



Exploring unsupervised pre-training for echo state networks

Peter Steiner¹ · Azarakhsh Jalalvand² · Peter Birkholz¹

Received: 14 November 2022 / Accepted: 22 August 2023 / Published online: 24 September 2023
© The Author(s) 2023

Abstract

Echo State Networks (ESNs) are a special type of Recurrent Neural Networks (RNNs), in which the input and recurrent connections are traditionally generated randomly, and only the output weights are trained. However, recent publications have addressed the problem that a purely random initialization may not be ideal. Instead, a completely deterministic or data-driven initialized ESN structure was proposed. In this work, an unsupervised training methodology for the hidden components of an ESN is proposed. Motivated by traditional Hidden Markov Models (HMMs), which have been widely used for speech recognition for decades, we present an unsupervised pre-training method for the recurrent weights and bias weights of ESNs. This approach allows for using unlabeled data during the training procedure and shows superior results for continuous spoken phoneme recognition, as well as for a large variety of time-series classification datasets.

Keywords ESN · RCN · Clustering · State machine

1 Introduction

Reservoir Computing (RC), particularly the Echo State Network (ESN), describes an emerging research area of randomly initialized Recurrent Neural Networks (RNNs), in which traditionally only a very small amount of all the weights are trained using linear regression [17]. ESNs have achieved state-of-the-art results in various directions, such as speech recognition [50, 51], image classification [19], music analysis [42, 43], plasma control [20, 21], and time-series forecasting [28, 30, 41, 52, 53, 57]. Furthermore, several toolboxes have recently been developed with different use-cases [29, 45, 54], which help to pave the way

for applying ESNs and other RC structures for a broader community.

In recent years, numerous approaches aimed to optimize the typically un-trained and fixed random input and recurrent weights, as there are rising arguments that a purely random initialization might not be the ideal way of setting up ESNs [27, 31, 38]. In fact, it was proposed in Lukoševičius et al. [27] that a data-driven way of weight initialization should lead to a superior behavior of such customized ESNs. However, of course, the main idea of the basic ESN with its simple training approach should not be forgotten.

Jalalvand et al. [18, 19] analyzed the hyper-parameters of an optimized ESN. Aceituno et al. [1] showed how the eigenspectrum (ordered absolute eigenvalues) of the recurrent weights influence the performance of an ESN. As a consequence, various optimization strategies [15, 19, 43] for ESNs were proposed to efficiently tune an ESN task-dependently.

Another venue of research has been the deterministic design of ESNs. Initially, in Rodan and Tino [36], very simple ESNs with a fixed reservoir were analyzed. Therefore, the authors proposed three types of ESNs, namely the delay line reservoir (DLR) with optional feedback (DLRB) and the simple cycle reservoir (SCR). The weight values were zero for all but for one specific value that was assigned to weights on the lower subdiagonal of the weight

✉ Peter Steiner
peter.steiner@tu-dresden.de
Azarakhsh Jalalvand
azarakhsh.jalalvand@princeton.edu
Peter Birkholz
peter.birkholz@tu-dresden.de

¹ Institute for Acoustics and Speech Communication, Technische Universität Dresden, 01069 Dresden, Saxony, Germany

² Mechanical and Aerospace Engineering Department, Princeton University, Princeton, NJ 08544, USA

matrix and another value on the upper subdiagonal in case of the DLRB. An entirely randomly initialized ESN was compared with these three simplified deterministic models for different tasks (time-series prediction and spoken digit classification), and it turned out that the proposed simplifications performed as well as the basic ESN model regardless of the task. In Rodan and Tiño [37] and in Sun et al. [47, 48], the SCR architecture (previously the most promising model) was expanded to include jump connections. In Strauss et al. [46], and in Griffith et al. [14], and in Verzelli et al. [56], yet other families of deterministic ESNs were designed. The results of these studies were similar — deterministic ESNs performed equivalently to randomly initialized ESNs and were better explainable, especially the eigenspectrum of the recurrent weights. Recently, the experiments from Rodan and Tino [36] were repeated for the deep and graph ESNs [8, 9] with similar results.

Overall, all these approaches either try to tune the reservoir parameters task-dependently or to initialize the weight matrices of an ESN in a deterministic way, independent of the task. Other approaches, such as [2, 3, 25, 40], still aimed to randomly initialize the weight matrices of an ESN but adapted the Self-Organizing Maps (SOM) algorithm [24] and Hebbian learning [11] to unsupervisedly adjust the weight matrices in an iterative fashion that is biologically plausible.

In Steiner et al. [44], we proposed an unsupervised method to pre-train the input weights of ESNs with the K -Means algorithm [26]. We showed that passing a feature vector to a neuron in the reservoir via the input weight matrix is closely related to computing the cosine similarity between the feature vector and the input weights that are connected to a specific neuron. By replacing the randomly initialized input weights by the K -Means centroids, we maximized the activation of those neurons, whose input weights are most similar to the current feature vector. This approach not only outperformed randomly initialized ESNs for many use-cases but also allows utilizing unlabeled data to learn the input weights [42].

In this paper, we propose an extension of our work in Steiner et al. [44]. Inspired by state machines from statistical signal processing, we aim to pre-train the bias weights and the recurrent weights of ESNs in an unsupervised fashion. In addition to our work in Steiner et al. [44], we not only consider the centroids of the K -means algorithm and use them as the input weights. In particular, we consider the clusters to which inputs were assigned, and propose to use their relative frequencies as the bias weights, and the transition probabilities between subsequent clusters as the reservoir weights. We show that this approach is motivated by the way in which the current reservoir state is computed from the current input and the previous reservoir state. Our approach

1. performs equally well or better than baseline models for the tasks of continuous phoneme recognition and multivariate time-series classification of many time series with different properties,
2. shows that the approach combines the two borderline cases of purely randomly or deterministically initialized ESNs,
3. proposes a way to interpret the learned weights given a dataset.

The reason for focusing on the first application of continuous phoneme recognition is that speech may follow phonotactic rules that define which phonemes follow each other. As illustrated in Sect. 3, the proposed training strategy tunes the recurrent weights in a way such that they approximate phonotactic rules. The reason for focusing on the second application of multivariate time-series classification is that this is a well-defined benchmark test for algorithms that deal with temporal data, and the broad variety of considered time-series covers many different use-cases and input data.

The remainder of this paper is organized as follows: In Sect. 2, we briefly review the basic ESN, the K -Means ESN and our proposed training strategy for the recurrent weights. In Sect. 3, we experimentally introduce, analyze and evaluate the method for phoneme recognition in speech data, and in Sect. 4, we extend the study to a broad variety of multivariate datasets with different characteristics. We conclude with a summary of the most important results in Sect. 5 and a possible outlook to future work in Sect. 6.

2 Methods

2.1 Basic echo state network

A basic ESN as depicted in Fig. 1 consists of three weight matrices: The input weights \mathbf{W}^{in} , the reservoir weights \mathbf{W}^{res} and the output weights \mathbf{W}^{out} .

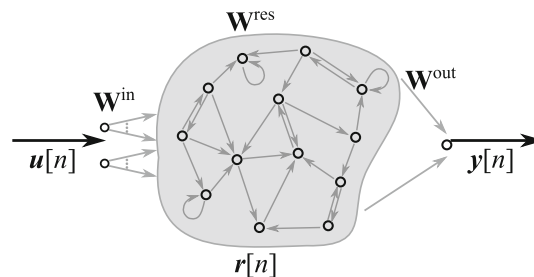


Fig. 1 Main components of a basic ESN: The input features $u[n]$ are fed into the reservoir using the fixed input weight matrix \mathbf{W}^{in} . The reservoir consists of unordered neurons, sparsely interconnected via the fixed reservoir matrix \mathbf{W}^{res} . The output $y[n]$ is a linear combination of the reservoir states $r[n]$ based on the output weight matrix \mathbf{W}^{out} , which is trained using linear regression

The input weight matrix \mathbf{W}^{in} has the dimension $N^{\text{res}} \times N^{\text{in}}$, where N^{res} and N^{in} are the size of the reservoir and the input dimension, respectively. \mathbf{W}^{in} is responsible to transfer a given input $\mathbf{u}[n]$ (time index n) to all reservoir neurons. The weight values are typically drawn from a uniform distribution between ± 1 and later scaled by the hyper-parameter “input scaling” α_u . According to Jalalvand et al. [19], it is sufficient to have only a limited number of connections from each input node to the reservoir neurons. In this paper, we define $K^{\text{in}} = \min(N^{\text{in}}, 10)$ as the number of connections from an input node to each neuron. Especially for high-dimensional inputs, this leads to a very sparse \mathbf{W}^{in} .

The reservoir weight matrix \mathbf{W}^{res} is a $N^{\text{res}} \times N^{\text{res}}$ square matrix, where the values are initialized from a standard normal distribution. Similar to the input weights, we connect each reservoir neuron only to $K^{\text{rec}} = 10$ randomly selected nodes in the reservoir and set the remaining weights to zero. Afterwards, \mathbf{W}^{res} is normalized by its largest absolute eigenvalue to fulfill the echo state property [17]. Finally, \mathbf{W}^{res} is scaled task-dependently by the hyper-parameter “spectral radius” ρ .

Since the input scaling α_u and the spectral radius ρ determine how strongly the network relies on the memorized past inputs compared with the current input, these hyper-parameters need to be tuned jointly in the optimization process.

Since the ESN consists of “traditional” analogue neurons, every neuron receives an additional constant bias input. The bias weight vector \mathbf{w}^{bi} with N^{res} entries is initialized from a uniform distribution between ± 1 , scaled by the hyper-parameter α_{bi} .

With these three weight matrices, the reservoir state $\mathbf{r}[n]$ can be computed with

$$\mathbf{r}[n] = (1 - \lambda)\mathbf{r}[n - 1] + \lambda f_{\text{res}}(\mathbf{W}^{\text{in}}\mathbf{u}[n] + \mathbf{W}^{\text{res}}\mathbf{r}[n - 1] + \mathbf{w}^{\text{bi}}), \tag{1}$$

which is a leaky integration of the reservoir neurons. This is equivalent to a first-order low-pass filter, where the leakage $\lambda \in (0, 1]$ determines, which amount of the past reservoir state is leaked over time. Together with ρ , the leakage λ determines the temporal memory of the reservoir.

The reservoir activation function $f_{\text{res}}(\cdot)$ determines the nonlinearity of the ESN. In this paper, we use the tanh nonlinearity.

