sensors

Article

EEG-Based Alzheimer's Disease Recognition Using Robust-PCA and LSTM Recurrent Neural Network

Michele Alessandrini [1], Giorgio Biagetti [1,*], Paolo Crippa [1], Laura Falaschetti [1], Simona Luzzi [2] and Claudio Turchetti [1]

1 Department of Information Engineering, Università Politecnica delle Marche, Via Brecce Bianche 12, I-60131 Ancona, Italy; m.alessandrini@univpm.it (M.A.); p.crippa@univpm.it (P.C.); l.falaschetti@univpm.it (L.F.); c.turchetti@univpm.it (C.T.)

2 Clinica Neurologica, Dipartimento di Medicina Sperimentale e Clinica, Università Politecnica delle Marche, Via Conca 71, I-60020 Ancona, Italy; s.luzzi@staff.univpm.it

* Correspondence: g.biagetti@univpm.it; Tel.: +39-071-220-4168

Abstract: The use of electroencephalography (EEG) has recently grown as a means to diagnose neurodegenerative pathologies such as Alzheimer's disease (AD). AD recognition can benefit from machine learning methods that, compared with traditional manual diagnosis methods, have higher reliability and improved recognition accuracy, being able to manage large amounts of data. Nevertheless, machine learning methods may exhibit lower accuracies when faced with incomplete, corrupted, or otherwise missing data, so it is important do develop robust pre-processing techniques do deal with incomplete data. The aim of this paper is to develop an automatic classification method that can still work well with EEG data affected by artifacts, as can arise during the collection with, e.g., a wireless system that can lose packets. We show that a recurrent neural network (RNN) can operate successfully even in the case of significantly corrupted data, when it is pre-filtered by the robust principal component analysis (RPCA) algorithm. RPCA was selected because of its stated ability to remove outliers from the signal. To demonstrate this idea, we first develop an RNN which operates on EEG data, properly processed through traditional PCA; then, we use corrupted data as input and process them with RPCA to filter outlier components, showing that even with data corruption causing up to 20% erasures, the RPCA was able to increase the detection accuracy by about 5% with respect to the baseline PCA.

Keywords: recurrent neural network (RNN); deep neural network (DNN); electroencephalography (EEG); Alzheimer's disease (AD); principal component analysis (PCA); robust PCA (RPCA)

check for updates

Citation: Alessandrini, M.; Biagetti, G.; Crippa, P.; Falaschetti, L.; Luzzi, S.; Turchetti, C. EEG-Based Alzheimer's Disease Recognition Using Robust-PCA and LSTM Recurrent Neural Network. *Sensors* **2022**, *22*, 3696. https://doi.org/10.3390/s22103696

Academic Editor: Ki H. Chon

Received: 14 March 2022
Accepted: 10 May 2022
Published: 12 May 2022

1. Introduction

Alzheimer's disease (AD) is a neurodegenerative disease resulting in cognitive impairments, functional deficits and loss of memory. It is one of the most widespread forms of dementia and has severe effects on all aspects of a patient's life. Currently no treatments are known for AD and the average survival time is about 4.5 years [1]. The number of subjects affected by AD has been sensibly increasing over the past years and it is expected to reach 15 million by 2050 [2].

AD can be diagnosed at its final stage or in a preclinical, mild stage called mild cognitive impairment (MCI), showing only isolated cognitive deficits that may later result in AD [3–5]. Diagnosing AD, especially at early stages, is difficult because symptoms are compatible with normal consequences of ageing or with other pathologies, and because an exact diagnosis requires histological analysis of the brain or other complex exams such as structural magnetic resonance imaging [6].

The use of electroencephalography (EEG) has recently grown as a means to diagnose neurodegenerative pathologies such as AD [7–13]. EEG is a non-invasive test that records the electrical activity of the brain measured at different sites on patient's scalp, resulting in indirect information about the physiological conditions of the brain. The resulting waveforms can be divided in different frequency bands, which in turn can be ascribed to different cerebral activities.

Several studies have shown that variations in frequency patterns of EEG signals can be related to a variety of neurodegenerative diseases, including AD and MCI [14–19]. However, analyzing EEG data can be a difficult task, because these exams typically generate large amounts of data, which must be precisely scanned for the specific pattern one is looking for. If done manually, this must be performed by trained medical professionals.

That is why a considerable amount of research exists in the application of machine learning techniques to automatically detect interesting patterns in EEG signals, in order to diagnose a specific disease. This can be treated as a pattern recognition problem and can take advantage of the continuous improvements of deep learning techniques. Existing algorithms can be categorized in two main groups: (i) statistical machine learning techniques, (ii) deep learning-based techniques.

In the first category various machine learning methods, such as *k*-nearest neighbors (*k*-NN) and support-vector machines (SVM), have been adopted [11,13,20–22]. The second category, i.e., deep learning-based techniques, includes deep neural networks (DNNs) and in particular recurrent neural networks (RNNs) [23–30]. The latter are especially suited for time-based data series, such as EEG.

Regarding the deep learning-based techniques, in [13] authors propose two EEG features, namely, epoch-based entropy (a measure of signal complexity) and bump modeling (a measure of synchrony) and demonstrate that these features are sufficient for efficient discrimination between subjective cognitive impairment (SCI) patients, MCI patients, possible AD patients, and patients with other pathologies, obtaining a classification accuracy of 81.8% to 88.8%. In [20], a novel analytical framework combining fuzzy learning and complex network approaches has been proposed for the identification of AD with multichannel scalp-recorded EEG signals, obtaining a highest accuracy of 97.12%. In [23,24] authors propose a strategy to use RNN that can handle missing data, which is common in healthcare data. Missing data are considered as arising due to various reasons, such as the nature of data collection procedures, subjects' dropping out of studies, or mistakes in data collection, so that entire examination results are missing, but these papers do not specifically handle the case of missing data segments within a signal trace as can happen if the measurement device has a wireless link. In [23], a multiclass area under the operating curve (mAUC) of 0.86 and a balanced class accuracy (BCA) of 0.79 was obtained, while in [24] a prediction performance of clinical diagnosis drop from a BCA of 0.935 in year 1 to a BCA of 0.810 in year 6 was achieved.

Special attention was also devoted to the extraction of features from the EEG signal, as shown in [31,32], where the authors propose a framework for the accurate identification of different motor-imagery (MI) tasks in brain-computer interface (BCI), exploring different feature extraction methods and obtaining good results with the application of the multiscale principal component analysis (MSPCA) method. The same aim, that is, increasing the classification outcome of different MI signals by selection of efficient features, was pursued in [33], applying the EEG preprocessing to several neural networks while a custom CNN was implemented in [34].

In the context of feature extraction, robust principal component analysis (RPCA) [35] is a method widely applied when the data observations are subject to sensible degradation. Thus, this method can be useful for the treatment of the EEG signal that can be affected by artifacts that can arise during the collection with, e.g., a wireless system that can lose packets.

The aim of this paper is primarily to validate the effectiveness of RPCA applied to corrupted EEG signals, in order to clear the signals and then extract features through PCA. Once the features are extracted, a custom RNN is implemented in order to classify patients as healthy or suffering from AD, training the network with a dataset obtained from EEG recordings of 35 hospitalized subjects belonging to both categories. In doing that we operate at three different levels of data elaboration, showing that performance of the network is progressively increased:

- Untransformed EEG data (time series);
- Features extracted from EEG data through principal component analysis (PCA);
- Corrupted signals, after filtering them with different algorithms, namely, robust PCA (RPCA) and multiscale PCA (MSPCA).

Specifically, the latter test showed that applying RPCA in a suitable way, compatible with the data representation, leads to better results in the case of corrupted input data with respect to other filtering algorithms, or no filtering at all.

