University of Coimbra

Machine Learning Report

Prediction and Detection of Epileptic Seizures

Authors:

Adolfo Pinto, nº 2013138622, uc2013138622@student.uc.pt
Jani Hilliaho, nº 2016167354, uc2016167354@student.uc.pt

Faculty of Sciences and Technology
Department of Informatics Engineering

November 2016

Introduction

Epileptic seizures are caused by a disturbance in the electrical activity of the brain. Although the epilepsy causes are still unknown and it can not be fully prevented, we can take a look at the seizures associated with it and reduce the risk of damaging the brain. The main purpose of this project is to predict and identify the occurrence of epileptic seizures with neural networks.

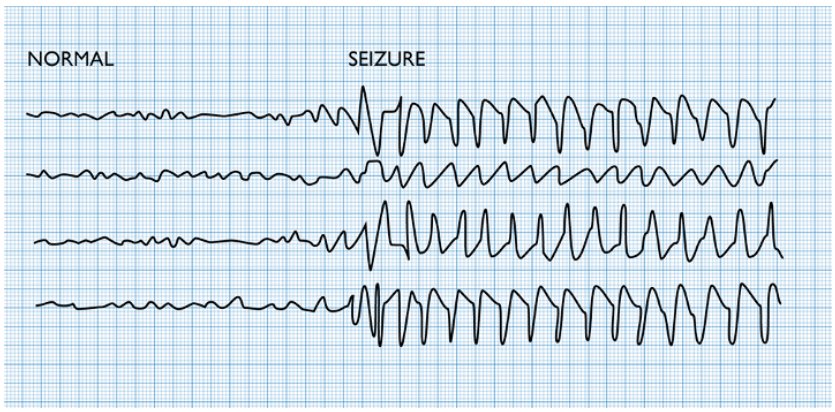

Fig. 1 - seizure happening. sample extracted from a Electroencephalogram

The data used in this project was extracted from patients with an electroencephalogram (EEG). Example of EEG curve can be seen in figure 1. Information about used two patients can be seen in Table 1.

ID	Sex	Age	Age	Location	Type	Hours rec.	Number
112502	F	11	3	RMT	CP(4), SP(4), UC(6)	155	14
54802	M	17	1	LMT, LLT, L-T	SP(25), SG(6)	142	31

Table 1 - Patient's seizure information.

Legend: CP - Complex Partial, SP - Simple Partial, UC - Unclassified, SG - Secondarily Generalized, RMT - right mesial temporal lobe, L-F - left frontal lobe,

As described in table 1, the first patient has 155 hours of EEG recording and the second one 142 hours. It also presents some technical information about the brain, namely in which lobe the seizures were detected and what kind of seizures they were.

To detect an predict the seizures, neural networks were created and trained using the neural network toolbox in MATLAB. The data from the patients was divided in four possible classes, representing the state of the brain: **inter-ictal,** corresponding to the normal brain state, **pré-ictal** meaning that a seizure is coming, **ictal,** when the seizure is happening and **pos-ictal,** when the seizure ended.

To evaluate the performance of the neural networks created, **specificity** and **sensibility** were computed for each case. *Sensitivity* (how many true seizures did it preview or detect, high number corresponds to high sensitivity) and *specificity* (how many false seizures did it predict or detect, high number corresponds to high specificity) can be calculated using the following formula:

$$SE = Sensivity = \frac{TP}{TP + FN},$$

$$SP = Specificity = \frac{TN}{TN + FP},$$

where TP is True Positives, TN, True Negatives , FP, False Positives and FN, False Negatives.

Neural Networks

During this project, we tested four different types of neural networks: *feedforward, distributed delay, time delay* and *layer recurrent* networks.

The **feedforward** neural network consist of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

We already know that the human brain is a dynamic system with memory. Following this idea, in the **layer recurrent** neural net we can add delays in some characteristics, in which the net can learn and predict any output based on previous inputs (considering that we feed the network a sufficient number of neurons and delays). They are similar to feedforward networks, except that each layer has a recurrent connection with a tap delay associated with it and that allows the network to have an infinite dynamic response to time series input data.

Distributed delay and **time delay** networks are similar to feedforward networks, adding tap delays to input or layer weights.

In this experiment, from the neural networks tested, we want to see which one(s) have the best performance and gives the best results.