The output weight matrix \mathbf{W}^{out} with the dimensions $N^{\text{out}} \times (N^{\text{res}} + 1)$ connects the reservoir state $\mathbf{r}[n]$, which is expanded by a constant intercept term of 1 for regression, to the output vector $\mathbf{y}[n]$ using

$$\mathbf{y}[n] = \mathbf{W}^{\text{out}}\mathbf{r}[n]. \tag{2}$$

The output weight matrix contains the only weights to be trained and is typically computed using ridge regression. Therefore, all reservoir states calculated for the training data are collected into the reservoir state collection matrix \mathbf{R} ; all desired outputs $\mathbf{d}[n]$ are collected into the output collection matrix \mathbf{D} .

With these matrices, ridge regression is solved as

$$\mathbf{W}^{\text{out}} = (\mathbf{R}\mathbf{R}^T + \epsilon\mathbf{I})^{-1}(\mathbf{D}\mathbf{R}^T), \tag{3}$$

where ϵ is the regularization parameter that needs to be tuned on a validation set. Since Eq. (3) can be solved in closed form, ESNs are quite efficient and fast to train compared to typical deep-learning approaches. Furthermore, they are adaptive, since new training data can always be used to update the output weights using incremental regression [20].

2.2 K-Means Echo State Network

In Steiner et al. [44], we introduced a novel way to initialize or pre-train the input weight matrix \mathbf{W}^{in} based on the K-Means algorithm.

The K-Means algorithm aims to group a set of feature vectors with N^{in} features in K clusters, and each observation is assigned to the cluster with the closest centroid according to the Euclidean distance [26]. This is visualized in the Voronoi diagram in Fig. 2, where a two-dimensional feature space is partitioned into four distinct regions. Each region is represented by the centroid μ_k of the k^{th} cluster C_k .

As we showed in Steiner et al. [44], passing a feature vector $\mathbf{u}[n]$ to the k^{th} reservoir neuron is equivalent to computing the dot product between $\mathbf{u}[n]$ and the weights to the k^{th} neuron. The dot product is closely related to the cosine similarity, and it gets large if $\mathbf{u}[n]$ and the weights are showing in similar directions, close to zero if they are orthogonal, and negative if $\mathbf{u}[n]$ and the weights are showing in opposite directions.

By replacing the random values of \mathbf{W}^{in} of a general ESN with the cluster centroids, each neuron is tuned to respond most strongly to a feature vector that lies within the area of the respective cluster described by its centroid μ_k . This resulted in a superior performance of the K-Means Echo State Network compared to the basic ESN for different kinds of data. In Steiner et al. [42], we showed that this approach can be used to include unlabeled data for the pre-training stage, because the K-Means algorithm does not require labeled data.

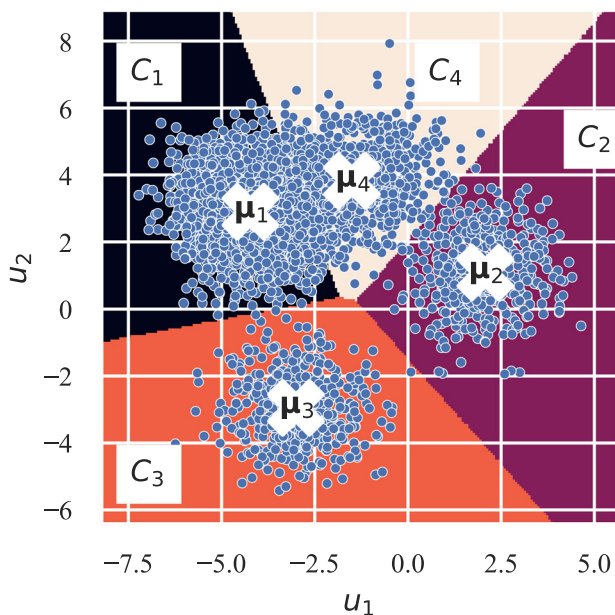


Fig. 2 Voronoi diagram for a two-dimensional feature space with four potential clusters

2.3 Unsupervised training of the reservoir weights

Now we consider the reservoir weights \mathbf{W}^{res} . From Eq. (1), it is known that the current state $\mathbf{r}[n]$ of ESNs as well as general RNNs only depend on the current input $\mathbf{u}[n]$ and the previous hidden/reservoir state $\mathbf{r}[n - 1]$. In this work, we propose a novel pre-training strategy for the reservoir weights. For this, we consider two neurons a and b that represent the clusters C_1 and C_4 in Fig. 2. The proposed pre-training scheme adjusts the weight between the two neurons in such a way that it reflects the probability that an input feature vector belonging to the cluster of b follows an input feature vector belonging to the cluster of a in the training data. We can approximate this probability by counting, how often the sequence of inputs $(\mathbf{u}[n - 1], \mathbf{u}[n])$ represents the transition between two neurons a and b . Hence, we can compute each recurrent weight w_{ij} in general using

$$w_{ij}^{\text{res}} = P(\mathbf{u}[n] \in C_j, \mathbf{u}[n - 1] \in C_i) \approx \frac{\#\{C_i, C_j\}}{\#\{C_i\}}, \quad (4)$$

where i and j are the indices of the previously and the currently selected clusters, respectively. The term $\#\{C_i, C_j\}$ denotes the absolute number of occurrences of successive input vectors $\mathbf{u}[n - 1]$ and $\mathbf{u}[n]$ belonging to the clusters C_i and C_j , respectively. The term $\#\{C_i\}$ denotes the absolute number of input vectors that belong to the cluster C_i . For the introductory example, this is visualized in the state transition graph (Fig. 3). We can, for example, observe that there are many transitions toward the cluster

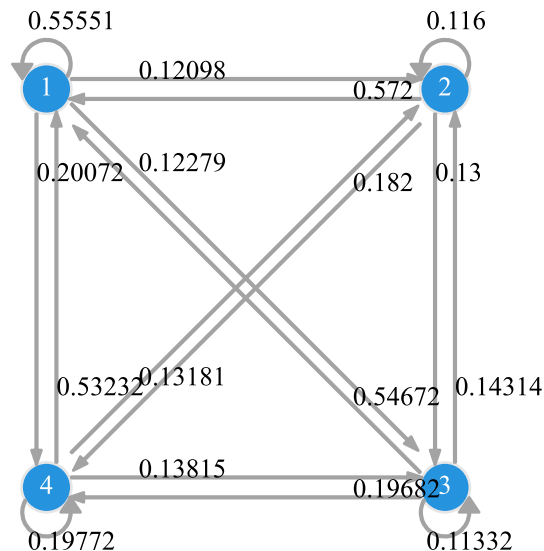


Fig. 3 State transition diagram for the clusters, where the numbers indicate transition probabilities. Since the clusters C_1 and C_4 lie closely together in the feature space, more transitions occur between these, than between the other clusters

C_1 (i.e., more than 50 %), which has the highest absolute frequency in this example. Furthermore, since the centroids μ_1 and μ_4 are close together, in general more transitions are happening between their clusters and there occur more self-transitions. An important aspect of this weight initialization scheme is that the weight values are always bounded between $[0, 1]$.

Overall, two factors determine the response of a reservoir neuron a at a given time:

1. The similarity of $\mathbf{u}[n]$ to the cluster centroid is represented by neuron a in terms of the input weights.
2. The degree to which the sequence of inputs $(\mathbf{u}[n - 1], \mathbf{u}[n])$ represents a probable transition between two neurons b and a .

This idea is loosely related to Hidden Markov models (HMMs), which were widely used for various recognition tasks with temporal dependencies, such as for speech recognition [34], alignment of musical performances and its scores [33] and for human activity recognition [32].

In this paper, we denote the models in which the input and recurrent weights are pre-trained as *KM-ESN 2-Gram*.

2.4 Unsupervised training of the bias weights

In addition to the reservoir weights, we propose a related strategy to initialize the bias weights \mathbf{w}^{bi} . From Eq. (1), it is known that each neuron receives the current input $\mathbf{u}[n]$ together with a constant 1, which is scaled by the bias weights. We still consider the two neurons a and b that represent the clusters C_1 and C_4 . We now adjust the bias

weights in a way that neurons, which represent frequently selected clusters, are emphasized more strongly than neurons that represent rarely selected clusters. Similar as in Sect. 2.3, this can be achieved in a probabilistic way. Therefore, we count how often each cluster was selected in the training data and divide it by the size of the training dataset according to Eq. (5)

$$w_i^{\text{bi}} = P(\mathbf{u}[n] \in C_i) \approx \frac{\#\{C_i\}}{\sum_{j=1}^{N^{\text{res}}} \#\{C_j\}}. \quad (5)$$

In Fig. 4, the relative frequencies of the different clusters of our toy example are visualized. It can be noticed that the clusters C_1 and C_4 occur more frequently than the other clusters. Consequently, the underlying neurons a and b get stronger bias weights (i.e., $w_1^{\text{bi}} \approx 0.55$, $w_1^{\text{bi}} \approx 0.20$) than the other neurons. Note that the bias weight values are also bounded between $[0, 1]$, as the reservoir weights.

In this paper, we denote the models in which the input and bias weights are initialized as *KM-ESN PB*. The term “PB” denotes “probabilistic bias”. If we combine our two newly proposed pre-training strategies, we refer to the models as *KM-ESN 2-Gram PB*.

3 Phoneme recognition

In the first experiment, we aim to demonstrate the effectiveness of our proposed pre-training strategy by applying it to continuous phoneme recognition from audio recordings. This task is particularly interesting because it allows us to analyze how the pre-trained weights capture the phonotactic rules that dictate which phonemes can follow others with specific probabilities in each language. Although we do not anticipate that clusters will strictly

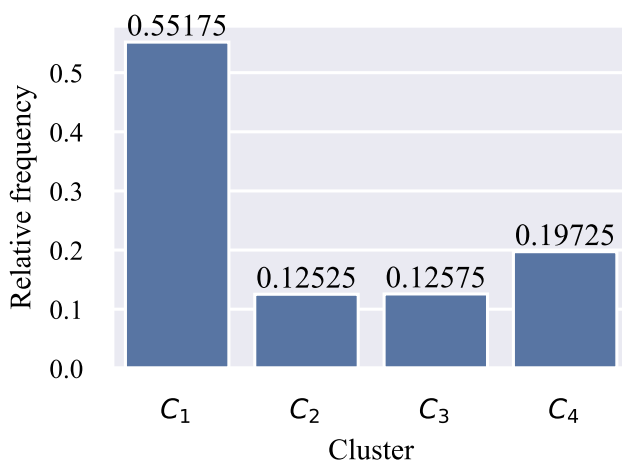


Fig. 4 Relative frequencies of the clusters, i.e., the bias weights. Since the clusters C_1 and C_4 occur more frequently than the other clusters, their bias weights are higher than the weights of the other clusters

correspond to phonemes, we expect that similar phonemes will be grouped together in a cluster. Moreover, we expect the pre-trained reservoir weights to exhibit a structure that shares similarities with the phonotactic rules. Specifically, we hypothesize that the recurrent weights will resemble the transition probabilities of a bigram language model.

