Clustered Echo State Networks for Signal Observation and Frequency Filtering

Laércio de Oliveira Junior[1], Florian Stelzer[2,3], Liang Zhao[1]

[1] Faculty of Philosophy, Science, and Letters at Ribeirão Preto (FFCLRP), University of São Paulo, Brazil
laercio.oliveira@usp.br, zhao@usp.br
[2] Department of Mathematics, Humboldt University of Berlin, Germany
[3] Institute of Mathematics, Technical University of Berlin, Germany
stelzer@math.tu-berlin.de

Abstract.

Echo State Networks (ESNs) are recurrent neural networks that map an input signal to a high-dimensional dynamical system, called *reservoir*, and possess adaptive output weights. The output weights are trained such that the ESN's output signal fits the desired target signal. Classical reservoirs are sparse and randomly connected networks. In this article, we investigate the effect of different network topologies on the performance of ESNs. Specifically, we use two types of networks to construct clustered reservoirs of ESN: the clustered Erdös–Rényi and the clustered Barabási-Albert network model. Moreover, we compare the performance of these clustered ESNs (CESNs) and classical ESNs with the random reservoir by employing them to two different tasks: frequency filtering and the reconstruction of chaotic signals. By using a clustered topology, one can achieve a significant increase in the ESN's performance.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Mathematics of computing** → **Graph theory**.

Keywords: clustered networks, complex networks, echo state networks, machine learning, neural networks

1. INTRODUCTION

The Echo State Networks(ESNs) proposed in [Jaeger 2001] are a type of Recurrent Neural Networks (RNNs), where the ESN contains one input layer, one hidden layer, and one output layer. The ESN's hidden layer is named *reservoir*, and it is generally a random network of artificial neurons.

ESN's are a commonly used possibility to perform *reservoir computing*, which is a framework based on dynamical systems. Reservoir computing is realized by a fixed and non-linear system, that maps inputs to a high-dimensional space [Tanaka et al. 2019]. This system also has a readout function, mapping the states in the reservoir to an output signal. [Tanaka et al. 2019]. The main advantage of this model is that the training part is fast because only the output connections are trained. Another advantage is the flexibility to choose the network in the reservoir.

Reservoir computing has an important role in machine learning study because it can be used to solve several applications, such as the classification of spoken digits [Appeltant et al. 2011], and the prediction of stock market prices [Lin et al. 2008]. The publication [Wen et al. 2015] demonstrates that ESNs combined with Convolutional Neural Networks can be used to recognize facial expressions. A recent study [Lu et al. 2017] has shown that ESNs can emulate chaotic systems, such as the Rössler system and the Lorenz system.

Some methods also used a clustered network for the reservoir, for example, in [Deng and Zhang 2007],

in an attempt to make the model closer to a biological brain. Some methods generate the reservoir network based on training data, as in [Li et al. 2015]. Despite the fact the methods to generate the clustered network are different, it was shown that the division of the reservoir in communities increases the accuracy.

These prior works show that using a clustered network in the reservoir can be better than using a random network. Whereas these prior works applied ESNs to find patterns in time-series, this work applies ESNs for observer problems [Lu et al. 2017] and signal separation tasks. Furthermore, two additional methods to generate the clusters are presented. Also, the role of the CESN's hyperparameters for the performance is studied.

The first investigated method consists of splitting the network into a given number of smaller networks, called *clusters*. Each of these clusters will is a random network, and the connections between the clusters also form a random network [Erdös and Renyi 1961]. The second method splits the network into smaller networks too, but each of these small networks has a scale-free distribution of the degree [Barabási and Albert 1999]. Furthermore, the connections between different clusters will also have a scale-free distribution.

2. METHODS

In this section, we describe the Clustered ESN (CESN) to be studied in this work. Firstly, we review the classical ESN. Then, we present the methods for constructing *clustered reservoirs*.

2.1 Echo State Networks

The ESNs contains a reservoir with N nodes (neurons on the hidden layer), K input entries, and L output entries. The state of the reservoir is ruled by the following equation:

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

where $x(t)$ is the N-dimensional reservoir state at a given time t, which contains a memory that is set by the variable α. While the matrix A is the weighted adjacency matrix of the reservoir, indicating which nodes are connected inside the reservoir, the matrix W^{in} contains the weight of the input nodes connected to each of the nodes inside the reservoir. Worth mentioning that A has dimensions $N \times N$, and the W^{in} has dimensions $K \times N$. The variable f represents the activation function, which usually is the hyperbolic tangent. The equation also has a bias parameter, ruled by $\gamma \mathbf{1}$, where *gamma* is a constant and $\mathbf{1}$ is a vector filled with ones.

