# Benchmarking Reservoir and Recurrent Neural Networks for Human State and Activity Recognition

Davide Bacciu, Daniele Di Sarli(✉), Claudio Gallicchio, Alessio Micheli, and Niccolò Puccinelli

University of Pisa, Pisa, Italy
bacciu@di.unipi.it, daniele.disarli@phd.unipi.it,
{claudio.gallicchio,alessio.micheli}@unipi.it,
n.puccinelli1@studenti.unipi.it

**Abstract.** Monitoring of human states from streams of sensor data is an appealing applicative area for Recurrent Neural Network (RNN) models. In such a scenario, Echo State Network (ESN) models from the Reservoir Computing paradigm can represent good candidates due to the efficient training algorithms, which, compared to fully trainable RNNs, definitely ease embedding on edge devices.

In this paper, we provide an experimental analysis aimed at assessing the performance of ESNs on tasks of human state and activity recognition, in both shallow and deep setups. Our analysis is conducted in comparison with vanilla RNNs, Long Short-Term Memory, Gated Recurrent Units, and their deep variations. Our empirical results on several datasets clearly indicate that, despite their simplicity, ESNs are able to achieve a level of accuracy that is competitive with those models that require full adaptation of the parameters. From a broader perspective, our analysis also points out that recurrent networks can be a first choice for the class of tasks under consideration, in particular in their deep and gated variants.

**Keywords:** Recurrent neural networks · Echo state networks · Human psychological state recognition · Human activity recognition

## 1 Introduction

The class of recurrent neural networks provides a comprehensive framework for learning in the context of sequential data. Besides canonical applications in the area of speech and text processing, recurrent neural networks are often considered the primary choice for learning tasks over time series data such as sensor streams. In particular, the inductive biases of recurrent networks [9] (such as

This work is supported by the EC H2020 programme under project TEACHING (grant n. 871385).


I. Rojas et al. (Eds.): IWANN 2021, LNCS 12862, pp. 168–179, 2021.
https://doi.org/10.1007/978-3-030-85099-9_14

their fading memory property) can be particularly suited for a variety of tasks. For example, an intuitively good fit of their bias is for tasks related to human state and activity recognition from physiological sensors.

Being able to accurately identify and predict the cognitive, physical or psychological state of a human is of fundamental relevance across multiple application scenarios, such as ambient assisted living, healthcare and wellness applications [2]. When humans are subjected to the decisions and operations of an autonomous application, a precise and timely characterization of their psycho-physical state becomes a necessary step to ensure their safety and comfort. Notable applications, in this sense, are autonomous transportation and cooperative human-robot assembly lines. Within such a context, the human state is not only a watchdog for controlling the execution of the autonomous routines, but it can also become a proxy to a non-supervised form of teaching signal that can drive the adaptation of the autonomous system, following the responses that its actions elicit on the humans. The work presented in this paper has been developed within the context of the H2020 project TEACHING[1] [3], which is an ongoing research endeavor targeting specifically the provisioning of innovative methods and systems to enable the development of the next-generation of adaptive autonomous applications. TEACHING puts forward a human-centric perspective based on a synergistic collaboration between the human and the autonomous application, mediated by human psycho-physical reactions, that becomes a driver for adaptation and personalization of the application. The work described in this paper is prodromal to the identification of the candidate learning model to realize a system capable of inferring human state from wearable and environmental sensing devices.

The prediction of the physical or psychological state of a human from sensors requires strong tolerance to noise [1], flexibility to the different personal characteristics of each subject, and, in most practical context, computational efficiency due to the embedding of the models on board of low-power devices. It has been previously shown that Echo State Networks (ESNs) [14,15], from the paradigm of Reservoir Computing [30], are efficient recurrent models [11] which can achieve high predictive performance in human activity recognition applications [4,21]. Recently, it has even been shown that ESN models are capable of an optimal form of federated learning [5], which is often instrumental in human-centric tasks. However, to the best of our knowledge, in the literature there exists no study comprehensively analyzing the performance of ESNs with that achievable by fully trainable alternatives on this kind of tasks or on those regarding the classification of the psychological state.

