Time Series Clustering with Deep Reservoir Computing*

Miguel Atencia[1][0000−0002−5158−5905], Claudio Gallicchio[2][0000−0002−6692−2564], Gonzalo Joya[1][0000−0001−9256−0870], and Alessio Micheli[2][0000−0001−5764−5238]

[1] Universidad de Málaga
Campus de Teatinos, 29071 Málaga, Spain
{matencia, gjoya}@uma.es
[2] Department of Computer Science, University of Pisa
Largo B. Pontecorvo, 3, 56127 Pisa, Italy.
{gallicch, micheli}@di.unipi.it

Abstract. This paper proposes a method for clustering of time series, based upon the ability of deep Reservoir Computing networks to grasp the dynamical structure of the series that is presented as input. A standard clustering algorithm, such as k-means, is applied to the network states, rather than the input series themselves. Clustering is thus embedded into the network dynamical evolution, since a clustering result is obtained at every time step, which in turn serves as initialisation at the next step. We empirically assess the performance of deep reservoir systems in time series clustering on benchmark datasets, considering the influence of crucial hyper-parameters. Experimentation with the proposed model shows enhanced clustering quality, measured by the silhouette coefficient, when compared to both static clustering of data, and dynamic clustering with a shallow network.

Keywords: Clustering · Time series · Echo State Networks · Reservoir Computing.

1 Introduction

Cluster analysis can be described as the *discovery* of groups or categories into data where class labels are not available or simply data is not known to be organised in classes [1], thus clustering belongs to the set of unsupervised learning algorithms. Numerous methods have been proposed to deal with the problem of clustering, from which k-means is probably the best known. In particular, clustering of time series introduces several critical issues. First of all, the series to be clustered may have different lengths, or may even be regarded as infinite, which is usual in the context of signal processing and data streams, e.g. audio or video sequences. Also, time series usually contain temporal dependencies of

* This work has been partially supported by the Spanish Ministry of Science and Innovation through the Project TIN2017-88728-C2-1-R, as well as the Universidad de Málaga.

arbitrary length that cannot be captured by selecting fixed-length windows of data. Finally, casting a sequence into a vector misses the temporal information that emerges from the ordering: simply put, the values of the series at different moments in the past have varying relevance for the prediction of the future value. One way to summarise all these difficulties is the fact that computing simply the Euclidean distance between the series samples is not a good measure of whether the series are alike. For instance, a small phase shift of the *same* series would produce completely different values, while preserving the significant dynamical features. Therefore, it is not obvious how to compare two sequences, and a critical issue is the choice of similarity measure of time series. Much work has been recently dedicated to the topic of clustering of time series [2], but there is still a significant margin for improvement, where this work aims to contribute.

The paradigm of Reservoir Computing (RC) has emerged in the last two decades (see e.g. [11] and references therein), being characterised by a reduction of weight adaptation, which is the usual meaning of learning in conventional machine learning algorithms. In particular, Echo State Networks (ESNs) [10] comprise a set of hidden units or neurons (the reservoir), linked by recurrent connections that have feedback loops, but the weights of connections are fixed at the beginning and remain constant. Only weights in the output or *readout* layer are adjusted by a fitting algorithm, which is usually very simple, even linear. Since ESNs are dynamical systems that evolve through time, they are often used to deal with prediction of time series. Applications of ESNs to classification and clustering have also been proposed [13], but much remains to be done in this direction. In a previous work [4], a method for time series clustering was proposed by embedding a conventional clustering algorithm during the evolution of an ESN. In this work we further explore this topic by using a deep RC architecture [7], which has been shown to improve the memorisation abilities of standard reservoir networks [5], enabling multiple time-scales and multi frequency representations of the driving input signals [7–9]. Hierarchical reservoir architectures go beyond the dynamical properties of shallow reservoirs by implementing a pool of diversified (fading) memories of the driving input time series. Here we aim at empirically exploring the impact of such diversified pool of dynamics in the context of time series clustering.