The output of a reservoir network has length an L, and it can be express by the following equation:

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

where W^{out} is the weighted output matrix with dimensions $N \times L$, and c is a correction vector. As mentioned earlier, the matrices A and W_{in} are generated initially with random values, and these values do not change over time.

In the training phase, the Echo State Network receives as input the sequence $u(1), u(2), \ldots, u(n_{\max})$, and then it will generate the reservoir internal states, $(x(1), x(2), \ldots, x(n_{\max})$. The key part is to find the elements W^{out} and c, such that the error rate is minimized for the Equation 2. To find a good estimator $\hat{s}$, the matrix W^{out} is calculated using linear regression, as stated in [Lu et al. 2017]. At the first, some pre-calculations are needed. We need to calculate the average of the states of the reservoir, $(x(1), x(2), ..., x(n_{\max})))$, and the average of the expected outputs, $(s(1), s(2), ..., s(n_{\max})))$, calculated

by the following equations:

$$\bar{x} = \frac{1}{n_{\max}} \sum_{t=1}^{n_{\max}} x(t), \tag{3}$$

$$\bar{s} = \frac{1}{n_{\max}} \sum_{t=1}^{n_{\max}} s(t), \tag{4}$$

After that, the estimator ($\hat{s}$), can be found by calculating the variables W^{out} e c:

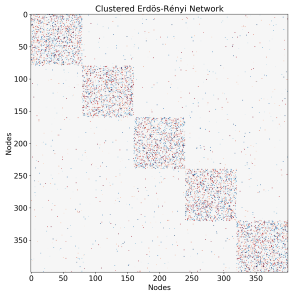
$$W^{out} = \delta S \delta X^T (\delta X \delta X^T + \beta I)^{-1}, \tag{5}$$

$$c = -[W^{out}\bar{x} - \bar{s}], \tag{6}$$

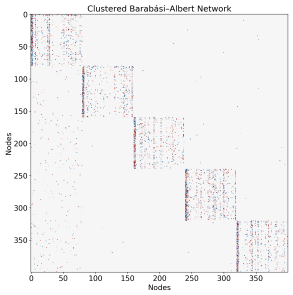
where the matrix I indicates the identity matrix with dimensions $N \times N$, δX indicates the concatenation of the states $x(t)$ in a matrix, where the i-th column contains the value for $x(t) - \bar{x}$. The same happens for the variable δS, but instead we concatenate the states $s(t)$, where the i-th column contains the value $s(t) - \bar{s}$. Using the equations stated above, we can find a good estimator $\hat{s}$ that best fits based on the input data. Worth mentioning that changing the reservoir matrix does not change anything in the training phase.

2.2 Clustered Networks

Two network models are proposed here to create the CESNs. The first model generates a network with clusters which are random networks themselves, and the connections between nodes of two different clusters are also chosen randomly. The second model is a clustered network, such that each cluster is a scale-free network, and the connections between nodes of different clusters also form a free scale network.



(a) Clustered Erdös-Rényi Network



(b) Clustered Barabási–Albert Network

Fig. 1. The image shows how is the final adjacency matrix for either the clustered networks.

Beforehand, some variables need to be defined. Let N, $\langle k \rangle$ and C, be the number of nodes in the reservoir network, the mean degree of the network, and the number of clusters of the network respectively. The mixture level means the density level of the connections between the nodes in the network. The variable P_{in} defines this mixture level of the connections inside the clusters, where $P_{\text{in}} = 0$ mean no connections between nodes of the same clusters, and $P_{\text{in}} = 1$ means that all nodes of the same cluster will be connected. The variable P_{out} defines the density level of the connections between nodes of different clusters, where P_{out} follows the same rule as P_{in}, but rather for connections between nodes from different clusters.

Note that $P_{\text{in}} + P_{\text{out}} = 1$, otherwise the mean degree of the network would be greater than D. Moreover, each cluster inside the reservoir network has the same number of nodes, which is N/C. For

the case where $P_{\text{in}} = 1$ and $P_{\text{out}} = 0$, a single connection is made between two consecutive nodes from different clusters to make the network connected.

