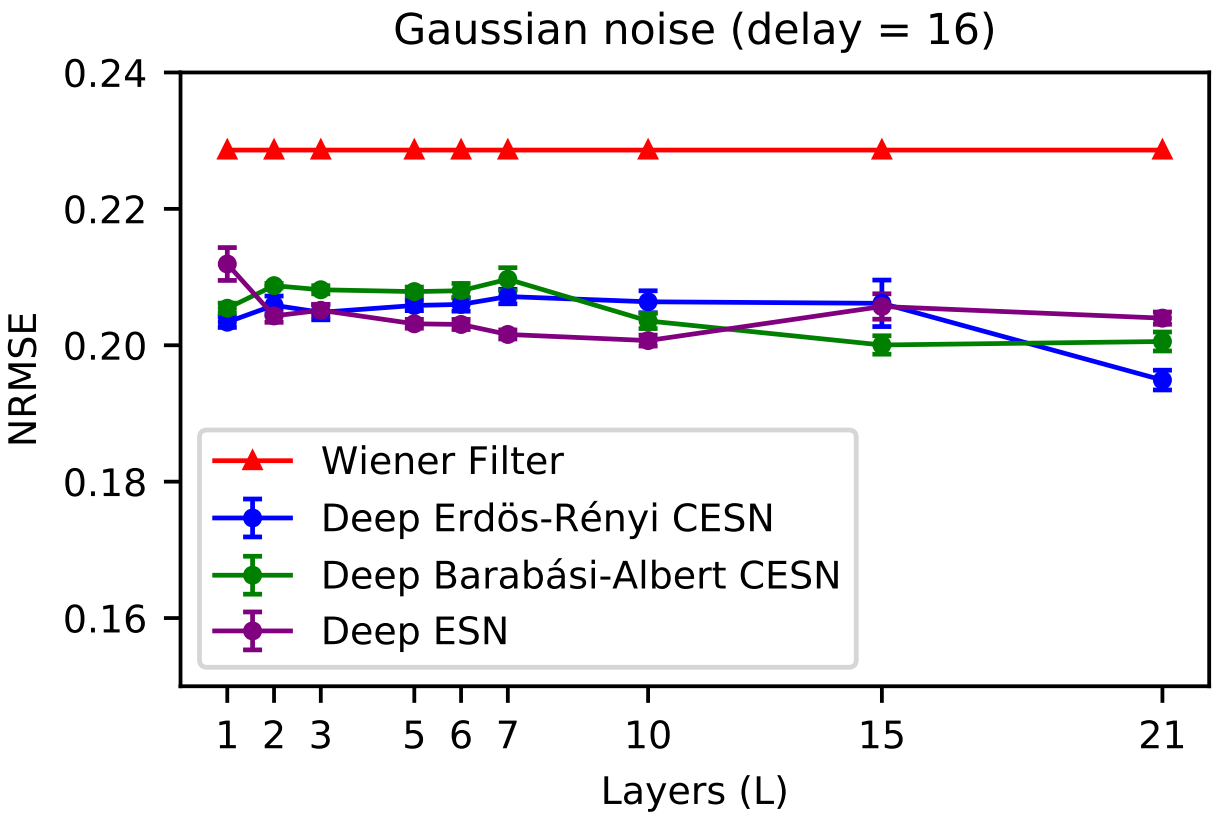


Figure 27 – NRMSE against number of layers in Gaussian noise reduction.