With this study we set the goal of benchmarking recurrent neural network models for tasks of human state and activity recognition. We do so in a uniform context, and through rigorous experimental settings, considering several datasets from the literature. For our comparison, in addition to the aforementioned ESNs, we will employ the most popular fully trainable recurrent neural architectures, namely the vanilla Recurrent Neural Network (RNN), the Long

[1] www.teaching-h2020.eu.

Short-Term Memory (LSTM) [13] and the Gated Recurrent Unit (GRU) [7]. Moreover, given the architectural and training differences of these models, we also discuss the effects of layering.

The rest of this paper is organized as follows. In Sect. 2 we introduce the neural network models under consideration, which will be used for the experimental comparison described in Sect. 3, where we also discuss the results of the study. Finally, in Sect. 4 we draw the conclusions.

## 2 Background

In this section we provide a brief overview of the recurrent neural network models under consideration. In doing so, we will consider general learning tasks based on sequence classification. In particular, we introduce Vanilla RNNs in Sect. 2.1, ESNs in Sect. 2.2, LSTMs in Sect. 2.3, and GRUs in Sect. 2.4. Note that, for simplicity of exposition, we omit the bias term from all equations.

### 2.1 Vanilla Recurrent Neural Networks

Vanilla Recurrent Neural Networks (RNNs) are one of the simplest and most widely known recurrent models [18]. The discrete-time evolution of the internal state $\mathbf{h}(t) \in \mathbb{R}^{N_R}$ is described by the recurrent equation

$$\mathbf{h}(t) = \tanh(\mathbf{W}\mathbf{x}(t) + \mathbf{U}\mathbf{h}(t-1)), \tag{1}$$

where $\mathbf{x}(t)$ is the driving input signal, and $\mathbf{W}$ and $\mathbf{U}$ are parameter matrices of the model. The output for a sequence of length $T$ can then be computed as

$$\mathbf{y} = \mathbf{V}\mathbf{h}(T), \tag{2}$$

where $\mathbf{V}$ is another parameter matrix. In the specific setting of time-series classification, which is the context of this work, the output $\mathbf{y} \in \mathbb{R}^{N_Y}$ can be interpreted as a vector of scores, one for each class. All the parameters ($\mathbf{V}$, $\mathbf{W}$, and $\mathbf{U}$) are jointly learned by gradient descent.

A multi-layer – or deep – RNN [12] can be constructed by stacking the recurrent layers. The first layer will take as input the time-series $\mathbf{x}(t)$ as in Eq. 1, while the successive layers will take as input the state of the preceding layer. The output is then computed as in Eq. 2, but using as state the one produced by the last layer.

For what concerns the (architectural) hyper-parameters of a RNN, the most important one is arguably the number $N_R$ of recurrent units. In the case of a multi-layer RNN, another crucial hyper-parameter to optimize is the number $L$ of layers.

### 2.2 Echo State Networks

Echo State Networks (ESNs) [14,15] are recurrent networks from the paradigm of Reservoir Computing [19,30]. A common way to instantiate the approach consists of using leaky-integrator neurons [16], which leads to the following state update equation:

$$\mathbf{h}(t) = (1 - \alpha)\mathbf{h}(t-1) + \alpha \tanh(\mathbf{W}\mathbf{x}(t) + \mathbf{U}\mathbf{h}(t-1)), \tag{3}$$

where $\mathbf{W}$ and $\mathbf{U}$ are (typically sparse and high-dimensional) matrices that encode the recurrent and input connections in the model, respectively, while $\alpha \in (0, 1]$ is a scalar leaking rate. The peculiar characteristic of ESNs is that the values in $\mathbf{W}$ and $\mathbf{U}$ are not learned. In fact, they are initialized from a random distribution according to mathematical properties of stability of the underlying dynamical system [14] and then they are left fixed. The only parameters that are trained are those in the (typically linear) output layer, or *readout*, whose output in case of a sequence of length $T$ can be computed as in Eq. 2.

