FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Universidade de Coimbra

Assignment 2B

Prediction and Detection of Epileptic Seizures

November 28, 2021

Author	*Student ID*
Francisco Fernandes	2018278239
João Marcelino	2018279700

Contents

1 Introduction

This report aims to explain the work done in Assignment 2b of Machine Learning course. Epilepsy is one of the most prevalent neurological diseases and can affect victims anytime, anywhere. Because of this, systems capable of detecting and predicting seizures in real time are highly valuable and being actively researched.

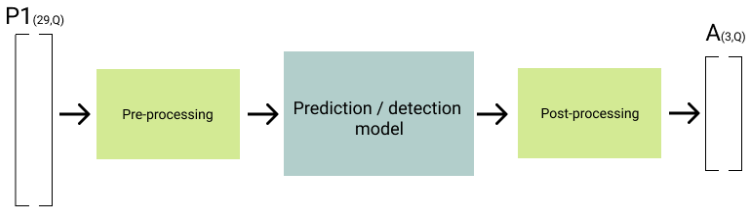


Figure 1: Overview of prediction/detection system for epileptic seizures.

During his PhD work, Doctor Mojtaba Bandarabadi monitored brain activity of 4 different patients and extracted time series with 29 band frequencies of EEG signals. The goal of this project is to develop different classification models capable of detecting pre-ictal, ictal and inter-ictal segments present in the time series provided (see figure 1). In this context, ictal segments correspond to seizure episodes, pre-ictal are the moments preceding those episodes and inter-ictal segments usually represent normal brain activity.

Different paradigms and architectures were considered for the classification model:

- Feed Forward NN
- Dynamic NN
- Convolutional NN
- Long Short Term Memory NN

2 Data set

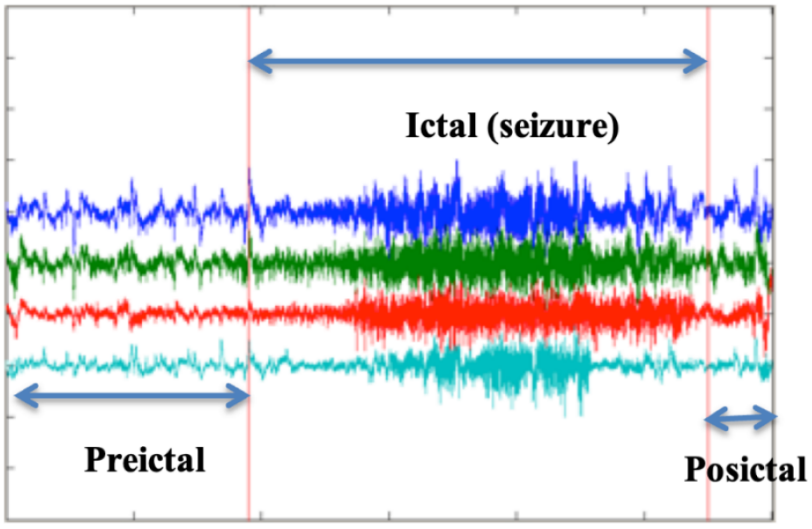


Figure 2: Time evolution of EEG signal. The frequency signature of the signal changes during an epileptic episode and during moments preceding it as well.

The dataset is divided into 4 time series of distinct patients with different characteristics such as gender, age, seizure type, location and severity. Each sequence consists of thousands of samples that, in turn, are composed of 29 frequency bands. Each sample is labeled as either inter-ictal or ictal.

It was decided that the dataset needed to be filtered in order to have samples of 3 different classes (pre-ictal (1), ictal (2) and post-ictal (3)) and equal number of samples for each class. To do that, each sequence was looped to find ictal segments. Then, for each ictal segment of sample size N, the preceding N samples were labeled as pre-ictal and the N samples after were labeled as post-ictal. The remaining samples were discarded.

-	**Patient 44202**		**Patient 63502**	
Class	Train	Test	Train	Test
Pre-ictal	2691	232	1662	314
Ictal	2691	232	1662	314
Post-Ictal	2691	232	1662	314
Total	8073	696	4986	942

Table 1: Dataset distribution of two patients. Perfect class balancing was achieved with the approach described in this section.

It is important to notice that, with this approach, the time order between consecutive samples was mostly preserved benefiting the training of dynamic networks, for example. In the other hand, a significant quantity of inter-ictal samples were rejected, potentially causing weaker performances across all networks.

3 Architectures

3.1 Feed Forward Neural Network

The first architecture created was the Feed Forward Neural Network. With this network it was tested the sensitivity and specificity of the results when the objective was to predict or detect seizures. In order to execute this test, we trained 3 different networks with different number of layers ([20 10],[20],[10]), and using different patients (44202, 63502).