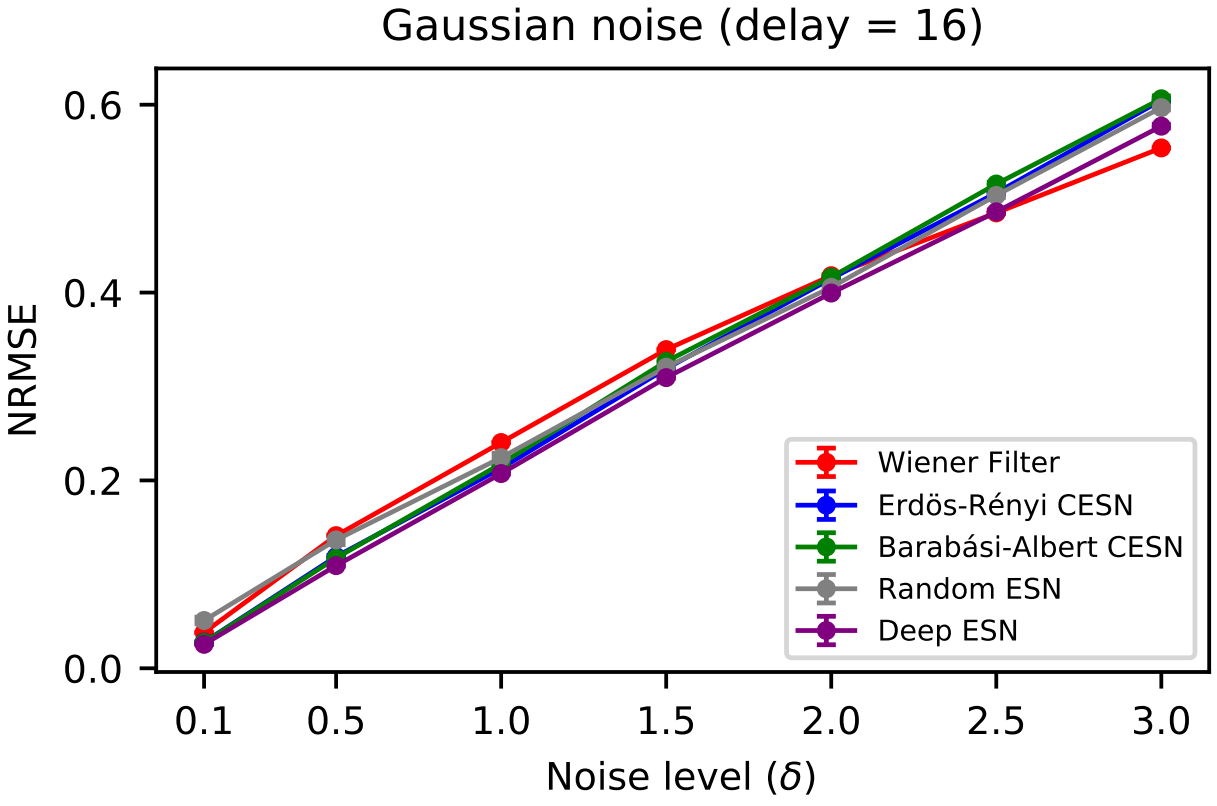


Figure 28 – NRMSE over different noise levels in Gaussian noise reduction.

4.3.3 ECG signal noise reduction

The ECG dataset is an opportunity to test the methods in real-world signals and check whether the ESN methods outperform the Wiener filter. To test whether the knowledge obtained in the previous experiments, the CESN's parameters found earlier will be used in this task. Since it is a harder task, the optimal parameters for the CESNs in the Gaussian noise will be used in this task.

First, the ESN parameters are fixed with the values obtained from the previous studies and vary the common parameter α for all types of ESNs. Figure 29 shows the performance of the ESN and its extensions versus the parameter α. The *CESN* methods and the *deep ESN* reach the best performance when $\alpha = 0.2$, on the other hand, the random ESN has the best performance for $\alpha = 0.4$. Note the deep ESN and the Erdös-Rényi CESN deliver more stable results when varying the α compared to the other methods.

