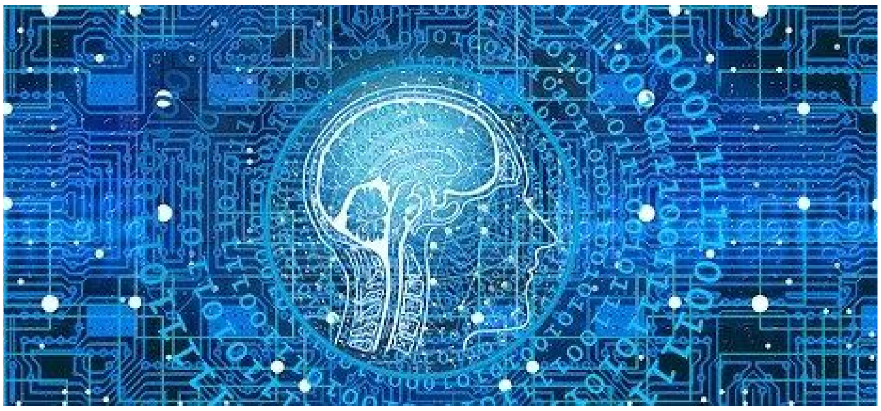
Informatics Engineering Department

Machine Learning
2020/2021 - 1º Semester

Assignment Nº2:

Prediction and Detection of Epileptic Seizures



Teacher:
António Dourado Pereira Correia

Students:

Renato Matos | 2015257602 | uc2015257602@student.uc.pt

Sérgio Machado | 2017265620 | uc2017265620@student.uc.pt

Index

Introduction

The present practical work main objective is to explore the use of dynamic neural network groups: shallow and deep neural networks. To accomplish this, we applied those tools on a difficult health problem that has already been the desire of many researches, to be able to detect and predict epileptic seizures. This is a difficult issue because despite being possible doing it for one person it is really hard to achieve a trained neural network with such a generalization capability to detect and/or predict seizures in every person.

We trained several neural networks: (1) feedforward, (2) layer-recurrent, (3) convolutional, (3) long short-term memory. This was done using the recorded brain signals from two patients, one had 31 seizures and the other 14.

In order to do this, we started by learning the structure and the idea behind each neural network with the help of theoretical classes and its materials. After that, we read the statement and did some research to see how to pre process the data. Lastly we trained all the networks with the processed data changing its parameters and using other tools like encoder to try to achieve trained NN with good sensitivity and specificity. We also used some post processing in order to improve those performance measurements.

Overall, we were able to get some good results both in detecting and predicting the seizures but it also felt like there was room for improvement as it will be shown later.

In this report we state all the steps that we did starting by giving an explanation of the patient's data pre processing and how it was done. Then we show how we implemented all the networks and resources to detect and predict the seizures and show what results we got. In the end we make an overall summary of what we concluded with this project.

Neural networks training architecture

In this section we make a brief description of the flow that the data has to go through since the patient data provided, until getting a neural network that is able to detect or predict epileptic seizures, or ideally, do both things.

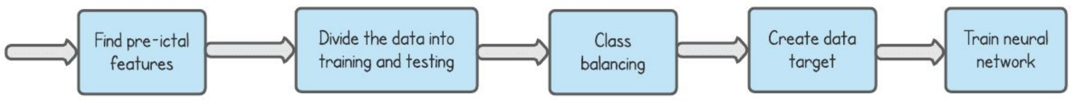
In order to solve this problem we used several neural networks, however each one has a unique purpose and/or training method so we had to adapt the data provided so we could understand the functionality of each dynamic neural network and obtain a trained network with the best performance possible.

In this chapter we divided the neural networks in both groups in each: **shallow** and **deep neural** networks. The first type is known for having few hidden layers while the second type usually has many layers and can be more computationally efficient. Also each group has similar characteristics and so has a similar train process each we will talk about later.

Shallow neural networks

In this work we tested two neural networks: (1) a **feedforward** neural network known for being a network that doesn't have cycles between the nodes and (2) a **layer-recurrent** neural network which has a feedback loop around each hidden layer of the network.

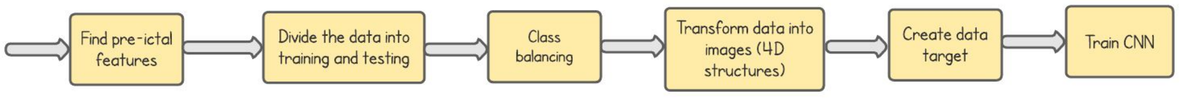
After loading the data we start by identifying the pre-ictal points (900s before the ictal), pos-ictal points which is later included in the inter-ictal class and then we divide the data into training and testing. We did this because it makes no sense to train a NN with data that has been used in training. Since there are a lot more points of interictal data we have to balance the dataset. Our method was that there must be as many inter-ictal points as pre-ictal and ictal combined in the training dataset. After that we create the target matrix, with targets [1 0 0]', [0 1 0]' and [0 0 1]' corresponding to inter-ictal, pre-ictal and ictal, respectively.