Predict	**Patient 44202**			**Patient 63502**		
Hidden Layers	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
20, 10	0.9167	0.9828	0.9073	0.6406	0.5262	0.8000
20	0.8578	0.7759	0.9547	0.6485	0.5166	0.8283
10	0.6466	0.1336	0.9828	0.6645	0.5604	0.8037

Detect	**Patient 44202**			**Patient 63502**		
Hidden Layers	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
20, 10	0.9167	0.8017	0.9871	0.6406	0.5765	0.9513
20	0.8578	0.8233	0.9871	0.6485	0.5947	0.9433
10	0.6466	0.8319	0.9849	0.6645	0.6043	0.9428

Table 2: Values of sensitivity and specificity for patients 44202 and 63502 for the three different layer configurations, in Predict and Detect focused networks. All networks were trained during 100 epochs.

After analysing Table 2, we can observe that the neural network with the best accuracy and sensitivity is the one that has 2 hidden layers with values 20 and 10 for the patient 44202. For patient 63502, the 3 neural networks showed similar behaviour, with accuracy values around 0.6, sensitivity around 0.5, and only specificity having better values around the 80 and 90 percent.

3.2 Dynamic Neural Network

Dynamic neural networks are similar to feed forward networks in the sense that a given sample is passed as input to the first layer, activating neurons throughout all hidden layers until the last layer produces an output. Dynamic NN distinguish themselves by not only feeding the current sample but also the preceding N samples and/or N outputs as inputs again to the network (feedforward-dynamic). There may also be recurrent connections between middle layers (recurrent-dynamic networks).

To create dynamic neural networks the group use the *narxnet* matlab function which feeds past inputs and outputs as input to the next training samples. The experiments conducted were very similar to the ones presented in the previous section. Only one hidden layer was considered but its size was changed three times (10,20,30). The number of delayed inputs and outputs was also changed (1,2,3 and 5).

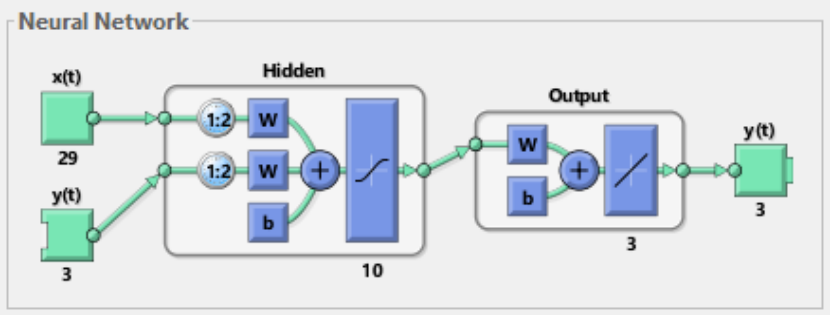


Figure 3: Example of a dynamic neural network with 10 neurons in the hidden layer and delay of 2 samples for both input and output.

Predict	**Patient 44202**			**Patient 63502**		
Hidden Layer	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
10	0.9957	0.9930	0.9919	0.9909	0.9906	0.9957
20	0.9957	0.9913	0.9914	0.9957	0.9882	0.9902
30	0.9957	0.9922	0.9916	0.9957	0.9904	0.9911

Detect	**Patient 44202**			**Patient 63502**		
Hidden Layer	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
10	0.9919	0.9914	0.9957	0.9906	0.9904	0.9949
20	0.9914	0.9914	0.9939	0.9904	0.9904	0.9949
30	0.9916	0.9914	0.9948	0.9904	0.9904	0.9942

Table 3: Measurements done for different hidden layer sizes. The performance of each architecture was measured on 5 random seeds and the average was computed. All networks were trained during 20 epochs and with 2 samples delay.

As observed in figure 3, this type of approach, where the time dependency between samples is taken into account, yielded far greater results than previous networks. Modifying the number of delayed inputs did not alter the performance in a significant way.