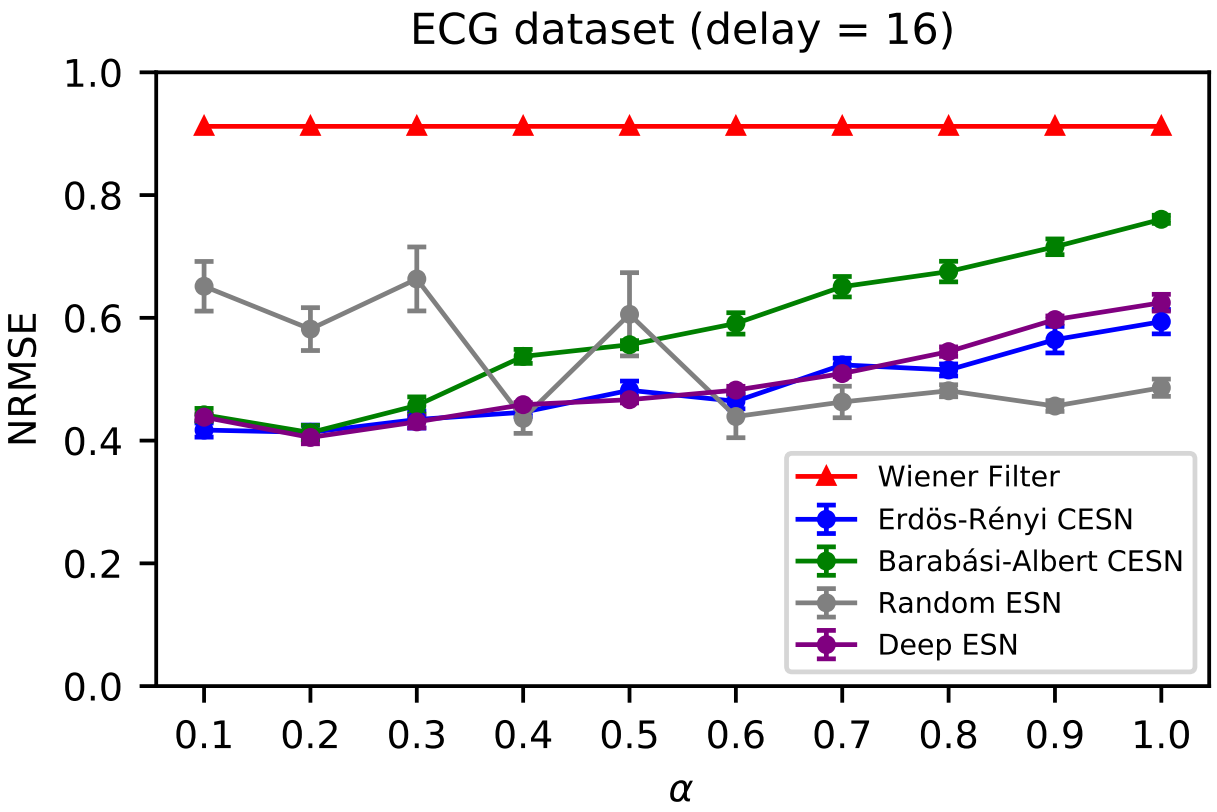


Figure 29 – NRMSE versus the parameter α.

This result is interesting because, for the Impulse noise and the Gaussian noise, the performance gets better as the α increases fixing the time-steps delay, but in this type of noise, smaller values of α yields better results. This means that when the network has a larger memory, i.e., more weight is given to past states, smaller is the error. The reason behind this is due to the different types of noise the signal contains.