The rest of this paper is structured as follows. Deep RC is introduced in Section 2, which also presents the standard shallow reservoir neural networks methodology as a sub-case. The proposed algorithm for time series clustering is formally described in Section 3, and then demonstrated on benchmark datasets in Section 4, thus providing an experimental validation of the method. Finally, some conclusions and directions for further research are presented in Section 5.

2 Deep Reservoir Computing

Reservoir Computing (RC) denotes a class of Recurrent Neural Networks in which the parameters of the recurrent hidden layer, the reservoir, are left untrained after initialisation. Deep RC extends this line of research considering

hierarchical organisations of the reservoir architectures, with the Deep Echo State Network (DeepESN) model [7] being a representative of the approach.

The dynamical component of a DeepESN is organised into a stacked composition of L reservoir layers. The external input signal drives the dynamics of the first layer, while each successive layer is driven by the activation of the previous layer in the stack. The state of the i-th reservoir layer at time step t, denoted by $\mathbf{h}^{(i)}(t)$ is computed by means of the following state update equation[3]:

$$\mathbf{h}^{(i)}(t) = (1 - \alpha^{(i)})\mathbf{h}^{(i)}(t-1) + \alpha^{(i)} \tanh\left(\mathbf{U}^{(i)}\mathbf{x}^{(i)}(t) + \mathbf{W}^{(i)}\mathbf{h}^{(i)}(t-1)\right), \quad (1)$$

where $\alpha^{(i)}$, $\mathbf{U}^{(i)}$ and $\mathbf{W}^{(i)}$ respectively indicate the leaking rate constant, the input weight matrix and the recurrent weight matrix for layer i in the deep reservoir architecture. Besides, $\mathbf{x}^{(i)}(t)$ indicates the driving input time series for layer i, which, for the first layer corresponds to the external input, i.e. $\mathbf{x}^{(i)}(t) \equiv u(t)$, which in this paper is considered as a one dimensional signal. For successive layers, the driving signal is the state of the preceding layer, i.e. $\mathbf{x}^{(i)}(t) \equiv \mathbf{h}^{(i-1)}(t)$ for $i > 1$. The hierarchical architecture of the deep reservoir is shown in Fig. 1. As initial condition, the state of each layer is set to a zero vector, i.e. $\mathbf{h}^{(i)}(0) = \mathbf{0}$ for all i. Noteworthy, when the reservoir architecture contains a single layer (i.e., $L = 1$) Equation 1 reduces to the state update equation of standard (shallow) reservoirs [10, 11].

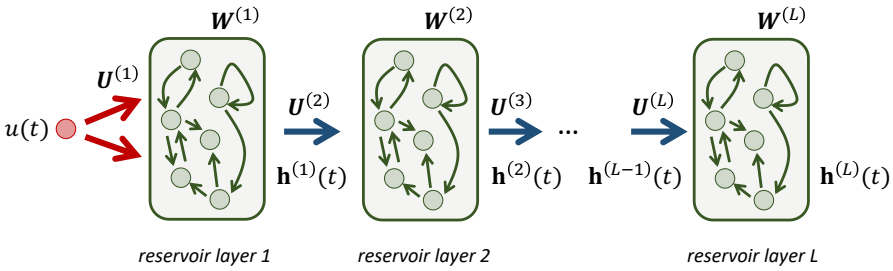

Fig. 1. Deep reservoir architecture.

.

The values in the weight matrices $\mathbf{U}^{(i)}$ and $\mathbf{W}^{(i)}$ are left untrained after initialisation under stability constraints expressed by the Echo State Property in [6]. For practical usage, this implies a random initialisation of the weight values, e.g. from a uniform distribution on $[-1, 1]$, followed by a re-scaling. In particular, the weights in $\mathbf{W}^{(i)}$ are scaled to control its effective spectral radius (the maximum among the eigenvalues in modulus), a hyper-parameter of the model indicated as $\rho^{(i)}$. Moreover, the input weight matrix for the first layer, i.e. $\mathbf{U}^{(1)}$, is re-scaled to have a maximum absolute weight value ω_{in}, which acts

[3] Bias terms are dropped to ease the notation.

as input scaling hyper-parameter. The weight matrices $\mathbf{U}^{(i)}$ for layers $i > 1$ are scaled similarly by a value ω_{il}, which is an inter-layer scaling hyper-parameter.