3.3 CNN

In order to create a convolutional neural network, our team decided use the Matlab App Deep Network Designer. With this application we created 3 CNN.
The first one starts with the image Input layer. After passing the first layer, it passes through a a 2D convolutional layer with filter size 5 and 20 filters. In the end it passed through a batch Normalization layer, a Rectified Linear Unit layer, a Fully Connected layer, a softmax layer to normalize the values and finally a classification layer.
The second one also starts with the image Input layer and end with the same 5 layers than the previous

one, from batch Normalization layer until the classification layer. The difference in this network is middle part. After the the image Input layer, it passes through a 2d convolutional layer with filter size 3 and 28 filters, then going to a 2-D max pooling layer that performs downsampling by dividing the input into rectangular pooling zones, then calculating the maximum of each zones. The network will perform again a 2d convolutional layer with filter size 3 but 13 filters, and end with the same 5 layers that the first network ended with.
The third one, is equal to the second CNN, only changing the 2d convolutional layers number of filters to 29 and 10, respectively. After the creation of the 3 networks, we decided to test them with the objetive to predict and detect, similar to what was done with the previous networks in this paper.

Predict	**Patient 44202**			**Patient 63502**		
CNN	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
1	0.9000	0.9333	0.8889	0.7073	0.5333	0.9231
2	0.9167	0.9333	0.9333	0.7317	0.6667	0.8462
3	0.9167	0.9333	0.9333	0.7805	0.6667	0.9615

Detect	**Patient 44202**			**Patient 63502**		
CNN	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
1	0.9000	0.8148	1.0000	0.7073	0.8571	0.8889
2	0.9167	0.8889	0.9394	0.7317	0.8571	0.9259
3	0.9167	0.8889	0.9394	0.7805	0.8571	0.9630

Table 4: Values of sensitivity and specificity for patients 44202 and 63502 for the three different layer configurations, in Predict and Detect focused convolutional neural networks. All networks were trained during 50 epochs.

After observing Figure 4, the group noticed that the networks that had 2 2d convolutional layers performed slightly better than the network that only had 1 layer, and that the second and third network had similar accuracy, sensitivity and specificity, although having different filter sizes. It's also worth noting the low sensitivity values of patient 63502 to all convolutional networks when the the objective is to predict ictals.

3.4 LSTM

Long short-term memory networks are a special type of recurrent networks. They are composed by cells arranged in series that compute an output (h_t) based on the input at time t (X_t) and the state of the previous cell (C_{t-1}). At each cell, the previous cell state is altered - information is deleted or added - based on multiple learnable layers, enabling the whole network to learn long term time dependencies between input samples. The cell state can have an arbitrary number of features.

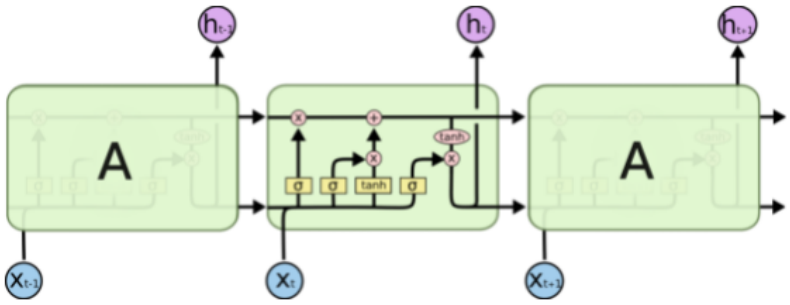


Figure 4: Schematic of an LSTM network. Taken from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

To test the performance of the LSTM architecture applied to our problem, different hidden state sizes

were used - 5, 30 and 100. Similarly to previous sections, accuracy, sensibility and specificity were computed on samples of patients 44202 and 63502.

Predict	**Patient 44202**			**Patient 63502**		
Hidden Layer	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
4	0.9233	0.8664	0.9668	0.5811	0.4503	0.7745
30	0.9164	0.8414	0.9681	0.5896	0.4873	0.7701
100	0.9351	0.9026	0.9659	0.5907	0.4605	0.7971
Detect	**Patient 44202**			**Patient 63502**		
Hidden Layer	Accuracy	Sensitivity	Specificity	Accuracy	Sensitivity	Specificity
5	0.9233	0.9466	0.9310	0.5811	0.5102	0.9669
30	0.9164	0.9578	0.9164	0.5896	0.5140	0.9669
100	0.9351	0.9569	0.9491	0.5907	0.5217	0.9653

Table 5: Measurements done for different hidden state sizes. The performance of each architecture was measured on 5 random seeds and the average was computed. All networks were trained during 50 epochs.

