Machine Learning (2022) 111:2885–2904
https://doi.org/10.1007/s10994-022-06135-6

Clustered and deep echo state networks for signal noise reduction

Laercio de Oliveira Junior[1,2] · **Florian Stelzer**[3,4,5] · **Liang Zhao**[1]

Received: 1 May 2021 / Revised: 11 November 2021 / Accepted: 6 February 2022 /
Published online: 11 March 2022

Abstract
Echo State Networks (ESNs) are Recurrent Neural Networks with fixed input and internal (hidden) weights, and adaptable output weights. The hidden part of an ESN can be considered as a discrete-time dynamical system, called *reservoir*. In classical ESNs, the internal connections are obtained from an Erdős-Rényi graph. A recent study proposed ESNs with clustered adjacency matrices (CESNs), where the clusters are either Erdős-Rényi graphs or Barabási-Albert-like graphs. In this work, we investigate the effectiveness of CESNs and apply them for signal denoising. In addition, we introduce and study deep CESNs with multiple clustered layers. We found that CESNs and deep CESNs can compete with deep ESNs for all tasks that we considered.

Keywords Echo state networks · Reservoir computing · Complex networks · Noise reduction

Editor: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann.

✉ Laercio de Oliveira Junior
laercio00@outlook.com

Florian Stelzer
florian.stelzer@ut.ee

Liang Zhao
zhao@usp.br

[1] Faculty of Philosophy, Science and Letters at Ribeirão Preto, University of São Paulo (USP), Ribeirão Preto, SP, Brazil

[2] Meta Platforms Inc, London, UK

[3] Institute of Computer Science, University of Tartu, Tartu, Estonia

[4] Department of Mathematics, Humboldt University of Berlin, Berlin, Germany

[5] Institute of Mathematics, Technical University of Berlin, Berlin, Germany

1 Introduction

Artificial Neural Networks (ANNs) are computing systems consisting of connected units, commonly called *neurons* or *nodes*. ANNs are inspired by biological neural systems (McCulloch and Pitts 1943; Hebb 1949) and constitute a class of powerful machine learning tools for a wide range of applications (Lecun et al. 2015; Schmidhuber 2015; Abiodun et al. 2018; Tealab 2018; Goodfellow et al. 2016). Recurrent Neural Networks (RNNs) are ANNs, which are useful to process sequential data. They contain at least three layers: an input layer, representing the input data, a hidden layer with recurrent connections, and an output layer. A given sequence of input data is fed into the RNN step by step. The recurrent connections of the hidden layer ensure that the network keeps information about past input elements.

A standard method to train RNNs is gradient descent, where the gradient is computed by backpropagation through time (Robinson and Fallside 1987; Mozer 1989; Goodfellow et al. 2016). This training method can suffer from vanishing or exploding gradients (Mozer 1989; Kolen and Kremer 2001; Goodfellow et al. 2016), which can be avoided by using advanced RNN models, such as long short-term memories (LSTMs) (Hochreiter and Schmidhuber 1997) or gated recurrent units (GRUs) (Cho et al. 2014). LSTMs and GRUs perform well, even on challenging tasks (Sak et al. 2014; Uhlich et al. 2017; Takahashi et al. 2018; Ravanelli et al. 2018), but are sophisticated and rather difficult to implement.

Another, earlier and elegantly simple approach to overcome the problem of vanishing or exploding gradients, are Echo State Networks (ESNs) (Jaeger 2001, 2002; Lu et al. 2017). ESNs are RNNs with fixed, randomly chosen input and internal weights and adaptable output weights. Since only the output weights are trained, there is no need to employ gradient descent. Instead, one can use a simple regression method. Fixed input and internal weights come at the price that ESNs perform worse than advanced methods, such as LSTMs or GRUs, on challenging tasks. But for solving rather simple or moderately challenging tasks, ESNs should still be considered as a possible solution due to their simplicity and their fast training process. ESNs have been successfully applied to chaotic signal prediction (Lu et al. 2017; Pathak et al. 2017), signal separation (Krishnagopal et al. 2020), stock price prediction (Lin et al. 2009), and for the simulation of cardiac electrical waves (Zimmermann and Parlitz 2018).

Furthermore, ESNs are a subclass of *reservoir computing*, which is the principle of employing an input driven dynamical system, called *reservoir*, with an adaptable readout transformation for machine learning problems. The hidden layer of an ESN can be considered as a discrete-time dynamical system and serves as the reservoir; the ESN's output layer realizes the readout transformation. The ESN is the reservoir computing concept that was introduced first. Meanwhile, there are further concepts, e.g., time-delay reservoir computing (Appeltant et al. 2011; Larger et al. 2012; Schumacher et al. 2013; Brunner et al. 2013).

Classical ESNs possess only one hidden layer, which obtains its connections from an Erdős-Rényi graph (Erdös and Renyi 1961). In addition, *deep ESNs* have been introduced (Gallicchio and Micheli 2016; Gallicchio et al. 2018; Gallicchio and Micheli 2020; Dettori et al. 2020). Deep ESNs possess multiple hidden layers, which are internally recurrent, but coupled to each other in a feed-forward way.

In spite of decent results obtained from ESNs with simple Erdős-Rényi topology, there is an interest in employing complex network topologies for further improvements. Complex networks are graphs with nontrivial connection patterns and provide a powerful tool

for modeling complex systems by unifying spatial, topological, functional, and evolutionary properties. One of their salient features is the presence of communities forming a clustered structure. While a community is a group of densely connected nodes, the connections between nodes of different communities are sparse. Interestingly, such clustered structures have been found in, e.g., the brains of humans and animals (Martens et al. 2017; Berry and Tkacik 2020). Data on both anatomical and functional connectomes have shown a small-world structure with highly clustered modules at different scales (Akiki and Abdallah 2019; Gleiser and Spoormaker 2010; Hagmann et al. 2008).

Various concepts of *clustered ESNs* (CESNs), i.e., ESNs with a reservoir described by a complex network with community structure, have been proposed, including geometrically motivated (Deng and Zhang 2007), data driven (Li et al. 2015), and knowledge-based (Yu et al. 2011) methods. The recent work (Oliveira et al. 2020) introduced two further types of CESNs, where the clusters are defined by equally sized blocks in the adjacency matrix. These blocks are generated using the Erdős-Rényi model or a Barabási-Albert-like model, respectively. However, only simple time-series prediction and frequency filtering problems on artificial datasets have been considered to evaluate the effectiveness of these methods.

In this article, we show that the CESNs from Oliveira et al. (2020) can also be applied for signal denoising. Specifically, we reconstruct signals from their noisy versions, which are corrupted by Gaussian, impulse, or ECG noise. To demonstrate the effectiveness of CESNs, we compare them to classical ESNs and to the Wiener filter (Wiener 1949). Furthermore, we introduce deep CESNs, i.e., ESNs with multiple clustered layers.

Our results show that CESNs and deep CESNs perform significantly better than the Wiener filter and classical ESNs. ESNs, in particular CESNs and deep CESNs, are a robust method providing a decent performance in signal denoising.

2 Review of echo state networks

In this section, we provide a general definition of ESNs and explain how to train them using ridge regression. We describe classical ESNs with a random Erdős-Rényi network topology as introduced in Jaeger (2001), which we call *random ESN* throughout this article. Further, we describe deep ESNs (Gallicchio and Micheli 2016, 2020).