3.1 Dataset

We use the DARPA TIMIT corpus [10], which consists of a broad variety of sentences spoken by native English speakers and is a common benchmark test in the area of speech recognition. It is already divided in training and test subsets. In total, 630 speakers (70 % male, 30 % female) from eight dialect groups each recorded ten sentences. Two of the sentences were equal for all speakers, and we did not use them in this study to stay in line with Triefenbach et al. [50] and Graves and Schmidhuber [13]. The training set included 3640 recordings and the test set 1336 recordings from 168 different speakers, which are not contained in the training set.

3.2 Feature extraction

The starting point for the feature extraction were the audio signals with a sampling frequency of 16 kHz. Each audio signal was first filtered with a pre-emphasis filter with a coefficient of 0.97 [58]. Afterwards, it was divided into frames with a length of 25 ms and with a window shift of 10 ms. Short-term spectra were extracted by applying a Hamming window to the frame and then computing the FOURIER transform. To reduce the dimensions of the resulting spectrograms, we used psycho-acoustic knowledge and applied a Mel filterbank with 40 filters between 0 kHz and 8 kHz. We use a logarithmic amplitude and finally add the Δ and $\Delta\Delta$ features. This leads to an input size $N^{\text{in}} = 120$. It has been shown in Huzafah [16] that the Mel spectrograms as used here are a well-performing alternative to Mel-Frequency Cepstral Coefficients (MFCCs). An example for a logarithmic Mel spectrogram is given in Fig. 5. For the sake of simplicity, the Δ and $\Delta\Delta$ features are not visualized here.

We normalized the features channel-wise to zero mean and unitary variance. Therefore, the constants for each Mel channel, its first and second derivatives were computed on the entire training dataset and then applied channel-wise on each spectrogram in the training and test set.

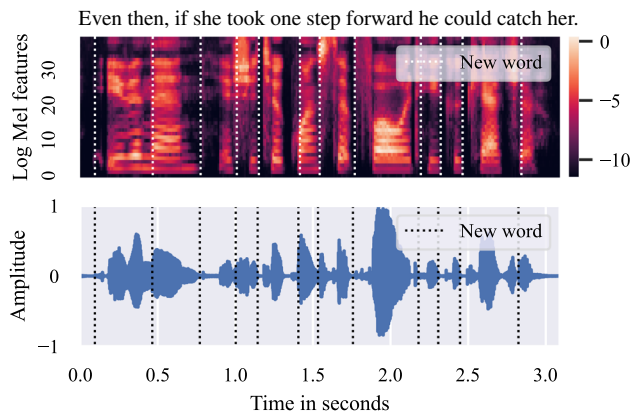


Fig. 5 Example for a raw audio signal and the extracted Mel spectrogram with 40 channels from one sentence of the TIMIT corpus

3.3 Target preparation and readout postprocessing

Each audio file has a time-aligned phoneme annotation. We followed [50] and used 39 phonemes. For each frame, the target was prepared in a one-hot encoding so that the output corresponding to the current active target phoneme is 1, all other outputs are zero.

The ESN serves as an *acoustic model* in this context and represents the relationship between acoustic features and the phonemes. The only postprocessing that was applied in this paper is to identify the output with the highest activation in each frame, which denotes the detected phoneme. The approximation of the likelihoods could also be fed in a subsequent language model, which, however, goes beyond the scope of this paper.

3.4 Measurements

We follow [50] and report the frame error rate (FER), which is the proportion of incorrect classifications to the total number of frames. The FER was used to optimize the hyper-parameters and for the final evaluation on the test set.

3.5 Optimization of the hyper-parameters of the ESNs

As discussed in our prior work [44] and in Hinaut and Trouvain [15], there exist intrinsic relationships between the most important hyper-parameters to be tuned in ESNs. Consequently, we use the strategy defined in Steiner et al. [44] and sequentially optimized the hyper-parameters using randomized searches that are specified in Table 1.

We start with a random search across the input scaling α_u and the spectral radius ρ , without using leaky integration and without the bias for each neuron. In a second step, we

Table 1 Setup of the hyper-parameter optimization. Each step was a randomized step with the specified distribution and the specified number of iterations

Parameter	Distribution	Range	#Iterations
Input scaling	Uniform	1×10^{-2} to 1	200
Spectral radius	Uniform	0 to 2	200
Leakage	Log uniform	1×10^{-3} to 1	50
Bias scaling	Uniform	0 to 3	50

optimized the leakage value using a random search, while the input scaling and spectral radius were fixed to the determined values from the previous step. Finally, the bias scaling was optimized using a random search.

For this optimization, the regression parameters were solely trained on the training set and evaluated on the validation set. The reservoir size was fixed to 50 neurons. In preliminary experiments, we compared the impact of different reservoir sizes for optimizing the hyper-parameters, and it turned out that the parameters obtained with 50 and with 500 neurons were almost the same. Thus, we chose 50 neurons, since this reduced the required optimization time.

The resulting hyper-parameters for the different ESN architectures are summarized in Table 2. There are substantial differences between the different models. Especially the input scaling and the spectral radius are different. In the following, we give insight in what the different ESN models learned from the audio data.

3.6 Reservoir analysis

In Fig. 6, the randomly initialized input weights are compared to the K -Means centroids. As opposed to the randomly initialized sparse input weights, the centroids are dense and three sections can be recognized, which belong to the Mel features and their first and second derivatives. The input weights leading to every neuron now represent mean values of specific feature vectors from the training dataset that were grouped together.

Similarly, we observe substantial differences between the random recurrent weights (Fig. 7a) and the normalized neuron transition matrix (Fig. 7b). In the first case, the weights are unordered, while in the neuron transition matrix the main diagonal is pronounced. This suggests that each neuron is more likely to be activated by itself than by other neurons. This is an expected phenomenon in speech data, where there is a high chance to see the same phoneme than a different one when moving from one frame to another, although we do not expect that the neurons strictly correspond to phonemes here. Furthermore, the 2-Gram transition matrix is sparse, because only a subset of

Table 2 Results of the hyperparameter optimization for the different ESN architectures for phoneme recognition

	Basic ESN	KM-ESN	KM-ESN 2-Gram	KM-ESN PB	KM-ESN 2-Gram PB
Input scaling	0.11	0.04	0.06	0.04	0.02
Spectral radius	0.04	0.07	0.65	0.07	1.02
Leakage	0.35	0.21	0.35	0.21	0.55
Bias scaling	2.89	2.73	1.12	0.88	1.30

Fig. 6 **a** Random input weights and **b** normalized *K*-Means centroids. While the random input weights are sparse, the centroids are dense and show three groups, which belong to the Mel features, to the Δ features, and to the $\Delta\Delta$ features, respectively

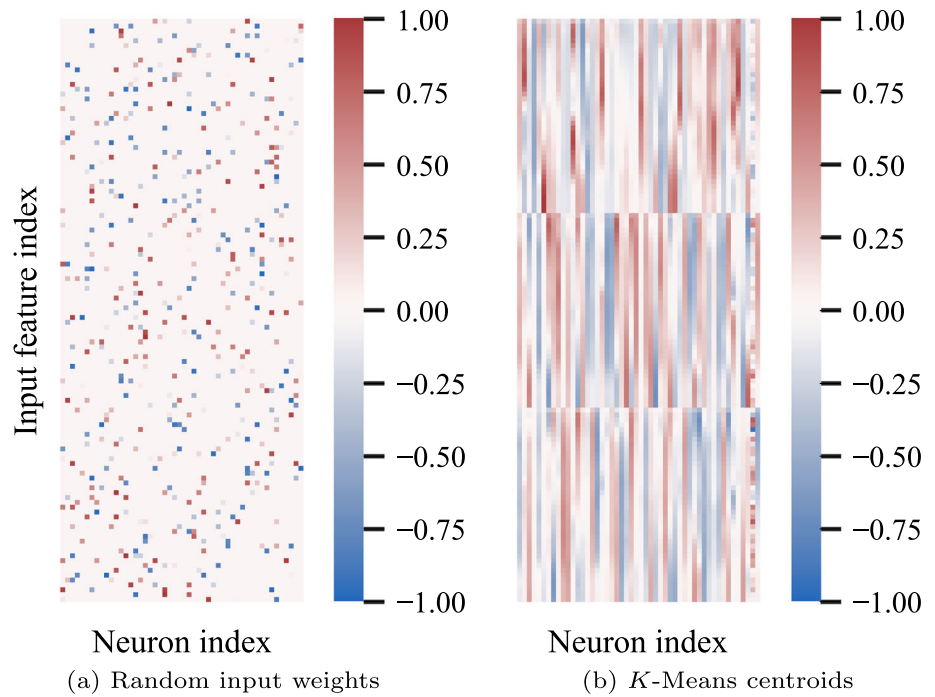
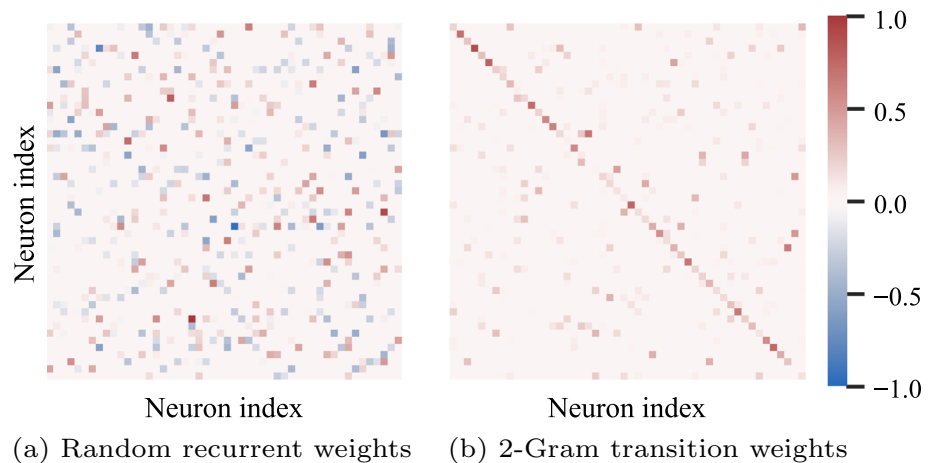


Fig. 7 **a** Random recurrent weights and **b** normalized neuron transition matrix. While the random recurrent weights are unordered, the main diagonal of the 2-Gram transition weights is pronounced. This means that it is likely that each neuron in the reservoir is activated by itself for multiple time steps



phoneme transitions occur in typical speech data. This subset, however, can have large non-zero values, which still allows a complex information flow between different neurons. Considering the complex eigenvalues of the 2-Gram transition weights, this means that a majority of the eigenvalues of these transition weights have larger imaginary parts.

