Human Activity Recognition

Giuseppe Lombardi
Computer Science (Artificial Intelligence)
g.lombardi11@studenti.unipi.it

CNS course (674AA), Academic Year: 2021/2022
Date: June 7, 2022

Abstract

Activity recognition plays a key role in providing activity assistance and care for users in smart homes. In this work, I present an activity recognition system that classifies in real-time a set of common daily activities. This system exploits the data sampled by sensors embedded in a smartphone carried out by the user and the reciprocal Received Signal Strength (RSS) values coming from worn wireless sensor devices. In order to achieve an effective and responsive classification, I model the RSS stream, using Recurrent Neural Networks (RNNs) implemented as efficient Echo State Networks (ESNs), within the Reservoir Computing (RC) paradigm. In this report, the performance of the proposed activity recognition system is assessed on a purposely collected real-world dataset, taking also into account two competitive neural network approachs for performance comparison. My results show that the proposed system reaches a good accuracy.

1 Introduction

In the recent years, **human activity recognition** is one of the most important and studied tasks in different research fields, including video surveillance systems, human-computer interaction, and robotics for human behavior characterization.

Particularly, in this project, I propose a human activity recognition system that is able to classify input data with a label among the following ones: *Lying*, *Sitting*, *Standing*, *Walking*, and *Bending* (both keeping the legs straight and keeping the legs folded, see Figure 1).

As input data, I use the *AReM* (Activity Recognition System based on Multisensor data fusion) dataset [1] which contains time-series generated by a Wireless Sensor Network (WSN), where each epoch time is large 250 milliseconds. For each time step, each row of the dataset corresponds to the mean and variance of the Received Signal Strenght (RSS) values (capturing by three sensors, see Figure 2).

The model used in this system to be able to classify the input data is the *echo state network* (ESN), exploiting the Reservoir Computing (RC) paradigm, suitable for processing time-series classification tasks, such as human activity recognition. Particularly,

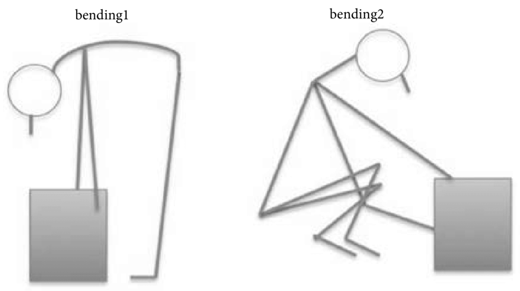


Figure 1: The two types of bending activity. Bending 1 on the left and bending 2 on the right.

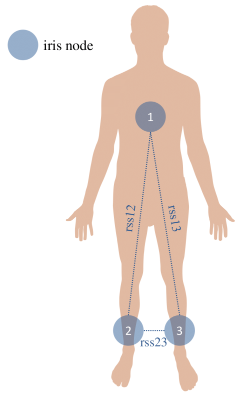


Figure 2: Position of the three sensors used to collect data.

in this project, I use two variation of the ESNs, the *Leaky Integrator ESN* (LI-ESN) and the *Euler State Network* (EuSN).

In order to emphasize that the ESNs are a good compromise between performance and efficiency, I compared the LI-ESN and the EuSN with two other models: *Input Delay Neural Network* (IDNN) and *Gated Recurrent Unit* (GRU). In addition, I decided to compare ESNs with the latter two models because, in the literature, they are widely used for the classification of time-series.

The report is organize as follow. In Section 2, I describe the design choices, the code structure and the validation schema; in Section 3, I report the results obtained with the different used models, in terms of performance on the training, validation and test set; finally, in Section 4, I discuss some conclusions.

2 Method

As mentioned in the Section 1, in this project, I use ESN in order to classify a stream of data captured by a WSN. I choose to design the model in order to classify each step of

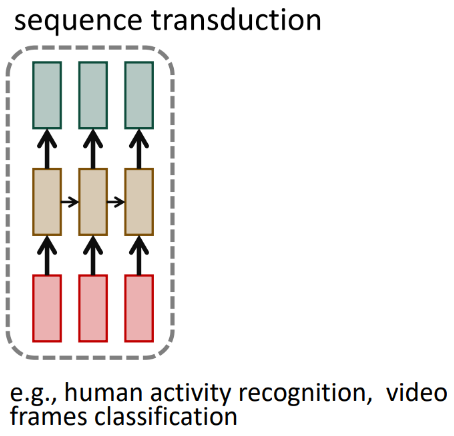

sequence transduction
e.g., human activity recognition, video frames classification

Figure 3: Sequence-to-sequence classification task.

sequence data (i.e., *sequence-to-sequence classification*, see Figure 3).