While in standard DeepESNs settings (in supervised learning contexts) the reservoir is coupled with a readout layer, in this paper we limit to consider only the network's states for the purposes of time series clustering.

3 Clustering with Deep Echo State Networks

The proposed algorithm for clustering of time series can be described as the sequential clustering of the states of a set of RC models. See a formal description in Algorithm 1. One reservoir architecture is built and evolved receiving as input each one of the time series within the data set, but all these reservoirs have the same values for all weights. Note that this means that if a total number of N neurons compose the reservoir, and there are n series in the dataset, clustering is performed on a matrix sized $N \times n$ **at each time step** t of the evolution of the networks. Arguably, the reservoirs allow for storing long-term dependencies on data that are lost if only the series samples themselves are used. For explanatory purposes, we consider that each basic clustering process is performed by the well-known k-means algorithm, but any iterative partition clustering method can be embedded, as long as the initial centroids of the clustering can be freely fixed. This is necessary because the key aspect of the proposed *dynamical* algorithm is that the centroids are initialised at time t with the result of the clustering at time $t-1$. The final result of each dynamic procedure is the clustering at the end of the time series. Contrarily to common usage, no readout layer performing supervised learning is included.

Algorithm 1 Dynamic clustering through evolution of the RC model.

Require: Dataset of n time series $\mathbf{u}_j$ with lengths l_j, $j = 1 \dots n$.
Ensure: k centroids
Initialise weight matrices $\mathbf{U}^{(i)}$, $\mathbf{W}^{(i)}$, $i = 1 \dots L$, and replicate n identical instances
Initialise all instances states $\mathbf{h}^{(i)}(0) = 0$, $i = 1 \dots L$
for $t = 1$ to $\max_j l_j$ **do**
  for $j = 1$ to n **do**
    if $t \leq l_j$ **then**
      Update the corresponding ESN instance by Equation (1)
    end if
  end for
  if $t = 1$ **then**
    Initialize centroids
  else
    Set initial centroids to centroids resulting from step $t-1$
  end if
  Build the dataset $\mathbf{Y}(t)$ of n reservoir states, where $\mathbf{Y}_{i\cdot}(t) = \left(\mathbf{h}^{(1)}, \dots \mathbf{h}^{(L)}\right)$
  Compute centroids at step t from clustering of dataset $\mathbf{Y}(t)$
end for

Algorithm 1 describes the application of a deep RC model to clustering of a time series. Note that the standard shallow model is recovered simply by setting $L = 1$, i.e. a fully connected single layer. The notation of line 14 represents that a single vector is formed with all units in the reservoir, simply by concatenating neurons belonging to each layer. An alternative clustering can be designed by using for clustering only the units of a layer. In this way, L independent clustering results can be obtained from the same deep architecture. We analyse below the performance of these three modalities: shallow ESN, deep ESN considering the full reservoir, and each one of the layers of the deep architecture.

4 Experimental results

In this section we aim at exploring the behaviour of RC neural networks with respect to the clustering of time series. To that end, we apply the Algorithm 1 described in the previous Section 3 to two datasets artificially built as described in Section 4.1, following the experimental settings reported in Section 4.2. The results for both shallow and deep reservoir architectures are presented in Sections 4.3 and 4.4, respectively on the two datasets used.

4.1 Data

Two different time series datasets are built for the evaluation of the proposed clustering method. The first one is constituted by different versions of the Multiple Superimposed Oscillator (MSO), which has already used as a challenging benchmark for learning with ESNs [14, 9]. The second dataset results from the evaluation of mathematical formulae with stochastic components, leading to different synthetic modes (SMs), and has been proposed in the context of time series similarity queries [3].