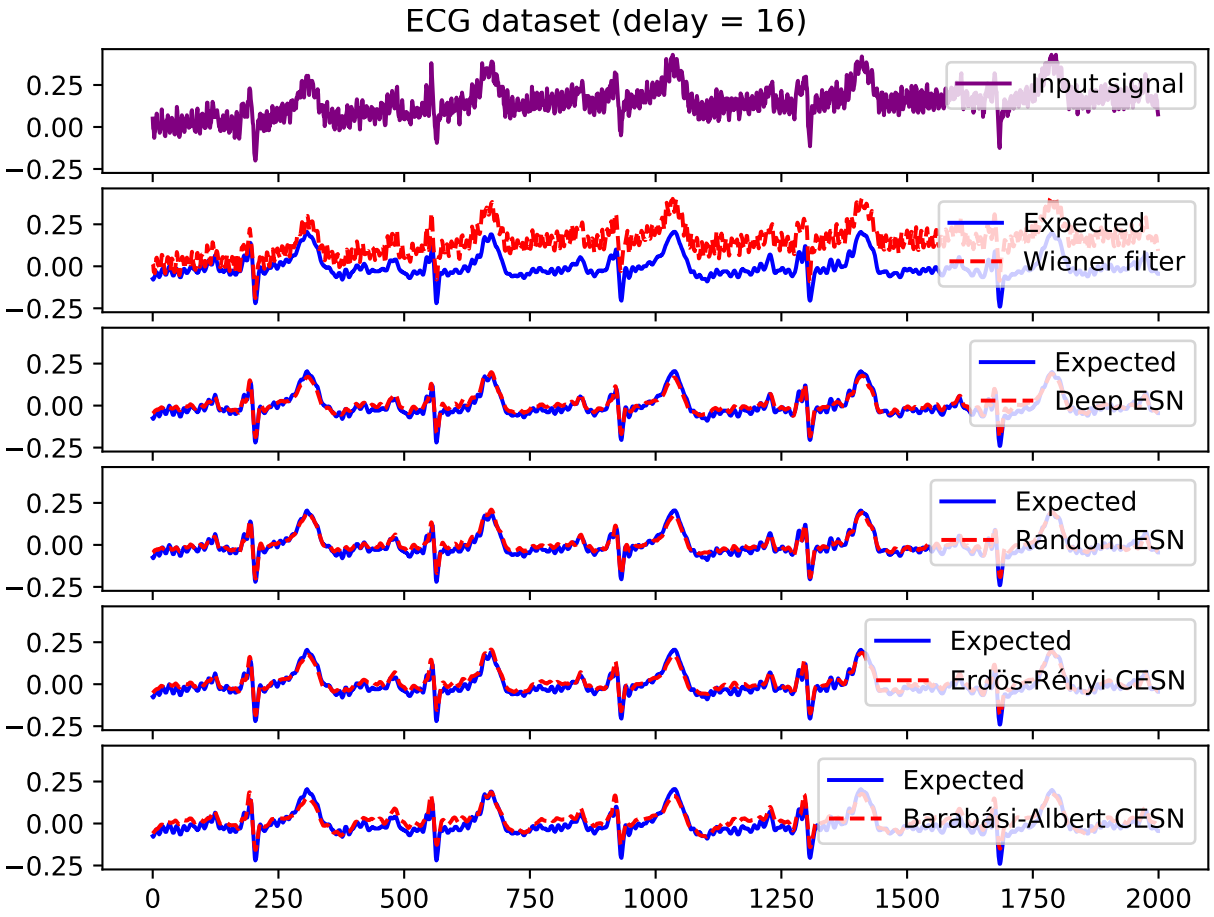


Figure 30 – The ECG signal with the noise and the filtered signals by Wiener filter and various ESNs.

A qualitative comparison between the ESN methods, and the Wiener filter can be observed in Figure 30. This Figure shows only a part of the output given by the methods. Whereas the Wiener filter is almost unable to detect the noise and produce the clean signal, the ESN and its extensions can filter the signal giving a good approximation of the clean signal.

Using the ESN methods with the optimal set of parameters for each of them, the NRMSE is decreased by more than half compared to the Wiener filter. This means that noises in real-world signals can vary a lot, i.e., they are not always Gaussian noises or impulse noises, which can be difficult for context-specific filters to remove these types of noises. Therefore, ESNs, specially *CESNs*, and *Deep ESNs* gives a robust solution for unknown noise reduction.

5

Conclusion

Past works using different tasks as referenced before has also shown the organization of the network in clusters improves the overall performance (APPELTANT et al., 2011; WEN; LI; LI, 2015; MARTENS et al., 2017; KAWAI; PARK; ASADA, 2019). Throughout this work several architectures of ESNs have been explored, starting with the most simple one, which is an ESN with a random network inside the reservoir, and then expanding this model to a little bit more sophisticated model that replaces the random network with a clustered network built based on complex network models. The model was expanded even more by using a deep ESN where each layer is a clustered network rather than a random network. Moreover, this work proposed to submit the ESN and its extensions to three different types of tasks to check if the performance given by the clustered network can be noticed when solving different types of problems.