We use the term *adjacency matrix* for the matrix indicating the number of connections between two nodes of a directed graph (including self-connections). The entries of an adjacency matrix are in general non-negative integers. In ESNs there is at most one connection per direction between two nodes, i.e., the reservoir's adjacency matrix contains only ones and zeros. Matrices describing weighted connections are referred to as *weight matrix*. We call the matrix of the weighted connections inside the reservoir *hidden weight matrix*.

2.1 General definition of echo state networks

ESNs are RNNs consisting of an input layer $u(t)$, an output layer $s(t)$, and one or multiple hidden layers defining the reservoir $x(t)$, where $t = 0, 1, \ldots, t_{\max}$ is a time index.

The input layer $u(t)$ represents a sequence of input vectors with dimension $M \in \mathbb{N}$, which are inserted step by step into the reservoir $x(t)$. The reservoir itself is a high-dimensional nonlinear discrete-time dynamical system with states in $\mathbb{R}^N$, where $N \gg M$. The high dimensionality ensures that a sufficiently large number of input elements can be stored (Jaeger 2002).

The dynamics of the reservoir is described by the equation

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

where $0 < \alpha \leq 1$ is a leakage rate, $A \in \mathbb{R}^{N\times N}$ is the hidden weight matrix, $W^{\text{in}} \in \mathbb{R}^{N\times M}$ is the input weight matrix, $u(t) \in \mathbb{R}^M$ is the input vector at time t, and $\gamma\mathbf{1}$ is a bias vector, where $\gamma \in \mathbb{R}$ is the bias strength and $\mathbf{1} \in \mathbb{R}^N$ denotes a vector filled with ones. The leakage rate α is a fixed hyperparameter that controls how fast the system state changes. If the ESN contains multiple hidden layers, they can all be described by just a single hidden weight matrix A with a certain block structure. See Sect. 2.5 about deep ESNs for details.

The output of an ESN is a time series with elements of dimension $P \in \mathbb{N}$ given by the equation

$$\hat{s}(t) = W^{\text{out}}x(t) + c, \tag{2}$$

where $W^{\text{out}} \in \mathbb{R}^{P\times N}$ is the output weight matrix and $c \in \mathbb{R}^P$ is a vector containing output bias weights.

All types of ESNs considered in this article differ only in the structure of the hidden weight matrix A. Consequently, they can all be trained with the same training procedure.

2.2 Training process

ESNs are applied for supervised sequential machine learning problems. That is, in addition to the sequence of inputs $u(t)$, we are given a sequence of target vectors $s(t) \in \mathbb{R}^P$. The training process aims to find a good estimator $\hat{s}(t)$ to approximate the target sequence $s(t)$. This is done by fitting the variables W^{out} and c using a regression method.

The work Lu et al. (2017) explains how to optimize W^{out} and c by ridge regression. We use the same method for our numerical studies presented below. For the purpose of self-containedness, we repeat the explanations from Lu et al. (2017).

For the training process, we assign the negative time index $t_0 < 0$ to the initial state $x(t_0)$ of the system. We solve Eq. (1) iteratively, to generate the system states $x(t)$. Only the states $x(1), \ldots, x(t_{\max})$ need to be stored. The period up to $t = 0$ is called *initial wash out phase* and ensures that the training results do not depend on the initial state (Jaeger 2001; Lu et al. 2017).

To perform the actual regression, we calculate the element-wise mean $\bar{x} \in \mathbb{R}^N$ of the reservoir states $x(1), \ldots, x(t_{\max})$ and the element-wise mean $\bar{s} \in \mathbb{R}^P$ of the target states $s(1), \ldots, s(t_{\max})$, i.e.,

$$\bar{x} = \frac{1}{t_{\max}} \sum_{t=1}^{t_{\max}} x(t), \quad \bar{s} = \frac{1}{t_{\max}} \sum_{t=1}^{t_{\max}} s(t). \tag{3}$$

Let δX be a matrix with N rows and $t_{\max}$ columns, where the tth column is the vector $x(t) - \bar{x} \in \mathbb{R}^N$. Analogously, let δS be a matrix with P rows and $t_{\max}$ columns, where the tth column is the vector $s(t) - \bar{s} \in \mathbb{R}^P$. Then, the output weight matrix $W^{\text{out}} \in \mathbb{R}^{P\times N}$ can be calculated using the equation

$$W^{\text{out}} = \delta S \delta X^{\text{T}} (\delta X \delta X^{\text{T}} + \beta I)^{-1}, \tag{4}$$

where I is the identity matrix and β is the ridge regression parameter. The output bias vector $c \in \mathbb{R}^P$ is calculated by

$$c = -(W^{\text{out}}\bar{x} - \bar{s}). \tag{5}$$

The required computational time for the training process is of order $\mathcal{O}(N^2 t_{\max})$, which is the computational complexity of the matrix multiplication of δX with δX^{T}.

The required memory space corresponds to the size of the largest matrix used by the algorithm and is of order $\mathcal{O}(N t_{\max})$. In particular due to the memory requirement, it can be advisable to truncate the training signal if a large amount of training data is available. This way the length of the training sequence $t_{\max}$ can be limited to a moderate value.

2.3 The Erdős-Rényi model

Below, we explain several types of ESNs that differ in the graph used for the construction of their hidden weight matrix A. Several of these ESN types, including the classical model introduced in Jaeger (2001), are based on Erdős-Rényi graphs (Erdös and Renyi 1961), which we briefly describe here.

Erdős-Rényi graphs are directed graphs with sparse connectivity, which are randomly generated. First, we choose two parameters: the number of nodes N and an average node degree D. The edges of the graph are randomly chosen in such a way that all possible connections can occur with the same probability. In practice, this means that we generate the random adjacency matrix of an Erdős-Rényi graph, which is contains the entries 1 and 0 to indicate whether there is a directed connection between two nodes or not. We call this adjacency matrix an Erdős-Rényi matrix.

Precisely speaking, there are two equally common ways to define an Erdős-Rényi matrix. One can either set all entries of the matrix independently of each other to one or zero with probability D/N or $1 - D/N$, respectively; or one can set the number of connections to exactly ND and choose them randomly. In this work, we use the second version.

2.4 Random echo state networks

We refer to the classical ESN model, introduced in Jaeger (2001), as *random ESN*. As illustrated in Fig. 1, random ESNs contain one hidden layer. An Erdős-Rényi matrix is employed as the adjacency matrix of the reservoir. Thus, the reservoir's connectivity is in general sparse and all possible connections occur with the same probability. We need to specify two parameters to generate the adjacency matrix: the number of hidden nodes N and the average node degree $D \leq N$. The adjacency matrix is then obtained by the algorithm described in Sect. 2.3.