The key advantage of ESNs is their training efficiency [15], which has been extensively confirmed by experimental comparisons [11]. Since only the linear readout is trained, it is possible to compute the weights in $\mathbf{V}$ without incurring in the problems associated with gradient descent training and backpropagation through time for the recurrent layers [6]. Moreover, since the linear readout is the only part of the model to be trained, it is also possible to employ closed form methods such as ridge regression for the computation of the optimal weights.

Deep versions of the ESN model can be constructed by stacking the recurrent layers while preserving the efficient training method [10]. In this case, it is common practice to concatenate the outputs of all layers into a vector in $\mathbb{R}^{LN_R}$ and use this as input to the readout. This is the setting that we use in this work.

*Initialization.* Initialization is a crucial phase for ESN training, as it is strictly related to the stable dynamics of the state. It is common to initialize both $\mathbf{W}$ and $\mathbf{U}$ in Eq. 3 with random values drawn from a uniform distribution on $[-1, 1]$, and then re-scaling $\mathbf{W}$ by an input scaling value $\omega_{in} \in \mathbb{R}^+$, and $\mathbf{U}$ to have a desired spectral radius[2] $\rho \in \mathbb{R}^+$. The values of $\omega_{in}$ and $\rho$, together with the leaking rate $\alpha \in (0, 1]$, constitute the major hyper-parameters for the design of reservoir layers in ESNs.

### 2.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) [13] has been introduced to overcome the classical gradient-propagation limitations of Vanilla RNNs. It introduces the concept of *gates* that regulate the flow of information across states. The evolution of the internal state $\mathbf{h}(t) \in \mathbb{R}^{N_R}$ is described by the equations

[2] I.e., its maximum eigenvalue in absolute value.

$$
\begin{aligned}
\mathbf{f}(t) &= \sigma(\mathbf{W}_f \mathbf{x}(t) + \mathbf{U}_f \mathbf{h}(t-1)) \\
\mathbf{i}(t) &= \sigma(\mathbf{W}_i \mathbf{x}(t) + \mathbf{U}_i \mathbf{h}(t-1)) \\
\mathbf{o}(t) &= \sigma(\mathbf{W}_o \mathbf{x}(t) + \mathbf{U}_o \mathbf{h}(t-1)) \\
\tilde{\mathbf{c}}(t) &= \tanh(\mathbf{W}_c \mathbf{x}(t) + \mathbf{U}_c \mathbf{h}(t-1)) \\
\mathbf{c}(t) &= \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \tilde{\mathbf{c}}(t) \\
\mathbf{h}(t) &= \mathbf{o}(t) \odot \sigma(\mathbf{c}(t)),
\end{aligned}
\tag{4}
$$

in which $\mathbf{f}(t)$, $\mathbf{i}(t)$, and $\mathbf{o}(t)$ respectively indicate the activations of the forget, input, and output gate. The matrices $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_c$ and $\mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_o, \mathbf{U}_c$ contain the model parameters. An output is computed from $\mathbf{h}(t)$ just like for the RNN as in Eq. 2, and all the parameters are trained by gradient descent.

A multi-layer – or deep – LSTM can be constructed in the same way as for a RNN. Likewise, the most important hyper-parameters are again the number $N_R$ of recurrent units and, for a multi-layer LSTM, the number $L$ of layers.

### 2.4 Gated Recurrent Unit

Gated Recurrent Unit (GRU) [7] is a gated recurrent network that has been introduced in the context of statistical machine translation. Compared to LSTM, it employs a fewer number of gates and a fewer number of parameters. The evolution of the internal state $\mathbf{h}(t) \in \mathbb{R}^{N_R}$ is described by the equations