The rest of the paper is organized as follows. In Section 2 we summarize the basic principles of RNNs. Section 3 describes the dataset adopted in the experiments. Section 4 describes the data processing applied to improve the RNN performance, including PCA and RPCA transformations. Section 5 reports the details of the proposed RNN architecture and its main features, together with a description of the hardware and software that was used. Experimental results are presented in Section 6 and discussed in Section 7. Finally, some conclusions are drawn in Section 8.

2. Brief of RNNs

In this work, we make use of RNNs with a structure similar to that employed in our previous work [36]; in this section, we thus only report a summary of their architecture and operational principles for easier reference.

While standard neural networks are characterized by a complete interconnection between adjacent layers, recurrent neural networks can map target vectors from the full history, as represented by the previous map. The structure of an RNN network is shown in Figure 1.

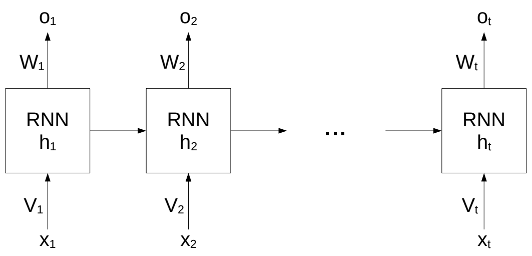

Figure 1. Structure of an RNN. Reproduced under CC-BY from [36].

In this architecture, each node maintains a current hidden state h_t and produces an output o_t by using the current input x_t and the previous hidden state h_{t-1} as follows:

$$h_t = f(W_h h_{t-1} + V_h x_t + b_h) \tag{1}$$

$$o_t = f(W_o h_t + b_o) \tag{2}$$

where W and V are the weights of the hidden layers in recurrent connections, b is the bias for hidden and output states and f is an activation function.

An RNN is very effective in modeling the dynamics of a continuous data sequence, but it may suffer from the problem of gradient disappearance and explosion [37] if modeling long sequences. In order to solve this problem, Hochreiter et al. [38] proposed a variant of RNN, based on the long short-term memory (LSTM) cell unit, which combines learning with model training, with no additional domain knowledge. The structure of the LSTM unit is shown in Figure 2.

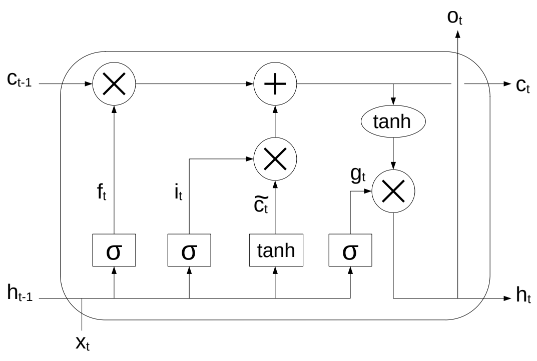

Figure 2. LSTM cell unit. Reproduced under CC-BY from [36].

The long-term state, short-term state and the output of each layer at each time step are described by the following equations:

$$f_t = \sigma(W_{x_f}^T\, x_t + W_{h_f}^T\, h_{t-1} + b_f) \tag{3}$$

$$i_t = \sigma(W_{x_i}^T\, x_t + W_{h_i}^T\, h_{t-1} + b_i) \tag{4}$$

$$\widetilde{c_t} = \tanh(W_{x_c}^T\, x_t + W_{h_c}^T\, h_{t-1} + b_c) \tag{5}$$

$$g_t = \sigma(W_{x_g}^T\, x_t + W_{h_g}^T\, h_{t-1} + b_g) \tag{6}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \widetilde{c_t} \tag{7}$$

$$o_t = h_t = g_t \otimes \tanh(c_t) \tag{8}$$

where W_{x_f}, W_{x_i}, W_{x_c}, W_{x_g} are the weight matrices associated to the matching connected input vector, W_{h_f}, W_{h_i}, W_{h_c}, W_{h_g} are the weight matrices for the short-term state from the previous time step, and b_f, b_i, b_c, b_g are bias terms. The symbol $\otimes$ means point-wise multiplication.

3. Dataset

We used a dataset obtained from EEG recordings of 35 hospitalized subjects, 20 of which suffered from Alzheimer's disease, and are denoted in the following as "AD", while 15 were healthy, indicated with "N" (Normal). The data were obtained from the subjects as part of ordinary medical diagnostic procedures performed in a hospital environment; all the data were collected according to the Declaration of Helsinki, had been properly anonymised, and informed consent was obtained at the time of original data collection.

EEG data are obtained with electrodes placed in several points along the subject's scalp, each recording a signal from a specific brain region. The specific equipment used to acquire the dataset is a Galileo BE Plus PRO Portable, Light version, providing 37 total inputs, of which 22 unipolar and 8 bipolar AC/DC inputs. The electrodes were applied in the standard 10–20 system (referring to distances between adjacent electrodes, expressed in percentage of the total available space on the skull).

Table 1 shows a summary of the consistency of the dataset, expressed in terms of the total duration of the recordings for the two classes of subjects. To perform our experiments, in the following the dataset will be split in 30 subjects for training and 5 for testing (see Section 6 for details).

Table 1. Dataset details.

Class	Subjects	Duration (s)
N	15	17,932
AD	20	28,586
Total	35	46,518

The demographic distribution of the involved subjects was chosen so to be as similar as possible in the two groups. The AD patients' mean age is 74.16 years (SD 4.64), their mean education is 7.55 years (SD 3.50) and the mean illness duration is 2.4 years (SD 0.64; range 1–3 years). The healthy subjects' mean age is 73.61 years (SD 5.81) and their mean education is 8.07 years (SD 3.92). The demographic variable do not differ in the two groups: age ($t = 0.29$, $p = 0.77$), education ($t = 0.38$, $p = 0.70$), sex ($\chi^2 = 0.40$, $d.f. = 2$, $p = 0.48$).

The dataset is composed of files in EDF format (European data format), a standard file format typically used for exchange and storage of medical data. Each recording includes a slightly different set of signals, all sampled at 128 Hz. All the signals for a specific subject have been acquired synchronously, so that all traces have the same duration, but of course the duration is different for different subjects. Data traces are organized in rows, with the number of different signals spanning the first dimension (rows, indeed) and time samples spanning the second dimension (columns).

4. Data Processing

A schematic diagram of the classification algorithm is shown in Figure 3. As previously mentioned, the first test has been performed with the original data, while in the second test the data have been corrupted and an additional filtering step has been applied using RPCA.

The individual steps of the algorithm are explained in detail in the following sections.

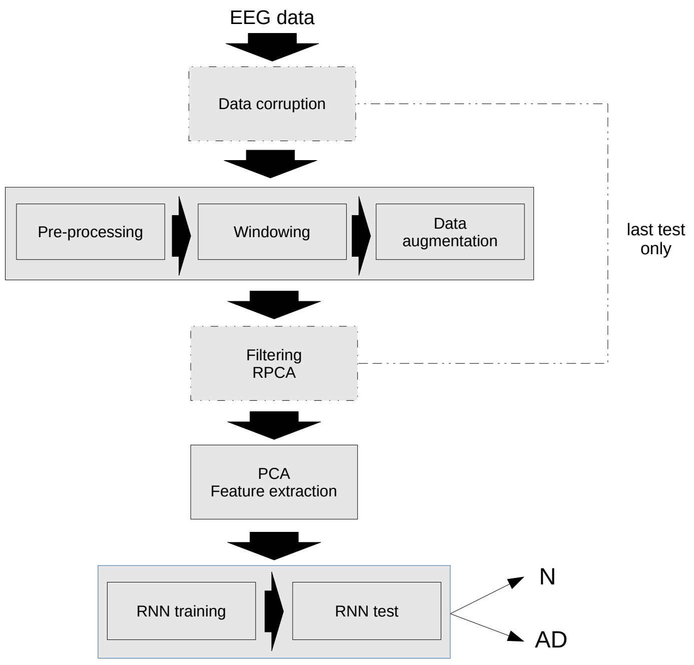

Figure 3. Flow chart of the proposed algorithm for Alzheimer's disease Recognition.