The architecture of a ESN is composed of an input layer with N_u units, reservoir layer with N_h recurrent non-linear units, and a readout layer with N_y feed-forward linear units. The reservoir encodes the input history of the driving input signal into a network state. The readout computes the output of the model by linearly combining the activation of the reservoir units.

The LI-ESN has a large and sparsely connected reservoir layer and at each time step t, the reservoir computes a state $\mathbf{x}(t) \in \mathbb{R}^{N_h}$ according to a state transition function:

$$\mathbf{x}(t) = (1-a)\mathbf{x}(t-1) + a\tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$$

where $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ is the input of the LI-ESN at time step t, $\mathbf{W}_{in} \in \mathbb{R}^{N_h \times N_u}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}} \in \mathbb{R}^{N_h \times N_h}$ is the recurrent reservoir weight matrix, tanh is the element-wise applied activation function, and $a \in [0,1]$ is the leaking rate parameter, used to control the speed of the reservoir dynamics. The weight matrix $\hat{\mathbf{W}}$ of the reservoir layer of the LI-ESN is sparsely connected. This allows me to have a high-dimensional hidden space while maintaining a low computational cost. To ensure the stability of the model the spectral radius of $\tilde{\mathbf{W}} = (1-a)I + a\hat{\mathbf{W}}$ should be less than 1, but even slightly higher values are experimentally effective. The stability of the model is given by the echo state property and in the initial part the state could be affected by initial condition [2]. So for this reason the orbits synchronize after a transient and the first states can be discarded.

The EuSN considers the operation of a continuous-time recurrent neural system modeled by the following ODE:

$$\mathbf{x}'(t) = \tanh(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t) + \mathbf{b}),$$

where $\mathbf{b} \in \mathbb{R}^{N_h}$ is the bias. The aim of EuSN is to force the system to be stable and non-dissipative, avoiding explosion of input perturbations that would result in poor generalization. Recalling that an ODE is stable if the real part of all eigenvalues of the corresponding Jacobian are ≤ 0, EuSN impose that the real part of all eigenvalues of the corresponding Jacobian are ≈ 0. This condition avoid that the resulting dynamical system is lossy. In this case the eigenvlaues are purely complex, and the orbits are circular around the stability point (Figure 4).

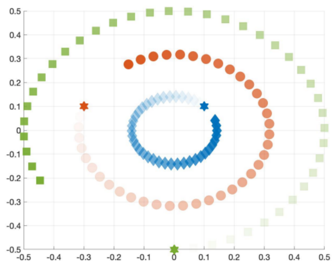


Figure 4: Circular orbits by dynamical system obtained by EuSN.

Solving the ODE by using the forward Euler method, we obtain the discretization:

$$\mathbf{x}(t) = \mathbf{x}(t-1) + \epsilon \tanh(\mathbf{W}_{in}\mathbf{u}(t) + (\hat{\mathbf{W}} - \gamma \mathbf{I})\mathbf{x}(t) + \mathbf{b})$$

where ϵ is the step size of integration, and γ is a diffusion coefficient used for stabilizing the discrete forward propagation [3]. A simple way to impose that real parts of the eigenvalues are 0 is that $\hat{\mathbf{W}}$ is anti-symmetric. For the EuSN is not required to scale the spectral radius of $\hat{\mathbf{W}}$, because the stability is due to the anti-symmetric structure of the matrix. To balance the contributions of the different terms in the state transition the recurrent, input and bias scaling coefficients ω_r, ω_{in}, and ω_b are introduced.

In an ESN model, the readout computes at each time step t the output of the model $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ through a linear combination of the elements in the stat $\mathbf{x}(t)$:

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t)$$

where $\mathbf{W}_{out} \in \mathbb{R}^{N_y \times N_h}$ is the readout-to-reservoir weight matrix. Since the number of classes in this case is $N_y = 7$ (see Section 1), the activity recognized i is defined as follow:

$$y_i(t) = max_{j=1}^{N_y}(y_j(t))$$

where $y_j(t)$ is the output vector at time step t. The weight matrix $\mathbf{W}_{out}$ is the only trainable part of a ESN model and it is trained offline by the direct solution of the least squares problem.

The entire code is written in `MATLAB`. The IDNN and GRU models are implemented by exploiting `Deep Learning Toolbox`.