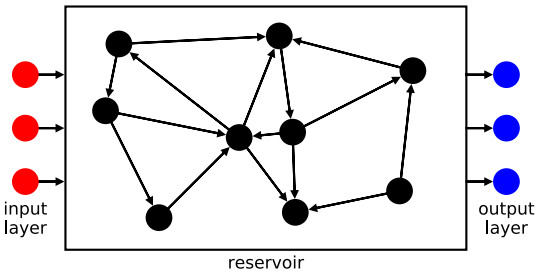

Fig. 1 Illustration of a random ESN. The reservoir is illustrated by a box containing the hidden nodes (black circles). The connections inside the reservoir are drawn randomly from an Erdős-Rényi graph. The ESN also contains multiple input nodes (red circles) and output nodes (blue circles) (Color figure online)

For the hidden weight matrix A, we need to specify one further parameter: the spectral radius ρ_A of the matrix. In order to generate the weight matrix, we draw the weights for all connections (given by the adjacency matrix) from a uniform distribution on the interval $[-1, 1]$. Then we compute the spectral radius ρ_{init} of the obtained matrix and multiply the matrix with the factor $\rho_A/\rho_{\text{init}}$. This way we obtain a hidden weight matrix A with Erdős-Rényi topology, uniformly distributed random weights, and the desired spectral radius ρ_A.

2.5 Deep echo state networks

Deep Echo State Networks (deep ESNs) are ESNs with multiple hidden layers (Gallicchio and Micheli 2016; Gallicchio et al. 2018; Gallicchio and Micheli 2020; Dettori et al. 2020), as illustrated in Fig. 2. We consider deep ESNs with equally sized layers, i.e., the total number of nodes N of a deep ESN is a multiple of the number of hidden layers L and the number of nodes per hidden layer is $N_L = N/L$, which is usually the case for deep ESNs. Although deep ESNs can have differently sized hidden layers, it requires the introduction of new parameters to the model and, therefore, this topic is left as a future work. The adjacency matrix of a deep ESN is a block matrix consisting of blocks with dimension $N_L \times N_L$. The hidden layers are given by the main-diagonal blocks, and the inter-layer connections are given by the blocks directly below the main diagonal. All other blocks are zeros matrices. Thus, the layers are connected in a feed-forward manner, i.e., the ith hidden layer is forward-connected to the $i+1$-st hidden layer, for $i = 1, \ldots, L-1$.

The main-diagonal blocks are generated by the Erdős-Rényi model with a given mean degree $D \leq N_L$. Also, the block determining the inter-layer connections are Erdős-Rényi matrices with a given mean degree $D_{\text{inter}} \leq N_L$.

Once we obtained the whole $N \times N$ adjacency matrix of the deep ESN's reservoir, the weights of the connections are drawn from a uniform distribution on the interval $[-1, 1]$. Again, we rescale the resulting matrix to obtain a hidden weight matrix A with the desired spectral radius ρ_A.

The input layer is only connected to the first hidden layer, i.e., only the first N_L rows of the input weight matrix W^{in} have non-zero entries. All nodes of the reservoir (i.e., of all

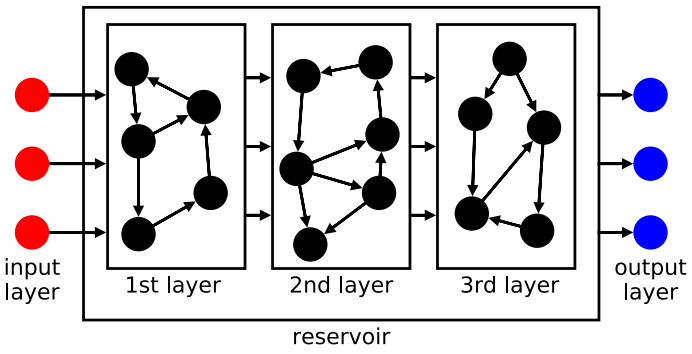

Fig. 2 Illustration of a deep ESN with three hidden layers. The reservoir is indicated by the large outer box and contains three hidden layers (smaller rectangles inside the box). Each hidden layer contains multiple nodes (black circles) which are recurrently connected inside the layers. The hidden layers are connected to each other in a feed-forward manner. The input nodes (red circles) are connected to the first hidden layer, and the output nodes (blue circles) are connected all hidden layers, i.e., to the whole reservoir (Color figure online)

hidden layers) are connected to the output layer. Hence, there are no restrictions for the output weight matrix W^{out}.

For the numerical tests presented in Sect. 5, we sometimes did not use exactly equally sized layers, but layers which can differ in size by one node. This way, the number of layers L could be chosen more flexibly.

3 Clustered echo state networks

Multiple works Deng and Zhang (2007), Li et al. (2015), Yu et al. (2011) and Oliveira et al. (2020) have proposed new types of ESNs based on complex networks: *Clustered Echo State Networks* (CESNs). Instead of choosing a simple random Erdős-Rényi graph to define the reservoir, connection patterns with clusters were chosen. See Fig. 3 for an illustration.

In this section, we focus on the models from Oliveira et al. (2020). The clusters are represented by equally sized blocks in the adjacency matrix, and the internal connections of each cluster are either realized using the Erdős-Rényi model or a custom model which is similar to the Barabási-Albert model. These new ESN types are called *Erdős-Rényi CESN* or *Barabási-Albert-like CESN*, respectively. Moreover, we propose a combination of CESNs and deep ESNs: deep ESNs with clustered layers, which we call *deep CESNs*.

For all CESNs and deep CESNs defined below, we denote the total number of hidden nodes by N. We let D be the mean degree of the hidden layer(s). We consider deep CESNs with L equally sized hidden layers, each consisting of $N_L = N/L$ nodes. Further, let $C \in \mathbb{N}$ be the number of clusters per hidden layer. Also the clusters are equally sized, i.e., N (or N_L, respectively) must be a multiple of C. Let $P_{\text{in}} \in [0, 1]$ be the share of cluster-internal connections and $P_{\text{out}} = 1 - P_{\text{in}}$ be the share of inter-cluster-connections within a hidden layer.

Since D is the mean degree of the hidden layer(s) and N (or N_L, for the deep case) is the number of nodes per hidden layer, one hidden layer contains ND (or $N_L D$) connections: specifically, NDP_{in} (or $N_L DP_{\text{in}}$) cluster-internal connections and NDP_{out} (or $N_L DP_{\text{out}}$) inter-cluster connections.

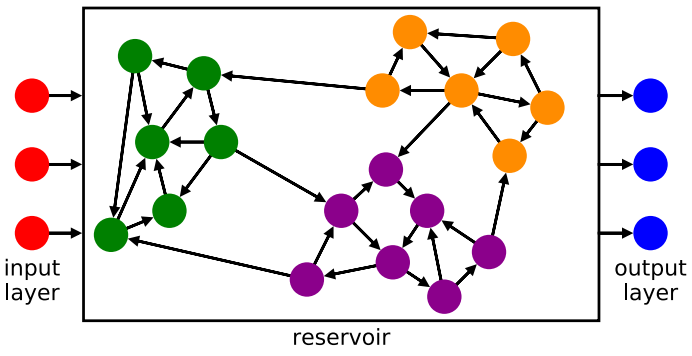

Fig. 3 Illustration of a clustered ESN. The reservoir (large box) contains three clusters, where the nodes belonging to the same cluster are drawn in the same color (green, purple or orange). The inter-cluster connectivity is sparser than the connectivity inside the clusters. The input and the output layer of the clustered ESN (red and blue) are connected to all clusters in the reservoir (Color figure online)

3.1 Erdős-Rényi clustered echo state networks

The clustered network topology of an Erdős-Rényi CESN is described by an adjacency matrix which is partitioned into blocks of size $N_C \times N_C$. Each cluster is represented by a block on the main-diagonal. The off-diagonal blocks contain the connections between nodes of different clusters. The connections inside the main-diagonal blocks are determined by an Erdős-Rényi graph with N_C nodes and NDP_{in}/C edges. To realize the inter-cluster connections, we randomly set NDP_{out} entries of the off-diagonal blocks to 1.