2.2.1 *Erdös-Rényi Clustered Network.* To build such clustered network, first let split the mean degree $\langle k \rangle$ into $\langle k_{\text{in}} \rangle$ and $\langle k_{\text{out}} \rangle$, such that $\langle k_{\text{in}} \rangle$ is the mean degree counting only connections between two nodes of the same cluster, and $\langle k_{\text{out}} \rangle$ is the mean degree counting only connections between two nodes of different clusters:

$$\langle k_{\text{in}} \rangle = \langle k \rangle P_{\text{in}}, \tag{7}$$

$$\langle k_{\text{out}} \rangle = \langle k \rangle P_{\text{out}}. \tag{8}$$

After that, a total of C random networks are created with mean degree equals to $\langle k_{\text{in}} \rangle$. To create these networks, a total of $(N/C) \times \langle k_{\text{in}} \rangle$ pairs (u, v) are chosen randomly, meaning that there will be a connection between the nodes u and v, such that u and v belong to the same cluster. To create the connections between different cluster, just repeat the process by choosing a total of $(N/C) \times \langle k_{\text{out}} \rangle$ pairs (u, v) randomly, such that u and v does not belong to the same cluster. The figure 1.a shows how looks like the adjacency matrix of a clustered network with $C = 5$ and $P_{\text{in}} = 0.9$.

2.2.2 *Barabási Albert Clustered Network.* The Barabási-Albert [Barabási and Albert 1999] model implies that the degree distribution of the nodes follows a power law. This type of network is also known as Scale-free networks. It means that the more connections a node has attached to it, the higher is the probability to make new connections. The classical model of this network assumes that the connections between nodes are undirected. To build a scale-free network with directed connections, the algorithm used is the one proposed by [Bollobas et al. 2003]. This algorithm guarantees that the degree distribution of the incoming edges and outcoming edges follows a power law.

Let C, P_{in} and P_{out} the parameters defined the same as above. To build the *CESN*, the first step is to C scale-free networks, such that each of these scale-free networks has a mean degree equals to $\langle k_{\text{in}}/C \rangle$. After that, the connections between the nodes of different clusters need to be created. The same algorithm is repeated, but now the clusters are treated as big nodes. i.e., at every step, a connection is made between two nodes, based on the degree distribution following the algorithm proposed by [Bollobas et al. 2003].

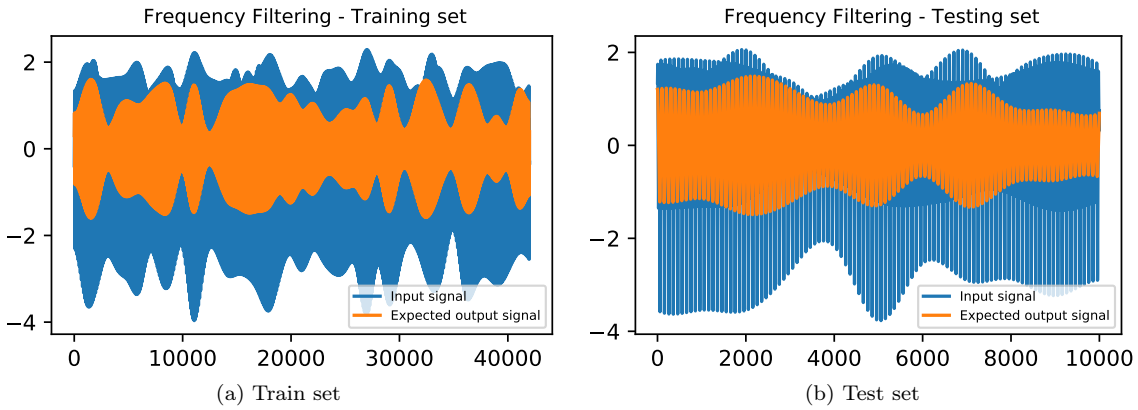
The mean degree split calculation will be the same for the scale-free network, but a new variable comes in, which is the number of initial vertices in the network. The rest of the algorithm will be the same as stated above, but instead of choosing the nodes randomly, they will be chosen according to the degree distribution [Barabási and Albert 1999]. The figure 1.b shows how looks like the adjacency matrix of a scale-free clustered network with $C = 5$ and $P_{\text{in}} = 0.9$.

2.3 Tasks

For each task, the dataset contains a training set and a testing set, and each of these sets contains an input signal and a test signal. The only change is on the Observer Problem, where the initial condition of the chaotic system is different for training and testing. The parameters to generate the signals are the same for training and testing. What makes the data different is due to the randomness the algorithms have to build the data, as it can be observed in Figure 2.

Two different tasks are employed to evaluate the performance of the Clustered ESNs (CESNs): the estimation of an unobserved variable of the chaotic system, specifically, Rössler system at chaotic regime, and a frequency filtering task. Both tasks are explained in detail below.