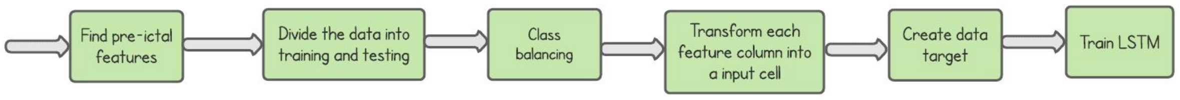
Deep neural networks

For deep neural networks we tested: (1) **convolutional** neural networks which are known for being good to classify images and (2) **long short-term memory** neural networks which take entire sequences as an input cell and can learn long-term dependencies between time steps.

For the CNN, after balancing the data and before creating the data target array which has to be of categorical type, we have to transform the data into a 4D sequence of squared images (each size as the length of the number of the features of the patient data). This process will be explained in the next section.


Find pre-ictal features
Divide the data into training and testing
Class balancing
Transform data into images (4D structures)
Create data target
Train CNN

For the LSTM neural network the process is much easier but still, we have to transform each array of features of the data set into a single cell. As the previous network the target array also has to be of categorical type.


Find pre-ictal features
Divide the data into training and testing
Class balancing
Transform each feature column into a input cell
Create data target
Train LSTM

Data transformation

As described in the previous section in order to train the neural networks to reach a good performance we had to process the data along several steps. Here we will explain how we did.

Shallow Neural Networks

Find pre-ictal points: This step is trivial, we have to go through the "Trg" which has the seizures instants marked as 1's and the rest of the data as 0's. What we did was create a new array of ones (which represent the interictal points), then find the first ictal instant after a non ictal sequence and we attributed the last 900 instants the number two. The ictal sequences were given the number three.

Data division: As said before it makes no sense to the neural network with data that it has seen in training, as so, we divided the dataset into two subsets. The first one is the training subset which must contain at least 70% of all seizures and the other is the testing subset which may have the rest of the seizures (30%) and related data (pre-ictal).

Class balancing: Despite dividing the dataset in the last step, we still would have a lot more of inter-ictal points. As so, we decided to balance only the training dataset in a way that there should be as many inter-ictal points and pre-ictal and ictal.

Target creation: As the first step this one is also trivial. Depending on the number that we put on the first step we now substitute the 1's , 2's and 3's by [1 0 0]', [0 1 0]' and [0 0 1]' which correspond to inter-ictal, pre-ictal and ictal, respectively.

Deep Neural Networks

Some steps in the data transformation of the deep neural networks are the same as the shallow NN which can be seen in their architecture in the first section. As so, we will only describe the differences in the CNN or LSTM which rely on transforming the final training data structure.

CNN: Transform training data into 4D structures

This seemed to be a challenging step but it turned out to be pretty simple. We started by creating a cell and in each index we put a number of instants that correspond to the number of features. Considering the example without autoencoders, the features length is 29 so we would put 29 columns of 29 features. Each index would correspond to an "image" to train the CNN. We did the same for pre-ictal and ictal points. After that we used the function cat so we would end up with a structure with the size of $feature\ x\ feature\ x\ 1\ x\ ImageNumber$ as referred in the work statement. Finally we only had to use the Matlab's function *categorical* in order to obtain a target to train the neural network.

LSTM: Transform training data into a input sequence

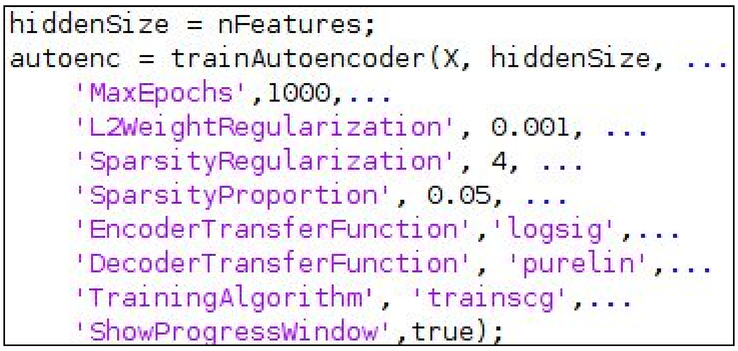
For the LSTM function what we did differently was putting each column of features in a cell index, once this neural network learns a sequence of features at once. The same was done for the target as in CNN.

Implementation

In this section we will describe what approaches we followed in order to detect and predict epileptic seizures for two different patients. Also, we make a detailed description of how we implemented the different dynamic networks and how they work so it might be reproduced by any person.