In particular, IDNN is implemented with `convolution1dLayer()`, which allows more flexibility in modifying parameters, instead of `timedelaynet()`.

2.1 Validation schema

I split the available data into a development set and a test set (for the evaluation). The number of sequences in the tr set and in the test set is 64 and 24, respectively. The values of hyper-parameters of LI-ENS, EuSN, IDNN and GRU are selected on a validation set, in order to maximize the K-class classification accuracy (1), according to a holdout model selection scheme over the development set. The number of sequences in the validation set is approximately the 25% of the number of sequences in the development set, i.e. 14. The sequences are shuffled so that the training, validation, and test sets have at least one sample per class.

$$accuracy_K = \frac{\sum_{i=1}^{K} tp_i}{\sum_{i=1}^{K} N_i}. \tag{1}$$

3 Experimental results

For each architecture (LI-ESN, EuSN, IDNN and GRU), the best model is chosen based on the K-class classifican accuracy on the validation set.

In order to select the best model for the LI-ESN, I implement a grid search on the following hyper-parameters: the number of hidden neurons N_h, the connectivity, the leaking rate a, the input scaling ω_{in}, the spectral radius ρ of the reservoir matrix $\hat{\mathbf{W}}$, the regularization λ_r and the transient. For each choice of hyper-parameters 5 different reservoir guesses are performed, averaging the K-class accuracy on the training and the validation set in order to better evaluate the model. The tried values of each hyper-parameter are reported in Table 1.

Hyper-parameter	**Values**
N_h	10, 50, 100, 300, 500
connectivity	10%
a	0.1, 0.3, 0.5, 0.7, 1
ω_{in}	0.4
ρ	0.9
λ_r	0.0001, 0.001, 0.01, 0.5, 1, 5, 10, 100, 1000
transient	0, 24

Table 1: Hyper-parameters used to select the best model (LI-ESN).

In order to select the best model for the EuSN, I implement a grid search on the following hyper-parameters: the number of hidden neurons N_h, the input scaling ω_{in}, the reservoir scaling ω_r, the bias scaling ω_b, the two factors ϵ and γ, the regularization λ_r and the transient (Table 2). For each configuration of hyper-parameters 5 different reservoir guesses are performed.

Hyper-parameter	**Values**
N_h	100, 200
ω_{in}	0.4
ω_r	0.4, 0.8
ω_b	0.2,0.4,0.6
ϵ	0.001, 0.0001
γ	0.001, 0.0001
λ_r	0.0001, 0.001, 0.01, 0.1
transient	0, 24

Table 2: Hyper-parameters used to select the best model (EuSN).

I also implement a grid search for selecting the best model for IDNN too. In this case, I use the stochastic gradient descent with momentum as optimization algorithm. For each choice of hyper-parameters, the performance on the training and validation set are averaged on 5 different runs, corresponding to different initialization of the weight matrix in the interval $[-0.001, 0.001]$. For this model, the hyper-parameters on which I do the grid search are: learning rate η, momentum α, weight decay regularization λ_{wd}, number of hidden units N_h, maximum number of epochs, early stopping. Their values are reported in Table 3.

Hyper-parameter	**Values**
η	0.0001, 0.002, 0.0025, 0.003, 0.005, 0.01, 0.1
α	0, 0.001, 0.0001
λ_{wd}	0, 0.01, 0.001, 0.0001, 0.00001
N_h	10, 50, 100, 500
max epochs	5000
early stopping	10

Table 3: Hyper-parameters used to select the best model (IDNN).

Finally, I train the GRU with the *Adam* optimization algorithm [4] to select the best model of GRU, the hyper-parameters that I fix and test are: N_h, initial η, λ_{wd}, maximum number of epochs, early stopping and the gradient threshold (if the gradient exceeds the threshold, then it is clipped). The values are reported in Table 4.

After running grid searches, I select the best models for each described architecture. Results in terms of best hyper-parameters and K-class classification accuracy for the training, validation and test set are reported in Table 5.

Now, I show the values of accuracy and F1 score on the test set for each class - corresponding to an activity (Table 6). The overall scores are given by the average of accuracy per-class and F1 macro score. All the mentioned measures are reported in [1]. The table shows that the best model in terms of performance is the GRU, instead the

Hyper-parameter	**Values**
initial η	0.001
N_h	100
λ_{wd}	0.0001
gradient threshold	2
max epochs	50, 100
early stopping	10

Table 4: Hyper-parameters used to select the best model (GRU).