$$
\begin{aligned}
\mathbf{z}(t) &= \sigma(\mathbf{W}_z \mathbf{x}(t) + \mathbf{U}_r \mathbf{h}(t-1)) \\
\mathbf{r}(t) &= \sigma(\mathbf{W}_z \mathbf{x}(t) + \mathbf{U}_r \mathbf{h}(t-1)) \\
\hat{\mathbf{h}}(t) &= \tanh(\mathbf{W}\mathbf{x}(t) + \mathbf{U}(\mathbf{r}(t) \cdot \mathbf{h}(t-1))) \\
\mathbf{h}(t) &= (1 - \mathbf{z}(t)) \cdot \mathbf{h}(t-1) + \mathbf{z}(t) \cdot \hat{\mathbf{h}}(t),
\end{aligned}
\tag{5}
$$

in which $\mathbf{z}(t)$ and $\mathbf{r}(t)$ respectively indicate the activations of the update gate and reset gate. The matrices $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}$ and $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}$ contain the parameters of the model. An output is computed from $\mathbf{h}(t)$ just like for the RNN as in Eq. 2, and all the parameters are trained by gradient descent.

The multi-layer (i.e., deep) architectural setup and the hyper-parameters considerations are as in the case of LSTM (Sect. 2.3).

## 3 Experimental Comparison

In the following we compare the performance of 4 recurrent models (RNN, ESN, LSTM, and GRU) over several datasets for human state and activity recognition. In Sect. 3.1 we describe the general experimental setup, then in Sect. 3.2 we provide details about the datasets and their preprocessing. In Sect. 3.3 we then present and discuss the results.

**Table 1.** Summary of the amount of data that we have selected from each dataset to support our study. The sequence lengths are to be intended after resampling.

| Dataset | | Sequences | Seq. length | Features | Classes |
|---|---|---|---|---|---|
| WESAD | [26] | 952 | 100 (3 s) | 14 | 4 |
| HHAR | [28] | 960 | 100 (2 s) | 12 | 6 |
| PAMAP2 | [22] | 930 | 350 (10 s) | 31 | 4 |
| OPPORTUNITY | [23] | 480 | 75 (3 s) | 130 | 4 |
| ASCERTAIN | [29] | 3360 | 160 (5 s) | 17 | 4 |

### 3.1 General Setup

For the experimental comparison of the models we have divided the data from each dataset into subsequences of fixed length, each associated to a given label (the target class), and we have trained the networks to classify them. In Sect. 3.2 we describe the details about the datasets and their preprocessing. For the convenience of the reader we have also reported a summary in Table 1.

For what concerns the experimental methodology, for each dataset we have split the data in different folds to perform hold-out validation. In particular, about 80% of the data of each dataset has been used for training, 10% for validation, and the remaining 10% for testing. For a fair comparison we have constrained the number of total trainable parameters to be the same for all models, and for each combination of model and dataset we have selected the numbers of recurrent units that allowed to meet the constraint. To choose the reference number of trainable parameters to use in all experiments we have taken the number of trainable parameters exhibited by the best performing ESN on the validation set, with a number of recurrent layers fixed to 1.

For uniformity of experimental conditions, all models including the ESN have been trained by the Adam algorithm [17] and regularized by a combination of L1 and L2 penalties. The loss function that has been employed is the cross-entropy.

The hyperparameters that have been explored are the number of layers ($L \in [1, 6]$), the learning rate of the optimizer ($\eta \in [1e^{-3}, 1e^{-2}]$), the decay rates of Adam ($\beta_1 \in [0.8, 0.99]$ and $\beta_2 \in [0.98, 0.999]$), and the regularization strength (kernel regularization, bias regularization, and activity regularization, all in $[1e^{-6}, 1e^{-4}]$). For the ESN-only hyperparameters, we have optimized the input scaling ($\omega_{in} \in [0.5, 2.0]$, also used as intra-layer scaling), leaking rate ($\alpha \in [0.3, 1.0]$) and spectral radius of the recurrent matrix $\mathbf{U}$ ($\rho \in [0.5, 1.5]$). The best configuration of the hyperparameters has been selected through 100 iterations of random search.

### 3.2 Datasets

We briefly describe here the datasets used and their preprocessing. For those datasets that are publicly available, we also publish our preprocessed data.[3]

[3] https://github.com/danieleds/Benchmarking_RNN_for_HSM.