Autoencoders

The implementation of the autoautocoder was easy to achieve since the Matlab documentation already gives some good examples. We decided to keep most of the parameters set with the **default settings**, since in this work we do not really want to know how well the autoencoder is able to restore the data to with minimum data loss but decrease the number of features. However, we were able to notice that the transfer function *logsig* was better than the *satlin* since if we have negative numbers we would lose the relation between the data. We used the following configuration:

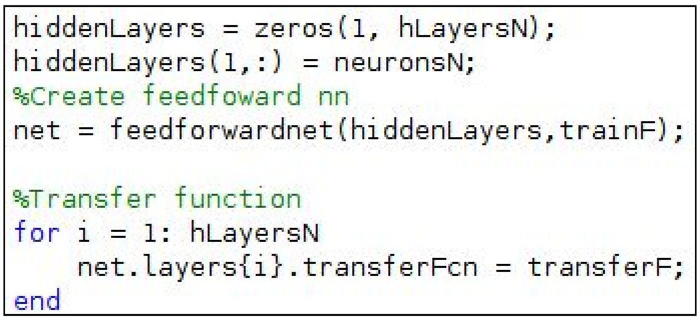


```
hiddenSize = nFeatures;
autoenc = trainAutoencoder(X, hiddenSize, ...
    'MaxEpochs',1000,...
    'L2WeightRegularization', 0.001, ...
    'SparsityRegularization', 4, ...
    'SparsityProportion', 0.05, ...
    'EncoderTransferFunction','logsig',...
    'DecoderTransferFunction', 'purelin',...
    'TrainingAlgorithm', 'trainscg',...
    'ShowProgressWindow',true);
```

Shallow Neural Networks

For the **feedforward** and **layer-recurrent** neural networks we decided to explain them as one since they have very similar training parameters and they are both shallow networks which means they usually have few hidden layers.

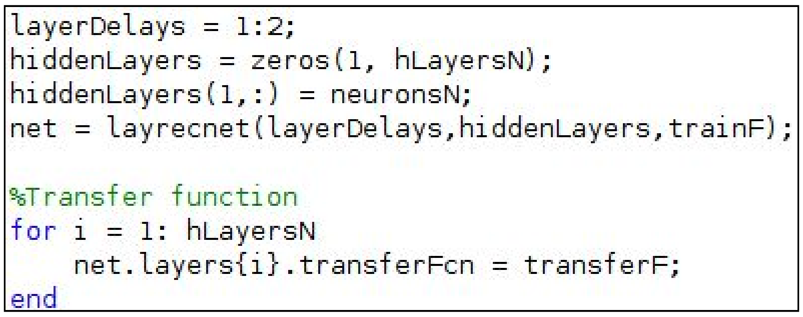
Feedforward neural network : to implement this neural network we started by creating an array to declare how many neurons there will be in each layer. Then we create the neural network in this case *feedforward* and it’s training function. Later we make a loop to define the transfer function of each layer.



```
hiddenLayers = zeros(1, hLayersN);
hiddenLayers(1,:) = neuronsN;
%Create feedfoward nn
net = feedforwardnet(hiddenLayers,trainF);

%Transfer function
for i = 1: hLayersN
    net.layers{i}.transferFcn = transferF;
end
```

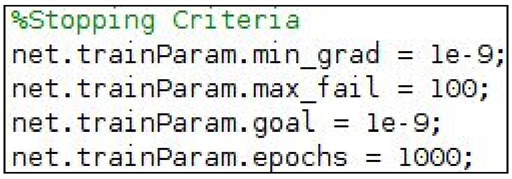
Layer-recurrent neural network: To implement this neural network the code is similar. Beyond the train and transfer function we also have to define the delay of the feedback loop for all the hidden layers. Also to define the network we use the *layrecnet*.



```
layerDelays = 1:2;
hiddenLayers = zeros(1, hLayersN);
hiddenLayers(1,:) = neuronsN;
net = layrecnet(layerDelays,hiddenLayers,trainF);

%Transfer function
for i = 1: hLayersN
    net.layers{i}.transferFcn = transferF;
end
```

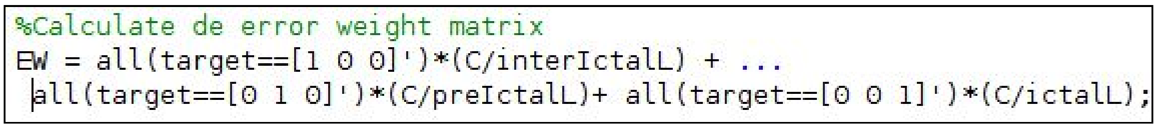
Now we have some code that it is similar to the **both implementations** which is the stopping criteria.



```
%Stopping Criteria
net.trainParam.min_grad = 1e-9;
net.trainParam.max_fail = 100;
net.trainParam.goal = 1e-9;
net.trainParam.epochs = 1000;
```