4.1. Data Pre-Processing

Before using the dataset as input for the following steps, some preliminary pre-processing was required. First of all, the set of signals included in every subject is slightly different, varying from 21 to 23 signals from the following list: Fp1, Fp2, Fpz, EKG, F7, F3, Fz, F4, F8, C3, Cz, C4, T3, T4, T5, T6, P3, Pz, P4, O1, O2, I, II, MK. Moreover, they are in different order for every subject.

For that reason, in order to have a coherent set of input data, we isolated the subset of common signals for all the subjects, resulting in a set of 16 signals, namely: Fp1, Fp2, F7, F3, F4, F8, T3, C3, C4, T4, T5, P3, P4, T6, O1, O2, and extracted and reordered them. In the process, data have been saved in "npy" format (Python NumPy numerical format) to be more conveniently processed in the following steps. Moreover, the matrices have been transposed, so as to have the time variable on the first dimension, as required by the RNN toolkit used.

It follows from the previous steps that every subject's data result in a matrix of dimension $n \times 16$, where n is the number of time points. It can be seen that a maximum of 7 tracks have been discarded from every subject. The resulting matrices have no associated information about the meaning of the specific tracks anymore, but they are used as 16-dimensional temporal series associated to every subject.

A further cleaning of the data has been performed for the presence, in some of the recordings, of artificial signals (square waves) at the beginning and at the end of actual EEG data. Those artificial signals are used to test or synchronize the equipment before starting acquiring real data from the electrodes. Leading and trailing intervals have been removed when needed to discard such spurious data. The duration of those intervals is indeed negligible with respect to the total duration.

Finally, since different signals exhibit significative differences in magnitude, statistical standardization has been applied to all the signals (columns), that is, data were scaled so that the resulting mean and standard deviation were 0 and 1, according to the formula

$$y[n] = \frac{x[n] - \mu}{\sigma} \tag{9}$$

with μ and σ being the original mean and standard deviation, respectively. This was experimentally proven to largely improve the final classification accuracy, especially when using PCA, which selects principal components based on the variance of the input data.

4.2. Data Windowing

In order to be processed by the RNN, input data are split along the time axis in windows of fixed size, possibly overlapping by a given amount. The window length and overlapping are important hyper-parameters in neural networks, as well as in many other machine learning algorithms [39,40]. Being w the number of samples in a window and o the number of overlapping samples between adjacent windows, the n^{th} data window corresponds to samples in the range

$$[n(w - o), n(w - o) + w - 1] \tag{10}$$

with $n >= 0$.

The resulting input matrix for the RNN has therefore dimension $N \times w \times 16$, where N is the number of resulting data windows and w is the window length.

To find the best combination for our particular network, we did a series of tests with various values of the two parameters (Section 6).

4.3. Data Augmentation

Since the number of inputs belonging to the two different classes are not equally represented, the network might end up being biased towards a specific class. A simple technique to address this problem is oversampling [41], a form of data augmentation where the data from classes with less occurrences are duplicated as needed, so that the data used for training are more uniformly distributed among the different classes.

Jittering with gaussian noise has been applied to the duplicated data, so not to have identical data in the training set.

Oversampling has been applied before each training phase to the subset of subjects used for training, leaving the test subjects unaltered. Table 2 shows an example of data distribution before and after data augmentation, for windows of 512 samples with 25% overlap, considering the subset of 30 training subjects.

Table 2. Number of data windows before and after oversampling (example for windows of 512 samples with 25% overlap).

Class	Original	Oversampled
N	5424	8099
AD	8099	8099
Total	13,523	16,198

4.4. Principal Component Analysis

A first set of tests has been performed using the original uncorrupted data, both in the time domain (no feature extraction, letting the RNN itself adapt to the raw input data) and applying PCA to the input data matrix. PCA is commonly used in a vast range of applications in order to reduce data dimensionality while simultaneously improving the signal-to-noise ratio of the resulting data. Consequently, dimensionality reduction not only simplifies the computation, but also generally improves the final accuracy, isolating the most relevant features of the signal.

PCA-based data reduction of a matrix can be performed by first factorizing the original matrix through singular value decomposition (SVD). Given a generic matrix $\mathbf{X}$ of size $m \times n$ and assuming $m > n$, SVD can factorize it as:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{11}$$

where $\mathbf{S}$ ($n \times n$) is a diagonal matrix containing the singular values, and $\mathbf{U}$, $\mathbf{V}$ have dimension $m \times n$, $n \times n$, respectively, and contain the singular vectors.

Dimensionality reduction can then be achieved by truncating $\mathbf{S}$ to the first p singular values ($p < n$), provided they are sorted in descending order, and consequently truncating $\mathbf{V}$ to the first p columns (let the new matrix be called $\mathbf{V}_p$). This decomposition ensures the preservation of the largest part of the variance of the original matrix for a given rank p.

A new matrix can then be substituted to $\mathbf{X}$:

$$\mathbf{X}^* = \mathbf{X}\mathbf{V}_p \qquad (m \times p) \tag{12}$$

which contains the principal components of $\mathbf{X}$ and has lower rank.

The choice of the p parameter must take into account the relative magnitude of singular values and can be chosen according to different criteria (see below for a possible choice).

A problem arising in the case of neural networks is that the input data matrix, once the original data are split into windows, is actually a 3-dimensional tensor, with size (in this particular case) $N \times w \times 16$, where N is the number of data windows and w is the window length, 16 being the number of selected EEG signals. As a concrete example, for the case with data windows of size 512 and 25% overlapping, using data from the 30 training subjects, the input tensor has size $16,198 \times 512 \times 16$.

Calling $\mathbf{A}$ the original input tensor, a strategy must therefore be devised to reshape $\mathbf{A}$ into a 2-dimensional matrix, in order to apply SVD and PCA.

A first strategy might consist in converting the original data to size $N \times (16w)$, that is flattening every data window to a single one-dimensional vector:

$$\mathbf{A} \in \mathbb{R}^{N \times w \times 16} \rightarrow \mathbf{A}' \in \mathbb{R}^{N \times (16w)} \tag{13}$$

However, experiments showed this to not be the best strategy, since the number of input features (N) is comparable to the dimension of the features themselves, leading to a poor statistical representation. In the previous example, indeed, the size of $\mathbf{A}'$ would be 16,198 $\times$ 8192.