MSOs are sums of sinusoidal functions:

$$s(t) = \sum_{i=1}^{n} \sin(\varphi_i t) \tag{2}$$

where the frequencies φ_i are assigned the following values: $\varphi_1 = 0.2$, $\varphi_2 = 0.331$, $\varphi_3 = 0.42$, $\varphi_4 = 0.51$, $\varphi_5 = 0.63$, $\varphi_6 = 0.74$, $\varphi_7 = 0.85$, $\varphi_8 = 0.97$, $\varphi_9 = 1.08$, $\varphi_{10} = 1.19$, $\varphi_{11} = 1.27$, $\varphi_{12} = 1.32$. In our experiments, each series is built by, first, randomly setting a value for $n \in \{1, 4, 8, 12\}$ and then building a series by $u(t) = s(t + T)$ where T is a uniform random variable. Finally, we obtain a dataset comprising 160 series of length 500. A realisation of these series for each value of n is shown in Figure 2. The difficulty of this task is that it requires to properly separate input time series with different frequency content, a known challenge for standard RC system.

In contrast to the oscillatory nature of the MSOs, SM time series result from the composition of simple linear functions. Time series are extracted by selecting

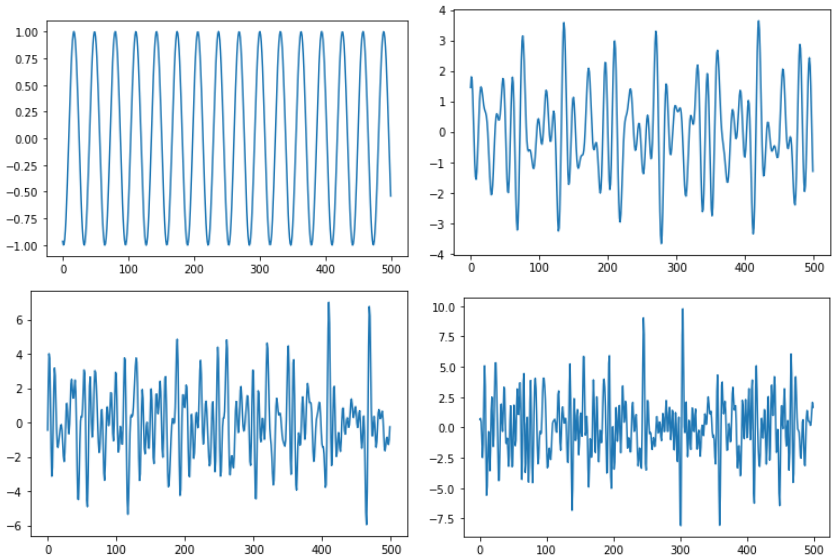

Fig. 2. Example realisations of Multiple Superimposed Oscillators for $n \in \{1, 4, 8, 12\}$.

one from the following 6 types:

$$\begin{aligned} s(t) &= 30 + 2\,R_t \\ s(t) &= 30 + 2\,R_t + A\,\sin 2\pi t/T \\ s(t) &= 30 + 2\,R_t + G_t\,t \\ s(t) &= 30 + 2\,R_t - G_t\,t \\ s(t) &= 30 + 2\,R_t + I_{\{t>K\}}\,t \\ s(t) &= 30 + 2\,R_t - I_{\{t>K\}}\,t \end{aligned} \tag{3}$$

where A, T, K, and each R_t, G_t are uniform distributions (over different intervals), and I_S is the indicator function of the set S. In the experiments, each one of these synthetically generated data will constitute the external input to the reservoir system (i.e., to the first layer in the deep setting), with $u(t) = s(t)$. We create a total of 180 independent series of length 100, chosen randomly from the six modes defined in Equation (3). In Figure 3 a realisation of each type is shown for illustration purposes. Here the challenge is to properly represent a variety of input signals with heterogeneous behaviours.

In order to establish a baseline, a conventional clustering algorithm has also been performed on data, considered as standard vectors. In other words, all the series belonging to a dataset are stacked together yielding a large matrix X, and then k-means is applied to the rows of X. This procedure is called "static" clustering in this paper, to distinguish it from the proposed algorithm that is executed during the dynamical evolution of the reservoir neural network.