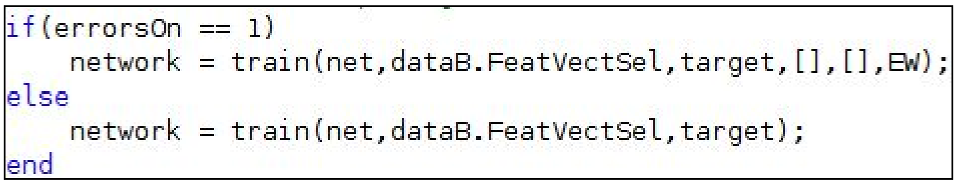
To train the networks we decrease the minimum performance value (goal) and the minimum gradient magnitude (min_grad) and increase the numbers of maximum number of validation increases (max fail), the number of training epochs stayed as default.

Also, as it will be shown further the result of the NN's can be improved not just by balancing the classes but also by using **error weights.** It was suggested that the errors must be the inverse to the quantity of each class. So what we did was dividing the number of total occurrences (C) by the number of each class occurrence. As fewer elements one class has the bigger will be the error.



```
%Calculate de error weight matrix
EW = all(target==[1 0 0]')*(C/interIctalL) + ...
 all(target==[0 1 0]')*(C/preIctalL)+ all(target==[0 0 1]')*(C/ictalL);
```

Finally the only thing left is training the neural network.



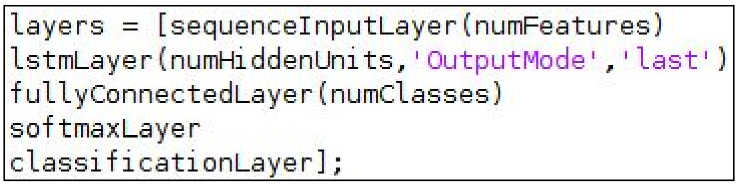
```
if(errorsOn == 1)
    network = train(net,dataB.FeatVectSel,target,[],[],EW);
else
    network = train(net,dataB.FeatVectSel,target);
end
```

Deep Neural Networks

The deep neural networks implementation is quite different from the ones seen at the before or in the other assignment because we have to describe exactly the composition of the neural network layers and then use the function trainNetwork by giving the layers defined before also as the training data and target.

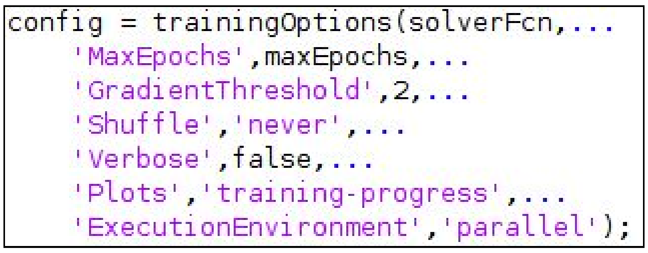
1. Long Short-Term Memory (LSTM) Neural Network

To build the LSTM neural network we started by the layers array containing the network structure definition. We start by defining the number of the sequence input layer (number of features), a lstm layer with the number of hidden units. Setting the outputMode to last since we want a **sequence-to-label** classification, finally we define the transfer function of the last layer.



```
layers = [sequenceInputLayer(numFeatures)
lstmLayer(numHiddenUnits,'OutputMode','last')
fullyConnectedLayer(numClasses)
softmaxLayer
classificationLayer];
```

Also we had to define the training options, we defined the max epochs, the gradient threshold we set it to a positive scalar (the default is Inf), we disallowed data shuffling otherwise we would lose the possibility of prediction.



```
config = trainingOptions(solverFcn,...
    'MaxEpochs',maxEpochs,...
    'GradientThreshold',2,...
    'Shuffle','never',...
    'Verbose',false,...
    'Plots','training-progress',...
    'ExecutionEnvironment','parallel');
```

Finally we just have to train the network using the function *trainNetwork* using the training data and the configurations defined above.

2. Convolutional Neural Network (CNN)

To define the layers of a convolution neural network the process is similar to the LSTM, we have to define the architecture of the network in an array and then the training options. However we must start by defining the size of the images which is $featuresNumber\ x\ featuresNumber\ x\ 1$. Then, in each convoltion2dLayer we must define the size of the filter size which is the size of each scan and the filter number which consists in the number of channels in the output of a convolutional layer also we must define the step size with the Stride options.

```
poolSize = 2;
poolStride = 2;
layerStride = 2;

if isequal(poolType,'average')
    poolingLayers = averagePooling2dLayer(poolSize,'Stride',poolStride);
end
if isequal(poolType, 'max')
    poolingLayers = maxPooling2dLayer(poolSize,'Stride',poolStride);
end
```