Finally, we analyze the bias weights. Since the phoneme set in the TIMIT corpus is not balanced, we expected that the pre-trained bias weights are not uniformly distributed as well. In fact, the pre-trained bias weights are similar as in the introductory example in Fig. 4. A few neurons have higher bias weights than most of the neurons. Interestingly,

regardless of the reservoir size, it turned out that particularly one neuron had the overall highest weight value.

In Fig. 8, we visualized the centroids of the four clusters with the highest bias weights. For the sake of this visualization, we restricted the plot to the Mel space and disregarded the first and the second derivative for now. Particularly for the cluster index 3, which denoted the neuron with the largest bias weight (0.157), we can see that the energy is rather low. This is exactly the description of a feature vector that belongs to silence. The orange line that belongs to cluster 5 denotes the most frequent phoneme symbol “s” [12] in the TIMIT corpus. From the Mel spectrum in Fig. 8, it can be observed that the high-frequency content of these phonemes was learned very well, because the energy toward high frequencies increased here. The green line that belongs to cluster 14 denotes the phone symbols “ix” and “ih” in the TIMIT corpus. These phones were grouped together to one phoneme [50], and together, they are the third most frequently occurring phoneme in the TIMIT corpus. As shown in Fig. 8, the green line has two maxima, which could approximately be the first and the second formant frequencies (240 Hz and 2400 Hz, respectively) of this phoneme group. The red line that belongs to cluster 11 was not as straightforward to be assigned to one phoneme. Since still two maxima might be recognized here, this could probably belong to the same phoneme as before, but with a lower energy.

In summary, we showed now that the proposed model is not only trained in an unsupervised way but can be further interpreted with prior knowledge about the data. From the transition weights, we have shown that, since the task of phoneme recognition needs to map from fast changing input sequences to slowly changing target output sequences, it is effective that especially the main diagonal of the learned transition weights is emphasized. Consequently, it

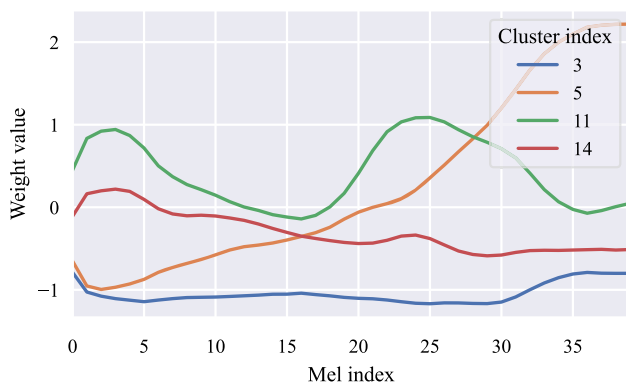


Fig. 8 Centroids of the strongest activated neurons. The first line with the label “3” shows that most of the frames belonged to silence, because it shows a low energy and derivatives that are approximately zero. Furthermore, other centroids that have local maxima show a formant structure, which is frequently represented in the dataset, as well

is likely to choose the same “winner neuron” for a longer amount of time. Secondly, the proposed probabilistic bias weights allow for an inspection of the model, since we can study which neurons have learned which part of the data.

3.7 Results and discussion

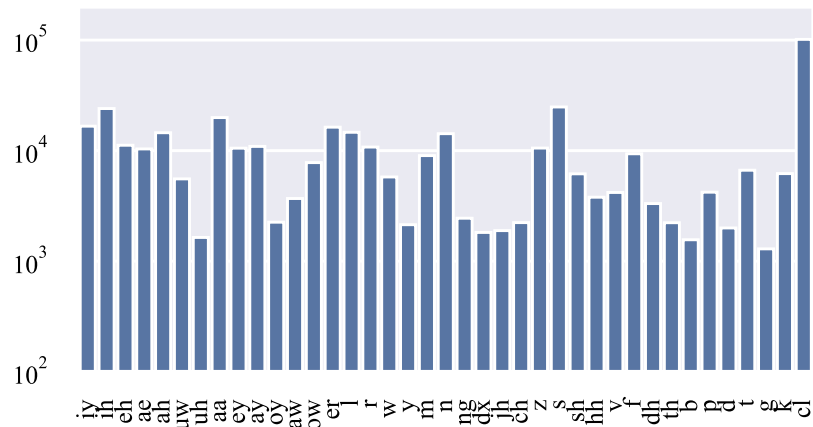
For the final evaluation, it is a common step for ESNs to report the model performance on the test set for different reservoir sizes. Therefore, we started with 50 neurons and increased the reservoir size up to 16 000.

For each reservoir size, we first of all tuned the regularization parameter α using a randomized search with 50 iterations between 1×10^{-5} and 10 in a logarithmic uniform distribution, then re-trained the model on the entire training set, and finally reported the performance on the test set.

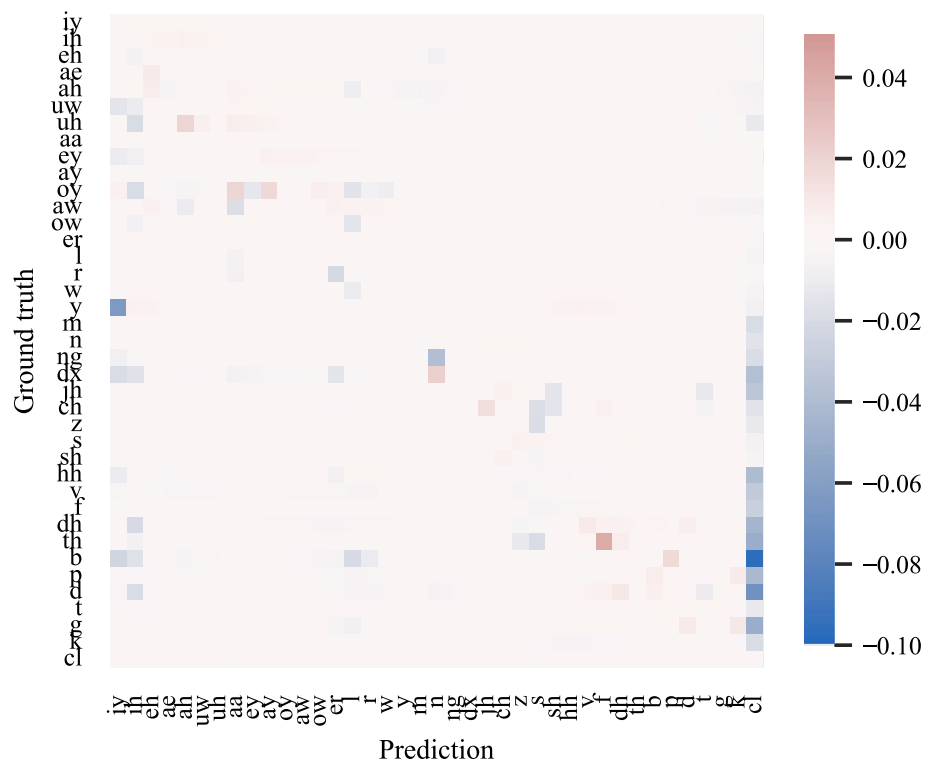
In Fig. 10, we summarized the FER for different reservoir sizes and architectures. In general, the FER decreased with larger reservoir sizes, and the basic ESN, particularly for small and medium reservoir sizes, showed the worst performances. The KM-ESN was one of the overall best models which confirms the effectiveness of the clustering approach proposed in Steiner et al. [44]. On the other hand, the models, in which we used the probabilistic bias weights, did not improve the KM-ESN performance. Only in case of larger reservoirs, the KM-ESN seems to benefit from the probabilistic bias weights. Overall, the model “KM-ESN 2-Gram” was the best performing model within this study, especially for reservoir sizes with more than 400 neurons. Interestingly, the combination of the probabilistic bias weights and the 2-Gram weight matrix decreased the performance of the model. A possible reason for this effect is that the neuron that represents silence is too strongly emphasized by both, input, recurrent and bias weights.

To get a better insight into the performance of the different models, we now compare the confusion matrices of the “KM-ESN” and of the “KM-ESN 2-Gram” models, both with 16 000 neurons. Figure 9 shows the difference between the confusion matrix obtained with the “KM-ESN” model and the confusion matrix obtained with the “KM-ESN 2-Gram” model on the test set, with the main diagonal set to zero for visualization purposes. The “KM-ESN” model tended to incorrectly recognize many fricatives and plosives as silence, whereas the “KM-ESN 2-Gram” model showed a better performance in recognizing these phonemes correctly. The “y” phoneme was also more frequently recognized by the “KM-ESN 2-Gram” model. However, the “th” phoneme was more frequently recognized as “f”, and a few similar-sounding vowels were more frequently confused. It is worth noting that the plosives and the “y” are relatively rare phonemes in the TIMIT corpus, and the plosives always occur after a

Fig. 9 Absolute label frequencies in the test set, and the difference between the confusion matrices of the “KM-ESN 2-Gram” and the “KM-ESN” models on the test set, respectively. The main diagonal was set to zero for the sake of visualization



(a) Absolute label frequencies



(b) Confusion matrix difference

closure (“cl”). The transition matrix of the “KM-ESN 2-Gram” model allows for better modeling of these dependencies compared to randomly initialized reservoir weights, which leads to better recognition rates in these rarely occurring classes.

We finally compare our results with Triefenbach et al. [50], in which the same data was used for randomly initialized ESNs, and with Graves and Schmidhuber [13], in which bidirectional LSTM networks were used. From Table 3, can be recognized that the proposed ESN models

with pre-trained input and recurrent weights performed better than the reference models, which both define the state-of-the-art in frame-wise continuous phoneme recognition. Of course, the training complexity increases in particular when moving from the basic ESN to the cluster-based ESN models. However, the pre-training is an important step, because it allows for an efficient usage of basically any recorded speech data that is suitable for a specific task. Since both, the input and recurrent weights are trained un- or self-supervisedly, not even labels are

Table 3 Final recognition errors on the TIMIT corpus. Obviously, the proposed ESN structure outperformed all reference results

Model	FER
Basic ESN 20000 Triefenbach et al. [50]	30.6 %
Basic ESN 16000	32.5 % \pm 0.3 %
KM-ESN 16000	30.4 % \pm 0.2 %
KM-ESN 16000 2-Gram	28.7 % \pm 0.2 %
BLSTM Graves and Schmidhuber [13]	29.8 %

The bold value marks the best results

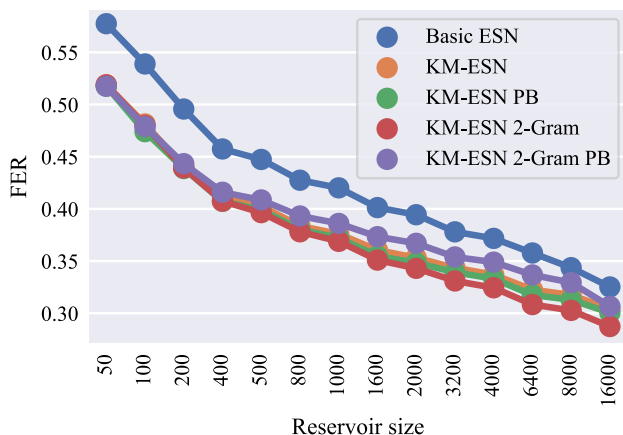


Fig. 10 Results for different ESN models and reservoir sizes. The KM-ESN outperformed the basic ESN as in prior work. The model “KM-ESN 2-Gram” with a transition matrix normalized between zero and one performed best. The models with the probabilistic bias performed equivalently well as the KM-ESN model

required for the training. Furthermore, computing the transition weights as well as the probabilistic bias weights is computationally not complex anymore but the impact could clearly be observed in this experiment. One more important aspect is the interpretability of the models. As discussed, it could easily be inspected, what happened inside the pre-trained ESN models.