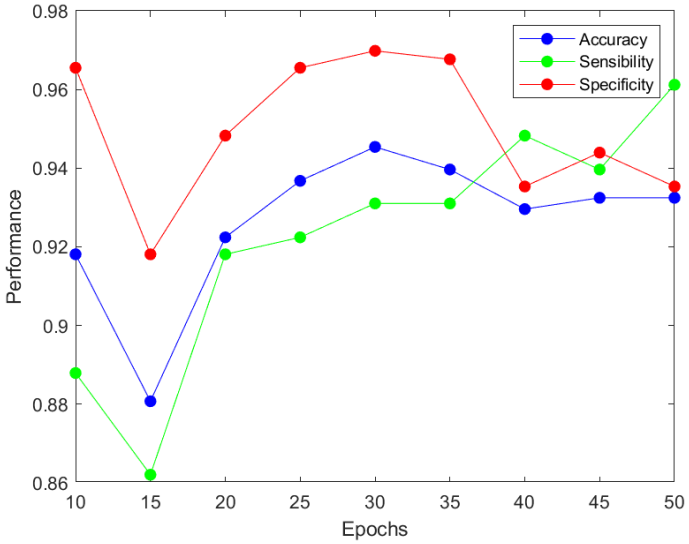
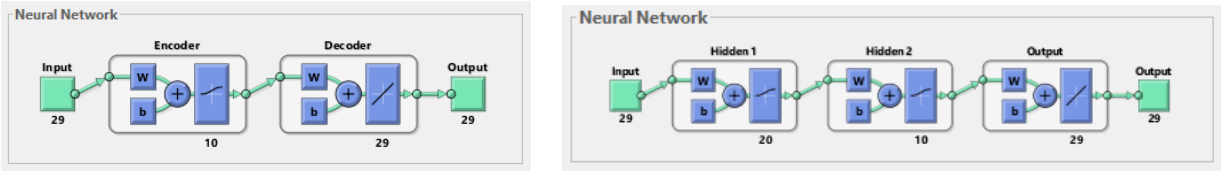


Figure 5: Performance of LSTM with hidden state size of 5, trained on patient 44202 for various epochs.

3.5 Autoencoder

An autoencoder is a feed forward network that aims to reduce the dimensionality of data. Usually the middle layer of this type of networks has less neurons than the outer most layers, forcing the network into coding the input features into less and highly representative new features. Ideally this reduction in the number of features incurs in a small loss of information which can be measured by computed the mean square errors between the input X and output of the network X', for example.

Two different configurations were tested for the autoencoder. They can be visualized in the following figures.



(a) (b)

Figure 6: Diagrams of distinct autoencoder configurations ((a) and (b)). The main different being the second network has an additional layer before the encoding layer of 10 neurons.

Configuration	**Configuration (a)**	**Configuration (b)**
MSE	0.1941	0.1825

Table 6: Effect of number of layers in performance. MSE measured after 500 training epochs with data of patient 63502.

The difference in measured performance between the two configurations is negligible (see table 6). After this experiment, the performance of configuration (b) was investigated further by varying the number of encoded features.

Nº Encoding neurons	**5**	**10**	**15**	**20**
MSE	0.2300	0.1825	0.1806	0.1735

Table 7: Effect of encoding layer size in performance. MSE measured with configuration (b), after 500 training epochs and with data of patient 63502.

Naturally, the autoencoder incurs in less loss of information when the number of encoded features tends to the number of original features (29). This effect can be observed in table 7.

Architecture	**Accuracy**	**Sensibility**	**Specificity**
LSTM	0.5907	0.5217	0.9653
Autoencoder + LSTM	0.5907	0.3803	0.9901

Table 8: Differences in performance of LSTM network with and without encoding training samples by the autoencoder. Training samples were encoded into 20 features. LSTM network was trained for 50 epochs and had a state size of 100, while the autoencoder was trained for 2000 epochs. Both used patient 63502 samples.

Interestingly, reducing the number of features using an autoencoder did not improve the LSTM accuracy measured on test data. In fact, the specificity suffered a slight increase from 0.9653 to 0.9901 at the cost of less sensibility.

4 Pre-processing and post-processing

Generally, one of the first steps in training neural networks is to normalize data to [0,1], for example, as it may lead to faster convergence and smaller training times.

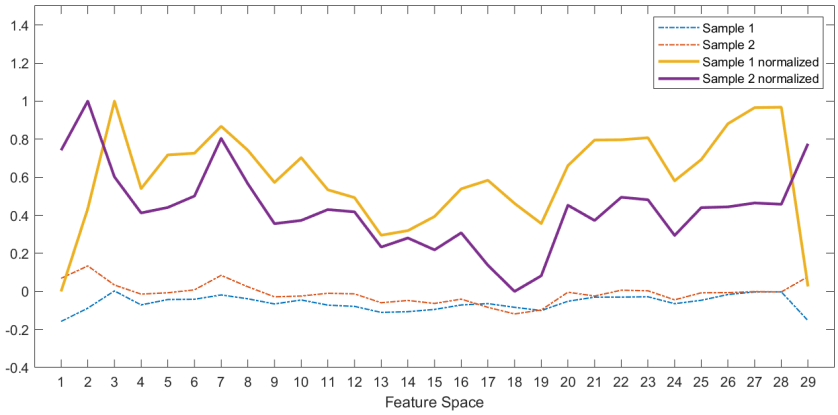


Figure 7: Raw data vs normalized feature values in [0,1] of two data points.