**WESAD.** The WESAD (*WEarable Stress and Affect Detection*) dataset [26] contains physiological measurements recorded from 15 subjects performing activities designed to induce different mental states, such as stress or amusement. The participants to the study have been exposed to different situation in order to induce a state of stress, amusement, or meditation. Recordings from a neutral mental state have also been collected as a baseline. We consider the aforementioned 4 states (baseline, stress, amusement, and meditation) as classes for a task of classification of the sensor data.

*Preprocessing.* The time series are resampled 32 Hz and standardized. After splitting we obtain a training set of 771 sequences, a validation set of 86 sequences, and a test set of 95 sequences.

**HHAR.** The HHAR (*Heterogeneity Human Activity Recognition*) dataset [28] contains inertial measurements recorded from 9 subjects performing physical activities. The sensors used (gyroscopes and accelerometers) were part of smartphones or smartwatches which were carried or worn by the subjects. To each subject it was requested to perform 6 different physical activities (in no particular order): standing, sitting, walking, waking up the stairs, walking down the stairs, and biking. The sensor data is labeled with the associated activity: we consider these activities as classes for a task of classification of the sensor data.

*Preprocessing.* The time series are resampled 70 Hz, and we consider 16,000 samples for each activity. After splitting we obtain a training set of 768 sequences, a validation set of 96 sequences, and a test set of 96 sequences.

**PAMAP2.** The PAMAP2 (*Physical Activity Monitoring*) dataset [22] contains physiological and inertial measurements recorded from a heart rate monitor and 3 inertial measurement units. To each of the 9 subjects it was requested to perform 18 activities (of which 6 were optional) such as running, watching TV, car driving, and so on. For our purposes we only select the top 4 activities in terms of how many participants did actually perform it. These are lying, sitting, standing, and walking, which have been used for a classification task.

*Preprocessing.* The features related to the second accelerometer are removed (it is defective). The orientation features are also removed. The time series are upsampled 100 Hz. After splitting we obtain a training set of 753 sequences, a validation set of 84 sequences, and a test set of 93 sequences.

**OPPORTUNITY.** The OPPORTUNITY Activity Recognition dataset [23] includes measurements from body-worn sensors (7 inertial measurement units, 12 3D acceleration sensors, 4 3D localization information) and other sensors associated to objects in a controlled environment. For the purposes of our study, we only consider the body-worn sensors, in particular the 3D acceleration, 3D

rate of turn, 3D magnetic field, and orientation of the sensors. The 4 participants were required to perform several activities in different contexts (e.g., making coffee, eating a sandwich, cleaning the room), and the data was labeled according to four basic motor activities (standing, walking, lying, and sitting). We employ these basic motor activities in a classification task.

*Preprocessing.* Sequences containing null data are excluded. The features are already at a uniform sampling rate 30 Hz. After splitting we obtain a training set of 389 sequences, a validation set of 43 sequences, and a test set of 48 sequences.

**ASCERTAIN.** The ASCERTAIN dataset [29] has been built to support tasks of emotion and personality recognition from commercially available sensors. Signals include electrocardiogram, electroencephalogram, galvanic skin response, and facial activity, which were recorded while the subjects were watching affective movie clips (36 clips per subject). After each clip, the 58 subjects were required to give a rating of their state of arousal (indicating a value between 0 and 6) and valence (–3 to 3). We consider these values as axes of the circumplex model [24] and we classify the quadrant. This yields the following four classes: high arousal - high valence, high arousal - low valence, low arousal - high valence, and low arousal - low valence.

*Preprocessing.* Sequences containing invalid data are excluded. The features are resampled 32 Hz. After splitting we obtain a training set of 2722 sequences, a validation set of 302 sequences, and a test set of 336 sequences.

### 3.3 Results and Discussion

In Table 2 we report the accuracy obtained by the four models under consideration. We highlight how the values of accuracy obtained by the different models are mostly above 90% on all datasets. In fact, the average accuracy across all models and datasets is 94.67%, with a standard deviation of 4.34%. This indicates that, in general, recurrent neural network models are indeed particularly fit for predicting the mental or physical state of human subjects from sensors.