A strategy that proved to perform better is instead the following: first we define a new input matrix $\mathbf{A}'$ having size $N \times 16 \times w$ (obtained by transposing data windows); then, it is converted to size $(16N) \times w$ (flattening the first two dimensions):

$$\mathbf{A} \in \mathbb{R}^{N \times w \times 16} \rightarrow \mathbf{A}' \in \mathbb{R}^{N \times 16 \times w} \rightarrow \mathbf{A}'' \in \mathbb{R}^{(16N) \times w} \tag{14}$$

This results in a matrix $\mathbf{A}''$ with the first dimension much greater than the second one, as desired, and as assumed in (11). In the previous example, $\mathbf{A}''$ has size 259,168 $\times$ 512. At this point we can apply SVD and PCA reduction as previously defined to $\mathbf{A}''$ and compute $\mathbf{V}_p$ of size $w \times p$. We can then compute the principal components of $\mathbf{A}''$ by multiplying it by $\mathbf{V_p}$, defining:

$$\mathbf{A}''' = \mathbf{A}''\mathbf{V}_p \qquad ((16N) \times p) \tag{15}$$

This new matrix can then be reverted to a 3-dimensional tensor ($N \times 16 \times p$) and finally, in order to have the 16 signal tracks correctly appearing on the last dimension, we can transpose the second and third dimensions again, to obtain the final tensor suitable to be used as an input the RNN application:

$$\mathbf{A}''' \in \mathbb{R}^{(16N) \times p} \rightarrow \mathbf{A}^* \in \mathbb{R}^{N \times p \times 16} \tag{16}$$

The previous steps of reducing the input data to a 2-dimensional matrix for application of the PCA and converting it back to a 3-dimensional tensor are shown schematically in Figure 4.

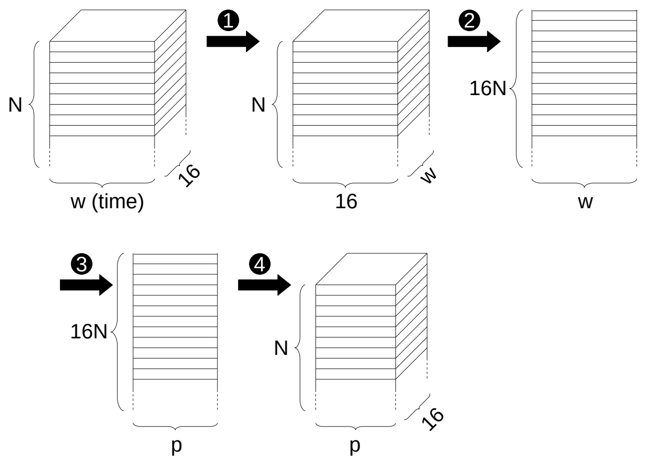

Figure 4. Steps to apply PCA reduction to 3-D input tensor: ❶ transposing second and third dimension, ❷ flattening first two dimensions, ❸ dimension reduction through PCA, ❹ converting back to original form.

As mentioned, one must choose the parameter p for the truncation of the number of principal components, as an hyper-parameter of the resulting system. An empirical value of 50 has been chosen for all the input matrices, after inspecting some of the results of the PCA decomposition for several input cases. Figure 5 shows, for example, the magnitude of the first 150 singular values for the input matrix used in the previous examples. In the previous example, therefore, the final input tensor $\mathbf{A}^*$ has size 16,198 $\times$ 50 $\times$ 16.

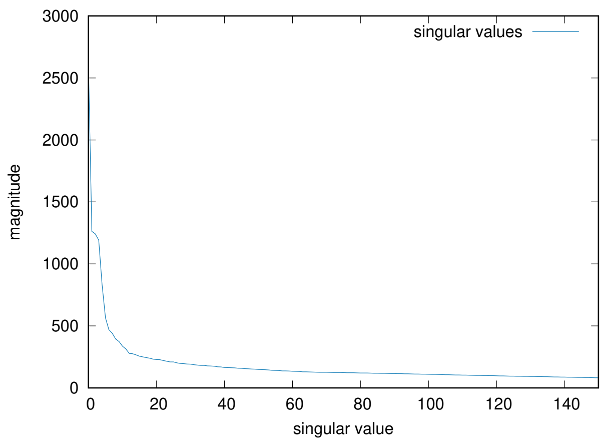

Figure 5. Singular value magnitude of an example input matrix, limited to the first 150 values.

4.5. Robust PCA

Robust PCA is an enhancement to the traditional PCA decomposition, that can be applied when the matrix contains corrupted observations, even for sensible levels of corruption [35].

Given a (possibly large) data matrix $\mathbf{M}$ and assuming that it can be decomposed as

$$\mathbf{M} = \mathbf{L_0} + \mathbf{S_0} \tag{17}$$

where $\mathbf{L_0}$ is a low-rank matrix and $\mathbf{S_0}$ is sparse, RPCA offers a method to recover $\mathbf{L_0}$ and $\mathbf{S_0}$ accurately and efficiently, even without prior information on their dimension and magnitude.

RPCA is widely used when the data observations are subject to sensible degradation; in many cases the low-rank $\mathbf{L_0}$ component models the original data, distributed in a low-dimensional subspace, while the sparse $\mathbf{S_0}$ matrix represents the corruption components (outliers). Typical applications are image recognition and classification, signal filtering, user ranking and many others [35,42–45].

The RPCA decomposition is performed with an iterative algorithm, estimating $\mathbf{L_0}$ and $\mathbf{S_0}$ within a desired tolerance. Details are shown in Algorithm 1. In the algorithm, $\mathbf{M}$ is the input matrix, $\mathbf{L}_k$ and $\mathbf{S}_k$ are the results as computed at the kth iteration;

$\mathcal{S}_\tau : \mathbb{R} \to \mathbb{R}$ is a shrinkage operator such that

$$\mathcal{S}_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0) \tag{18}$$

extended to matrices by applying it to each element;

$\mathcal{D}_\tau$ is a threshold operator such that

$$\mathcal{D}_\tau(\mathbf{X}) = \mathbf{U}\mathcal{S}_\tau(\mathbf{\Sigma})\mathbf{V}^T \tag{19}$$

where $\mathbf{X} = \mathbf{U\Sigma V}^T$ (singular value decomposition);

μ and λ, finally, are parameters that must be chosen according to the nature of the data.

In this work, we apply the RPCA to compensate for degradation in the original EEG signals, and show that it can improve the RNN performance in those cases. Operatively, we apply RPCA to the matrix obtained in (14), that is the matrix obtained by reducing the dimension of the original 3-dimensional tensor input; subsequently, dimensional reduction by PCA is applied to the $\mathbf{L_0}$ matrix as computed by the RPCA.

Algorithm 1 RPCA algorithm

```
Input: $\mathbf{M}$
1: procedure COMPUTE RPCA
2:     $\mathbf{S}_0 = \mathbf{Y}_0 = \mathbf{0}$
3:     repeat
4:         Compute $\mathbf{L}_k = \mathcal{D}_{1/\mu}(\mathbf{M} - \mathbf{S}_{k-1} + \mu^{-1}\mathbf{Y}_{k-1})$
5:         Compute $\mathbf{S}_k = \mathcal{S}_{\lambda/\mu}(\mathbf{M} - \mathbf{L}_k + \mu^{-1}\mathbf{Y}_{k-1})$
6:         Compute $\mathbf{Y}_k = \mathbf{Y}_{k-1} + \mu(\mathbf{M} - \mathbf{L}_k - \mathbf{S}_k)$
7:     until $\|\mathbf{M} - \mathbf{L}_k - \mathbf{S}_k\| <$ tolerance
8: end procedure
Output: $\mathbf{L}_k$, $\mathbf{S}_k$                    ▷ $k$ = last computed step
```

5. RNN Architecture

The RNN used in this paper is depicted in Figure 6, for the case of PCA applied and 50 principal components retained. It is based on architectures commonly used with time-based sensor data [36,40,41,46] and consisting of a mix of LSTM cells and fully connected layers.