The results in the experiments presented here show the extensions proposed can yield better performance than the random ESN. One key point to achieve good performance with the extensions is the parameter optimization phase. After the optimization phase, a good performance can be achieved using the ESN extensions, which are the CESNs, the deep ESNs, and the deep CESNs.

In the observer problem, due to the low NRMSE that the ESN gives (around 10^{-4}), it is hard to measure whether the extensions are better than the classic model of the ESN. Although, there is a small improvement of the CESNs over the ESNs in this task. In the frequency filtering task, the difference in the performance between the ESNs extensions and the classic model becomes larger. In this task, compared to the random ESN, one can achieve (**30** to **40**) percent smaller NRMSE.

Moreover, in the novel approach for the ESNs, the noise filtering task, the ESN and its extensions present a robust approach for signal noise reduction. The reason behind this is that ESNs learn from the input-output of the data, then the performance depends much less on the types of noise. The result given by this approach can shed a light on how ESNs can be used to solve different types of problems providing good results. When

submitted against real-world signals, the ESNs can be a good approach since it can learn from different types of noise and remove them easily. Since the Wiener filter was designed for a specific type of noise, it may have a bad performance for real-world signals, where the types of noises are usually unknown.

In all tasks, a notable superiority of the *Deep ESN* and the *CESNs* over the classic ESNs in general cases, indicating that the organization of reservoirs in clustered or layered networks can improve the learning performance of ESNs. These improvements are probably due to the selective nature of the clusters in the reservoir, where different clusters are responsible to capture different signal properties.

This point deserves to be further investigated. Therefore, as future work, we will analyze the collective dynamics of neurons in the reservoir, such as frequency synchronization, phase synchronization, or generalized synchronization, and try to find out the correlation between the performance of ESNs and the firing patterns of neurons in the reservoir.

References

AKIKI, T. J.; ABDALLAH, C. G. Determining the hierarchical architecture of the human brain using subject-level clustering of functional networks. *Sci Rep*, v. 9, p. 19290, 2019.

ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, v. 74, n. 1, p. 47–97, jan. 2002.

A.P., L. T. N. Biometric human identification based on electrocardiogram. *Proc. XII-th Russian Conference on Mathematical Methods of Pattern Recognition*, p. 387–390, June 2005.

APPELTANT, L. et al. Information processing using a single dynamical node as complex system. *Nature Communications*, Nature Publishing Group, v. 2, p. 1–6, 9 2011. ISSN 2041-1723.

BARABÁSI, A. L.; ALBERT, R. Emergence of scaling in random networks. *Science*, v. 286, n. 5439, p. 509–512, 1999.

BERRY, M. J.; TKACIK, G. Clustering of neural activity: A design principle for population codes. *Frontiers in Computational Neuroscience*, v. 14, p. 20, 2020. ISSN 1662-5188. Disponível em: <https://www.frontiersin.org/article/10.3389/fncom.2020.00020>.

BOCCALETTI, S. et al. Complex networks: Structure and dynamics. *Physics Reports*, v. 424, n. 4, p. 175 – 308, 2006. ISSN 0370-1573. Disponível em: <http://www.sciencedirect.com/science/article/pii/S037015730500462X>.

BOCCATO, R. A. L.; ZUBEN, F. J. V. Self-organization and lateral interaction in echo state network reservoirs. *Neurocomputing*, v. 138, p. 297—309, 2014.

BOLLOBAS, B. et al. Directed scale-free graphs. In: *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. [s.n.], 2003. p. 132–139. Disponível em: <https://www.microsoft.com/en-us/research/publication/directed-scale-free-graphs/>.

BREDE, M. Networks—an introduction. mark e. j. newman. (2010, oxford university press.). *Artificial life*, v. 18, p. 241–2, 02 2012.

CARMICHAEL, Z.; SYED, H.; KUDITHIPUDI, D. Analysis of wide and deep echo state networks for multiscale spatiotemporal time series forecasting. In: *Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop*. New York, NY, USA: Association for Computing Machinery, 2019. (NICE '19). ISBN 9781450361231. Disponível em: <https://doi.org/10.1145/3320288.3320303>.