The adjacency matrix of an Erdős-Rényi CESN is similar to a stochastic block model (Holland et al. 1983). It is, however, not exactly the same because the main-diagonal and off-diagonal blocks are constructed following different rules.

Given the adjacency matrix, we obtain the weight matrix of the reservoir by drawing random $\mathcal{U}([0, 1])$-distributed weights and rescaling them to achieve the desired spectral radius.

3.2 Barabási-Albert-like clustered echo state network

The reservoir of a Barabási-Albert-like CESN is described by a block matrix, which has the same block structure as the adjacency matrix of the Erdős-Rényi CESN defined above. However, the main-diagonal blocks, which represent the clusters, are obtained from a different graph model, and have themselves a clustered structure. This graph model is inspired by the model introduced in Bollobas et al. (2003) and the well-known Barabási-Albert model Barabási and Albert (1999). We apply a modified model instead of the original Barabási-Albert model or the model from Bollobas et al. (2003) because we require directed graphs and we are given an exact number of connections.

To generate the main-diagonal blocks, we initialize all N_C nodes of a cluster at once (in contrast to the models from Barabási and Albert (1999) and Bollobas et al. (2003)). We seed the graph with one connection from one node to itself (a loop). Then we repeat the following procedure until the graph contains NDP_{in}/C directed edges:

1. With probability P_1 do: (a) Choose a node u with uniform probability. (b) Choose a node v with a probability proportional to the node's in-degree. (c) Add an edge from u to v.
2. With probability P_2 do: (a) Choose a node u with a probability proportional to the node's out-degree. (b) Choose a node v with a probability proportional to the node's in-degree. (c) Add an edge from u to v.
3. With probability P_3 do: (a) Choose a node u with a probability proportional to the node's out-degree. (b) Choose a node v with uniform probability. (c) Add an edge from u to v.

Per step only one of the procedures 1), 2) or 3) is conducted; note that the probabilities add up to 1. If there is already an edge between the randomly selected nodes, we do not add another edge with the same direction. Here, we choose $P_1 = 0.41$, $P_2 = 0.54$, and $P_3 = 0.05$.

Subsequently, we add inter-cluster connections in two steps:

1. We construct a graph with C nodes and NDP_{out} edges using the modified Barabási-Albert-like model, described above, with one difference: this time we allow multiple edges per direction between two nodes. That is, the entries of the resulting adjacency

matrix can be any non-negative integer. Moreover, we do not allow loops in this case, i.e., the entries on the diagonal are zeros.

2. According to the adjacency matrix generated in step 1), we decide how many connections we draw from one cluster to another. Recall that the reservoir's adjacency matrix is a block matrix with $C \times C$ blocks of size $N_C \times N_C$. If the adjacency matrix obtained in step 1) has the entry n at position (i, j), we generate the block with the index (i, j) of the reservoir's adjacency matrix using the Erdős-Rényi model with N_C nodes and n edges.

Again, the hidden weight matrix A is obtained by drawing the weights from a uniform distribution on [0, 1] and rescaling.

3.3 Deep clustered echo state networks

We propose combinations of deep ESNs and CESNs: deep Erdős-Rényi CESNs and deep Barabási-Albert-like CESNs. The basic structure of these networks is the same as for the deep ESN defined in Sect. 2.5: the hidden weight matrix is a block matrix consisting of $L \times L$ blocks of size $N_L \times N_L$, where $N_L = N/L$. The main-diagonal blocks, which represent the hidden layers, are constructed according to the models described in the Sects. 3.1 and 3.2, respectively. That is, each hidden layer has a clustered Erdős-Rényi or Barabási-Albert-like structure, and is itself a block matrix. The connections between the hidden layers are constructed in the same way as for the classical deep ESN, Sect. 2.5.

Note that equally sized clusters and layers require the total number of nodes N to be a multiple of L, and N_L to be a multiple of C. For a more flexible choice of these parameters, we allowed the clusters and layers to differ in size by one node for our numerical test presented in Sect. 5.

4 Materials and tasks

The performance of the ESN methods described above was evaluated using three different tasks—two based on artificial data and the third based on real-world signals. These tasks are: *Gaussian noise reduction* We add Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 1$ to a randomly generated wave signal. The wave signal consists of four sine signals with random phases, slow frequencies 0.005, 0.01, 0.02, 0.03, and fluctuating (between 0.5 and 1.5) amplitudes determined by randomly generated envelope functions. The objective of the task is to remove the noise and to reconstruct the original artificial wave signal. For training and testing we use the same parameters for the noise, frequencies, and amplitude, but independent realization of the randomly generated signal. *Impulse noise reduction* We use the same kind of randomly generated wave signal as for task 1), but we add impulse noise instead of Gaussian noise. For a given number $n_{\mathrm{dp}} \in \mathbb{N}$ and a given noise amplitude $\delta > 0$, we randomly choose n_{dp} data points and deviate each of them by a random value drawn from a uniform distribution on the interval $[-\delta, \delta]$. For our numerical tests, we chose $\delta = 1$ and $n_{\mathrm{dp}} = K/20$, where K is the number of data points of the original discrete-time signal. Again, the objective of the task is to reconstruct the original wave signal. For training and testing we use the same parameters for the noise, frequencies, and amplitude, but independent realization of the randomly generated signal.

ECG signal noise reduction For this task, we were using the dataset from Lugovaya (2005); Goldberger et al. (2000), which contains ECG records from 90 persons

(multiple records per person). Each record consists of two different signals: the noisy, raw ECG signal, and a filtered signal without noise. The signals have a length of 20 seconds and were recorded with a sampling rate of 500 Hz. The amplitude was measured in units of mV. The raw signals serve as input and the filtered signals serve as target signals.

In order to take account of non-causal relationships between input and target data, i.e., the fact that $s(t)$ may depend on $u(t')$ with $t' > t$, we introduce the delay parameter d. If $d > 0$, the input signals are shifted by d time steps relative to the target signal. Consequently, the ESN methods can take into account d input data points ahead of the target signal.

For the ECG signal noise reduction task, we did not use the complete dataset. Instead we randomly selected 50 records as training dataset, and 20 records as test dataset. All records were taken from different people. For the training process, we concatenated the 50 training signals to one long sequence of data points. The test signals were processed in the same way. Each of the 50 training records contains 10,000 data points, i.e., the complete training signal contains 500,000 data points, of which 5000 are used for the initial wash out phase and 495,000 contribute directly to the training. This amount of data is sufficient to train an ESN. It might be possible to increase the prediction performance for the ECG signal noise reduction task by increasing the amount of training data. However, the right choice of this amount is a trade-off between prediction performance and memory requirement (see Sect. 2.2).