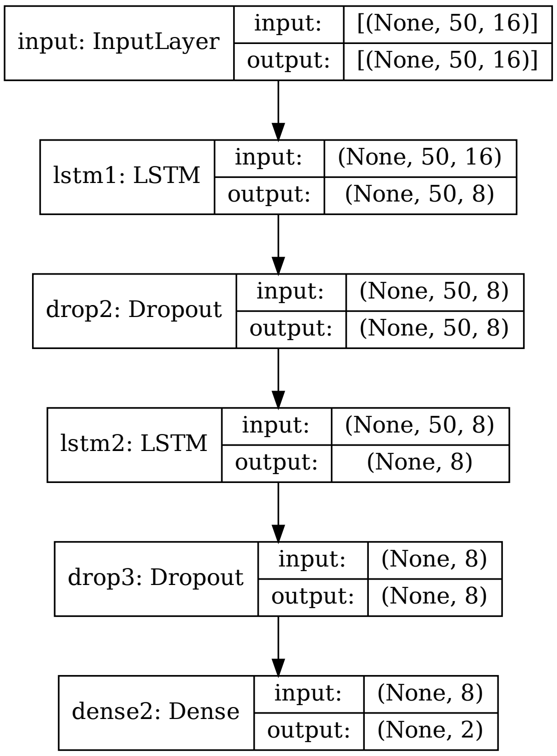

Figure 6. RNN architecture.

Input data consist in a series of $m \times n$ matrices, depending on whether PCA reduction is used or not. When using original data with no PCA, input data are time series of size $w \times 16$, with w being the size of data windows as described in Section 4.2. When using PCA or RPCA, input data are the principal components extracted from the original data, of size $p \times 16$ as in (16).

The core of the recurrent neural network consists of two cascaded LSTM layers, whose internal architecture was briefly explained in Section 2. Each LSTM layer is followed by a dropout layer that randomly discards some of the input data. All the intermediate layers have size 8; this hyper-parameter has been chosen experimentally, starting with a larger value and decreasing it until the network accuracy varied significantly.

Finally, there is a fully-connected layer (dense) of size 2. In this layer the generic n^{th} neuron produces an output y_n depending on the $x_1, \dots x_m$ inputs to the layer and the w_{nj} neuron weights associated to every input, specifically:

$$y_n = \phi\left(\sum_{j=1}^{m} w_{nj}x_j + b_n\right) \tag{20}$$

where ϕ is an activation function and b_n is a bias value.

The fully-connected layer performs the classification in one of the 2 classes, according to the sparse categorical crossentropy loss function assigned to the network. The loss function, or cost function in general terms of an optimization problem, represents the error to be minimized by the training process. The specific formula for the error according to the categorical crossentropy function is

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{21}$$

where w is the set of model parameters, e.g. the weights of the RNN, N is the number of input test features, y_i and $\hat{y}_i$ are the actual and predicted classes, respectively, expressed numerically.

One of the problems to be aware of when designing a neural network is overfitting, that is, the generated model might be performing well on the training dataset, but poorly on new, unseen data. We employed a combination of several techniques to reduce overfitting: use of dropout layers, as just mentioned, to discard random portions of data; data augmentation with random noise applied (Section 4.3); randomly mixing input data from training and validation subjects (Section 6) so to have representative samples from all the subjects for both training and validation.

As an example, Table 3 shows the details of the individual layers in the case where PCA has been performed and 50 principal components retained ($p = 50 \rightarrow m, n = 50, 16$), matching the configuration of Figure 6. The RNN in this configuration has 1394 trainable parameters.

Table 3. Details of RNN layers.

Layer	Input Size	Output Size	Parameters
LSTM 1	(-, 50, 16)	(-, 50, 8)	832
Dropout 1	(-, 50, 8)	(-, 50, 8)	0
LSTM 2	(-, 50, 8)	(-, 8)	544
Dropout 2	(-, 8)	(-, 8)	0
Dense	(-, 8)	(-, 2)	18

Hardware and Software

The RNN was developed with TensorFlow 2.6.1 and Keras 2.6.0; the rest of the computations were performed using Python NumPy and SciPy suites, unless otherwise specified in Section 6. For the purpose of design and hyper-parameter optimization, the network and the related algorithms were initially developed on the Google Colaboratory platform; the final computations were then performed on a computer with an Intel Core i7-6800K CPU, 32 GiB of RAM and an NVIDIA GeForce GTX 1080 GPU.

All the software developed for this article is publicly available at https://github.com/MAlessandrini-Univpm/rnn-eeg-ad, published 14 April 2022.

6. Experimental Results

As mentioned, the first set of tests has been performed using both original temporal data and principal components extracted through PCA. Every test has been repeated for different combinations of window length and overlapping, to find the best results.

It is common practice, when training a neural network, to split the dataset in subjects used for training and subjects for independent test, and to further split the training data in two sets: data actually used to fit the network weights, and validation data to monitor the performance of the network during the various training epochs.

Since the number of different subjects in the dataset is not large, and different subjects inevitably have substantial differences in their data, the statistical distribution of the data might not be uniform enough, and so choosing a single partition for training and validation might not lead to representative results.

So we decided to split the training input data in two sets for training and validation, respectively, with a ratio of 75%/25%, after shuffling the input data windows, so to better statistically integrate data from different training subjects. The subjects used for test are isolated from the beginning and not participating in the training/validation phase. The exact procedure is showed in Algorithm 2.

Algorithm 2 Testing algorithm

```
 1: procedure COMPUTE TESTING ACCURACIES
 2:     $S_1$ = set of training subjects
 3:     $S_2$ = set of testing subjects
 4:     for all cases No-PCA / PCA do
 5:         for all combinations of window and overlap do
 6:             Train RNN with subjects $S_1$, splitting in training + validation
 7:             Test trained network on subjects $S_2$
 8:         end for
 9:     end for
10: end procedure
```

The RNNs were trained with the following parameters:

- 5 different window sizes ranging from 1 to 5 s;
- window overlapping of 0, 25% and 50%;
- data augmentation (Section 4.3);
- 50 principal components (when PCA is used);
- 30 subjects used from training and validation, 5 for testing;
- 20 training epochs.

Tables 4 and 5 show the test results when using original temporal data and when applying PCA, respectively. It can be seen that using PCA results in sensibly higher accuracy (97.9% versus 79.3% for the best cases), as it is expected for the benefits of PCA on this class of data.

Figure 7 shows the progress of accuracy and loss (computed on the validation subset) with respect to the training epochs, for the case with a window of size 512 and 25% overlap.

Table 4. Experimental results for original temporal data, best result is displayed in bold.

window samples	128			256			384			512			640		
window duration (s)	1			2			3			4			5		
window overlap (%)	0	25	50	0	25	50	0	25	50	0	25	50	0	25	50
input features	36,493	48,642	72,961	18,240	24,310	36,468	12,157	16,198	24,294	9115	12,148	18,214	7290	9711	14,562
training time (s)	399.3	527.7	787.5	288.5	402.8	603.4	270.7	359.7	539.9	250.3	334.3	500.6	236.7	315.7	472.3
test time (s)	1.3	1.9	2.8	1.6	1.4	2.1	1.0	1.2	1.9	0.9	1.2	1.8	0.9	1.2	1.7
test accuracy (%)	56.7	77.5	**79.3**	51.9	45.2	42.3	50.7	68.6	57.3	56.9	51.3	48.1	59.5	67.5	52.4

Table 5. Experimental results for PCA data using 50 principal components, best result is displayed in bold.

window samples		128			256			384			512			640	
window duration (s)		1			2			3			4			5	
window overlap (%)	0	25	50	0	25	50	0	25	50	0	25	50	0	25	50
input features	36,493	48,642	72,961	18,240	24,310	36,468	12,157	16,198	24,294	9115	12,148	18,214	7290	9711	14,562
training time (s)	267.5	354.4	531.1	124.8	179.3	279.5	91.7	120.3	179.4	69.3	90.9	135.1	55.6	73.5	109.2
test time (s)	0.5	1.2	1.7	1.0	0.3	0.7	0.2	0.2	0.4	0.2	0.2	0.6	0.1	0.2	0.2
test accuracy (%)	62.4	73.6	87.0	92.8	96.0	**97.9**	96.9	90.5	90.1	94.6	88.4	95.6	93.4	96.3	93.2

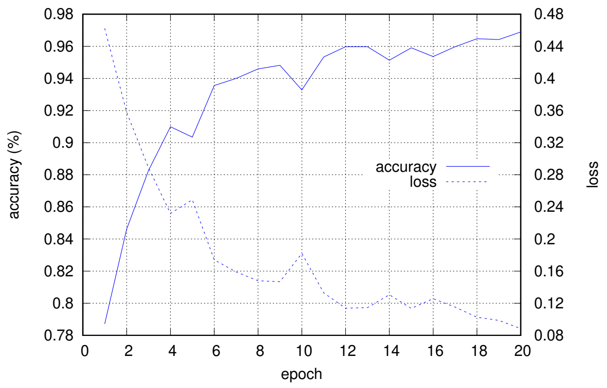

Figure 7. Accuracy and loss progress for validation data with respect to training epochs.