After normalizing the data provided from patient 63502, the performance of one of the feed forward networks discussed in section 3.1 was evaluated again. The network chosen had two hidden layers of sizes 20 and 10 and was trained for 100 epochs. The model trained with normalized samples suffered a 7.65% increase in accuracy, 13.76% increase in sensibility and 4.23 decrease in specificity in comparison to the model trained with the raw dataset.

On the other hand, the predictions made by the trained models can also be filtered to yield better results. For example, a moving window strategy may be considered, where the predicted class for the N-th sample ca be altered to match the class that appears most in the predictions for the preceding samples. To work, this approach requires that the samples given to the model need to be ordered in time and the tunning of the window size. The following table reflects the results observed for different window sizes applied to the output of the feed forward network used before.

Window Size	**Accuracy**	**Sensibility**	**Specificity**
Raw predictions	0.6008	0.4586	0.9857
3	0.6359	0.4490	0.9841
5	0.6476	0.4427	0.9809
10	0.6444	0.4363	0.9729
30	0.5977	0.3662	0.9411

Table 9: Effect of post-processing of outputs on the performance of the system composed by a simple feed forward network.

5 GUI

In order to train and test all the networks in this paper, our group used Matlab Design App to design a simple Graphical User Interface (GUI).

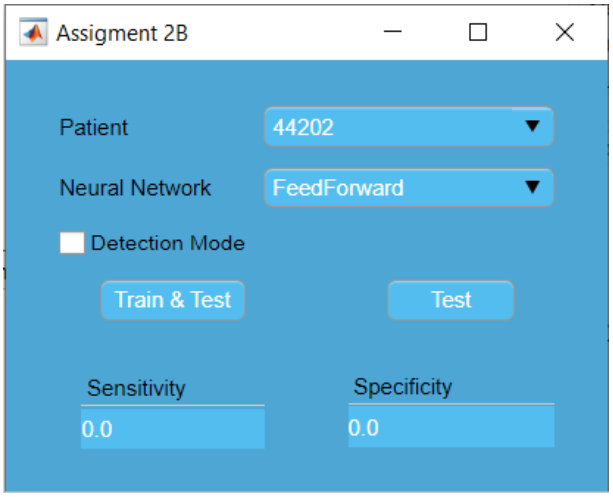


Figure 8: Graphical user interface.

As we can observe by Figure 8, we created 2 dropdown buttons, one to choose the number of patient that will test the neural network, and the other to choose what type of neural network we will be testing. In the window there's also a checkbox to choose Detection Mode or Prediction mode.
After choosing all the 3 parameters, the user needs to choose between training and testing the neural network, or just testing, depending if the network is already save or not.
After clicking in either of the buttons, the values of sensitivity and specificity will appear in the boxes underneath.
The GUI helped simplify the training and testing for all the values present in the paper.

6 Results

Dynamic neural networks outperformed the remaining types of models across all three metrics. They achieved accuracies greater than 99% regardless of the number of delayed inputs or outputs. This may indicate high temporal correlation between data sequences that can be exploited by neural network models given that feed forward neural networks, which receive no delayed inputs and outputs, were one type of models that scored worst in the experiments.

However, LSTM networks that, supposedly, can learn this correlation between samples in a more complex manner did not perform on par with dynamic networks. That was especially true when they were trained and tested with data from patient 63502.

This trend of networks trained and tested with data from patient 63502 under performing comparatively with networks trained with data of patient 44202 was found regardless of the type of model chosen. That may be caused by differences in training dataset sizes (8073 vs 4986 samples), differences in medical conditions of the subjects, etc.

7 Conclusion

This project allowed us to better understand the strengths, weaknesses and applications of various neural network architectures such as feed forward networks, dynamic NNs, LSTMs, CNNs and autoencoders with a real world problem in mind. It also enabled the deployment of creative solutions to tackle the problem at hand, such as the use of convolutional models, that are mostly used in image recognition tasks, in prediction and detection of events in time series.

In addiction, the results obtained highlight the need for multiple performance metrics to evaluate the quality of the models. Often times, a seemingly good accuracy values were accompanied by low sensibility scores that prevent the incorporation of such models in critical systems found in medical environments, for example.