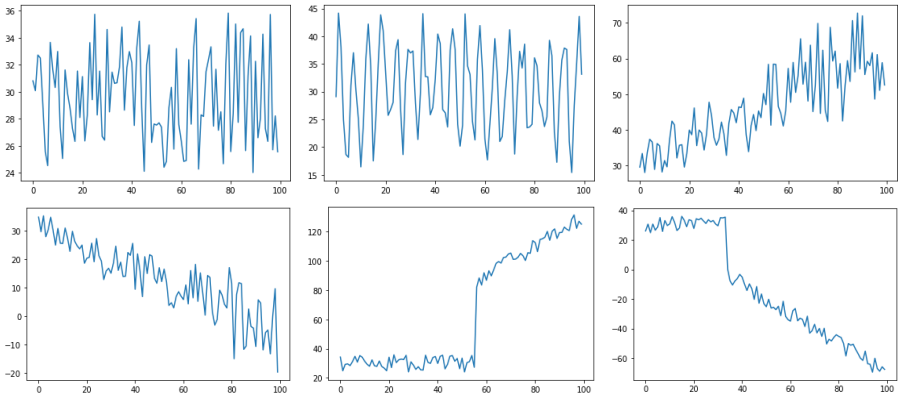

Fig. 3. Example realisations of Synthetic Modes for each class of Equation (3).

4.2 Experimental Settings

In order to apply the proposed algorithm to the datasets described in the previous section, shallow and deep reservoir architectures are built, with the total number of recurrent units being 100 in both cases. For the DeepESN, these 100 units are structured forming 10 layers with 10 units each, i.e., $L = 10$. For the shallow ESN case, the 100 recurrent units are organised in one single layer, i.e., $L = 1$. Preliminary experimentation revealed that results were strongly dependent on the choice of hyper-parameters. Therefore, an exhaustive cross-validation was performed in order to determine optimal settings of the four hyper-parameters that are considered most important: spectral radius ρ, leaking rate α, input scaling ω_{in}, and inter-layer scaling ω_{il} (the last one, only for deep reservoirs). Hyper-parameters are chosen from values shown in Table 1, leading to 192 possible settings. To simplify the construction, all layers share the same spectral radius for the self-connection (recurrent) matrices, the same leaking rate, and the same inter-layer scaling. The number of clusters is set to $k = 10$ for all experiments.

Table 1. Values of hyper-parameters used in the experiments with both tasks. All combinations are generated leading to 192 hyper-parameter sets.

Spectral radius	ρ	0.6	0.9	1.2	1.5
Leaking rate	α	0.5	0.7	0.9	
Input scaling	ω_{in}	0.5	1	1.5	2
Inter-layer scaling	ω_{il}	0.5	1	1.5	2

For each experiment, the goodness of the clustering is measured by the silhouette coefficient [12]: a value between -1 and 1 that can be roughly considered as the probability that a particular data instance belongs to the assigned cluster.

Then, this value is averaged for all data, so the closer the value to 1, the more compact the clustering. For each model, and for each one of the hyper-parameter combinations, 100 runs are performed on the same dataset with different, independent weight initialisations of the reservoir architectures, as well as centroid initializations of the k-means method (at $t = 1$), to account for stochastic fluctuations due to initialisation of network weights and cluster centroids.

4.3 Clustering of Multiple Superimposed Oscillators

In order to ascertain the relation between the dynamical behaviour of the network and the clustering results, under different combinations of hyper-parameters, we apply the described algorithm to the MSOs data. Both the shallow and deep architecture are implemented and, besides, as mentioned above, for the deep architecture the clustering may be performed by using either the whole reservoir or each one of the layers independently. Finally, the silhouette coefficient is computed and the hyper-parameters that led to the best result for each one of the architectures are recorded. The results are summarised in Table 2, which also reports the corresponding values of the hyper-parameters.

Table 2. Best results for each architecture and layer, measured by the silhouette coefficient, with $k = 10$, for the MSO task. For each result, the values of the corresponding hyper-parameters are shown.