It is interesting to analyze the difference in accuracy for the two classes. Figure 8 shows the confusion matrix for the mentioned example. It can be seen that the accuracy for the "Normal" class is sensibly worse. This is likely due to the smaller number of subjects in that category, and also for the choice of subjects in the dataset: subjects in the "Normal" category come from hospitalized patients, too, who either were treated for other diseases or belong to risk groups.

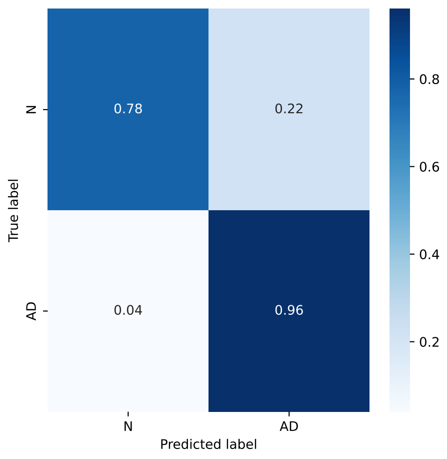

Figure 8. Confusion matrix for a sample case.

Test of Robustness with RPCA

To test the robustness of the RNN with respect to corruption of original signals, we simulated a situation where some of the data are lost, for example due to bad connections of the electrodes, limited capacity of the elaboration system with respect to the data bandwidth, or packets lost in a hypothetical wireless connection between the electrodes and the acquisition system.

To this end, we artificially corrupted the data at several increasing levels by creating "holes" in the original signals, before any other elaborations; such holes are sequences of zeros, applied simultaneously to all the 16 tracks. Holes are applied at random times with a probability p chosen in the set $1/1000, 1/500, 1/200, 1/100$. Single holes have a random length, too, with a normal distribution of mean length 20, resulting in average erasure rates of 2%, 4%, 10%, and 20%, respectively. Figure 9 shows an example of a hole created in one of the signals.

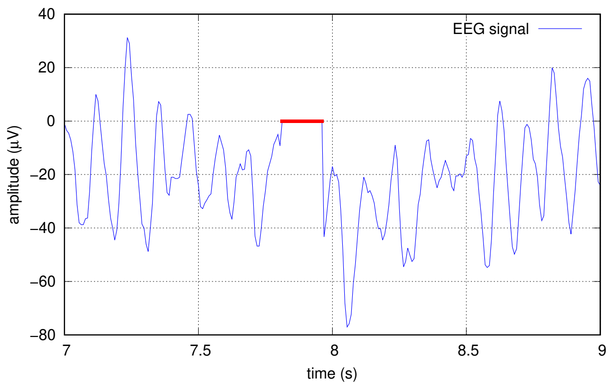

Figure 9. Example of data corrupted with a hole (red).

The corrupted signals are filtered with RPCA, as explained in Section 4.5, before the actual PCA and dimension reduction. As a matter of comparison, another test has been done using MSPCA as a filtering technique.

MSPCA [47] combines the ability of PCA to extract the crosscorrelation or relationship between the variables, with that of orthonormal wavelets to separate deterministic features from stochastic processes, and approximately decorrelate the autocorrelation among the measurements. In MSPCA, several PCA decompositions are performed at different resolution levels, keeping a different number of components at every level, according to a suitable algorithm. Every decomposition level can highlight different signal components, such as noise.

MSPCA has shown to be an effective method for noise removal in automated detection systems, possibly combined with other dimension reduction techniques, such as empirical wavelet transform or multivariate variational mode decomposition [31–34].

In this article, we used the Matlab MSPCA implementation [48], running on Matlab R2022a. As in the RPCA, we apply MSPCA to the matrix obtained in (14), then dimensional reduction by PCA is applied to the matrix obtained by the MSPCA.

For RPCA, a fundamental aspect is choosing the right parameters, especially the μ parameter (Section 4.5). For every value of p (probability of data corruption) several tests have been performed to find the optimal value of μ. In real applications, this parameter must be evaluated according to the statistical properties of the analyzed signal and noise.

For MSPCA, the best results were obtained experimentally as follows. Decomposition is performed at 5 levels, using the Daubechies least-asymmetric wavelet with 4 vanishing moments (sym4). A first run is performed to compute the retained principal components according to the Kaiser's rule. Then a second run is performed after removing the principal components of level 1, in order to filter out the signal noise. See [48] for implementation details.

Table 6 and Figure 10 show the results of these tests, where all the other parameters and procedures are the same as in the previous experiments, comparing the three cases: no filtering applied before PCA, filtering through MSPCA, filtering through RPCA.

It can be seen that applying RPCA always results in better final accuracies, holding good performances for the RNN even with severely corrupted input data.

Table 6. Test accuracy for corrupted signals with different filtering techniques (p is the corruption probability, data loss represent the average number of missing samples).

p (‰)	Data Loss (%)	No Filter (%)	MSPCA (%)	RPCA (%)
1	2	89.0	90.8	94.5
2	4	91.1	92.6	94.6
5	10	80.0	81.6	85.7
10	20	76.9	72.1	81.5

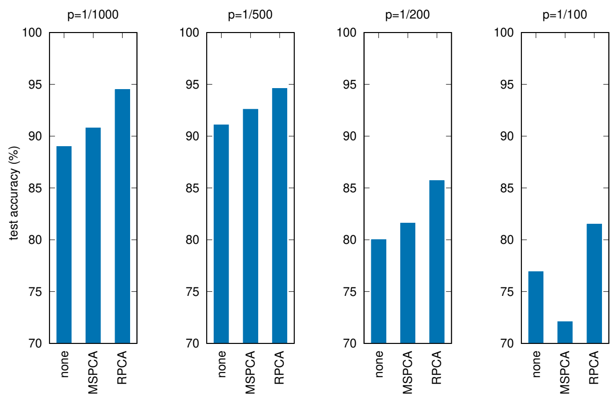

Figure 10. Test accuracy for corrupted signals with different filtering techniques (p is the corruption probability).

7. Discussion

The performed experiments have shown that the designed RNN can recognize subjects with AD, from the dataset used for the tests, with a good accuracy (97.9%), under optimal choice of hyper-parameters and after computing the principal components of input data with PCA (as opposed to untransformed time-based data). These results confirm the effectiveness of DNNs, and specifically RNNs, for time-based data series, in finding significant patterns in classes of data with no previous knowledge of the underlying system.

Moreover, the last experiment has shown that for corrupted data, specifically with bursts of missing information, applying RPCA as a filtering stage can maintain a good accuracy, even in the case of high corruption rate. This is due to the property of RPCA of removing outlier components from the main signals, filtering out noise and other artifacts with a different statistical distribution with respect to the relevant information.