Our convolutional NN has 3 layers: the convolutional layer has 16 filters with the size of 5 x 5, the second has 32 filters with 3x3 size and the last is a 3 x 3 layer with 64 filters. Each layer is followed by a batch normalization layer, a hyperbolic tangent layer and then a max/average pooling layer. Lastly we have a softmax layer to perform the classification. Then we define the training options.

```
layers = [
imageInputLayer([features features 1])
convolution2dLayer(5,16,'Stride',layerStride,'Padding','same')
batchNormalizationLayer
tanhLayer
poolingLayers
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
tanhLayer
poolingLayers
convolution2dLayer(3,64,'Padding','same')
batchNormalizationLayer
tanhLayer
poolingLayers
fullyConnectedLayer(3)
softmaxLayer
classificationLayer];
```

```
config = trainingOptions(solverFcn,...
    'MaxEpochs',maxEpochs,...
    'Shuffle','never',...
    'Verbose',false,...
    'InitialLearnRate', 0.001,...
    'Plots','training-progress',...
    'ExecutionEnvironment','parallel');
```

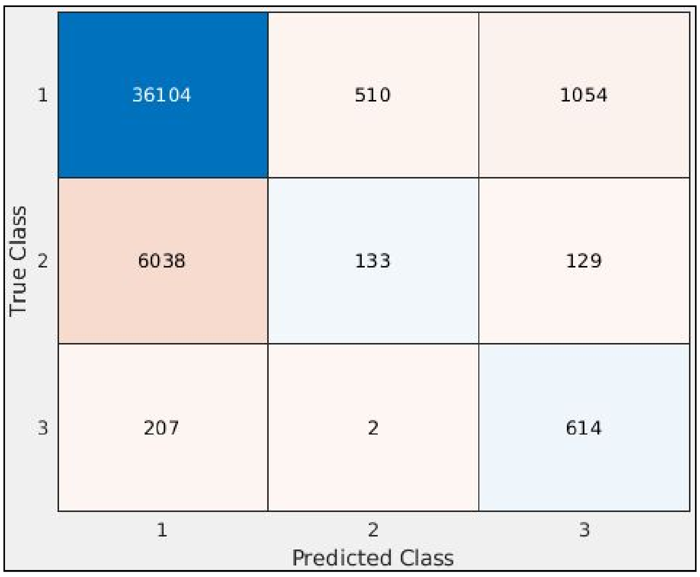
Training

What we tested

In order to obtain trained networks with a good performance we made several tests: **(1)** we applied class balancing, **(2)** we applied error weights inversionally proportional to the number of occurrences and **(3)** we used a simple autoencoder to decrease the number of features. The results are shown in the next section. As in this assignment we had many tests to do and with heavy computational requirements we had to take conclusions from early tests and from Matlab documentation as we stated in the results section. However we made tests to train the performance impact of some variables such as the training function, transfer function, training parameters, layers and neurons number.

How was the performance measured

To measure the performance we used the Matlab function *confusionmat* gives us the confusion matrix which is a table that allows us to visualize the performance of a classification algorithm. From this matrix we can calculate the sensitivity and specificity of a trained NN. Each row represents the instance true class and each column the class predicted by the neural network. Below we can see a plotted example from one of our training tests.



True Class \ Predicted Class	1	2	3
1	36104	510	1054
2	6038	133	129
3	207	2	614

As suggested in the statement we did measure the performance in two ways: point by point and seizure by seizure. The point by point approach just requires us to use the confusion matrix and calculate the true positives, false positives, false negatives and true negatives. On the other hand, the seizure by seizure approach requires us to do some pos-processing. We followed the suggestion in the statement and we checked if in the next 10 classification at least 5 (threshold) are equal. If they are then all 10 results are considered to be from the same class as the 5 or more.