Architecture	Clustering	silh	ρ	α	ω_{in}	ω_{il}
Static	Data vector	0.12				
Shallow 100	Full reservoir	0.44	0.60	0.90	2.00	
Deep 10 x 10	Full reservoir	**0.84**	1.50	0.70	0.50	2.00
Deep 10 x 10	Layer 1	0.90	1.50	0.50	0.50	1.00
Deep 10 x 10	Layer 2	0.91	1.50	0.50	1.00	0.50
Deep 10 x 10	Layer 3	0.92	1.50	0.50	0.50	0.50
Deep 10 x 10	Layer 4	0.92	1.50	0.70	0.50	0.50
Deep 10 x 10	Layer 5	0.96	1.50	0.70	1.50	0.50
Deep 10 x 10	Layer 6	0.94	0.60	0.50	2.00	0.50
Deep 10 x 10	Layer 7	0.94	1.50	0.70	0.50	0.50
Deep 10 x 10	Layer 8	0.96	1.20	0.50	1.50	1.00
Deep 10 x 10	Layer 9	**0.98**	1.50	0.70	1.50	0.50
Deep 10 x 10	Layer 10	0.96	1.20	0.50	2.00	1.50

The analysis of results in Table 2 shows, first of all, that a conventional clustering considering the series as a standard vector does not provide a satisfactory performance, measured in terms of the silhouette coefficient. We interpret that this is due to the fact that casting a sequence into a vector misses the temporal information that is contained in the ordering. The clustering performed on the states of the shallow architecture with 100 neurons, as described in Section 3, considerably improves with respect to the vectorized data, but it is still outperformed by the the deep architecture. Remarkably, the deep network (full

reservoir setting) almost doubles the performance of the shallow counterpart, indicating a substantially improved ability to represent input series with multiple frequency content. Moreover, in the deep architectural organisation, a differential behaviour is observed when layering is taken into account in the clustering: deeper layers (farther from input) exhibit increasing silhouette coefficient, evidencing a more complex dynamics that is grasping information about the temporal input features. Beside the best result obtained by each architecture in Table 2 it is indicated the hyper-parameter set that led to such result.

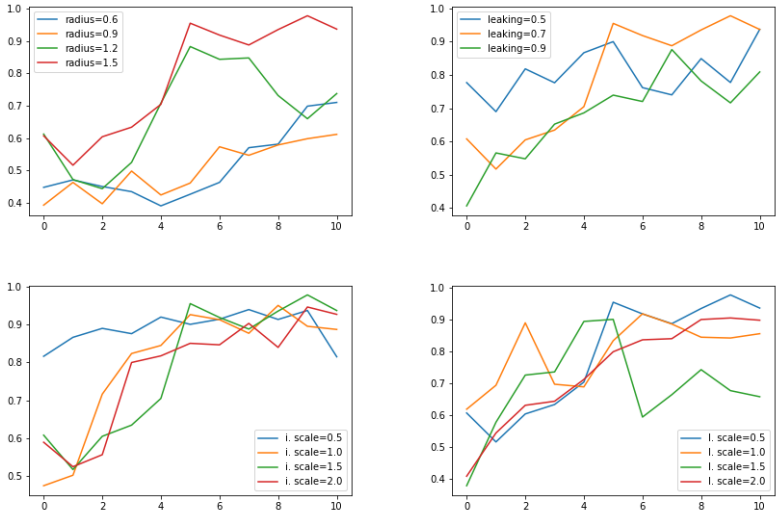

Fig. 4. Silhouette coefficient for increasing layer depth on the X-axis, for different values of spectral radius (top, left), leaking rate (top, right), input scaling (bottom, left), and interlayer scaling (bottom, right) on the task MSO. For each plot, the other three hyper-parameters are as in the penultimate row of Table 2.

As a final investigation on the clustering performance of the ESN under the influence of different choices of hyper-parameters, we plot in Figure 4 the silhouette coefficients obtained by each layer, where clustering on the full reservoir is represented as layer 0. Starting from the configuration that achieved the global maximum, namely the shown in the penultimate row of Table 3, each hyper-parameter is iterated along the possibilities in the considered range (shown in Table 1). Observing for instance the influence of the spectral radius in the top left plot, looking at the graph for the value 1.5, it is hardly surprising that the maximum is achieved by level 9, since this particular hyper-parameter combination was precisely selected by this feature. It is though more interesting that a general increasing trend is apparent, instead of a random distribution with a peak at level 9 that could have resulted from overfitting. Also, a stronger

and more consistent influence of spectral radius and leaking rate compared to the scaling hyper-parameters is observable, which is coherent with the general knowledge on dynamics of reservoir computing.