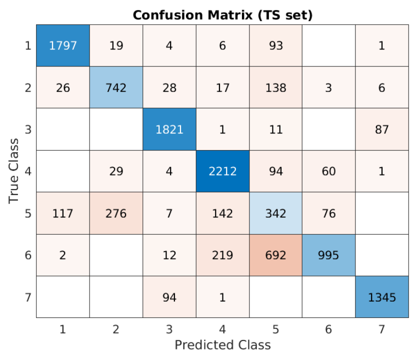
Models	**Best hyper-parameters**	**TR Acc.**	**VL Acc.**	**TS Acc.**
LI-ESN	N_h: 500; a: 0.1; λ_r: 0.1; transient: 0	0.834	0.688	0.803
EuSN	N_h: 100; ω_{in}: 0.4; ω_r: 0.4; ω_b: 0.2; ϵ: 0.0001; γ: 0.001; λ_r: 0.1;	0.840	0.762	0.812
IDNN	N_h: 100; η: 0.002; α: 0.0001; λ_{wd}: 0.00001	0.751	0.670	0.766
GRU	max epochs: 100	0.977	0.833	0.911

Table 5: Values of the best hyper-parameters and of the K-class classification accuracy for training (TR), validation (VL) and test (TS) sets, for each tested architecture.

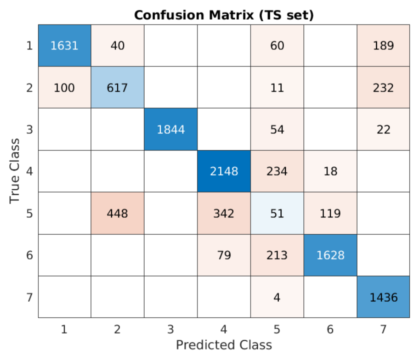
worst is the IDNN. The worst performance of all fours models is on the *standing*, *sitting* and *bending2* classes. As can be seen by the confusion matrices (Figure 5) the activities standing and sitting are confused by models, due to the fact that the phase of sitting down and standing up are not included in the signal, making the RSS streams similar [1]. Futhermore, all models often wrongly predict the sitting activity as bending2 (Figure 5), probably due to the fact that folded legs are also in the bending2 position (Figure 1). The LI-ESN and the EuSN model have very similar overall performance, but they have different performance per-class. The EuSN is better in standing and cycling class, but it is very bad in sitting class.

Activity	**Accuracy**				**F1**			
	LI-ESN	EuSN	IDNN	GRU	LI-ESN	EuSN	IDNN	GRU
Bending1	0.977	0.966	0.963	0.998	0.931	0.894	0.888	0.996
Bending2	0.953	0.928	0.932	0.958	0.733	0.598	0.628	0.799
Cycling	0.979	0.993	0.982	0.998	0.936	0.980	0.944	0.998
Lying	0.950	0.942	0.934	0.998	0.885	0.865	0.848	0.995
Sitting	0.857	0.871	0.841	0.913	0.294	0.064	0.265	0.330
Standing	0.908	0.962	0.897	0.954	0.651	0.884	0.580	0.863
Walking	0.984	0.961	0.985	0.999	0.934	0.865	0.941	0.998
Overall	0.944	0.946	0.933	0.975	0.775	0.739	0.743	0.863

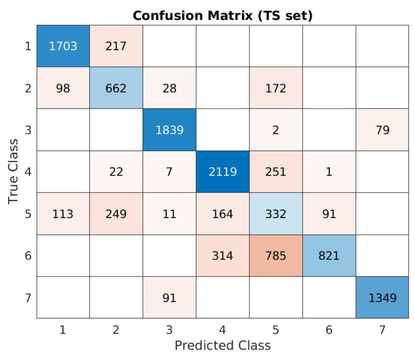
Table 6: Model performance (for each class) on the test set in terms of accuracy and F1 score per-class. The overall performance is given by the average accuracy an the F1 macro score.



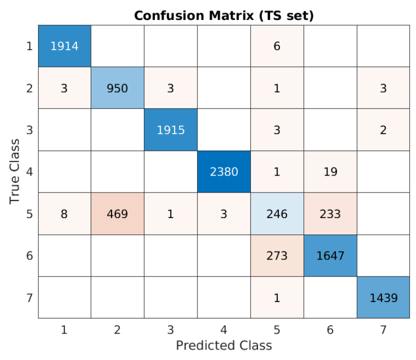
(a) LI-ESN



(b) EuSN



(c) IDNN



(d) GRU

Figure 5: Confusion matrices on test set.