Note that for all tasks we have one-dimensional input and target time series, i.e., $M = 1$ and $P = 1$. Thus, the input and output matrices $W^{\text{in}} \in \mathbb{R}^{N \times M}$ and $W^{\text{out}} \in \mathbb{R}^{P \times N}$ become vectors and the output bias vector $c \in \mathbb{R}^P$ is actually a scalar. Other variables than W^{in}, W^{out} and c are not affected by the input dimension M or the target dimension P. In particular, the hidden weight matrix A does not depend on M or P. Therefore, adapting our approach to multivariate time series is straightforward. Ref. Lu et al. (2017) contains examples of random ESNs applied to multivariate time series. Since the other ESN types described in Sect. 2 and 3 differ only in the choice of A, they can be easily applied to multivariate time series too.

5 Experimental results

We applied the CESNs and deep CESNs, described in Sect. 3, to the tasks from Sect. 4, and compared them to the classical ESN methods, described in Sect. 2. We used the hyperbolic tangent as the activation function f. The default parameters for our experiments are listed in Table 1. The results were evaluated by the Normalized Root Mean Squared Error (NRMSE) of the output signal $\hat{s}(t)$ in comparison to the target signal $s(t)$:

$$\text{NRMSE}(\hat{s}, s) = \sqrt{\frac{\sum_{t=1}^{t_{\text{test}}} (\hat{s}(t) - s(t))^2}{\text{Var}(s) t_{\text{test}}}} \tag{6}$$

where $\text{Var}(s)$ is the variance of the target signal and t_{test} is the number of time steps of the test time series. All presented results are an average over 8 executions.

Table 1 Default parameters for the numerical experiments

Parameter	Default value
Total number of hidden nodes N	1050
Mean degree of the hidden layer(s) D	$20^{a,b,c,d}$, $15^{e,f}$
Mean deg. of deep ESN's inter-layer connections	20^{b}, $5^{e,f}$
Spectral radius ρ_A	1.1
Number of layers for the deep ESN L	4
Number of clusters C (Erdős-Rényi CESN)	$6^{g,i}$, 42^{h}
No. of clusters C (Barabási-Albert-like CESN)	$25^{g,i}$, 3^{h}
No. of clusters C (deep CESN)	6^{e}, 3^{f}
Ratio of connections inside the clusters P_{in}	$0.98^{c,e}$, $0.82^{d,f}$
Ratio of connections outside the clusters P_{out}	$0.02^{c,e}$, $0.18^{d,f}$
Leakage rate α	$0.95^{g,h}$, 0.2^{i}
Bias strength γ	0.1
Max. amplitude of the random input weights	0.1
Delay parameter d	$16^{g,i}$, 4^{h}
Length of the training time series (excluding washout phase) t_{max}	$50{,}000^{g,h}$, $495{,}000^{i}$
Length of the test time series (excluding washout phase) t_{test}	$30{,}000^{g,h}$, $195{,}000^{i}$
Number of steps for the initial washout phase $-t_0$	5000
Ridge regression parameter β	2×10^{-7}

[a]Random ESN
[b]Deep ESN
[c]Erdős-Rényi CESN
[d]Barabási-Albert-like CESN
[e]Deep Erdős-Rényi CESN
[f]DBarabási-Albert-like CESN
[g]Gaussian noise reduction
[h]Impulse noise reduction
[i]ECG noise reduction

5.1 Gaussian noise reduction

We conducted numerical tests to compare the ESN methods to each other and to the causal Wiener filter. We applied the Wiener filter implementation from scipy.signal.wiener with optimal window size and noise power parameter. We found that the random ESN performs as good as the Wiener filter on Gaussian noise reduction and that all other ESN methods perform even better if their parameters are properly chosen.

Figure 4 shows the performance of various ESN methods on the Gaussian noise reduction task for different leakage rate values α. For all methods, we need rather large values of α to obtain good results. This can be explained by the fact that for removing Gaussian noise from our artificially created time series the local structure of the time series is more important than its states in the more distant past. The leakage rate controls how fast the system state changes: a larger leakage rate leads to a faster changing system state which makes the ESN more sensitive to short-term fluctuations of the input time series. The Wiener filter achieved an NRMSE of approximately 0.23. For $\alpha \geq 0.95$, the deep ESN and the CESNs

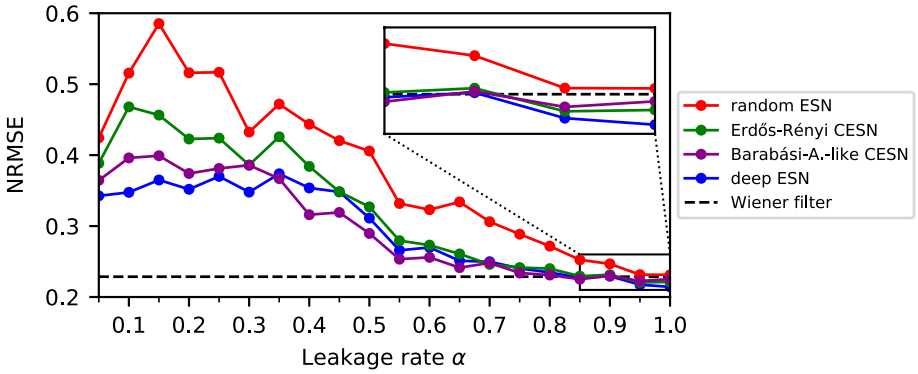

Fig. 4 Gaussian noise reduction NRMSE depending on α for the random ESN (red), the Erdős-Rényi CESN (green), the Barabási-Albert-like CESN (purple), and the deep ESN (blue). For comparison, the Wiener filter's NRMSE is indicated by a dashed black line. The deep ESN and the CESNs perform better than the Wiener filter if the leakage rate is large enough. The random ESN's performance is similar to that of the Wiener filter if $\alpha = 1$ (Color figure online)

yield better results than the Wiener filter. Moreover, we found that the deep ESN and the CESNs yield better results than the classical random ESN for all α.

Figure 5a shows the NRMSE obtained from the deep ESN with different values of the delay parameter d and α. One can see that it is important to set d not too small. The results for α correspond to the results shown in Fig. 4.

Figure 6 compares the performance of the Erdős-Rényi and the Barabási-Albert-like CESN for different numbers of clusters C. Both CESN methods work similarly well if C is at least 15. The performance of the Barabási-Albert-like CESN drops strongly when we decrease C. For the Erdős-Rényi CESN we obtain a similar NRMSE over the whole range of C.