4.4 Clustering of Synthetic Modes

A similar set of experiments has been performed on data coming from the SM task as defined in Equation (3), covering the same combinations of hyper-parameters as in Table 1. In this way, cross-validation results, shown in Table 3, allow to select optimal hyper-parameter settings for each architecture. Clustering with reservoir networks is again obviously advantageous with respect to the baseline static clustering, even for the shallow network. Also in this case we observe the beneficial effect of layering, although albeit less pronounced than in the previous MSO case. In this respect, results indicate that even in a case in which a shallow reservoir network is able to achieve an excellent clustering performance, a deep reservoir architecture is still (slightly) advantageous (and adding more layers improves, though not greatly, the achieved results).

Table 3. Best results for the SM task with each architecture and layer, measured by the silhouette coefficient, with $k = 10$. For each result, the values of the corresponding hyper-parameters are shown.

Architecture	Clustering	silh	ρ	α	ω_{in}	ω_{il}
Static	Data vector	0.57				
Shallow 100	Full reservoir	0.90	0.60	0.90	2.00	
Deep 10 x 10	Full reservoir	**0.91**	0.60	0.70	2.00	1.00
Deep 10 x 10	Layer 1	0.96	0.60	0.70	2.00	1.00
Deep 10 x 10	Layer 2	0.95	1.50	0.50	1.50	2.00
Deep 10 x 10	Layer 3	0.95	1.50	0.50	2.00	1.00
Deep 10 x 10	Layer 4	0.93	0.90	0.90	1.00	2.00
Deep 10 x 10	Layer 5	0.94	1.20	0.50	1.00	1.00
Deep 10 x 10	Layer 6	0.94	1.20	0.50	2.00	0.50
Deep 10 x 10	Layer 7	0.95	1.50	0.70	0.50	2.00
Deep 10 x 10	Layer 8	0.95	1.50	0.70	1.00	1.50
Deep 10 x 10	Layer 9	0.95	1.20	0.50	1.50	1.00
Deep 10 x 10	Layer 10	**0.97**	1.20	0.50	1.50	1.00

It is also remarkable that, for the SM dataset, several results had to be discarded because they provided meaningless results. For instance, in some cases all reservoir states converged to zero or saturated to one, regardless of the input. Then, the corresponding series were all assigned to a single cluster leaving most other clusters depopulated. When this occurred, the silhouette coefficient was approximately one yet this could not be considered as a satisfactory clustering. This phenomenon can be attributed to an insufficient excitation of the network due to the information provided by the input becoming negligible, and deserves further exploration in future works.

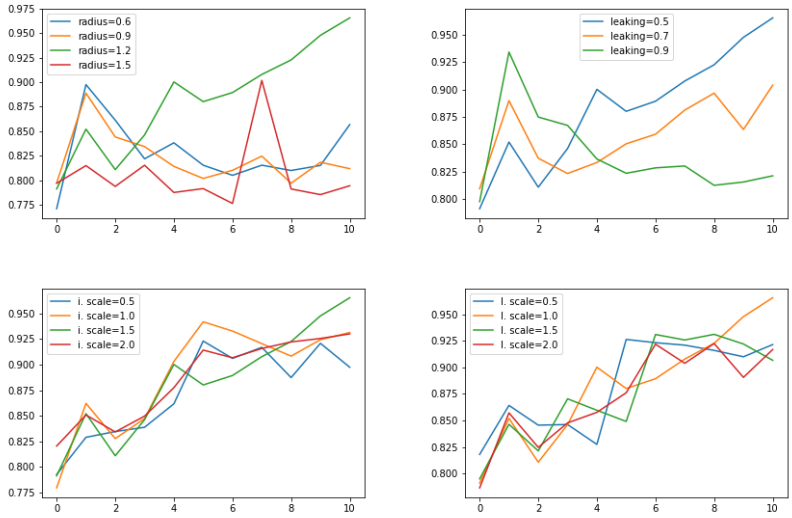

Fig. 5. Silhouette coefficient for increasing layer depth on the X-axis, for different values of spectral radius (top, left), leaking rate (top, right), input scaling (bottom, left), and interlayer scaling (bottom, right) on the task SM. For each plot, the other three hyper-parameters are as in last row of Table 3.