CHATZIS, S. P.; DEMIRIS, Y. Echo state gaussian process. *IEEE Transactions on Neural Networks*, v. 22, p. 1435—-1445, 2011.

References

CHEN, J. et al. New insights into the noise reduction wiener filter. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 14, p. 1218 – 1234, 2006.

CHEN, X. et al. Brain-wide organization of neuronal activity and convergent sensorimotor transformations in larval zebrafish. *Neuron*, v. 100, n. 4, p. 876 – 890.e5, 2018. ISSN 0896-6273. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0896627318308444>.

COHEN, R.; HAVLIN, S. *Complex Networks: Structure, Robustness and Function*. [S.l.: s.n.], 2010. ISBN 978-0521841566.

CORNELIS, B.; MOONEN, M.; WOUTERS, J. Performance analysis of multichannel wiener filter-based noise reduction in hearing aids under second order statistics estimation errors. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*, v. 19, n. 5, p. 1368–1381, 2011.

COSTA F. A. RODRIGUES, G. T. L. F.; BOAS, P. R. V. Characterization of complex networks: A survey of measurements. In: *Advances in Physics*. [S.l.: s.n.], 2006. v. 56, p. 167–242.

DENG, L.; YU, D. *Deep Learning: Methods and Applications*. [S.l.], 2014. Disponível em: <https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/>.

DENG, Z.; ZHANG, Y. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks*, v. 18, n. 5, p. 1364–1375, Sep. 2007. ISSN 1045-9227.

DETTORI, S. et al. Deep echo state networks in industrial applications. In: MAGLOGIANNIS, I.; ILIADIS, L.; PIMENIDIS, E. (Ed.). *Artificial Intelligence Applications and Innovations*. Cham: Springer International Publishing, 2020. p. 53–63. ISBN 978-3-030-49186-4.

ERDÖS, P.; RENYI, A. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, v. 12, p. 261–267, 1961.

FORTUNATO, S. Community detection in graphs. *Physics Reports*, v. 486, p. 75–174, 2010.

GALLICCHIO, C.; MICHELI, A. Deep reservoir computing: A critical analysis. In: *ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. [S.l.: s.n.], 2016. Available from http://www.i6doc.com/en/.

GLEISER, P. M.; SPOORMAKER, V. I. Modelling hierarchical structure in functional brain networks. *Phil. Trans. R. Soc. A*, v. 368, p. 5633–5644, 2010.

GOLDBERGER, A. L. et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, v. 101, n. 23, p. e215–e220, 2000. Circulation Electronic Pages: https://physionet.org/content/ecgiddb/1.0.0/ https://doi.org/10.13026/C2J01F.

HAGMANN, P. et al. Mapping the structural core of human cerebral cortex. *PLoS Biol.*, v. 6, p. e156, 2008.

References

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HAZAN, H.; MANEVIT, L. M. Topological constraints and robustness in liquid state machines. *Expert Systems with Applications*, p. 1597—-1606, 2012.

JAEGER, H. 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*, v. 148, 01 2001.

JAISWAL, A.; UPADHYAY, J.; SOMKUWAR, A. Image denoising and quality measurements by using filtering and wavelet based techniques. *AEU - International Journal of Electronics and Communications*, v. 68, n. 8, p. 699 – 705, 2014. ISSN 1434-8411. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1434841114000442>.

KAWAI, Y.; PARK, J.; ASADA, M. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, v. 112, p. 15 – 23, 2019. ISSN 0893-6080. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608019300115>.

LACY, S. E.; SMITH, S. L.; LONES, M. A. Using echo state networks for classification: A case study in parkinson's disease diagnosis. *Artificial Intelligence in Medicine*, v. 86, p. 53 – 59, 2018. ISSN 0933-3657. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0933365717303482>.

LATORA, V.; NICOSIA, V.; RUSSO, G. *Complex Networks: Principles, Methods and Applications*. [S.l.]: Cambridge University Press, 2017.

LEE, Y.; KASSAM, S. Generalized median filtering and related nonlinear filtering techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 33, n. 3, p. 672–683, 1985.