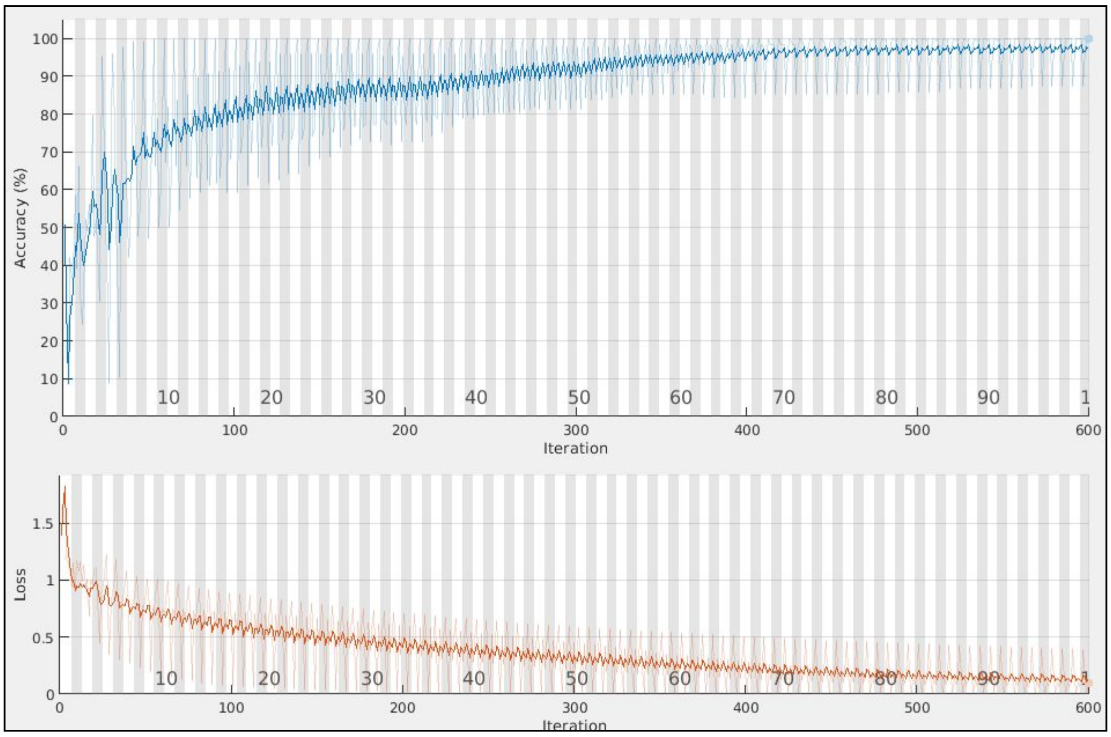
Results and Discussion

We started by doing tests with just class balancing so we would have a way to compare and choose the one with the best performance. Also we did this for both patients which have datasets completely different: one has 31 recorded seizures and the other has 14. As so, we also made the tests for both patients so we could observe the impact of the data on training.

Once that we wouldn't be able to test all training functions, we had to choose the ones that, theoretically, have better performance and test between those. As so, we consulted Matlab's tests done with all the training functions available on different pattern recognition datasets [2]. We conclude that the best ones would be Levenberg-Marquardt (trainlm), Scaled Conjugate Gradient (trainscg) and Conjugate Gradient with Powell/Beale Restarts (traincgb), so we tested those.

Best Results

Overall the trained neural network that achieved the best results was CNN. The combination of the 3 convolutional 2d layer with the batch normalization layers, the hyperbolic tangent layers and average 2x2 pooling layers. This network was able to detect **75%** of the cases with a specificity of almost **100%**. Also we could predict **67%** of the pre ictal instants with a high specificity of **85%**. We defined this as our best network because of the difficulty that is to have a good network that is good at both detecting and predicting seizures. These results are also shown in the results section.



Shallow Neural Networks

Class Balancing

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	-	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	0	56.50	100	95.81	0	54.19	100	99.76
			purelin	0.32	65.74	99.73	91.40	0	68.29	100	98.55
			tansig	1.08	61.19	99.47	96.55	0	62.45	99.98	99.73
		trainlm	purelin	1.91	66.34	98.54	91.12	0	70.84	99.98	98.42
		traincgb	purelin	0.49	65.74	99.51	92.40	0	68.89	99.99	98.98
Layer-Recurrent	1/10	trainscg	logsig	0	72.18	100	88.53	0	75.58	100	97.54
			purelin	4.37	61.36	96.03	94.17	0.11	60.75	99.92	99.36
			tansig	0.14	53.10	99.90	98.39	0	50.79	100	99.84
		trainlm	purelin	3.22	67.19	97.18	90.13	0.03	75.58	99.78	98.12
		traincgb	purelin	2.69	52.86	97.92	97.36	0	54.68	99.93	99.83

Just using class balancing we could obtain significant performance in both sensitivity and specificity in seizures detection. The best results show that the network was able to detect 75% of seizures with a good specificity. The main observation is that with this configuration we were almost unable to predict any seizure except for some values with the layer-recurrent. In this case having pos process gave better results.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	-	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	0	20.23	99.99	99.46	0	20.23	100	99.89
			purelin	0	16.03	99.97	99.88	0	17.56	100	99.97
			tansig	0	22.14	99.73	99.58	0	21.37	99.98	99.90

		trainlm	purelin	0	14.50	99.85	99.92	0	17.56	100	99.97
		traincgb	purelin	0	15.65	99.98	99.82	0	17.18	100	99.91
Layer-Recurrent	1/10	trainscg	logsig	0	21.76	99.99	99.72	0	20.99	100	99.91
			purelin	0	14.89	100	99.85	0	15.65	100	99.92
			tansig	0	23.28	100	99.69	0	21.76	100	99.91
		trainlm	purelin	0	14.89	99.97	99.89	0	16.79	99.99	99.94
		traincgb	purelin	0	14.12	100	99.93	0	16.03	100	99.97

With some preliminary tests we were able to realize that the tests for this patient were giving bad results due to having a low number of seizures when compared to the other patient. As so, we **decided not to use validation** so we could have more data for training. Still, we got worse performances at detecting seizures with this patient and bad sensitivity performances overall. The prediction was also bad. From this data we could conclude that it is harder to train a good neural network with the data from a patient with few seizures but we still made more tests for the two patients as it will be shown further. Also the best results show that we were only able to detect ⅕ of seizure instants but almost never gave a false negative. The networks were unable to predict the seizures.