It is interesting to compare the results with the other studies mentioned in the introduction (Section 1). Starting with methods based on RNNs, Refs. [23,24] address the problem of missing data, even if not in the form of whole data segments as we do, and reach a classification accuracy of 79% and 93.5%, respectively, the latter decreasing to 81% on a longer timescale. Refs. [27,28] combine RNNs with wavelet preprocessing, obtaining an accuracy under 90%. Even if the data were processed in different ways, such as explicitly adding noise in our case, it can be seen that our results are consistent with the cited ones, while also addressing the possible problem of signal degradation.

The same considerations hold when comparing to methods based on strictly statistical machine learning, such as k-NN and SVM, especially with regard to the added robustness to corrupted data. Ref. [11] analyzes data in the frequency domain, using the fast Fourier transform (FFT), in order to link variations in spectral components to AD; three alternative classification algorithms are then compared, namely, SVM, decision trees and rule-based classifiers, in the end obtaining a prediction accuracy up to 90%. Ref. [13] extracts two specific features from EEG data— epoch-based entropy (a measure of signal complexity) and bump modeling (a measure of synchrony)—and, together with orthogonal forward regression (OFR) and SVM algorithms, shows an accuracy of 91.6%. In our previous work [22] we also used RPCA at the feature extraction phase, before applying several machine learning algorithms, like *k*-NN, decision tree, SVM, naïve Bayes, and obtained a maximum accuracy of 93.18% with SVM. This proves that RPCA is useful as a pre-processing stage for EEG signals since it improves recognition rates in both traditional machine learning algorithm and, as the current paper shows, with deep learning methods.

Finally, the accuracy obtained by our algorithm is comparable to more expensive and complex techniques, such as magnetic resonance imaging (MRI), of which a comprehensive review of its application to machine learning for AD diagnosis can be found in [49].

8. Conclusions

In this paper, an RNN is built to classify medical subjects as healthy or suffering from Alzheimer's disease, using EEG data obtained by a sample of subjects from both categories. Several elaborations are applied to the original data to obtain a better accuracy, in particular principal component analysis, that results in an accuracy of more than 97% on the test data, using optimal hyper-parameters. Most importantly, it is shown that using corrupted input data and applying RPCA to them, a good accuracy can be maintained, and a consistent improvement of around 5% with respect to baseline PCA, can be achieved even when the corruption rate is sensibly high, with up to 20% erasures in the input samples.

As a possible future evolution of the proposed algorithms, their application could be extended to the classification of multiple neurodegenerative pathologies besides AD, for which data collection is already in progress, or to the identification of contributions of the different brain regions to the various patologies.

Author Contributions: Conceptualization, M.A., G.B., P.C., L.F., S.L. and C.T.; investigation, M.A., L.F. and C.T.; methodology, M.A., G.B., L.F., S.L. and C.T.; project administration, P.C. and C.T.; software, M.A., G.B. and L.F.; supervision, P.C. and C.T.; validation, M.A. and G.B.; visualization, M.A. and L.F.; writing—original draft, M.A., L.F. and C.T.; writing—review and editing, M.A., G.B., P.C., L.F., S.L. and C.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: The dataset was obtained from subjects as part of ordinary medical diagnosis in hospital environment; data were collected according to the Declaration of Helsinki, were properly anonymised and informed consent was obtained at the time of original data collection.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xie, J.; Brayne, C.; Matthews, F.E. Survival times in people with dementia: Analysis from population based cohort study with 14 year follow-up. *BMJ* **2008**, *336*, 258–262. [CrossRef] [PubMed]
2. Jeong, J. EEG dynamics in patients with Alzheimer's disease. *Clin. Neurophysiol.* **2004**, *115*, 1490–1505. [CrossRef] [PubMed]
3. Petersen, R.C. Early diagnosis of Alzheimer's disease: Is MCI too late? *Curr. Alzheimer Res.* **2009**, *6*, 324–330. [CrossRef] [PubMed]
4. Petersen, R.C. Mild cognitive impairment as a diagnostic entity. *J. Intern. Med.* **2004**, *256*, 183–194. [CrossRef]
5. Gauthier, S.; Reisberg, B.; Zaudig, M.; Petersen, R.C.; Ritchie, K.; Broich, K.; Belleville, S.; Brodaty, H.; Bennett, D.; Chertkow, H.; et al. Mild cognitive impairment. *Lancet* **2006**, *367*, 1262–1270. [CrossRef]
6. Biagetti, G.; Crippa, P.; Falaschetti, L.; Luzzi, S.; Santarelli, R.; Turchetti, C. Classification of Alzheimer's disease from structural magnetic resonance imaging using particle-Bernstein polynomials algorithm. *Smart Innov. Syst. Technol.* **2019**, *143*, 49–62.
7. Tsolaki, A.; Kazis, D.; Kompatsiaris, I.; Kosmidou, V.; Tsolaki, M. Electroencephalogram and Alzheimer's disease: Clinical and research approaches. *Int. J. Alzheimer's Dis.* **2014**, *2014*, 349249. [CrossRef]
8. Kulkarni, N.N.; Bairagi, V.K. Electroencephalogram based diagnosis of Alzheimer Disease. In Proceedings of the 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 9–10 January 2015; pp. 1–5.
9. Falk, T.H.; Fraga, F.J.; Trambaiolli, L.; Anghinah, R. EEG amplitude modulation analysis for semi-automated diagnosis of Alzheimer's disease. *EURASIP J. Adv. Signal Process.* **2012**, *2012*, 192. [CrossRef]
10. Dauwels, J.; Vialatte, F.; Cichocki, A. Diagnosis of Alzheimer's disease from EEG signals: Where are we standing? *Curr. Alzheimer Res.* **2010**, *7*, 487–505. [CrossRef]