Architecture

Using the two data sets **54802.m** and **112502.m** (made available by the professor), two scripts were created to divide the data for interictal, preictal, ictal and posictal testing and training data files. The gui.m file calls two other scripts to aid in the process, as described in figure 2.

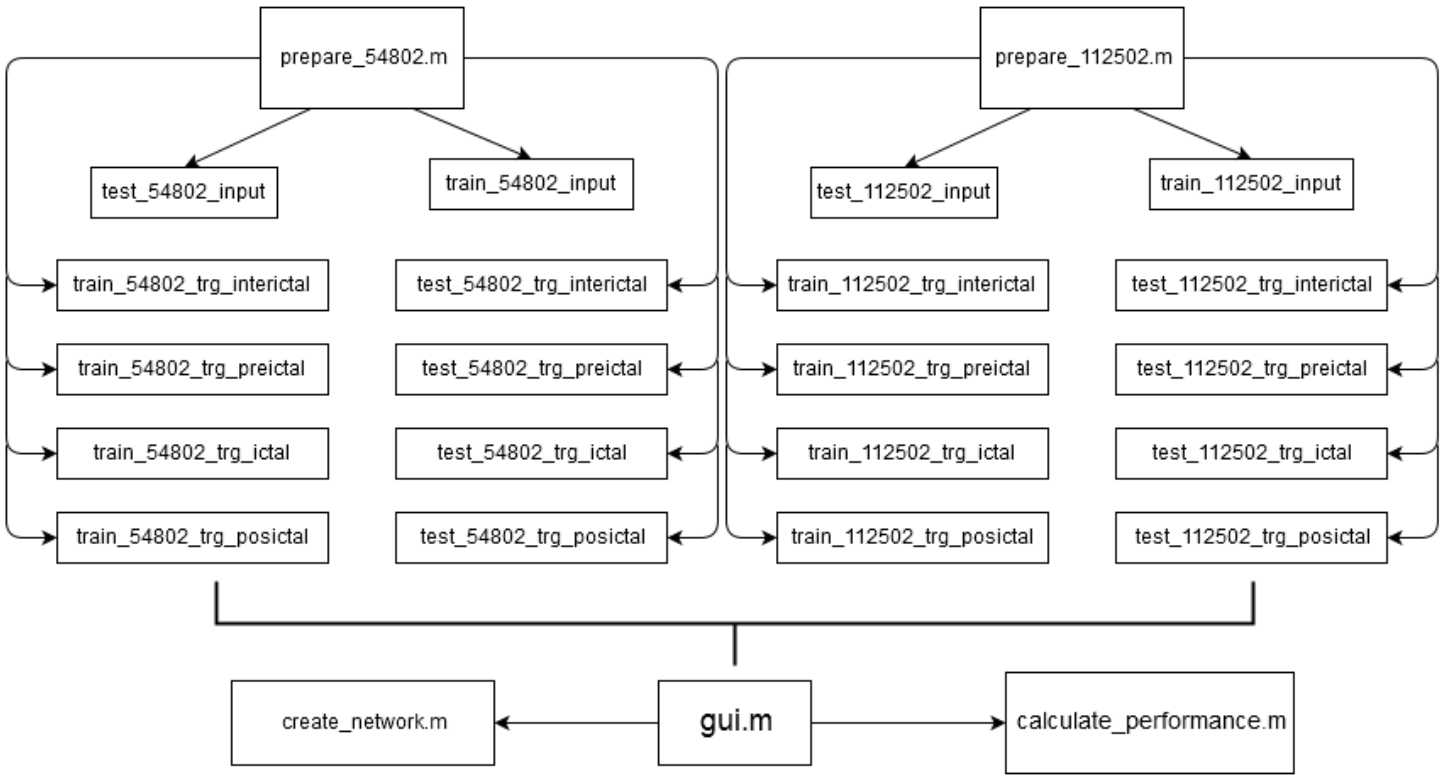

Figure 2. Internal architecture of the application

Matlab implementation

During the project, a custom MATLAB application was implemented for creating, training and testing different neural networks. The application is divided to four logic parts, which are data preparation and creation, training and testing networks.