The interrelations of layer depth and each hyper-parameter setting are illustrated by Figure 3. As observed in the graphs of the bottom row, results are better for deeper layers regardless the value of the scaling hyper-parameters. Although this trend is also visible for the optimal combination of hyper-parameters in the plots of spectral radius and leaking rate, for other settings a more erratic behaviour remains to be explained.

5 Conclusions

In this work we have implemented a clustering algorithm on the states of reservoir computing architectures along their dynamical evolution. The results suggest that this idea contributes to grasp the temporal features of a time series, which are lost when the series is formulated as a static vector. When comparing shallow to deep reservoir algorithms, it was observed that the latter presented an enhanced capability to provide a good clustering, as measured by the silhouette coefficient. The fact that this effect was more evident for series with multiple oscillations reinforces the notion that layered recurrent architectures are naturally most suitable to deal with several time scales by developing diversified fading memories in the different layers.

Looking ahead to future developments, an interesting direction is to explore extensions in the context of supervised problems. Interestingly, in cases in which

data with class labels are available, such supervision information could be used to drive the clustering, for instance by exploiting output feedback connections from the output to the reservoir units. This could be useful in real-world problems of time series classification, e.g. for medical diagnosis from ECG data. Remarkably, our algorithm can deal with series of variable length, simply by stopping evolution when input is finished, whereas a conventional clustering method must be applied on a rectangular matrix, which cannot be formed from variable-length rows. In light of such considerations, extensive applications to real-world problems are also planned as future works.

References

1. Aggarwal, C.C., Reddy, C.K.: Data clustering : algorithms and applications. CRC Press (2014)
2. Aghabozorgi, S., Seyed Shirkhorshidi, A., Ying Wah, T.: Time-series clustering - A decade review. Information Systems **53**, 16–38 (2015)
3. Alcock, R.J., Alcock, R.J., Manolopoulos, Y.: Time-series similarity queries employing a feature-based approach. 7th Hellenic Conference on Informatics pp. 27—-29 (1999)
4. Atencia, M., Stoean, C., Stoean, R., Rodríguez-Labrada, R., Joya, G.: Dynamic Clustering of Time Series with Echo State Networks. In: Lecture Notes in Computer Science. vol. 11507 LNCS, pp. 73–83. Springer, Cham (2019)
5. Gallicchio, C.: Short-term memory of deep rnn. In: 26th European Symposium on Artificial Neural Networks (ESANN). pp. 633–638. i6doc. com publ (2018)
6. Gallicchio, C., Micheli, A.: Echo state property of deep reservoir computing networks. Cognitive Computation **9**(3), 337–350 (2017)
7. Gallicchio, C., Micheli, A., Pedrelli, L.: Deep reservoir computing: A critical experimental analysis. Neurocomputing (2017)
8. Gallicchio, C., Micheli, A., Pedrelli, L.: Design of deep echo state networks. Neural Networks **108**, 33–47 (2018)
9. Gallicchio, C., Micheli, A., Pedrelli, L.: Hierarchical temporal representation in linear reservoir computing. In: Smart Innovation, Systems and Technologies, vol. 102, pp. 119–129. Springer Science and Business Media Deutschland GmbH (2019)
10. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. science **304**(5667), 78–80 (2004)
11. Jaeger, H., Lukoševičius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky- integrator neurons. Neural Networks **20**(3), 335–352 (2007)
12. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**(C), 53–65 (1987)
13. Tanisaro, P., Heidemann, G.: Time series classification using time warping invariant echo state networks. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 831–836 (2016)
14. Wierstra, D., Gomez, F.J., Schmidhuber, J.: Modeling systems with internal state using evolino. In: GECCO 2005 - Genetic and Evolutionary Computation Conference. pp. 1795–1802. ACM Press, New York, New York, USA (2005)