2.3.1 *Chaotic System 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



(a) Train set (b) Test set

Fig. 2. The image shows one of the training and testing frequency filtering datasets.

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:

$$\begin{aligned} \frac{dx}{dt} &= -y - z, \\ \frac{dy}{dt} &= x + ay, \\ \frac{dz}{dt} &= b + z(x - c). \end{aligned} \tag{9}$$

In this paper, 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 4th 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 testing, are chosen randomly from a uniform distribution on $[0, 1]^3$.

2.3.2 *Frequency Filtering.* For the second task which we employ for our comparison, 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 CESN's tasks are to let a certain frequency pass and filter out the others.

3. EXPERIMENTAL RESULTS

The main goal is to compare the performance between the ESNs and the Clustered ESNs (CESNs). To better evaluate the performance of both methods, a numerical analysis will be made to find the best set of parameters for the networks that lead to the best performance, and then compare the results. While evaluating a particular variable, some default values will be set to the other parameters as stated in the table I. To measure the performance it will be used the Normalized Root Mean Square Error (NRMSE). 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 System, and get as

Table I. 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}

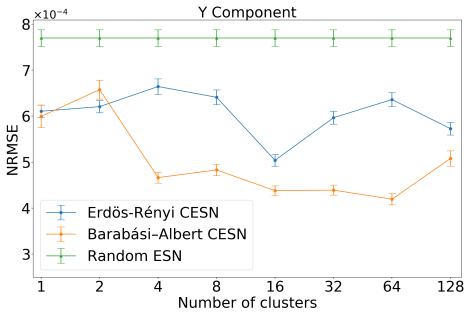
output the components y and z. The ESNs can solve this problem with high precision as shown in [Lu et al. 2017], but the goal here is to compare the performance of the ESNs with a random network in the reservoir and CESNs.

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 ($P_{\text{in}} = 0.75$, therefore $P_{\text{out}} = 0.25$). Moreover, for the Barabási clustered network the number of *initial vertices* was set to 1. Note that the image 3 shows only the results for the component y. The results for the component z were omitted since they are quite similar.

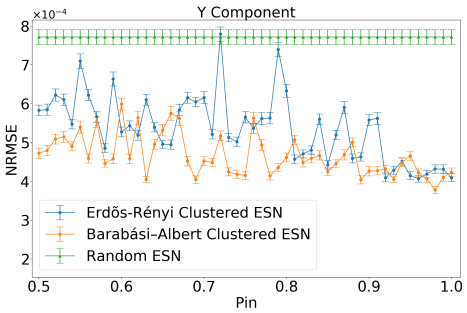
The first parameter to optimize is the number of clusters (C). As the Figure 3.a shows the value of this parameter is different for each of the CESNs. This experiment demonstrates that a better performance can be achieved once the optimal number of clusters is found. After fixing the number of clusters for each CESN, a new experiment is performed to find the optimal values for the mixture level ($P_{\text{in}}, P_{\text{out}}$).

As the image 3.b 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 random networks in the reservoir. Worth mentioning that the CESN with Barabási-Albert clusters has a slight advantage over the other two methods.

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 2.3.2. The idea is the same as in the previous experiment, which is to find the set of parameters that yields the best performance.



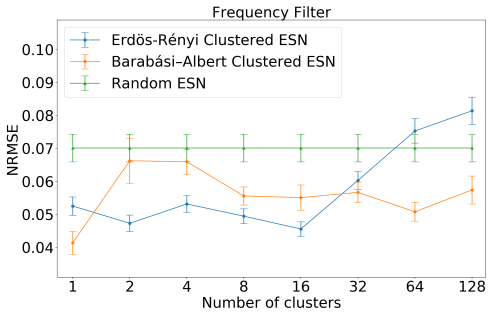
(a) Finding the optimal number of clusters.



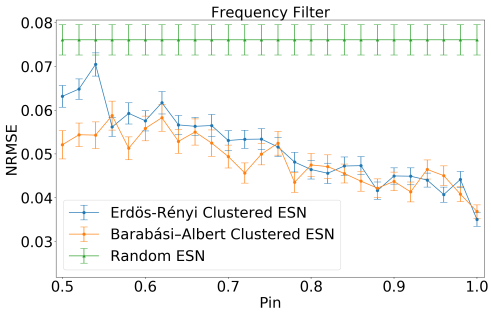
(b) Finding the optimal value for P_{in}.

Fig. 3. In the Image (a), the performance of the random ESN is constant as the number of cluster changes, and the performance of the CESNs varies depending on the number of clusters. The CESNs' performance increases once the optimal P_{in} is found.