Data preparation

The data set of each patient consists of input matrix and output vector. The input matrix consists of 29 different features of brain signals collected every second, and the output vector consists of boolean values showing the seizures. The data preparation script marks 600 data points as preictal before the start of every seizure, and 300 points as posictal after the end of every seizure. After that it divides the data for training and testing sets, and also divides output vectors into files containing boolean values for interictal, preictal, ictal and posictal states. The data preparation is divided to two scripts by the patient: *prepare_54802.m* and *prepare_112502.m*.

GUI

A graphical user interface (GUI) was implemented in *gui.m* for creating, training and testing the networks.

It uses the function in *create_network.m* to create networks, and *calculate_performance.m* for calculating the specificity, sensitivity and F-score from the test results of the tested network. Used networks are saved to *networks* folder. All networks in *networks* folder can be trained and tested in GUI. The GUI is shown in figure 3.

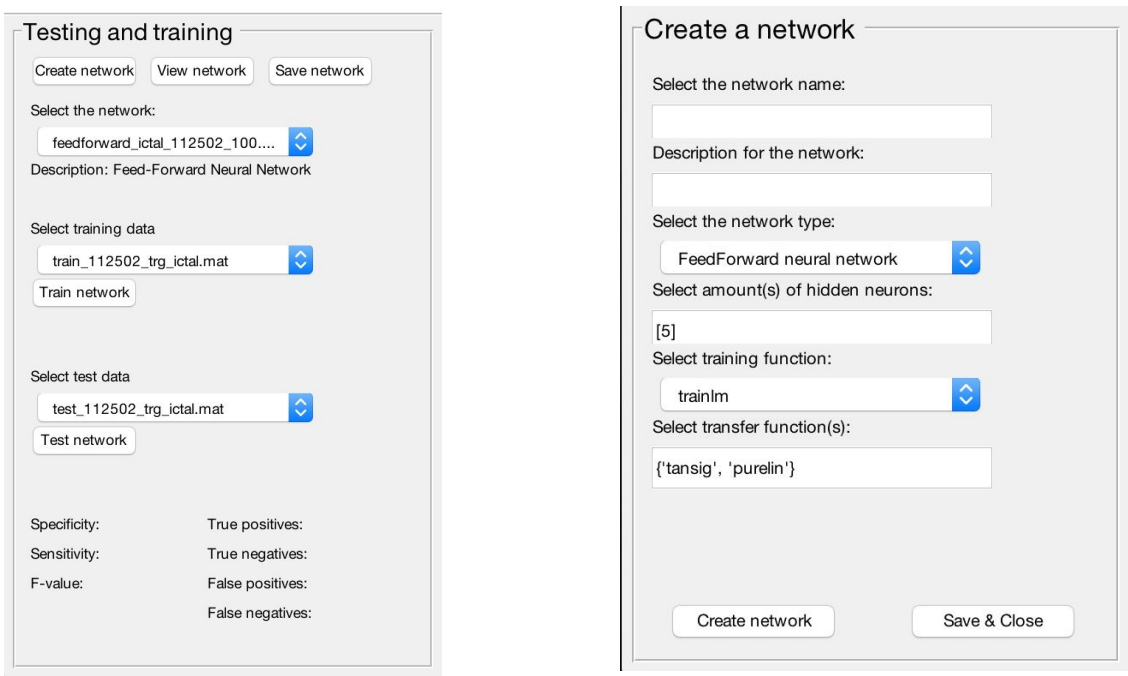

Fig. 3: Graphical User Interface presented to the user

Testing scripts

Because testing a high amount of different networks is slow with the GUI, some scripts for testing were created. *test_feedforward.m* and *test_layrec.m* include functions for testing any feedforward and layer recurrent networks and *tests_feedforward.m* and *tests_layrec.m* include for-loops for creating, testing and saving the results of different networks.

Testing

Feedforward and layer recurrent networks were tested systematically with scripts mentioned in 'Testing scripts' section to find possible correlation between network complexity and the results of it. Both architectures were used to train networks for detecting and predicting seizures with both of patients. The networks used in these tests were trained 500 epochs with training function *trainscg*. The tests were started with quite simple networks, with e.g. 20 neurons in hidden layer and delays of 1:2, and the networks grew more complex at every test. Running these training and testing scripts took a full day using a single CPU thread in a basic laptop.