Class Balancing + Error Weights

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	Yes	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	58.87	82.87	34.55	65.41	65.49	82.26	25.36	74.40
			purelin	60.25	80.80	31.97	68.25	67.08	81.90	22.54	77.09
			tansig	45.24	83.84	43.62	55.54	45.11	83.23	36.69	61.58
		trainlm	purelin	71.80	66.95	22.92	85.65	87.62	68.116	10.50	92.17
		traincgb	purelin	66.98	67.80	30.60	74.18	82.13	69.26	18.25	83.36
Layer-Recurrent	1/10	trainscg	logsig	38.4	85.544	49.35	49.59	37.67	85.45	43.78	54.32
			purelin	53.66	81.4	36.50	63.00	56.84	82.38	27.86	70.96
			tansig	39.51	85.29	50.4	48.57	38.17	84.33	55.21	42.84
		trainlm	purelin	61.76	79.22	30.19	69.57	69.87	80.68	20.60	79.16

		traincgb	purelin	52.65	82.14	37.45	61.97	55.52	82.87	29.10	69.70

By using the proportional inverse error weights we were able to reach some good results in predicting the seizures both in sensitivity and specificity. However the feedforward neural network seems to give better results detecting and the layer-recurrent network predicting. We reached these conclusions by taking in count the specificity (true negatives) and sensitivity (true positives). Also the results show that a network tends to be better in predictions or detection but it is hardly good at both.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	Yes	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	99.63	45.42	3.00	97.17	100	42.37	0.49	99.64
			purelin	99.67	29.39	1.12	99.02	100	28.24	0.24	99.83
			tansig	98.04	47.33	13.26	96.04	100	46.56	5.32	98.77
		trainlm	purelin	0.11	27.09	98.96	98.53	0	25.19	99.99	99.76
		traincgb	purelin	99.62	25.57	1.59	98.53	100	22.52	0.37	99.67
Layer-Recurrent	1/10	trainscg	logsig	99.44	45.04	3.13	96.92	100	41.98	0.46	99.58
			purelin	99.37	26.34	1.59	98.78	100	22.52	0.30	99.72
			tansig	0	45.42	99.91	97.21	0	42.37	100	99.58
		trainlm	purelin	99.26	41.98	3.15	96.90	100	37.41	0.48	99.55
		traincgb	purelin	99.52	27.10	1.90	98.94	100	23.28	0.29	99.80

For patient 112502, we were able to get good results detecting the seizures with the feedforward network being able to detect almost 50% of all seizures with a good specificity considering that this patient had few seizures recorded. Also, we were able to predict some seizures but the results showed that this came with the cost of having a lot of false positives so we cannot say that the results were good.

Configuration	Patient Number	Class balancing	Error Weights	Encoder

	54802	Yes	Yes	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/200	trainscg	logsig	49.95	79.34	40.33	59.55	51.89	78.73	32.41	66.31
			purelin	60.79	77.89	35.69	71.80	72.60	82.13	23.26	78.55
			tansig	49.80	82.62	40.66	60.89	51.95	83.23	32.13	66.71
Layer-Recurrent	1/200	trainscg	logsig	65.39	81.05	28.20	71.94	75.89	81.53	10.29	82.06
			purelin	56.11	67.68	35.4	69.93	62.39	75.69	35.41	74.78
			tansig	5.52	82.74	94.35	58.80	0.57	83.96	99.31	60.40

We made the tests with 200 neurons to see what the results would be like similar to the test done with 10 neurons. Also they confirmed the fact that in order to have good results predicting correctly that comes with the cost of also more false negatives.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	Yes	-

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/200	trainscg	logsig	0.19	33.97	98.87	98.55	0	33.59	99.96	99.81
			purelin	0.48	25.19	98.47	99.05	0	23.28	99.97	99.79
			tansig	0.56	29.77	98.59	98.75	0	28.63	99.97	99.77
Layer-Recurrent	1/200	trainscg	logsig	0.04	26.34	99.39	99.14	0	25.57	99.91	99.83
			purelin	0.22	23.66	99.67	99.03	0	21.75	99.99	99.82
			tansig	1.78	40.46	97.85	97.11	0	37.02	99.83	99.52

For the patient 112502 with 200 neurons we were able to detect 40% of the seizures, however we already got better performances. Also, the results were similar to the ones with just 10 neurons so from this point we decided to not do any more tests with 200 neurons.