Since the performance was evaluated on five random initializations and the results of some models are very similar, we performed a variety of significance tests to compare the effect of the different proposed models. Considering the TIMIT corpus, we firstly conducted tests for different mean values of the FER in case of the different models. The null hypothesis for these tests was that there is no significant difference in the results reported for the different models. With this null hypothesis, we conducted a one-way ANOVA, which revealed that there was a statistically significant difference in the performance of at least two models on the TIMIT corpus with an F -value of 141.73 and a p -value of 2.25×10^{-14} . We could also reject the null hypothesis in case of the Kruskal–Wallis H -test for

independent samples (Kruskal–Wallis H statistic of 21.57 and p -value of 0.000 24) and in case of the Alexander–Govern approximation test (A statistic of 40.14 and p value of 4.04×10^{-8}).

Next, we used the Tukey’s honestly significant difference (HSD) test as a post hoc test to compare the means of the FER values reported for each model. We set the significance level to 0.05, and the null hypothesis was that there is no significant difference in the results. The results of this test are summarized in Fig. 11. As can be seen from the p -values, all pre-trained models performed significantly different to the basic ESN. Furthermore, the model “KM-ESN 2-Gram” performed significantly different to the other pre-trained models. However, in combination with the probabilistic bias weights, there is no significant difference between the KM-ESN and these variants. Hence, we ensured that the model “KM-ESN 2-Gram” is performing significantly better than the other models.

4 Multivariate time-series classification

The goal of this experiment is the evaluation of the different proposed ESN models on a broad variety of time-series classification tasks with different characteristics. We jointly evaluate the sequence classification error CER and the frame error rate FER. We compare the performance of the models with probabilistic bias weights and with 2-Gram transition weights with the baseline models.

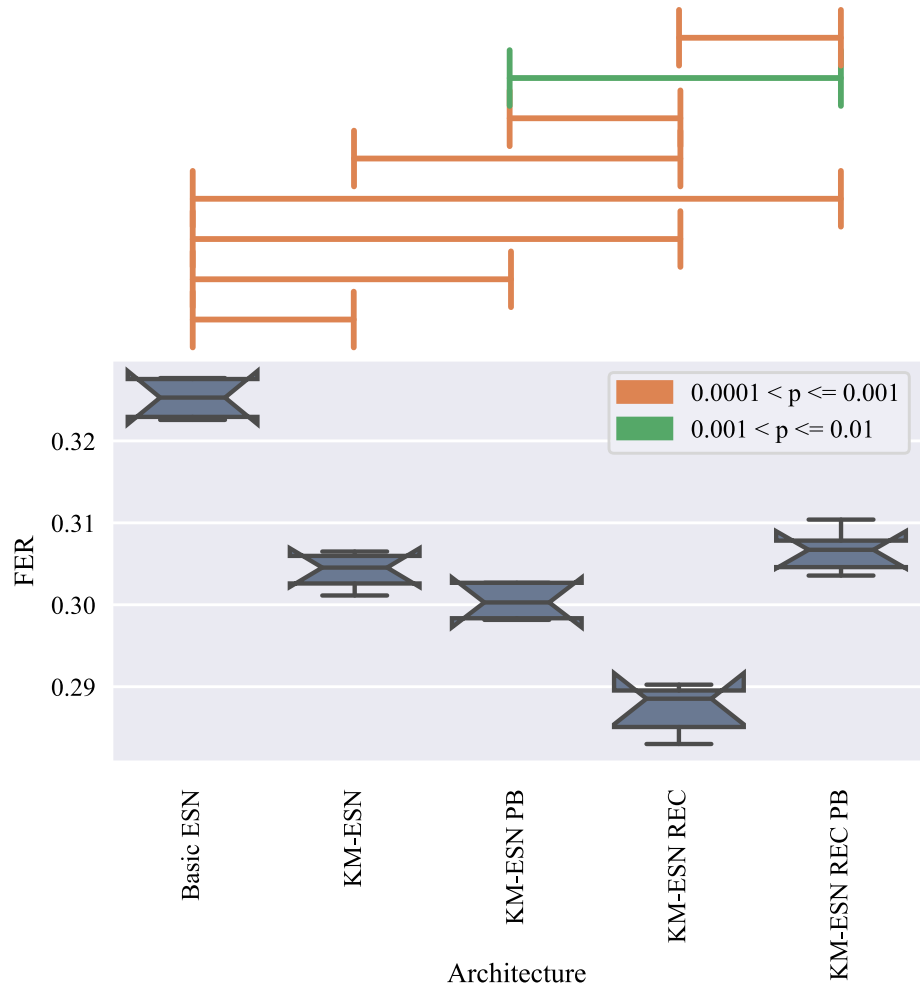
4.1 Datasets

We used the datasets from Toth and Oberhauser [49], which differ in the number of samples, input dimension, sequence length and data type. The main characteristics about the different datasets are summarized in Table 4. We used the default split into training and test sets in our experiments and optimized the hyper-parameters based on K -fold cross-validation on the training set. For most of the datasets, we used fivefold cross-validation. Only for the datasets with only a few training sequences, we used threefold cross-validation. For details, we refer to the column “CV” in Table 4.

4.2 Readout postprocessing and measurements

For each dataset, the target outputs (classes) were one-hot encoded across the sequence (0 for the inactive classes and 1 for the active class).

Fig. 11 Tukey’s honestly significant difference (HSD) post hoc test to compare the mean values of the results of the analyzed models on the TIMIT corpus



During inference, the class scores were obtained by searching for the class with the highest score in the entire sequence.

To optimize the hyper-parameters, the Frame Error Rate FER, which is the proportion of incorrect classifications to the total number of frames, was used as a measure.

To measure the overall classification results, the classification error rate CER was used:

$$CER = \frac{N_{error}}{N_{seq}} \tag{6}$$

where N_{error} and N_{seq} were the number of incorrectly classified sequences and the overall number of sequences, respectively.

4.3 ESN hyper-parameters

The optimization of the ESN models followed the same strategy as in Sec. 3.5 with one exception: Instead of starting with a default leakage of 1.0, we chose 0.1, because the target outputs were constant across the entire sequence. Thus, we expected a lower leakage. The

subsequent optimization procedure was then the sequential optimization of the hyper-parameters, during which we minimized the cross-validation FER. During the optimization, we fixed the reservoir size to 50 neurons.

As discussed for the previous experiments, the hyper-parameters significantly influence the performance of the ESN models and they need to be tuned task-dependently. Thus, in contrast to Bianchi et al. [6], we used models with optimized hyper-parameters to report our final results for each dataset. As can be expected and without reporting all the hyper-parameters for every task, we observed that these parameters were significantly different across the datasets.

In Fig. 12, the FER of the different ESN models and reservoir sizes are summarized for all of the analyzed datasets.

In most of the cases, we observed that the FER decreased with an increasing reservoir size. Exceptions were the datasets “KICK” and “DS”, in which the performance of some models decreased with the reservoir size or was unstable.

Overall, the pre-trained ESN models performed better and more stable than the basic ESNs, except for the

Table 4 Details about the datasets for the multi-dataset evaluation with the number of input features (N^{in}), number of outputs (N^{out}), mean, minimum and maximum sequence duration (T_{mean} , T_{min} and T_{max}), the entire number of feature vectors (observations) in the

training set and test set (N_{train} , N_{test}), the number of sequences in the training and test set (#Train, #Test), and the number of folds for the cross-validation (CV)

Dataset	Abbreviation	N^{in}	#Train	#Test	CV	N^{out}	T_{mean}	T_{min}	T_{max}	N_{train}	N_{test}
Arabian Digits	ARAB	13	6600	2200	5	10	39.8	4	93	263,256	87,063
Auslan	AUS	22	1140	1425	5	95	57.3	45	136	63,371	83,578
Character Trajectories	CHAR	3	300	2558	5	20	170.5	109	205	51,465	435,812
CMU subject 16	CMU	62	29	29	3	2	305.0	127	580	8462	9229
Digit Shapes	DS	2	24	16	3	4	66.5	30	98	1625	1035
ECG	ECG	2	100	100	5	2	89.5	39	152	9020	8887
Japanese Vowels	JPVOW	12	270	370	5	9	15.6	7	29	4274	5687
Kick vs. Punch	KICK	62	16	10	5	2	426.7	274	841	6413	4682
Libras	LIB	2	180	180	5	15	45	45	45	8100	8100
NetFlow	NET	4	803	534	5	2	230.7	50	997	182,881	125,506
PEMS	PEMS	963	267	173	5	7	144	144	144	38,448	24,912
Pen Digits	PEN	2	300	10,692	5	10	8	8	8	2400	85,536
Shapes	SHAPES	2	18	12	3	3	80.6	52	98	1442	977
uWave	UWAV	3	896	3582	5	8	315	315	315	282,240	1,128,330
Wafer	WAF	6	298	896	5	2	136.8	104	198	40,833	122,450
Walk vs. Run	WALK	62	28	16	3	2	368.0	128	1918	10,930	5261

datasets “AUS”, “LIB” and “NET”. Since the “AUS” dataset has a very large number of classes (95), it is likely that more centroids would have been required to describe this dataset better. The other datasets (“LIB” and “NET”) are rather low-dimensional (4D and 2D, respectively) and difficult to get clustered. For the movement datasets (“CMU”, “KICK”, “WALK”), as well as “JPVOW” and “PEMS”, the pre-trained models outperformed the basic ESN. These are all rather high-dimensional datasets that can be clustered well.