Due to the complex preprocessing of the datasets, which involves many choices that are difficult to replicate without the source code available, it is not always easy to compare the results that we have obtained with those found in the literature for non-recurrent models. That said, we have observed that the recurrent models that we have tested almost consistently surpass non-recurrent models in the literature on the same datasets. This is especially true for the datasets of emotional state (WESAD and ASCERTAIN) and OPPORTUNITY, where the improvement can be as high as 15% [20,25,26,29]. In particular, for WESAD, the studies in [20,26] reach an accuracy of 80% while limiting to the classification of 3 classes, which is significantly lower than the multi-layer recurrent models which we have shown to be able to achieve an accuracy above 94% for the classification of all 4 classes (with a peak at 98% for GRU). On PAMAP2

**Table 2.** Average accuracy and standard deviation on the test set. The averages and standard deviations are computed by retraining the models 5 times, each with a different random initialization of the weights.

| | WESAD | | HHAR | | PAMAP2 | | OPPORTUN. | | ASCERTAIN | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Avg | Std | Avg | Std | Avg | Std | Avg | Std | Avg | Std |
| RNN | 94.62 | 2.84 | 78.54 | 2.04 | 96.00 | 3.39 | 96.84 | 2.58 | 94.77 | 0.78 |
| ESN | 94.96 | 2.60 | 89.79 | 3.81 | 97.50 | 2.74 | 94.74 | 5.77 | 96.54 | 0.77 |
| LSTM | 95.48 | 1.17 | 92.71 | 2.72 | 96.50 | 1.22 | 93.08 | 2.88 | 94.63 | 0.00 |
| GRU | 98.13 | 1.16 | 98.54 | 0.83 | 98.50 | 2.00 | 96.84 | 2.58 | 94.63 | 0.00 |

[22], we can observe accuracies that are in line with our results (97–98% for our RNNs and 94–99% for the non-recurrent models reported in the study). For HHAR, the accuracy obtained by the ESN (90%) and the LSTM (93%) is similar to the one obtained by k-NN (91%) [28] and a feedforward DNN (93%) [27], but the GRU (98.5%) shows superior performance by more than 5% points. We also point out that for DNN the number of trainable parameters is 4 times larger than those employed by our RNNs (24,000 vs 6000).

Still from Table 2, it can be particularly interesting to focus on the performance of ESN. For this relatively simple model, in which the recurrent and input connections are not trained, we can observe how the accuracy is often comparable to that of the fully trained recurrent models. On average, in the results reported in Table 2 the accuracy of ESN is only 2.65% points below that of the best performing model, and in the case of ASCERTAIN the ESN actually surpasses all other models.

As a further observation, we can also see that gated models perform generally better than non-gated models. In fact, across all datasets, the gated networks (LSTM and GRU) achieve an average accuracy of 95.90%, while the non-gated networks (ESN and RNN) achieve a lower 93.43%. This difference shows that gating mechanisms are indeed useful in the context of sensor data for human state and activity recognition, even if the long-term dependencies in this kind of data are arguably less complicated than those found in other contexts such as natural language processing [7]. Since the use of gates appears to provide a critical advantage in terms of predictive performance, our results motivate further research on how to construct gated ESNs [8].

In Table 3 we have reported the number of layers, recurrent units, and trainable parameters for each combination of model and dataset. From the table we can observe how in almost all cases the reported models have more than one layer. Since the configurations that have been reported are the best performing ones, this means that to achieve maximum performance – under the constraint of a fixed number of trainable parameters – the architecture needs to exploit layering. In order to quantify the advantage of layering we also report in Table 4 the same results of Table 2, but only selecting the best performing non-layered models instead of the overall best performing models. In this case the average

**Table 3.** Number of layers (L), recurrent units per layer ($N_R$), and trainable parameters (P) for the best performing hyperparametrization of each model, across all tasks under consideration. The number of recurrent units has been scaled to obtain almost the same number of trainable parameters across models for each given task.