11. Fiscon, G.; Weitschek, E.; Felici, G.; Bertolazzi, P.; De Salvo, S.; Bramanti, P.; De Cola, M.C. Alzheimer's disease patients classification through EEG signals processing. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Orlando, FL, USA, 9–12 December 2014; pp. 105–112.
12. Lehmann, C.; Koenig, T.; Jelic, V.; Prichep, L.; John, R.E.; Wahlund, L.O.; Dodge, Y.; Dierks, T. Application and comparison of classification algorithms for recognition of Alzheimer's disease in electrical brain activity (EEG). *J. Neurosci. Methods* **2007**, *161*, 342–350. [CrossRef]
13. Houmani, N.; Vialatte, F.; Gallego-Jutglà, E.; Dreyfus, G.; Nguyen-Michel, V.H.; Mariani, J.; Kinugawa, K. Diagnosis of Alzheimer's disease with Electroencephalography in a differential framework. *PLoS ONE* **2018**, *13*, e0193607. [CrossRef] [PubMed]
14. Coben, L.A.; Danziger, W.L.; Berg, L. Frequency analysis of the resting awake EEG in mild senile dementia of Alzheimer type. *Electroencephalogr. Clin. Neurophysiol.* **1983**, *55*, 372–380. [CrossRef]
15. Arenas, A.M.; Brenner, R.P.; Reynolds, C.F. Temporal slowing in the elderly revisited. *Am. J. EEG Technol.* **1986**, *26*, 105–114. [CrossRef]
16. Cibils, D. Dementia and qEEG (Alzheimer's disease). In *Supplements to Clinical Neurophysiology*; Elsevier: Amsterdam, The Netherlands, 2002; Volume 54, pp. 289–294.
17. Kowalski, J.W.; Gawel, M.; Pfeffer, A.; Barcikowska, M. The diagnostic value of EEG in Alzheimer disease: Correlation with the severity of mental impairment. *J. Clin. Neurophysiol.* **2001**, *18*, 570–575. [CrossRef]
18. Besthorn, C.; Förstl, H.; Geiger-Kabisch, C.; Sattel, H.; Gasser, T.; Schreiter-Gasser, U. EEG coherence in Alzheimer disease. *Electroencephalogr. Clin. Neurophysiol.* **1994**, *90*, 242–245. [CrossRef]
19. Locatelli, T.; Cursi, M.; Liberati, D.; Franceschi, M.; Comi, G. EEG coherence in Alzheimer's disease. *Electroencephalogr. Clin. Neurophysiol.* **1998**, *106*, 229–237. [CrossRef]
20. Yu, H.; Zhu, L.; Cai, L.; Wang, J.; Liu, J.; Wang, R.; Zhang, Z. Identification of Alzheimer's EEG With a WVG Network-Based Fuzzy Learning Approach. *Front. Neurosci.* **2020**, *14*, 641. [CrossRef]
21. Tanveer, M.; Richhariya, B.; Khan, R.; Rashid, A.; Khanna, P.; Prasad, M.; Lin, C. Machine learning techniques for the diagnosis of Alzheimer's disease: A review. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2020**, *16*, 1–35. [CrossRef]
22. Biagetti, G.; Crippa, P.; Falaschetti, L.; Luzzi, S.; Turchetti, C. Classification of Alzheimer's disease from EEG signal using robust-PCA feature extraction. *Procedia Comput. Sci.* **2021**, *192*, 3114–3122. [CrossRef]
23. Nguyen, M.; Sun, N.; Alexander, D.C.; Feng, J.; Yeo, B.T. Modeling Alzheimer's disease progression using deep recurrent neural networks. In Proceedings of the 2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI), Singapore, 12–14 June 2018; pp. 1–4.
24. Nguyen, M.; He, T.; An, L.; Alexander, D.C.; Feng, J.; Yeo, B.T.; Initiative, A.D.N. Predicting Alzheimer's disease progression using deep recurrent neural networks. *NeuroImage* **2020**, *222*, 117203. [CrossRef]
25. Gong, S.; Xing, K.; Cichocki, A.; Li, J. Deep Learning in EEG: Advance of the Last Ten-Year Critical Period. *arXiv* **2020**, arXiv:2011.11128.
26. Li, G.; Lee, C.H.; Jung, J.J.; Youn, Y.C.; Camacho, D. Deep learning for EEG data analytics: A survey. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5199. [CrossRef]
27. Petrosian, A.; Prokhorov, D.; Lajara-Nanson, W.; Schiffer, R. Recurrent neural network-based approach for early recognition of Alzheimer's disease in EEG. *Clin. Neurophysiol.* **2001**, *112*, 1378–1387. [CrossRef]
28. Petrosian, A.A.; Prokhorov, D.; Schiffer, R.B. Early recognition of Alzheimer's disease in EEG using recurrent neural network and wavelet transform. In Proceedings of the Wavelet Applications in Signal and Image Processing VIII, San Diego, CA, USA, 30 July–4 August 2000; Volume 4119, pp. 870–877.
29. Wang, T.; Qiu, R.G.; Yu, M. Predictive modeling of the progression of Alzheimer's disease with recurrent neural networks. *Sci. Rep.* **2018**, *8*, 1–12. [CrossRef] [PubMed]
30. Yang, S.; Chen, H.C.; Wu, C.H.; Wu, M.N.; Yang, C.H. Forecasting of the prevalence of dementia using the lstm neural network in Taiwan. *Mathematics* **2021**, *9*, 488. [CrossRef]
31. Sadiq, M.T.; Yu, X.; Yuan, Z.; Aziz, M.Z. Motor imagery BCI classification based on novel two-dimensional modelling in empirical wavelet transform. *Electron. Lett* **2020**, *56*, 1367–1369. [CrossRef]
32. Sadiq, M.T.; Yu, X.; Yuan, Z.; Aziz, M.Z.; ur Rehman, N.; Ding, W.; Xiao, G. Motor Imagery BCI Classification Based on Multivariate Variational Mode Decomposition. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**. [CrossRef]
33. Sadiq, M.T.; Yu, X.; Yuan, Z. Exploiting dimensionality reduction and neural network techniques for the development of expert brain–computer interfaces. *Expert Syst. Appl.* **2021**, *164*, 114031. [CrossRef]
34. Sadiq, M.T.; Aziz, M.Z.; Almogren, A.; Yousaf, A.; Siuly, S.; Rehman, A.U. Exploiting pretrained CNN models for the development of an EEG-based robust BCI framework. *Comput. Biol. Med.* **2022**, *143*, 105242. [CrossRef]
35. Candès, E.J.; Li, X.; Ma, Y.; Wright, J. Robust principal component analysis? *J. ACM (JACM)* **2011**, *58*, 1–37. [CrossRef]
36. Alessandrini, M.; Biagetti, G.; Crippa, P.; Falaschetti, L.; Turchetti, C. Recurrent Neural Network for Human Activity Recognition in Embedded Systems Using PPG and Accelerometer Data. *Electronics* **2021**, *10*, 1715. [CrossRef]
37. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **1998**, *6*, 107–116. [CrossRef]
38. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

39. Biagetti, G.; Crippa, P.; Falaschetti, L.; Focante, E.; Martínez Madrid, N.; Seepold, R. Machine Learning and Data Fusion Techniques Applied to Physical Activity Classification Using Photoplethysmographic and Accelerometric Signals. *Procedia Comput. Sci.* **2020**, *176*, 3103–3111. [CrossRef]
40. Musci, M.; De Martini, D.; Blago, N.; Facchinetti, T.; Piastra, M. Online Fall Detection using Recurrent Neural Networks on Smart Wearable Devices. *IEEE Trans. Emerg. Top. Comput.* **2020**. [CrossRef]
41. Eddins, S. Classify ECG Signals Using LSTM Networks. 2018. Available online: https://blogs.mathworks.com/deep-learning/2018/08/06/classify-ecg-signals-using-lstm-networks/ (accessed on 16 April 2021).
42. Hubert, M.; Engelen, S. Robust PCA and classification in biosciences. *Bioinformatics* **2004**, *20*, 1728–1736. [CrossRef] [PubMed]
43. Rahmani, M.; Li, P. Outlier detection and robust PCA using a convex measure of innovation. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
44. Su, J.; Tao, H.; Tao, M.; Wang, L.; Xie, J. Narrow-band interference suppression via RPCA-based signal separation in time–frequency domain. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2017**, *10*, 5016–5025. [CrossRef]
45. Xu, H.; Caramanis, C.; Mannor, S. Outlier-robust PCA: The high-dimensional case. *IEEE Trans. Inf. Theory* **2012**, *59*, 546–572. [CrossRef]
46. Chevalier, G. LSTMs for Human Activity Recognition. 2016. Available online: https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition (accessed on 16 April 2021).
47. Bakshi, B.R. Multiscale PCA with application to multivariate statistical process monitoring. *AIChE J.* **1998**, *44*, 1596–1610. [CrossRef]
48. MathWorks. wmspca—Multiscale Principal Component Analysis. 2022. Available online: https://it.mathworks.com/help/wavelet/ref/wmspca.html (accessed on 7 April 2022).
49. Lazli, L.; Boukadoum, M.; Mohamed, O.A. A Survey on Computer-Aided Diagnosis of Brain Disorders through MRI Based on Machine Learning and Data Mining Methodologies with an Emphasis on Alzheimer Disease Diagnosis and the Contribution of the Multimodal Fusion. *Appl. Sci.* **2020**, *10*, 1894. [CrossRef]