4.4 Results and discussion

For the final evaluation, we collected the best results for each model and dataset and compared the error rates on frame level and on sequence level. In Fig. 13, the FER is summarized. The main conclusion was that no single model performed best in all tasks:

- The *Basic ESN* performed particularly well on the “ARAB”, “AUS”, “LIB” and on the “NET” datasets. With the exception of the “ARAB” and “AUS” datasets, these were mostly low-dimensional datasets with uniformly distributed features. This hinders the clustering; thus the pre-trained models do not have much advantage against the *basic ESN*.
- The *KM-ESN* showed the best performance on the datasets “CHAR”, “CMU”, “JPVOW”, “KICK” and on “PEN”. The dimensions of the datasets range from

3D in case of “CHAR” and larger in the other cases. All datasets are better linearly separable than the datasets, in which the basic ESN showed the best performance. Although the “CHAR” dataset is still low-dimensional, the separation gets easier, because the dataset contains fewer classes than most of the other low-dimensional datasets. Although the movement datasets “CMU” and “KICK” both have the same features, the performance is quite different. The reason therefore is that the features of the “KICK” dataset are partially mixed if it is projected in lower dimensions using PCA. This makes the two classes difficult to get separated properly.

- Probabilistic bias weights (i.e., the *KM-ESN PB*) helped to improve the results in case of the datasets “ECG”, “UWAV” and “WAF”. Particularly the first two datasets have distinct temporal orders of the features. The probabilistic bias weights here were useful to identify the most important neurons, similar as we observed for the phoneme recognition.
- The *KM-ESN with 2-Gram transition weights* showed the best performance in the datasets “ARAB” and “CMU”. These are all datasets with a medium number of features. Here, the order of subsequent winner neurons play an important role. This results in transition weights similar as in the phoneme experiment in Sect. 3, where the main diagonal was emphasized and the complex eigenvalues mostly had a positive real part.

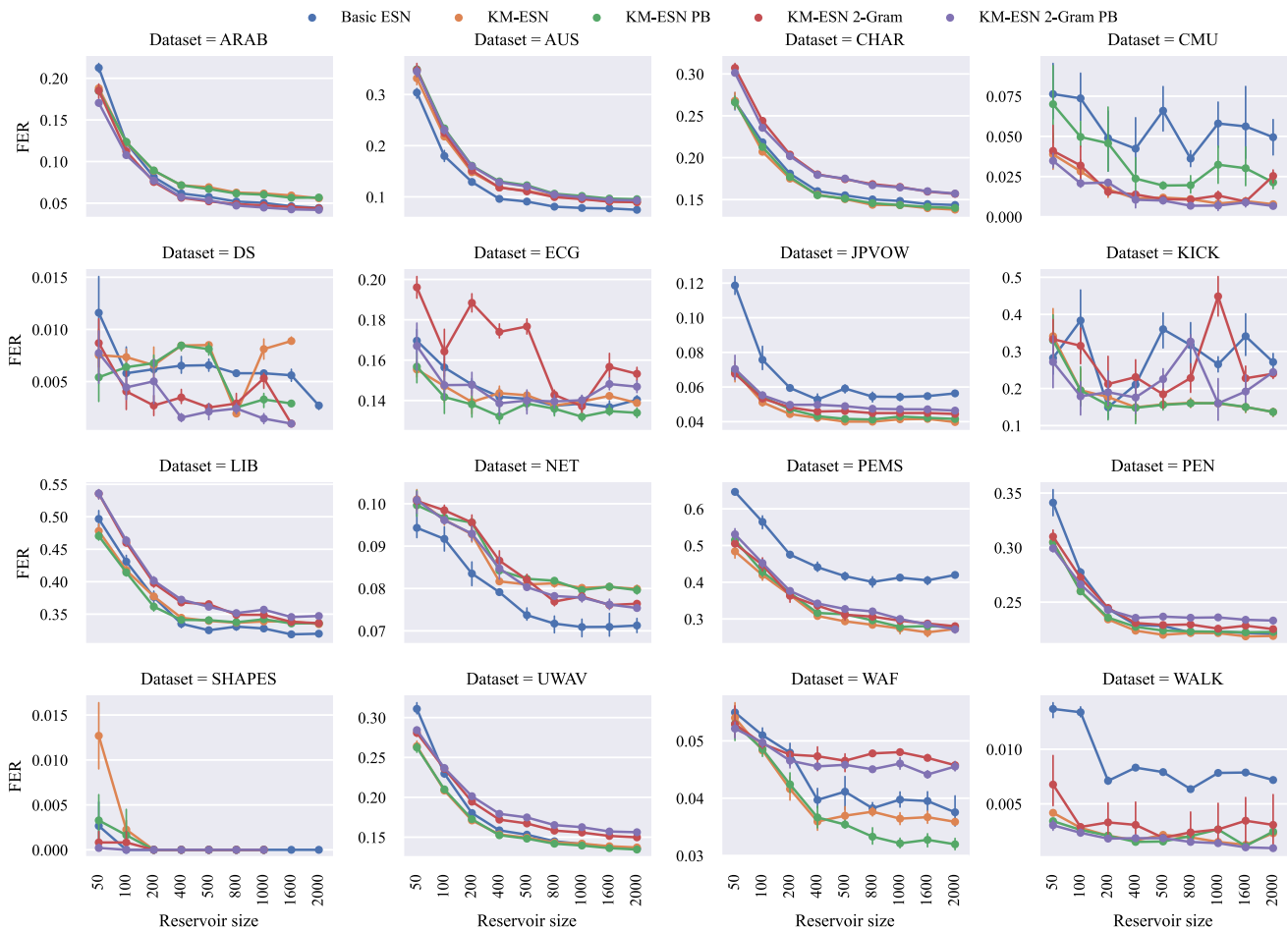


Fig. 12 FER with respect to reservoir size and models for the analyzed datasets. In general, we observed that increasing the reservoir size improves the performance. “KICK” and “DS” are exceptions, in which the performance of some models decreased with

the reservoir size or was unstable. Overall, the pre-trained ESN models performed better than the basic ESNs in most of the cases and lead to more stable results

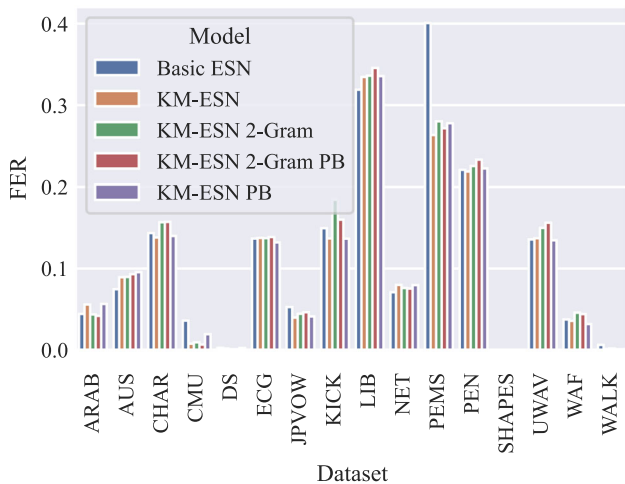


Fig. 13 FER for different datasets and evaluated models. For high-dimensional data, we can see that the models with probabilistic bias weights and the 2-Gram models in many cases outperformed the basic ESN and the KM-ESN as baseline models

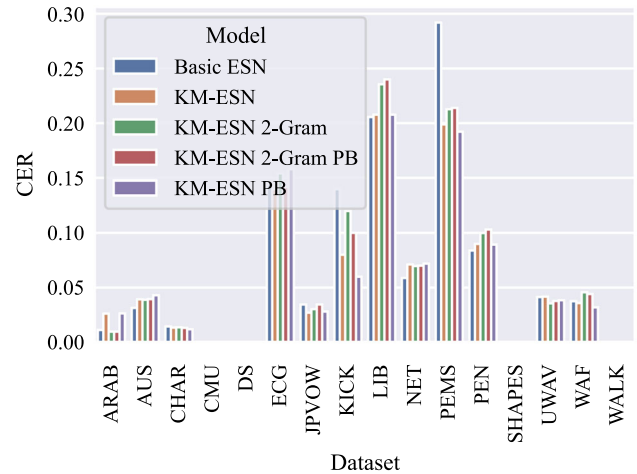


Fig. 14 CER for different datasets and evaluated models. For high-dimensional data, we can see that the models with probabilistic bias weights and the 2-Gram models in many cases outperformed the basic ESN and the KM-ESN as baseline models

The equally good performances of all the models on the remaining datasets do not provide enough room for discussion and comparison on those tasks.

In Fig. 14, the final CER is summarized.

- The *basic ESN* performed particularly well on the “ARAB”, “AUS”, “LIB” and on the “NET” datasets. These results are in line with the previously reported results using the FER.
- The *KM-ESN* performed better than the *basic ESN* in various cases. Particularly on the “ECG” “JPVOW” datasets, the performance of the *KM-ESN* was superior compared to the other pre-trained models. This supports that the proposed pre-training strategies are beneficial for the overall performance.
- Probabilistic bias weights (i.e., the *KM-ESN PB*) helped to improve the results in case of the datasets “CHAR”, “KICK”, “PEMS” and “WAF”. It again shows that, for high-dimensional data, a few neurons are more important than others to solve a given task.
- The *KM-ESN 2-Gram* showed a superior behavior on the “ARAB” and “UWAV” datasets, which also have the longest samples among all the studied datasets.

In Table 5, the mean and standard deviation of the accuracies on the test sets over five random initializations are computed for the best performing models. We compare our results with Toth and Oberhauser [49], where exactly the same evaluation scheme was used and where various state-of-the-art algorithms for time-series classification were benchmarked.

As expected, the ESN architectures mostly benefited from utilizing the centroids of the *K*-Means algorithm. Using pre-trained input weights in general led to better results compared to the basic ESN. Using probabilistic bias weights led to slightly more stable results, i.e., to a lower standard deviation of the recognition rates. However, the probabilistic transition weights even led to a decreased performance of the *KM-ESN REC* and of the *KM-ESN REC PB* architectures. The *K*-Means algorithm mostly benefits from larger datasets, meaning that any of the models *KM-ESN*, its deterministic or probabilistic variants performed significantly better than the other considered models for large datasets, such as the AUS, CHAR, and the UWAV datasets. Additionally, if the inputs underlie a multivariate distribution, as it is the case for, e.g., the UWAV dataset or in the small LIB, PEN datasets with many classes, the *K*-Means algorithm is able to extract a lot of information from the data. The *KM-ESN* models with 2-Gram transition weights did not perform well on this task of multivariate time-series classification. In fact, these models only performed better than the *KM-ESN* in a few cases. We noticed that the transition weights are mostly dominated by self-transitions, which hinder a complex information flow between the reservoir neurons.