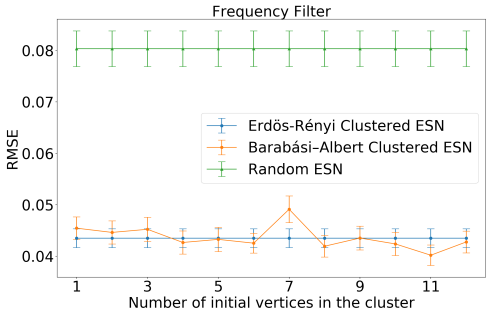
The next task to test the methods is the frequency filtering. To compare the performance of the



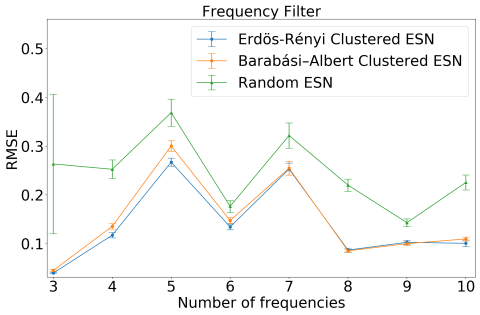
(a) Finding the optimal number of clusters.



(b) Finding the optimal value for P_{in}.



(c) Finding the optimal number of nodes.



(d) Performance for different number of frequencies.

Fig. 4. The four figures show the performance of the CESNs and the ESN over different parameter evaluations.

methods, the CESNs parameters need to be optimized. Comparing the results in figure 4.a, the CESNs also perform better than the random networks in the reservoir by only changing the number of clusters. For both CESNs methods, the number of clusters does have an impact on the performance.

The figure 4.b shows the performance of the clustered networks as the mixture level changes. The CESNs seem to improve 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.

Finding the optimal value for the parameters C and the P_{in} yields an increase in the performance as shown above. While some parameter optimizations are important in the overall performance, others not. As the initial number of nodes in the Barabási-Albert networks. Figure 4.c shows that NRMSE does not decrease as the initial number of nodes changes.

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 submitted to 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 $P_{\text{in}} \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

The performance of the methods follow the same trend, as shown in figure 4.d, oscillating in some cases. Since the set of parameters was smaller, there might be some room for improvement. Note the 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.

It is interesting that the tasks tested here have different characteristics, and the ESNs and its extensions can give good results in both of them. In all the experiments, the CESNs can outperform the ESNs after the parameter optimization phase.

4. DISCUSSION

ESNs are powerful tools due to their simplicity, the fast training process, and the flexibility to choose the network for the reservoir. The network can be chosen such that it is suitable for the according to application. ESNs with clustered networks as reservoirs (CESNs) can deliver much better results than classical ESNs for observer problems and frequency filtering tasks.

The key to achieving good performance for each task is the optimization of the CESN's hyperparameters. Comparing the presented CESN methods individually, their optimized performances in the experiments are equal.

These improved performances of CESNs are probably due to the selective nature of the clusters in the reservoir, where different clusters are responsible to capture different signal properties. As future works, a theoretical analysis will be conducted to get a better understanding of CESNs. Specifically, we will investigate the collective dynamics, like synchronization, among neurons in the reservoir related to the learning results.

REFERENCES

Appeltant, L., Soriano, M., Van Der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C., and Fischer, I. Information processing using a single dynamical node as complex system. *Nature Communications* vol. 2, pp. 1–6, 9, 2011.

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

Bollobas, B., Borgs, C., Chayes, J., and Riordan, O. Directed scale-free graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) ed. pp. 132–139, 2003.

Deng, Z. and Zhang, Y. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks* 18 (5): 1364–1375, Sep., 2007.

Erdös, P. and Renyi, A. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica* vol. 12, pp. 261–267, 1961.

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* vol. 148, 01, 2001.

Li, X., Zhong, L., Xue, F., and Zhang, A. A Priori Data-Driven Multi-Clustered Reservoir Generation Algorithm for Echo State Network - Fig 9. , 4, 2015.

Lin, X., Yang, Z., and Song, Y. The application of echo state network in stock data mining. In *Advances in Knowledge Discovery and Data Mining*, T. Washio, E. Suzuki, K. M. Ting, and A. Inokuchi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 932–937, 2008.

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., and Ott, E. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27 (4): 041102, 2017.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. Recent advances in physical reservoir computing: A review. *Neural Networks* vol. 115, pp. 100 – 123, 2019.

Wen, G., Li, H., and Li, D. An ensemble convolutional echo state networks for facial expression recognition. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*. pp. 873–878, 2015.