Moreover, we evaluated the performance of the deep ESN and deep CESNs for different numbers of hidden layers L. The results are presented in Fig. 7. All three methods achieve similar NRMSEs, mostly between 0.2 and 0.21, which is significantly better than the Wiener filter's NRMSE. The choice of the parameter L does not have a strong influence on the

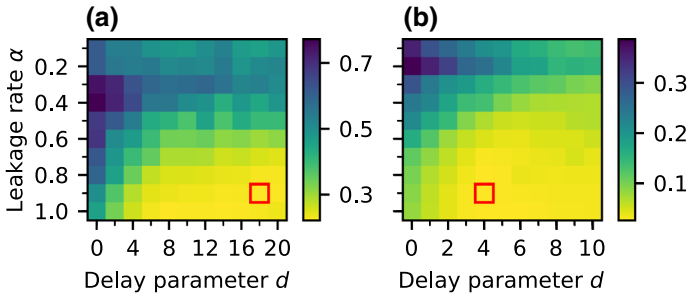

Fig. 5 NRMSE of the deep ESN for different values of α and d for **a** the Gaussian noise reduction task, and **b** the impulse noise reduction task. For both tasks, optimal results are achieved for a relatively wide range of the delay parameter d. Only if d is very small, the performance drops significantly. The leakage rate α should be larger than 0.8 to obtain optimal results for the Gaussian noise reduction tasks and larger than 0.6 for the impulse noise reduction task. The NRMSE minima are indicated by red squares (Color figure online)

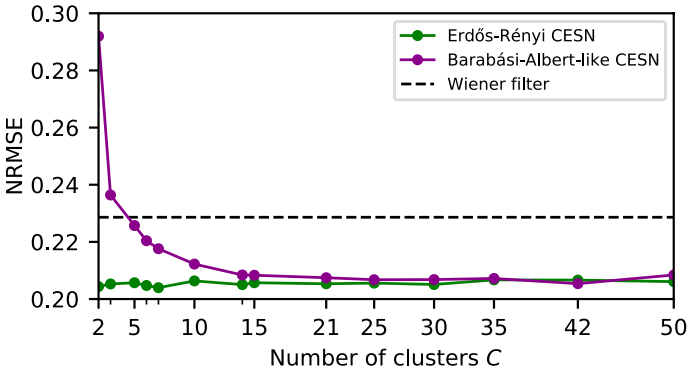

Fig. 6 Gaussian noise reduction performance of the CESNs depending on the number of clusters C. For the Erdős-Rényi CESN (green), the NRMSE remains almost unchanged if we vary C. The Barabási-Albert-like CESN (purple) has a high NRMSE for small C, but can compete with the Erdős-Rényi CESN when C is sufficiently large. The Wiener filter's NRMSE is indicated by a dashed black line (Color figure online)

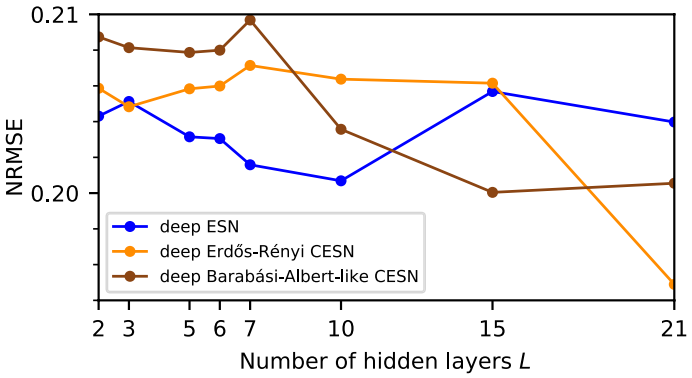

Fig. 7 Gaussian noise reduction with the deep ESN (blue), the deep Erdős-Rényi CESN (orange) and the deep Barabási-Albert-like CESN (brown). The NRMSE tends to be slightly better for a larger number of layers L (Color figure online)

performance, but one can notice a slight trend that the deep CESNs do better with a larger number of hidden layers.

5.2 Impulse noise reduction

As for the reduction of Gaussian noise, we require sufficiently large values of α for the impulse noise reduction task; see Fig. 8. With $\alpha \geq 0.7$, we obtain near optimal results for the deep ESN and the CESNs. For the random ESN α should be near 1 to obtain good results. Again, we can explain the need for a rather large leakage rate by the local nature of the task. For any choice of α, the deep ESN and the CESNs perform better than the random ESN.

A 2D plot showing the NRMSE of the deep ESN for the impulse noise reduction task for different values of α and d is presented in Fig. 5b. In comparison to the Gaussian

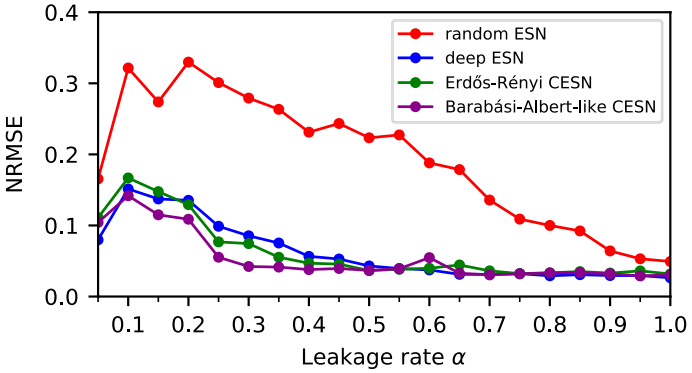

Fig. 8 Impulse noise reduction NRMSE depending on the leakage rate α for the random ESN (red), the deep ESN (blue), the Erdős-Rényi CESN (green), and the Barabási-Albert-like CESN (purple). The deep ESN and the CESN methods perform substantially better than the random ESN over the whole range of α. For all methods, α should be chosen to be rather large (Color figure online)

noise reduction task, a smaller delay parameter d is sufficient. Further, the NRMSE remains small for a wider range of α, which corresponds to the findings shown in Fig. 8.

Figure 9 shows how the CESNs perform depending on the number of clusters C. For the Erdős-Rényi CESN, C should be large, whereas for the Barabási-Albert-like CESN, fewer clusters are beneficial. Overall, we obtained significantly better results from the Barabási-Albert-like CESN.

We tested the deep ESN and deep CESNs with different numbers of layers L; see Fig. 10. In all cases, the NRMSE is widely independent of L. Moreover, the results are not better than the results obtained from the CESNs (Fig. 9). These findings suggest that the classical deep ESN, CESNs and deep CESNs are equally suitable for removing impulse noise.

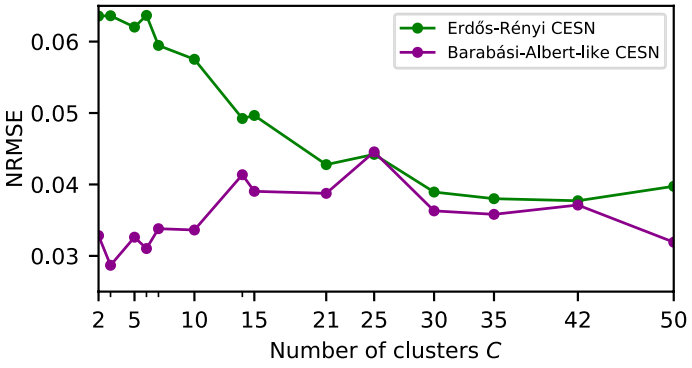

Fig. 9 Impulse noise reduction performance of the CESNs depending on the number of clusters C. For the Erdős-Rényi CESN (green), the NRMSE decreases as the number of clusters increases. In contrast, for the Barabási-Albert-like CESN (purple), we obtained the best NRMSE for the rather small value $C = 3$. Over the whole range of C, the Barabási-Albert-like CESN performs as good or better than the Erdős-Rényi CESN (Color figure online)

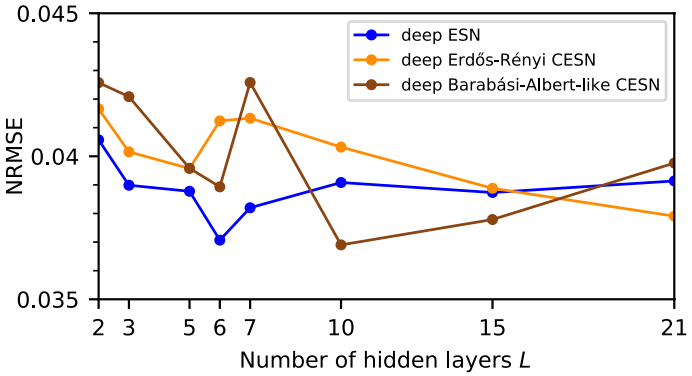

Fig. 10 Impulse noise reduction with the deep ESN (blue), the deep Erdős-Rényi CESN (orange) and the deep Barabási-Albert-like CESN (brown). All methods perform similarly well and the NRMSE seems not to depend on L (Color figure online)

5.3 ECG signal noise reduction

The numerical studies presented above have been conducted using artificially generated data. To demonstrate that the investigated ESN methods are also useful for real world problems, we apply them for the reduction of ECG noise (Lugovaya 2005; Goldberger et al. 2000).