The used feedforward networks had two hidden layers, and they were tested with 20, 40, 60, 80 and 100 neurons in both of layers. Layer recurrent networks were trained with one hidden layer of 20, 40, 60 and 80 neurons, and with delays of 1:2, 1:4, 1:8, 1:16 and 1:32. All networks had *tansig* as activation functions between layers, and *purelin* in the last layer. The result vectors from tested networks were rounded with threshold producing the best F-scores before comparing them to the target data. The results are shown in figures 4-11.

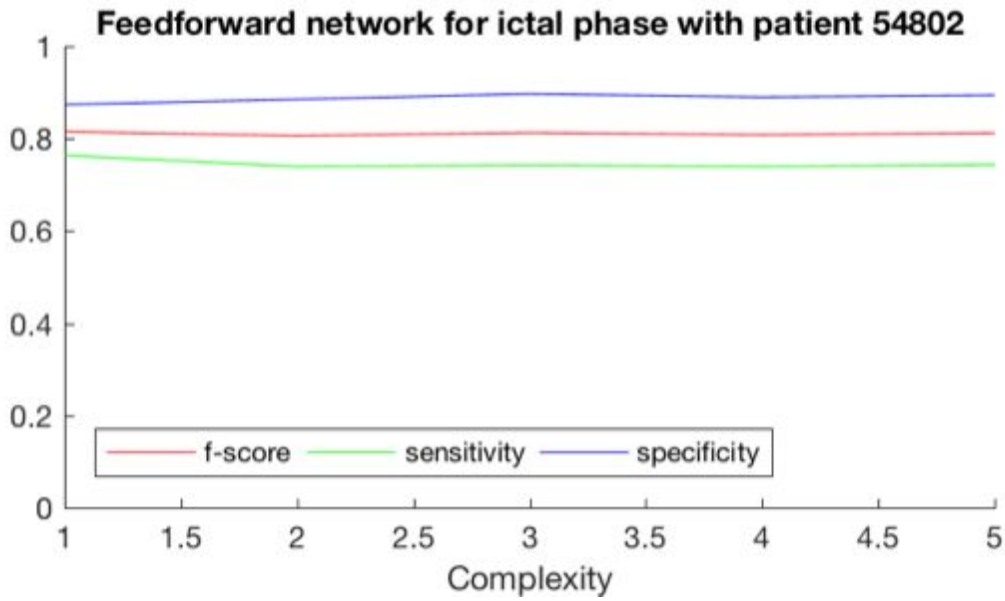

Figure 4: Feedforward network for ictal phase with patient 54802

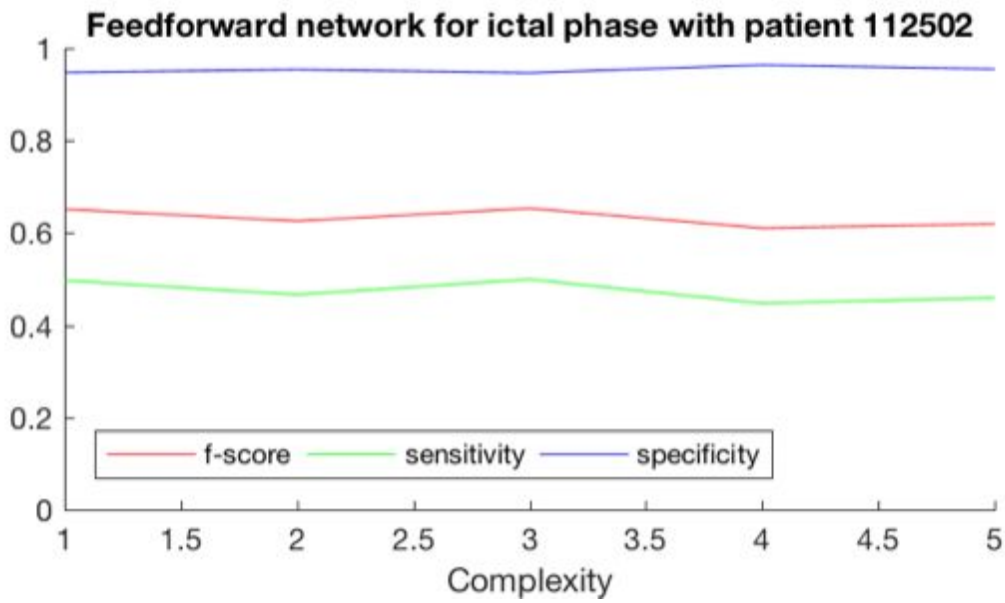

Figure 5: Feedforward network for ictal phase with patient 112502

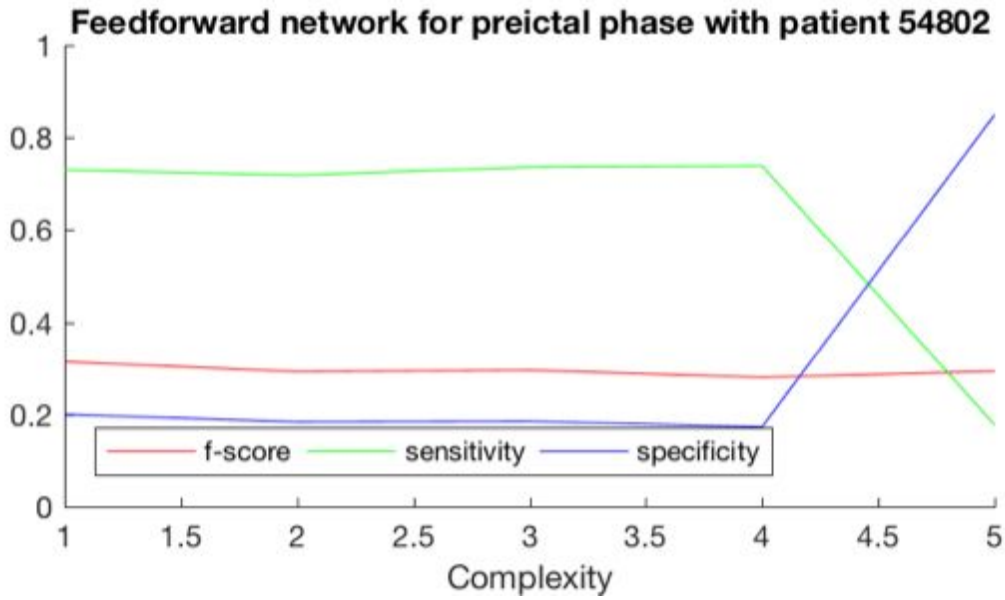

Figure 6: Feedforward network for preictal phase with patient 54802

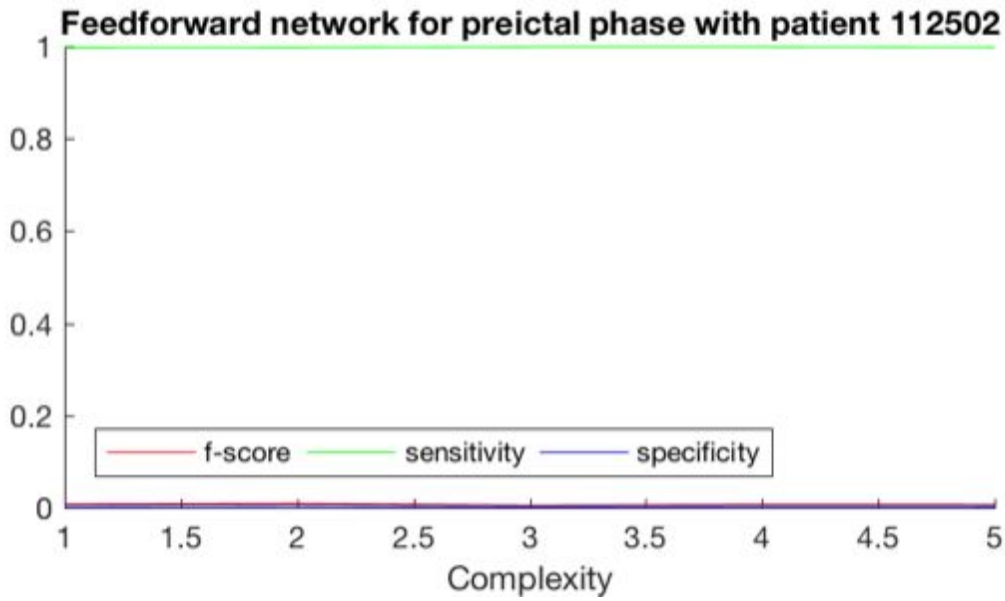