| Model | WESAD | | | HHAR | | | PAMAP2 | | | OPPORTUN. | | | ASCERTAIN | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L | $N_R$ | P | L | $N_R$ | P | L | $N_R$ | P | L | $N_R$ | P | L | $N_R$ | P |
| RNN | 3 | 26 | 3930 | 4 | 28 | 6110 | 3 | 21 | 3007 | 3 | 18 | 4090 | 2 | 30 | 3394 |
| ESN | 3 | 750 | 4004 | 2 | 668 | 6018 | 3 | 550 | 2932 | 3 | 750 | 4004 | 4 | 625 | 3124 |
| LSTM | 2 | 16 | 4164 | 3 | 16 | 6182 | 2 | 11 | 2952 | 1 | 6 | 4260 | 5 | 8 | 3044 |
| GRU | 2 | 18 | 3964 | 3 | 18 | 5946 | 2 | 14 | 3294 | 3 | 8 | 4252 | 2 | 16 | 3380 |

**Table 4.** Average accuracy and standard deviation on the test set, restricting the number of recurrent layers to 1. The averages and standard deviations are computed by retraining the models 5 times, each with a different random initialization.

| | WESAD | | HHAR | | PAMAP2 | | OPPORTUN. | | ASCERTAIN | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Avg | Std | Avg | Std | Avg | Std | Avg | Std | Avg | Std |
| RNN | 83.09 | 8.00 | 74.78 | 5.04 | 95.27 | 1.45 | 87.59 | 1.69 | 91.83 | 0.14 |
| ESN | 92.44 | 0.75 | 88.96 | 2.24 | 95.64 | 0.89 | 87.50 | 3.73 | 91.76 | 0.00 |
| LSTM | 94.23 | 3.49 | 91.50 | 4.37 | 96.36 | 1.99 | 93.08 | 2.88 | 91.76 | 0.00 |
| GRU | 97.73 | 1.01 | 97.08 | 1.21 | 96.36 | 0.00 | 92.31 | 2.43 | 91.76 | 0.00 |

accuracy of all models across all tasks is 91.55%, while allowing layering as in Table 2 yields an average of 94.67% (again, while keeping the total number of trainable parameters fixed). It is clear then how layered architectures allow to better model the different time-scales that are present in the sensor data.

## 4 Conclusions

In this paper, we have benchmarked recurrent neural network models on tasks on human state and activity recognition from streams of sensor-gathered data. In particular, our analysis comprised 4 different recurrent neural network models and 5 datasets of human state and activity recognition. In most cases, the recurrent models are able to obtain a level of accuracy well above 90%: this allows to state that this kind of models is particularly suited to the task and can be used as a reasonable starting point to build effective predictive systems.

We have shown how a simple and efficient recurrent model such as the ESN is still highly effective for this kind of tasks. This is a significant result in light of practical applications of human state and activity recognition, which often require the implementation of predictive models on small low-power devices and thus can benefit from the efficient training process of ESNs.

We can thus conclude that ESNs are valid alternatives to the most popular (fully trainable) recurrent models and allow effective predictive performances. This is accompanied by advantages in terms of training efficiency and optimal federation: both these aspects are unique to ESNs, have previously been demonstrated in the literature, and are often key to human-centric tasks. Additional advantages in practical applications include privacy preservation (there is no need to transmit the data to a central server, since training can happen locally) and light bandwidth requirements for model transmission over the network (since most of the weights are fixed).

Furthermore, our results suggest that employing deep *and* gated architectures is particularly relevant for these tasks. While the techniques for layering recurrent models are well established for RNNs, LSTMs, GRUs and ESNs [10], applying an ESN-like training methodology to a gated architecture is more tricky and subject of ongoing research efforts [8].

In conclusion, we have shown the effectiveness of recurrent neural networks for tasks of human state and activity recognition, and of ESNs in particular, which thanks to their characteristics can serve as the first choice in human-centric tasks.