If we compare the results with the state-of-the-art methods from Toth and Oberhauser [49], it turns out that the *KM-ESN PB* model can compete with most of the models both in terms of accuracy and the standard deviation. It is worth to emphasize that many of the reference

Table 5 Overall comparison of the proposed models with state-of-the-art algorithms for time-series classification based on the mean and standard deviation of the accuracies on the test sets over five random initializations

Model	Accuracy	Standard deviation
Basic ESN	0.932	0.083
KM-ESN	0.939	0.066
KM-ESN PB	0.940	0.066
KM-ESN 2-Gram	0.933	0.074
KM-ESN 2-Gram PB	0.935	0.074
GP-SIG-LSTM Toth and Oberhauser [49]	0.948	0.068
GP-SIG-GRU Toth and Oberhauser [49]	0.935	0.070
GP-SIG Toth and Oberhauser [49]	0.947	0.052
GP-LSTM Toth and Oberhauser [49]	0.853	0.193
GP-GRU Toth and Oberhauser [49]	0.824	0.218
GP-KConv1D Toth and Oberhauser [49]	0.898	0.107
SMTS Baydogan and Runger [4]	0.945	0.059
LPS Baydogan and Runger [5]	0.933	0.073
mv-ARF Tuncel and Baydogan [55]	0.949	0.055
DTW Rakthanmanon et al. [35]	0.899	0.111
ARKERNEL Cuturi and Doucet [7]	0.938	0.073
gRSF Karlsson et al. [23]	0.955	0.058
MLSTM-FCN Karim et al. [22]	0.970	0.039
MUSE Schäfer and Leser [39]	0.962	0.043

The bold values marks the best results

methods included pre-processing steps that went out of the scope of this paper. The generalized Random Shapelet Forests (gRSF) [23], for instance, requires a complex feature extraction pipeline including shapelet features. Autoregressive Random Forests for multivariate time-series modeling (mv-ARF) [55] propose a tree ensemble trained on autoregressive models with different lags of the MTS. The reference model MLSTM-FCN [22] consists of parallel Fully Convolutional Neural Network (FCN) and LSTM layers and is thus more complex than the approach proposed here. MULTivariate Symbolic Extension (MUSE) [39] computes histograms from the time-series based on an advanced feature extractor and then classifies the time series based on the histograms.

Again, we used a variety of significance tests to determine significant differences between the models. Firstly, we tested whether different models showed a globally significant different performance irrespective of different datasets. Therefore, we used a one-way ANOVA test, the Kruskal–Wallis H -test for independent samples, and the Alexander–Govern approximation test. Since none of these tests allowed us to reject the null hypothesis, there is no globally best performing model. As discussed earlier, the performance of the models depends on the particular dataset, and the accuracy values are distributed across a long range, regardless of the individual models. In order to identify datasets where the models show significant differences, we repeated the tests for the models on the individual datasets. If there was a significant difference between different models in a dataset, we used the Tukey’s honestly significant difference (HSD) test as a post hoc test to compare the means of the Acc values reported for each model. The test results suggest that significant differences mostly occurred between the performance of the Basic ESN and its pre-trained variants. Furthermore, many significant differences occurred between the “KM-ESN PB” model and other derivatives of the KM-ESN. This is in line with the performance comparison in Table 5, where the results of the KM-ESN and the KM-ESN PB models are similar and better than the other investigated ESN models.

5 General discussion

We introduced a variety of novel ESN models that are derived from the KM-ESN [44]. These models are relatively easy to interpret. We computed a transition matrix that describes the transition probability between two neurons, and we proposed to replace the randomly initialized reservoir weights by these transition weights and obtained the model *KM-ESN 2-Gram*. Under the assumption that some neurons in the reservoir are generally more often activated by the input data than others, we proposed to

weight the frequently activated neurons more strongly. Therefore, we computed the absolute frequencies of the “winner neurons” that were most strongly activated by the input data, divided it by the size of the dataset and used this a-priori probability as a bias input for each neuron in the model *KM-ESN PB*.

We evaluated the models based on two inherently different tasks, i.e., frame-wise phoneme recognition and time-series classification.

For the first task (phoneme recognition), we deeply investigated the impact of the different pre-training methods on the weight matrices of the ESN models. It turned out that the values of all pre-trained weight matrices can be interpreted with prior knowledge about the underlying task:

- The pre-trained input weights approximate different phonemes that occur in the dataset.
- The pre-trained bias weights can be related to relative phoneme frequencies in the dataset.
- The pre-trained recurrent weights approximately model the transition probability in the dataset.

For all different proposed models, we optimized the hyper-parameters and showed significant differences between the hyper-parameters of the different models. It turned out that the model “KM-ESN 2-Gram” was the best performing model for this task, significantly outperforming the other models. We compared the confusion matrices of the two best performing models and showed that especially phonemes that rarely occur in the training dataset were better recognized with the best performing model than with the baseline models, which frequently assigned these phonemes to silence, i.e., the most frequently occurring “phoneme” in the training dataset.

In the second experiment, we optimized and evaluated the proposed ESN models on an inherently different task, i.e., multivariate time-series classification. We used a set of 16 datasets that were already used in Toth and Oberhauser [49] to benchmark a broad variety of state-of-the-art algorithms, with which we directly compared our results. The results showed that the KM-ESN and its extension with the probabilistic bias weights achieved the best results that came close to the state of the art. We again tested the significance of the results, and it turned out that there were no significant differences in the performance of the different models. This is due to the long value range on which the accuracy is distributed. On the one hand, some datasets were classified entirely correctly, whereas the accuracy for other datasets was below 80 %. Nevertheless, we identified use-cases in which the pre-trained models should be considered, particularly for datasets with higher dimensions, in which the features are not uniformly distributed. Overall, the models “KM-ESN” and “KM-ESN PB” achieved results comparable to most of the benchmarked state-of-

the-art algorithms. This shows that the proposed models are interesting candidates.

A key advantage of the proposed models is that all the input, bias, and recurrent weights can be obtained from unlabeled data. This makes these models interesting candidates for utilizing a large amount of available data that cannot be used for supervised learning. Furthermore, it is interesting to investigate unsupervised data augmentation strategies for the typically un-trained ESN weights. This could be interesting for very challenging problems, such as fusion data [20, 21], where the amount of labeled data is limited. Regarding the time-series classification, it would be very interesting to use the proposed ESN models within the approaches [22, 39, 49].

6 Future work

As explained in Sect. 1, various publications proposed ESNs with deterministic recurrent weights that have specific spectral properties. For example, the delay line reservoir (DLR) do not have any nonzero eigenvalue, and the complex eigenvalues of the simple cycle reservoir (SCR) lie on a circle around the origin. The DLR with feedback connections or with jumps [37] finally has complex eigenvalues that allow for a more sophisticated reservoir design, depending on the nonzero values in the recurrent weights [36, 37]. However, in Rodan and Tiño [37] the question, to which extent the distribution of the complex eigenvalues has an impact on the performance, could not be answered. A detailed analysis of the complex eigenvalues goes beyond the scope of this paper. However, some observations point in a direction that the transition matrix has interesting properties, such as an intrinsic stability (i.e., a default maximum eigenvalue of exactly 1), a distinct distribution of the eigenvalues in the complex plane, a more or less flat eigenspectrum or some eigenvalues that are exactly 0. In the future, this needs to be deeply investigated, because this could be one way to answer the question from Scardapane and Wang [38], to which extent ESNs benefit from purely randomly or deterministically initialized recurrent weights.

Furthermore, it would be interesting whether the proposed method can be transferred toward deep-learning methods, for example to fully-trained RNNs. It might lead to a more stable training behavior or to a faster convergence. Of course, one further point is to show that the proposed method can be applied to different kinds of data.

Acknowledgements The parameter optimizations were performed on a Bull Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden. This research was also partially supported by Ghent University under the Special Research Award number BOF19/PDO/134.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials Supplemental material, such as the code to reproduce the results presented in this paper, will be found on our website (www.pyrcn.net). The datasets analyzed during the current study for the multivariate time-series classification are available from the corresponding author on reasonable request. The TIMIT data that support the findings of this study are available from the Linguistic Data Consortium of the University of Pennsylvania but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available.

Declarations

Conflict of interest Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Aceituno PV, Yan G, Liu YY (2020) Tailoring echo state networks for optimal learning. *iScience* 23:101–440. <https://doi.org/10.1016/j.isci.2020.101440>
2. Basterrech S, Snášel V (2013) Initializing reservoirs with excitatory and inhibitory signals using unsupervised learning techniques. In: Proceedings of the Fourth Symposium on Information and Communication Technology. Association for Computing Machinery, New York, NY, USA, SoICT '13, pp 53–60. <https://doi.org/10.1145/2542050.2542087>
3. Basterrech S, Fyfe C, Rubino G (2011) Self-organizing maps and scale-invariant maps in echo state networks. In: 2011 11th International Conference on Intelligent Systems Design and Applications, pp 94–99. <https://doi.org/10.1109/ISDA.2011.6121637>
4. Baydogan MG, Runger G (2014) Learning a symbolic representation for multivariate time series classification. *Data Min Knowl Disc* 29(2):400–422. <https://doi.org/10.1007/s10618-014-0349-y>
5. Baydogan MG, Runger G (2015) Time series representation and similarity based on local autopatterns. *Data Min Knowl Disc* 30(2):476–509. <https://doi.org/10.1007/s10618-015-0425-y>
6. Bianchi FM, Scardapane S, Løkse S et al (2021) Reservoir computing approaches for representation and classification of multivariate time series. *IEEE Trans. Neural Netw. Learn. Syst.* 32(5):2169–2179. <https://doi.org/10.1109/TNNLS.2020.3001377>
7. Cuturi M, Doucet A (2011) Autoregressive kernels for time series. <https://doi.org/10.48550/ARXIV.1101.0673>, arXiv:1101.0673
8. Gallicchio C, Micheli A (2019) Reservoir topology in deep echo state networks. In: Tetko IV, Kůrková V, Pavel K et al (eds)