Figure 7: Feedforward network for preictal phase with patient 112502

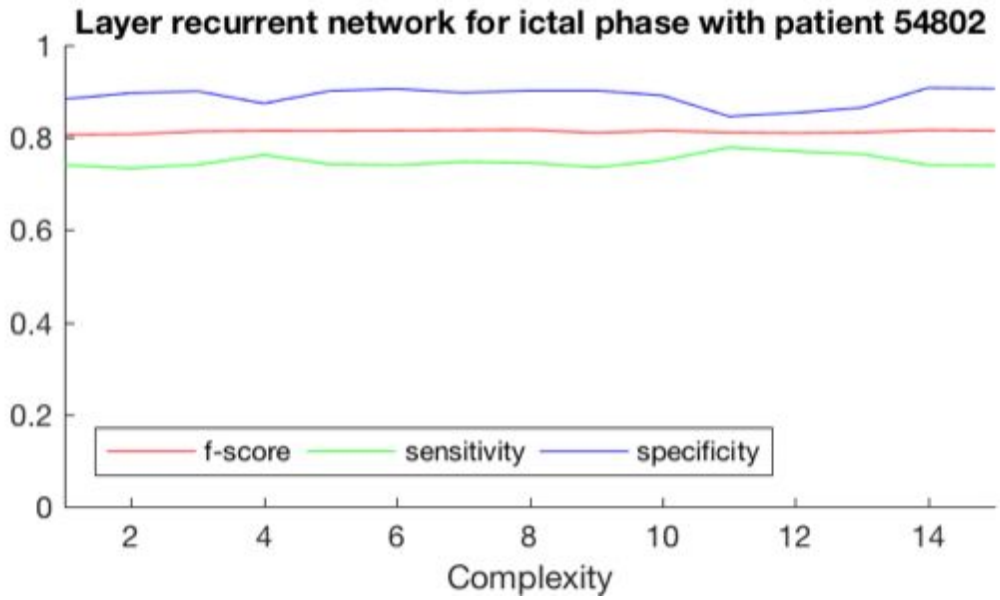

Figure 8: Layer recurrent network for ictal phase with patient 54802

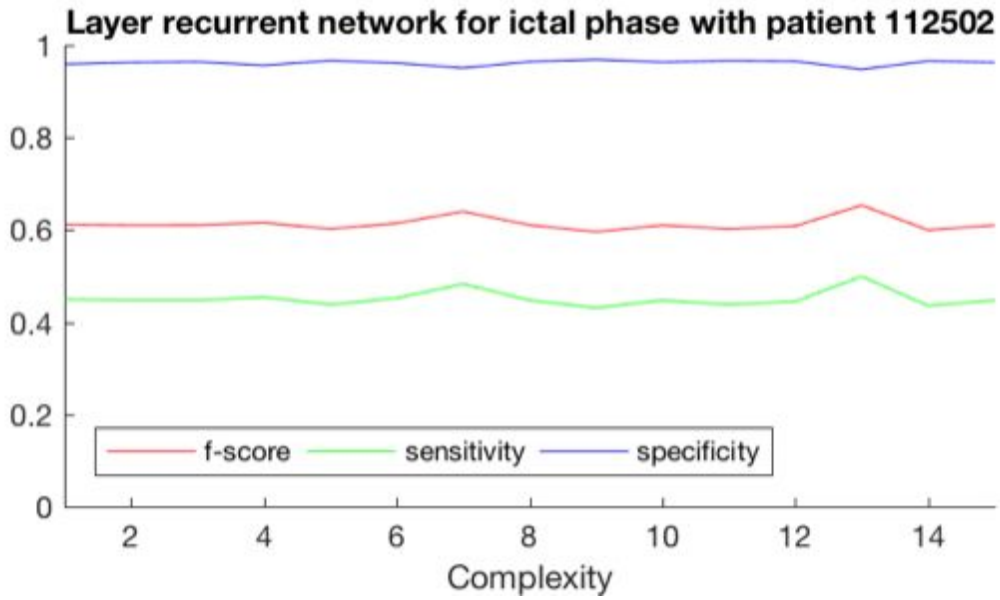

Figure 9: Layer recurrent network for ictal phase with patient 112502

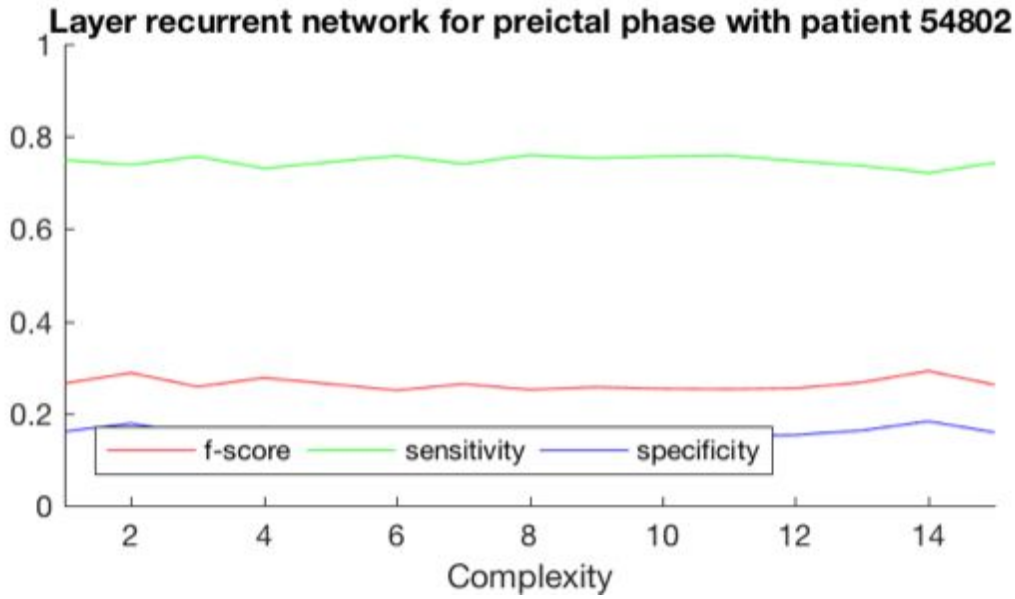

Figure 10: Layer recurrent network for preictal phase with patient 54802

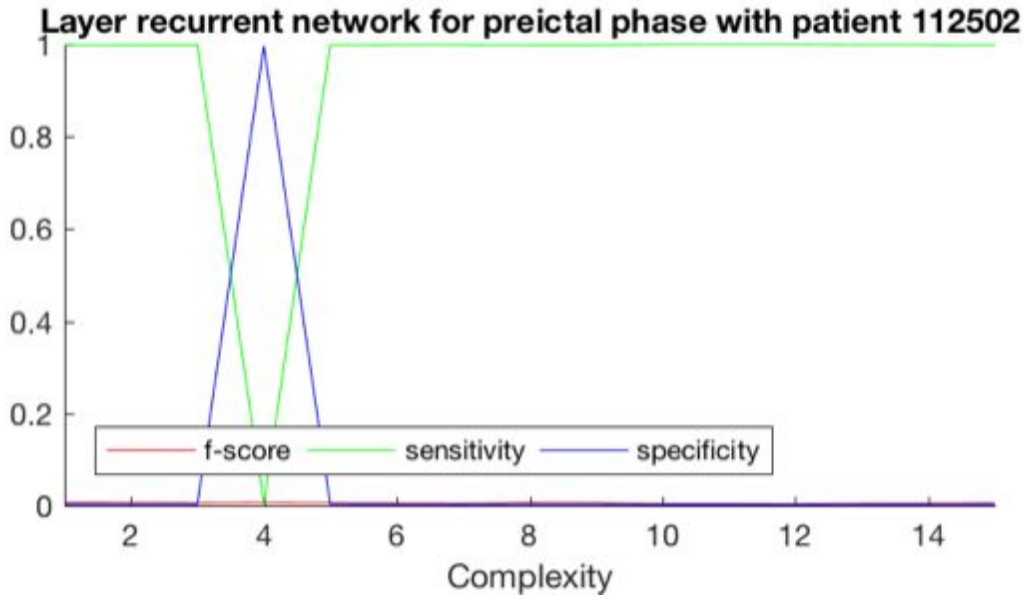

Figure 11: Layer recurrent network for preictal phase with patient 112502

As can be seen in Figures 3-11, the complexity of used networks didn't have any effect in results.

Conclusions

The automated tests didn't show any correlation between the complexity of used networks and the results. It means that the networks weren't trained enough, or the networks weren't complex enough. During this work, the more complicated networks could not be trained and tested because the lack of computing power.