LI, X. et al. A priori data-driven multi-clustered reservoir generation algorithm for echo state network. *PLOS ONE*, Public Library of Science, v. 10, n. 4, p. 1–15, 04 2015. Disponível em: <https://doi.org/10.1371/journal.pone.0120750>.

LIN, X.; YANG, Z.; SONG, Y. The application of echo state network in stock data mining. In: WASHIO, T. et al. (Ed.). *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 932–937. ISBN 978-3-540-68125-0.

LIN, X.; YANG, Z.; SONG, Y. Short-term stock price prediction based on echo state networks. *Expert Syst. Appl.*, v. 36, p. 7313–7317, 2009.

LU, Z. et al. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, v. 27, n. 4, p. 041102, 2017. Disponível em: <https://doi.org/10.1063/1.4979665>.

MARTENS, M. et al. Brain network clustering with information flow motifs. *Applied Network Science*, v. 2, 08 2017.

NEWMAN, M. E. J. Fast algorithm for detecting community structure in networks. *Physical Review*, v. 66133, n. 1–5, 2004.

NEWMAN, M. E. J. *Networks: An Introduction*. [S.l.]: Oxford University Press, 2010.

References

NEWMAN, M. E. J.; BARABÁSI, A.-L.; WATTS, D. J. *The Structure and Dynamics of Networks*. [S.l.]: Princeton University Press, 2006.

NEWMAN, M. E. J.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical Review*, v. 69, p. 026113–1, 2004.

PATHAK, J. et al. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, v. 27, n. 12, p. 121102, 2017. Disponível em: <https://doi.org/10.1063/1.5010300>.

PONGHIRAN, W.; SRINIVASAN, G.; ROY, K. Reinforcement learning with low-complexity liquid state machines. *Frontiers in Neuroscience*, v. 13, p. 883, 2019. ISSN 1662-453X. Disponível em: <https://www.frontiersin.org/article/10.3389/fnins.2019.00883>.

QUILES, M. G. et al. *Particle competition for complex network community detection*. Chaos (Woodbury: [s.n.], 2008. 033107–1 p.

SCHUBERT, F.; GROS, C. Local homeostatic regulation of the spectral radius of echo-state networks. *bioRxiv*, Cold Spring Harbor Laboratory, 2020. Disponível em: <https://www.biorxiv.org/content/early/2020/07/25/2020.07.21.213660>.

SILVA, T. C.; ZHAO, L. Stochastic competitive learning in complex networks. *IEEE Transactions on Neural Networks and Learning Systems*, v. 23, p. 385–398, 2012.

SILVA, T. C.; ZHAO, L. Machine learning in complex networks. In: *Machine Learning in Complex Networks*. [S.l.]: Springer, 2016. p. 10–1007.

SKARDA, C. A.; FREEMAN, W. J. How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, v. 10, p. 161–195, 1987.

STROGATZ, D. J. W. e S. H. Collective dynamics of smallworld networks. *Nature*, v. 393, p. 440–442, 1998.

STROGATZ, S. H. Exploring complex networks. *Nature*, v. 410, p. 268–276, 2001.

TRAPPENBERG, T. P. *Fundamentals of Computational Neuroscience*. [S.l.]: Oxford University Press, 2002.

VASEGHI, S. V. *Advanced Digital Signal Processing and Noise Reduction*. [S.l.]: John Wiley & Sons, Ltd, 2001.

WEN, G.; LI, H.; LI, D. An ensemble convolutional echo state networks for facial expression recognition. In: *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*. [S.l.: s.n.], 2015. p. 873–878. ISSN 2156-8111.

YU, P.; MIAO, L.; JIA, G. Clustered complex echo state networks for traffic forecasting with prior knowledge. In: *2011 IEEE International Instrumentation and Measurement Technology Conference*. [S.l.: s.n.], 2011. p. 1–5.

ZAMORA-LÓPEZ, G.; BRASSELET, R. Sizing complex networks. *Communications Physics*, v. 2, p. 144, 11 2019.

Zheng, K. et al. Long-short term echo state network for time series prediction. *IEEE Access*, v. 8, p. 91961–91974, 2020.