The Figure 6 shows the elapsed time to refit the best configuration of each model on the development set and thus the model efficiency in terms of computational time.

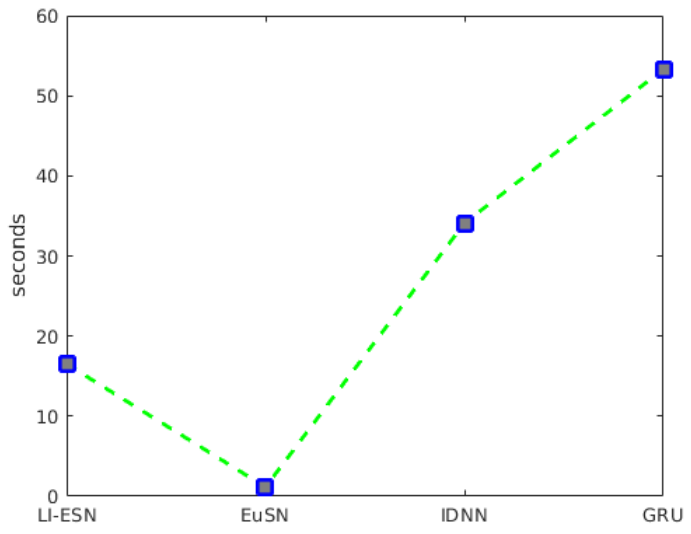


Figure 6: Elapsed time in second for refit the best configuration of each model on the development set.

The two ESNs are faster than the two other RNNs. The GRU is the slowest (53.309 s) neverthless the low number of epochs due to the use of Adam [4]. The EuSN is the fastest (1.150 s) thanks to the reservoir paradigm and the smaller number of hidden neurons with respect to the LI-ESN.

Finally, I predict a multi-class signal - obtained by concatenating dinstict signal - with the LI-ESN. As expected, the prediction of the sitting phase (class 5) is the worst. The LI-ESN configuration is more than 10 times slower than the EuSN configuration (Figure 6), but it can predict a received signal at time t 171 ms on average which is a comparable to the arrival time signal.

4 Conclusions

In conclusion, although an extensive grid search was not implemented on GRU as it was for the other models, it is the model with the best performance on the test set for all 7 target classes of the task under consideration in this project.

Nevertheless, since this system is designed to process and classify data streams in real time, the use of GRU could affect the time performance of the system. Therefore, although the performance of ESNs in terms of accuracy and F1 score is worse than GRU, it could be the right solution for the human activity recognition task, especially because embedded systems often have limited memory.

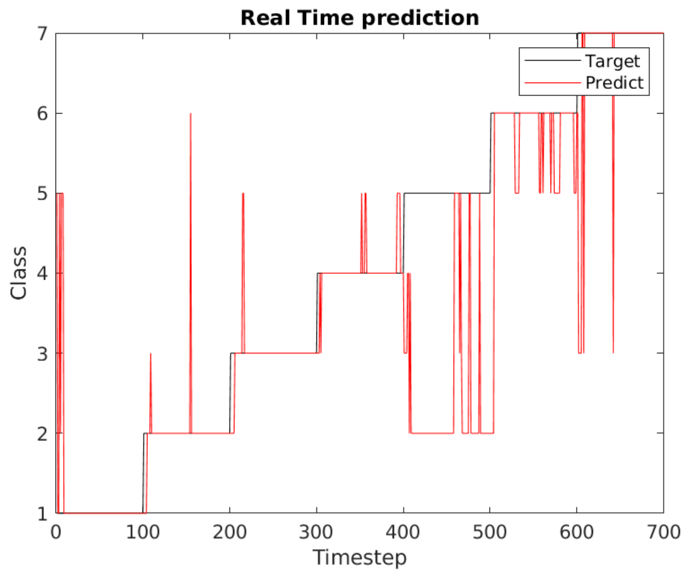


Figure 7: Prediction of LI-ESN on a multi-class signal.

Bibliography

[1] Palumbo F., Gallicchio C., Pucci R., and Micheli A. Human activity recognition using multisensor data fusion based on reservoir computing. *Journal of ambient intelligence and smart environments (Print)*, 8:87–107, 2016.

[2] Claudio Gallicchio. Chasing the echo state property. *CoRR*, abs/1811.10892, 2018.

[3] Claudio Gallicchio. Euler state networks, 2022.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.