Figure 11 shows the NRMSE obtained from the random ESN, the deep ESN, and the CESNs depending on the leakage rate α. The deep ESN and the CESNs show a similar behavior. For these methods, $\alpha = 0.2$ is optimal and yields an NRMSE of roughly 0.4. For the random ESN larger values of α are needed and the optimal NRMSE is about 0.45. In contrast to the Gaussian and impulse noise removal tasks which we considered above, the ECG signal noise reduction task requires a smaller leakage rate for optimal results. This indicates that considering a wider time window of the noisy input improves the quality of

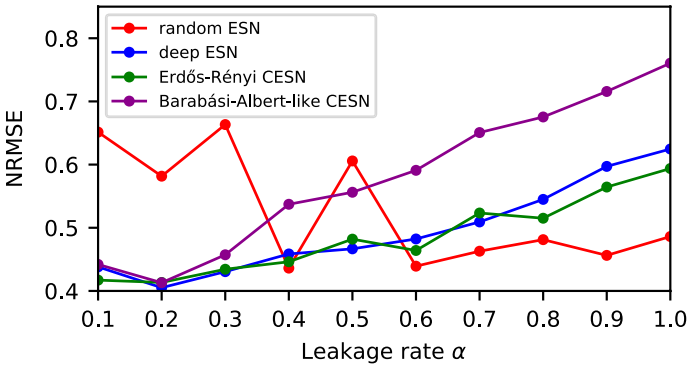

Fig. 11 ECG noise reduction NRMSE depending on the leakage rate α for the random ESN (red), the deep ESN (blue), the Erdős-Rényi CESN (green), and the Barabási-Albert-like CESN (purple). The deep ESN and the CESNs achieve an optimal NRMSE for $\alpha = 0.2$. The random ESN requires a larger value for α for an optimal NRMSE and performs slightly worse than the other methods (Color figure online)

the prediction of the signal without noise. Although these results are not competitive to more advanced methods, they show that ESNs are in principle suitable for real world tasks.

5.4 Computing time

ESNs are trained by processing a training input series once and using linear regression to fit the ESN's output to a given target series. See Sect. 2.2 for a detailed explanation of the training process. For the tasks with artificial time series (Gaussian and impulse noise reduction), the length of the training time series is 50,000 and the CPU time needed for the training is between 4.3 and 4.8 seconds for all tested ESN models. For the ECG noise reduction tasks, the length of the training time series is 495,000 and the measured CPU time for the training is between 38 and 42 seconds for all models.

Regarding the inference time ESNs do not offer a speed benefit. As inference time for the artificial datasets (test series length 30,000) we measured 2.2 to 2.5 seconds for each ESN model, for the ECG noise reduction task (test series length 195,000) we measured 14.5 to 15.6 seconds.

In summary, ESNs are in many cases a suitable method when fast training is required.

The CPU time measurements were performed on a HP Z1 computer with Intel Core i7-10700 processor.

5.5 Comparison of echo state networks to alternative methods

We performed numerical tests with a selection of alternative denoising methods: a low pass filter implemented using fast Fourier transform (FFT filter), the wavelet filter (implemented in skimage.restoration.denoise_wavelet) (Kohler 2005), and the Wiener filter (scipy.signal.wiener). In Table 2 we compare the results of these classical methods with the results of the ESN methods.

Table 2 Average NRMSE of the ESNs and classical methods for each task

	Gaussian Noise	Impulse Noise	ECG
Random ESN	0.2315	0.0531	0.5817
Deep ESN	0.2174	0.0295	**0.4052**[a]
Erdős-Rényi CESN	0.2206	0.0360	0.4134
Barabási-Albert-like CESN	0.2226	**0.0291**[a]	0.4131
Deep Erdős-Rényi CESN	**0.1949**[a]	0.0379	0.7547
Deep Barabási-Albert-like CESN	0.2000	0.0398	0.8124
FFT	**0.1318**[b]	**0.0194**[b]	0.9117
Wavelet	0.2362	0.0307	**0.9114**[b]
Wiener filter	0.2365	0.0424	0.9119

Bold values indicate optimal results

For the ESN methods we applied the parameters listed in Table 1. For the three additional filter methods we were using optimal parameters found by grid search

[a]Lowest NRMSE of all considered ESN methods

[b]Lowest NRMSE of all considered classical methods

For the tasks based on artificial time series data (Gaussian and impulse noise reduction), the FFT fiter has the best performance among all considered methods. This result is expected because the target time series for these tasks are obtained by discretization of signal with a discrete frequency spectrum containing only 4 frequencies. For the Gaussian and the impulse noise reduction tasks, ESNs perform similarly well as the Wiener filter and the wavelet filter.

It is noteworthy that the ESN methods perform significantly better than the classical filters if we consider real world data (the ECG noise reduction task). These results show that the ESN methods are quite robust to deal with real-world problems where the signals usually contain wide noise frequency bands and the frequency bands vary from one signal to another.

6 Conclusion

We demonstrated that Echo State Networks (ESNs), deep ESNs, and the Erdős-Rényi and Barabási-Albert-like clustered Echo State Networks (CESNs), introduced in Oliveira et al. (2020), are adequate tools to denoise time series. Further, we introduced deep CESNs, which are CESNs with multiple hidden layers.

All considered ESN methods can compete with the Wiener filter and the wavelet filter in removing Gaussian noise and impulse noise from an artificial wave signal. Moreover, ESN reveal significantly better results in comparison to classical denoising methods for the ECG noise reduction task. In comparison to common machine learning methods for sequential data, such as LSTMs, ESNs are significantly simpler in terms of their architecture and training process, which makes an implementation from scratch easier. Taking the ESN method's simplicity into account, we also obtained satisfactory results on the ECG noise reduction task. Overall, the deep ESN, the CESNs, and the deep CESNs have shown a significantly better performance than the classical ESN with simple Erdős-Rényi connectivity. This implies that ESNs can be benefited from both clustered and layered structures of the internal network. For the deep ESN, this result was expected because prior studies have shown its advantage over classical ESNs (Gallicchio and Micheli 2016, 2020). Our contribution is the finding that CESNs perform as good as deep ESNs for certain tasks. Hence, CESNs may be considered as an equivalent alternative to deep ESNs. Moreover, in some situations deep CESNs can provide an additional benefit over deep ESNs or CESNs. Despite the optimal NRMSE of deep CESNs is similar to the results obtained by deep ESNs and CESNs for the tasks that we investigated, deep CESNs are able to achieve this NRMSE with a smaller number of total connections within and in between the hidden layers (see Table 1). Using algorithms for sparse matrices, this leads to faster computation during inference time.