- Artificial neural networks and machine learning - ICANN 2019: Workshop and special sessions. Springer International Publishing, Cham, pp 62–75
9. Gallicchio C, Micheli A (2020) Ring reservoir neural networks for graphs. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp 1–7. <https://doi.org/10.1109/IJCNN48605.2020.9206723>
 10. Garofolo JS, Lamel LF, Fisher WM, et al. (1993) Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1. NASA STI/Recon technical report n 93:27,403
 11. Gerstner W, Kistler WM (2002) Mathematical formulations of hebbian learning. *Biol Cybern* 87(5–6):404–415. <https://doi.org/10.1007/s00422-002-0353-y>
 12. Glackin C, Wall J, Chollet G, et al. (2018) Convolutional neural networks for phoneme recognition. In: Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods - ICPRAM, INSTICC. SciTePress, pp 190–195. <https://doi.org/10.5220/0006653001900195>
 13. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Netw* 18(5):602–610. <https://doi.org/10.1016/j.neunet.2005.06.042>
 14. Griffith A, Pomerance A, Gauthier DJ (2019) Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplin J Nonlinear Sci* 29(12):123,108. <https://doi.org/10.1063/1.5120710>
 15. Hinaut X, Trouvain N (2021) Which hype for my new task? hints and random search for echo state networks hyperparameters. In: Farkaš I, Masulli P, Otte S et al (eds) Artificial neural networks and machine learning - ICANN 2021. Springer International Publishing, Cham, pp 83–97
 16. Huzaifah M (2017) Comparison of time-frequency representations for environmental sound classification using convolutional neural networks. <https://doi.org/10.48550/ARXIV.1706.07156>, [arXiv:1706.07156](https://arxiv.org/abs/1706.07156)
 17. Jaeger H (2001) The “echo state” approach to analysing and training recurrent neural networks. Tech. Rep. GMD Report 148, German National Research Center for Information Technology, <http://www.faculty.iu-bremen.de/hjaeger/pubs/EchoStatesTechRep.pdf>
 18. Jalalvand A, Triefenbach F, Demuyneck K et al (2015) Robust continuous digit recognition using reservoir computing. *Comput Speech & Language* 30(1):135–158. <https://doi.org/10.1016/j.csl.2014.09.006>
 19. Jalalvand A, Demuyneck K, De Neve W et al (2018) On the application of reservoir computing networks for noisy image recognition. *Neurocomputing* 277:237–248. <https://doi.org/10.1016/j.neucom.2016.11.100>
 20. Jalalvand A, Abbate J, Conlin R, et al (2021a) Real-time and adaptive reservoir computing with application to profile prediction in fusion plasma. *IEEE Transactions on Neural Networks and Learning Systems* pp 1–12. <https://doi.org/10.1109/TNNLS.2021.3085504>
 21. Jalalvand A, Kaptanoglu AA, Garcia AV et al (2021) Alfvén eigenmode classification based on ECE diagnostics at DIII-d using deep recurrent neural networks. *Nucl Fusion* 62(2):026,007. <https://doi.org/10.1088/1741-4326/ac3be7>
 22. Karim F, Majumdar S, Darabi H et al. (2019) Multivariate lstm-fns for time series classification. *Neural Networks* 116:237–245 <https://doi.org/10.1016/j.neunet.2019.04.014>, <http://www.science-direct.com/science/article/pii/S0893608019301200>
 23. Karlsson I, Papapetrou P, Boström H (2016) Generalized random shapelet forests. *Data Min Knowl Disc* 30(5):1053–1085. <https://doi.org/10.1007/s10618-016-0473-y>
 24. Kohonen T (1990) The self-organizing map. *Proc IEEE* 78(9):1464–1480. <https://doi.org/10.1109/5.58325>
 25. Lazar A, Pipa G, Triesch J (2009) SORN: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience* 3 <https://doi.org/10.3389/neuro.10.023.2009>, <https://www.frontiersin.org/article/10.3389/neuro.10.023.2009>
 26. Lloyd S (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28(2):129–137. <https://doi.org/10.1109/TIT.1982.1056489>
 27. Lukoševičius M, Jaeger H, Schrauwen B (2012) Reservoir computing trends. *KI - Künstliche Intelligenz* 26(4):365–371. <https://doi.org/10.1007/s13218-012-0204-5>
 28. Mansoor M, Grimaccia F, Leva S et al (2021) Comparison of echo state network and feed-forward neural networks in electrical load forecasting for demand response programs. *Math Comput Simulation* 184:282–293. <https://doi.org/10.1016/j.matcom.2020.07.011>
 29. Martinuzzi F, Rackauckas C, Abdelrehim A, et al. (2022) Reservoircomputing.jl: An efficient and modular library for reservoir computing models. <https://doi.org/10.48550/ARXIV.2204.05117>, [arXiv:2204.05117](https://arxiv.org/abs/2204.05117)
 30. Moreno SR, da Silva RG, Mariani VC et al (2020) Multi-step wind speed forecasting based on hybrid multi-stage decomposition model and long short-term memory neural network. *Energy Convers Manage* 213(112):869. <https://doi.org/10.1016/j.enconman.2020.112869>
 31. Ozturk MC, Xu D, Príncipe JC (2007) Analysis and design of echo state networks. *Neural Comput* 19(1):111–138. <https://doi.org/10.1162/neco.2007.19.1.111>
 32. Panahandeh G, Mohammadiha N, Leijon A et al (2013) Continuous hidden markov model for pedestrian activity classification and gait analysis. *IEEE Trans Instrum Meas* 62(5):1073–1083. <https://doi.org/10.1109/TIM.2012.2236792>
 33. Pardo B, Birmingham W (2005) Modeling form for on-line following of musical performances. In: Proceedings of the National Conference on Artificial Intelligence, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, pp 1018–1023
 34. Rabiner L (1989) A tutorial on hidden markov models and applications in speech recognition. *Proc IEEE* 77(2):257–286. <https://doi.org/10.1109/5.18626>
 35. Rakthanmanon T, Campana B, Mueen A, et al (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA, KDD '12, pp 262–270. <https://doi.org/10.1145/2339530.2339576>
 36. Rodan A, Tino P (2011) Minimum complexity echo state network. *IEEE Trans Neural Netw* 22(1):131–144. <https://doi.org/10.1109/TNN.2010.2089641>
 37. Rodan A, Tiño P (2012) Simple deterministically constructed cycle reservoirs with regular jumps. *Neural Comput* 24(7):1822–1852. https://doi.org/10.1162/NECO_a_00297
 38. Scardapane S, Wang D (2017) Randomness in neural networks: an overview. *WIREs Data Mining and Knowl Discov* 7(2):e1200. <https://doi.org/10.1002/widm.1200>
 39. Schäfer P, Leser U (2017) Multivariate time series classification with weasel+muse. <https://doi.org/10.48550/ARXIV.1711.11343>
 40. Schrauwen B, Wardermann M, Verstraeten D et al (2008) Improving reservoirs using intrinsic plasticity. *Neurocomputing* 71(7):1159–1171. <https://doi.org/10.1016/j.neucom.2007.12.020>
 41. da Silva RG, Ribeiro MHD, Moreno SR et al (2021) A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting. *Energy* 216(119):174. <https://doi.org/10.1016/j.energy.2020.119174>
 42. Steiner P, Jalalvand A, Birkholz P (2021) Unsupervised pre-training of echo state networks for onset detection. In: Farkaš I,

- Masulli P, Otte S et al (eds) Artificial neural networks and machine learning - ICANN 2021. Springer International Publishing, Cham, pp 59–70
43. Steiner P, Stone S, Birkholz P, et al. (2021b) Multipitch tracking in music signals using echo state networks. In: 2020 28th European Signal Processing Conference (EUSIPCO), pp 126–130, <https://doi.org/10.23919/Eusipco47968.2020.9287638>
 44. Steiner P, Jalalvand A, Birkholz P (2022a) Cluster-based input weight initialization for echo state networks. IEEE Transactions on Neural Networks and Learning Systems pp 1–12. <https://doi.org/10.1109/TNNLS.2022.3145565>
 45. Steiner P, Jalalvand A, Stone S et al (2022) Pyrcn: A toolbox for exploration and application of reservoir computing networks. Eng Appl Artificial Intell 113(104):964. <https://doi.org/10.1016/j.engappai.2022.104964>
 46. Strauss T, Wustlich W, Labahn R (2012) Design strategies for weight matrices of echo state networks. Neural Comput 24(12):3246–3276. https://doi.org/10.1162/NECO_a_00374
 47. Sun L, Yang X, Zhou J, et al. (2018) Echo state network with multiple loops reservoir and its application in network traffic prediction. In: 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), pp 689–694, <https://doi.org/10.1109/CSCWD.2018.8465335>
 48. Xc Sun, Hy Cui, Rp Liu et al (2012) Modeling deterministic echo state network with loop reservoir. J Zhejiang Univ SCI C 13(9):689–701. <https://doi.org/10.1631/jzus.C1200069>
 49. Toth C, Oberhauser H (2020) Bayesian learning from sequential data using Gaussian processes with signature covariances. In: III HD, Singh A (eds) Proceedings of the 37th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 119. PMLR, pp 9548–9560, <https://proceedings.mlr.press/v119/toth20a.html>
 50. Triefenbach F, Jalalvand A, Schrauwen B, et al. (2010) Phoneme recognition with large hierarchical reservoirs. In: Advances in Neural Information Processing Systems 23. Curran Associates, Inc., p 2307–2315, <http://papers.nips.cc/paper/4056-phoneme-recognition-with-large-hierarchical-reservoirs.pdf>
 51. Triefenbach F, Jalalvand A, Demuynck K et al (2013) Acoustic modeling with hierarchical reservoirs. IEEE Trans Audio Speech Lang Process 21(11):2439–2450. <https://doi.org/10.1109/TASL.2013.2280209>
 52. Trierweiler Ribeiro G, Guilherme Sauer J, Fraccanabbia N et al (2020) Bayesian optimized echo state network applied to short-term load forecasting. Energies. <https://doi.org/10.3390/en13092390>
 53. Trierweiler Ribeiro G, Alves Portela Santos A, Cocco Mariani V et al (2021) Novel hybrid model based on echo state neural network applied to the prediction of stock price return volatility. Expert Syst Appl 184(115):490. <https://doi.org/10.1016/j.eswa.2021.115490>
 54. Trouvain N, Pedrelli L, Dinh TT et al (2020) Reservoirpy: An efficient and user-friendly library to design echo state networks. In: Farkaš I, Masulli P, Wermter S (eds) Artificial neural networks and machine learning - ICANN 2020. Springer International Publishing, Cham, pp 494–505
 55. Tuncel KS, Baydogan MG (2018) Autoregressive forests for multivariate time series modeling. Pattern Recogn 73:202–215. <https://doi.org/10.1016/j.patcog.2017.08.016>
 56. Verzelli P, Alippi C, Livi L (2021) Learn to synchronize, synchronize to learn. Chaos: An Interdisciplin J Nonlinear Sci 31(8):083,119. <https://doi.org/10.1063/5.0056425>
 57. Wang L, Lv SX, Zeng YR (2018) Effective sparse adaboost method with ESN and FOA for industrial electricity consumption forecasting in China. Energy 155:1013–1031. <https://doi.org/10.1016/j.energy.2018.04.175>
 58. Young S (1994) The htk hidden markov model toolkit: design and philosophy. Entropic Cambridge Rese Lab Ltd 2:2–44

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.