## References

1. Bacciu, D., Bertoncini, G., Morelli, D.: Randomized neural networks for preference learning with physiological data. Neural Computing Applications (2021)
2. Bacciu, D., Colombo, M., Morelli, D., Plans, D.: Randomized neural networks for preference learning with physiological data. Neurocomputing **298**, 9–20 (2018)
3. Bacciu, D., et al.: Teaching - trustworthy autonomous cyber-physical applications through human-centred intelligence. In: Submitted (2021)
4. Bacciu, D., Barsocchi, P., Chessa, S., Gallicchio, C., Micheli, A.: An experimental characterization of reservoir computing in ambient assisted living applications. Neural Comput. Appl. **24**(6), 1451–1464 (2013). https://doi.org/10.1007/s00521-013-1364-4
5. Bacciu, D., Di Sarli, D., Faraji, P., Gallicchio, C., Micheli, A.: Federated reservoir computing neural networks. In: IJCNN. IEEE (2021)
6. Bengio, Y., Simard, P.Y., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
7. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP, pp. 1724–1734. ACL (2014)
8. Di Sarli, D., Gallicchio, C., Micheli, A.: Gated echo state networks: a preliminary study. In: INISTA, pp. 1–5. IEEE (2020)
9. Gallicchio, C., Micheli, A.: Architectural and Markovian factors of echo state networks. Neural Netw. **24**(5), 440–456 (2011)
10. Gallicchio, C., Micheli, A., Pedrelli, L.: Design of deep echo state networks. Neural Netw. **108**, 33–47 (2018)
11. Gallicchio, C., Micheli, A., Pedrelli, L.: Comparison between DeepESNs and gated RNNS on multivariate time-series prediction. In: ESANN (2019)
12. Hermans, M., Schrauwen, B.: Training and analysing deep recurrent neural networks. Adv. Neural Inf. Process. Syst. **26**, 190–198 (2013)

13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. German National Research Center for Information Technology GMD Technical Report, Bonn (2001)
15. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667), 78–80 (2004)
16. Jaeger, H., Lukoševičius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw. **20**(3), 335–352 (2007)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
18. Kolen, J.F., Kremer, S.C.: A field guide to dynamical recurrent networks. John Wiley & Sons (2001)
19. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**(3), 127–149 (2009)
20. Nkurikiyeyezu, K., Yokokubo, A., Lopez, G.: The effect of person-specific biometrics in improving generic stress predictive models. arXiv:1910.01770 (2019)
21. Palumbo, F., Gallicchio, C., Pucci, R., Micheli, A.: Human activity recognition using multisensor data fusion based on reservoir computing. J. Ambient Intell. Smart Environ. **8**(2), 87–107 (2016)
22. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: ISWC, pp. 108–109. IEEE Computer Society (2012)
23. Roggen, D., et al.: Collecting complex activity datasets in highly rich networked sensor environments. In: INSS, pp. 233–240 (2010)
24. Russell, J.A.: A circumplex model of affect. J. Pers. Soc. Psychol. **39**(6), 1161 (1980)
25. Sagha, H., et al.: Benchmarking classification techniques using the opportunity human activity dataset. In: SMC, pp. 36–40. IEEE (2011)
26. Schmidt, P., Reiss, A., Dürichen, R., Marberger, C., Laerhoven, K.V.: Introducing wesad, a multimodal dataset for wearable stress and affect detection. In: ICMI, pp. 400–408. ACM (2018)
27. Sozinov, K., Vlassov, V., Girdzijauskas, S.: Human activity recognition using federated learning. In: ISPA/IUCC/BDCloud/SocialCom/SustainCom, pp. 1103–1111. IEEE (2018)
28. Stisen, A., et al.: Smart devices are different: assessing and mitigatingmobile sensing heterogeneities for activity recognition. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 127–140. SenSys 2015, Association for Computing Machinery, New York (2015)
29. Subramanian, R., Wache, J., Abadi, M.K., Vieriu, R.L., Winkler, S., Sebe, N.: ASCERTAIN: emotion and personality recognition using commercial sensors. IEEE Trans. Affect. Comput. **9**(2), 147–160 (2018)
30. Verstraeten, D., Schrauwen, B., d'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Netw. **20**(3), 391–403 (2007)