Author Contributions LOJ, FS and LZ designed this research, made the analysis, and revised the paper. LOJ and FS implemented the solution, conducted the computational simulations, and wrote the paper.

Funding This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the Sao Paulo Research Foundation (FAPESP) under Grant Number: 2019/07665-4 and by the IBM Corporation. This work is also supported in part by FAPESP under Grant Numbers 2015/50122-0, the Ministry of Science and Technology of China under Grant Number:G20200226015, and the Deutsche Forschungsgemeinschaft (DFG) in the framework of IRTG 1740.

Availability of data and materials The script to generate the artificial datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request. The ECG dataset used is available in the PhysioNet repository, https://doi.org/10.13026/C2J01F

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Code availability The code used for experiments and/or analysis is available at https://github.com/laerc/esn-noise-filtering.

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938.

Akiki, T. J., & Abdallah, C. G. (2019). Determining the hierarchical architecture of the human brain using subject-level clustering of functional networks. *Science and Reports, 9,* 19290.

Appeltant, L., Soriano, M., Van Der Sande, G., Danckaert, J., Massar, S., Dambre, J., et al. (2011). Information processing using a single dynamical node as complex system. *Nature Communications, 2,* 1–6.

Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science, 286*(5439), 509–512.

Berry, M. J., & Tkacik, G. (2020). Clustering of neural activity: A design principle for population codes. *Frontiers in Computational Neuroscience, 14,* 20.

Bollobas, B., Borgs, C., Chayes, J., & Riordan, O. (2003). Directed scale-free graphs. In *Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms (SODA)*, pp. 132–139.

Brunner, D., Soriano, M. C., Mirasso, C. R., & Fischer, I. (2013). Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature Communications, 4,* 1364. https://doi.org/10.1038/ncomms2368.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Deng, Z., & Zhang, Y. (2007). Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks, 18*(5), 1364–1375. https://doi.org/10.1109/TNN.2007.894082.

Dettori, S., Matino, I., Colla, V., & Speets, R. (2020). Deep echo state networks in industrial applications. In I. Maglogiannis, L. Iliadis, & E. Pimenidis (Eds.), *Artificial Intelligence Applications and Innovations* (pp. 53–63). Cham: Springer International Publishing.

Erdös, P., & Renyi, A. (1961). On the strength of connectedness of a random graph. *Acta Mathematica Hungarica, 12,* 261–267.

Gallicchio, C., & Micheli, A. (2016) Deep reservoir computing: A critical analysis. In *ESANN 2016 proceedings, European symposium on artificial neural networks, computational intelligence and machine learning*.

Gallicchio, C., & Micheli, A. (2020). Deep echo state network (deepesn): A brief survey.

Gallicchio, C., Micheli, A., & Pedrelli, L. (2018). Design of deep echo state networks. *Neural Networks,108*, 33–47. https://doi.org/10.1016/j.neunet.2018.08.002, https://www.sciencedirect.com/science/article/pii/S0893608018302223.

Gleiser, P. M., & Spoormaker, V. I. (2010). Modelling hierarchical structure in functional brain networks. *Philosophical Transactions of the Royal Society A, 368,* 5633–5644.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., et al. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation, 101*(23), e215–e220.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Hagmann, P., Cammoun, L., Gigandet, X., Meuli, R., Honey, C. J., Wedeen, V. J., & Sporns, O. (2008). Mapping the structural core of human cerebral cortex. *PLoS Biology, 6,* e156.
Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York, NY: Wiley.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9,* 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.
Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic block models: First steps. *Social Networks, 5*(2), 109–137.
Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks—with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148**.
Jaeger, H. (2002). Short term memory in echo state networks.
Kohler, D. (2005). A comparison of denoising methods for one dimensional time series.
Kolen, J. F., & Kremer, S. C. (2001). Gradient flow in recurrent nets: The difficulty of learning long term dependencies, pp. 237–243. Wiley. https://doi.org/10.1109/9780470544037.ch14.
Krishnagopal, S., Girvan, M., Ott, E., & Hunt, B. (2020). Separation of chaotic signals by reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*. https://doi.org/10.1063/1.5132766.
Larger, L., Soriano, M., Brunner, D., Appeltant, L., Gutierrez, J., Pesquera, L., Mirasso, C., & Fischer, I. (2012). Photonic information processing beyond Turing: An optoelectronic implementation of reservoir computing. *Optics Express, 20*(3).
Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539.
Li, X., Zhong, L., Xue, F., & Zhang, A. (2015). A priori data-driven multi-clustered reservoir generation algorithm for echo state network. *PLoS ONE, 10*(4), 1–15.
Lin, X., Yang, Z., & Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications, 36,* 7313–7317.
Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., & Ott, E. (2017). Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos An Interdisciplinary Journal of Nonlinear Science, 27*(4), 041102. https://doi.org/10.1063/1.4979665.
Lugovaya, T. S. (2005). Biometric human identification based on electrocardiogram. Master's thesis, Faculty of Computing Technologies and Informatics, Electrotechnical University "LETI", Saint-Petersburg, Russian Federation.
Martens, M., Meier, J., Hillebrand, A., Tewarie, P., & Mieghem, P. (2017). Brain network clustering with information flow motifs. *Applied Network Science, 2*. https://doi.org/10.1007/s41109-017-0046-z.
McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115–133. https://doi.org/10.1007/BF02478259.
Mozer, M. C. (1989). A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems,3*(4).
Oliveira, L., Jr., Stelzer, F., & Zhao, L. (2020). Clustered echo state networks for signal observation and frequency filtering. Mining and Learning*Anais do VIII symposium on knowledge discovery* (pp. 25–32). Porto Alegre, RS, Brasil: SBC.
Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*. https://doi.org/10.1063/1.5010300.
Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computing, 2* . https://doi.org/10.1109/TETCI.2017.2762739.
Robinson, A. J., & Fallside, F. (1987). *The utility driven dynamic error propagation network*. Tech. rep.: Engineering Department, Cambridge University, Cambridge, UK.
Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the annual conference of the international speech communication association, INTERSPEECH*, pp. 338–342.
Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61,* 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.
Schumacher, J., Toutounji, H., & Pipa, G. (2013). An analytical approach to single node delay-coupled reservoir computing. *Conference: 23rd international conference on artificial neural networks*.
Takahashi, N., Goswami, N., & Mitsufuji, Y. (2018). Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *2018 16th International workshop on acoustic signal enhancement (IWAENC)*, pp. 106–110. https://doi.org/10.1109/IWAENC.2018.8521383.

Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal, 3*(2), 334–340. https://doi.org/10.1016/j.fcij.2018.10.003.

Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., & Mitsufuji, Y. (2017). Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 261–265. https://doi.org/10.1109/ICASSP.2017.7952158.

Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series*. Wiley.

Yu, P., Miao, L., & Jia, G. (2011). Clustered complex echo state networks for traffic forecasting with prior knowledge. In *2011 IEEE international instrumentation and measurement technology conference*, pp. 1–5.

Zimmermann, R. S., & Parlitz, U. (2018). Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*. https://doi.org/10.1063/1.5022276.