Class Balancing + Error Weights + Encoder

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	Yes	Yes -> 10 features

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	58.08	79.10	37.23	62.88	69.33	79.10	24.84	76.34
			purelin	50.71	80.19	40.19	59.29	52.97	80.80	31.01	67.64
			tansig	39.84	86.03	49.90	49.31	39.75	85.42	44.30	54.18
		trainlm	purelin	59.44	78.49	33.10	66.68	66.65	79.34	22.43	77.07
Layer-Recurrent	1/10	trainscg	logsig	66.62	79.64	30.23	70.21	81.21	79.47	16.36	84.48
			purelin	49.84	82.14	41.05	58.42	51.60	82.50	32.39	66.26
			tansig	68.35	76.31	28.76	71.72	83.29	79.10	14.82	86.11
		trainlm	purelin	55.19	79.59	36.37	63.23	59.73	79.95	26.49	72.53

We thought that using where the autoencoder would result in a loss of information but actually we were able to get better results than by not using and the training was faster because we had less features. Actually they turned out to be better and we were able to improve the prediction performance, however the layer recurrent network still gave better detection and prediction accuracy

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	Yes	Yes -> 10 features

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	0	32.44	99.86	97.72	0	29.01	99.89	99.65
			purelin	0	27.86	99.83	98.92	0	24.43	99.88	99.79
			tansig	0.56	35.11	98.70	97.45	0	32.06	99.90	99.43
		trainlm	purelin	0	27.10	99.92	98.90	0	25.19	100	99.80
Layer-Recurrent	1/10	trainscg	logsig	1.48	30.53	88.86	97.94	1.48	29.77	97.81	99.34
			purelin	99.48	29.01	1.96	98.61	100	27.48	0.46	99.70
			tansig	22.44	33.97	80.91	97.43	12.48	32.44	94.52	99.16
		trainlm	purelin	99.63	27.10	1.36	98.67	100	23.66	0.37	99.65

However, on the patient that has less recorded seizures the autoencoder worsens the performance as the results show bad prediction specificity.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	Yes	Yes -> 3 features

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	26.98	81.17	65.81	60.63	25.22	84.08	64.11	54.86
			purelin	61.43	75.21	33.08	66.95	73.90	76.31	19.92	80.27
			tansig	64.81	77.16	31.58	68.75	78.19	79.71	18.32	82.33
		trainlm	purelin	72.06	73.63	24.69	75.85	88.11	75.33	10.20	90.82
Layer-Recurrent	1/10	trainscg	logsig	59.32	79.83	36.13	64.04	69.95	80.80	24.05	76.16
			purelin	38.43	79.10	52.74	60.55	40.19	81.65	45.91	57.96
			tansig	0	80.93	100	61.91	0	81.17	100	73.00
		trainlm	purelin	51.98	78.01	40.24	59.39	57.38	78.37	30.14	68.98

The results show that neural networks trained with instants of 3 features gave better results then with 10 features and similar ones with those that had no autoencoder which surprised us.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	Yes	Yes -> 3 features

NN Type	Hidden layers/ Neurons	Training Function	Transfer Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
Feedforward	1/10	trainscg	logsig	0.37	32.82	98.69	97.60	0	30.92	99.96	99.52
			purelin	0.07	24.43	99.77	98.90	0	21.76	99.99	99.78
			tansig	0.22	33.59	99.32	97.55	0	29.77	99.95	99.58
		trainlm	purelin	0	37.40	99.91	97.02	0	33.97	99.91	99.57
Layer-Recurrent	1/10	trainscg	logsig	0	32.06	99.91	97.76	0	29.01	100	99.91
			purelin	3.78	24.43	90.15	98.87	0.04	21.37	97.72	99.62
			tansig	0.074	32.83	99.82	97.66	0	29.01	99.98	99.65

		trainlm	purelin	0.04	24.05	99.72	99.08	0	21.37	100	99.80

For the second patient the results were similar to the ones with 10 features. The detection performance decreased and the prediction became inexistant.

Deep Neural Networks

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	-	No

NN Type	Hidden layers/ Hidden Units	Transfer Function	Training Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
CNN	3/-	tanh/ softmax	sgdm	67.74	67.85	65.93	100	67.74	75	84.98	99.5
			rmsprop	69.12	60.71	54.57	95.23	87.09	78.57	56.77	99.35
			adam	69.58	57.14	33,70	96.96	85.71	67.85	27.83	98.79
LSTM	2/100	softmax	sgdm	2.32	75.06	98.04	91.65	0.05	78.13	99.97	99.08
			rmsprop	18.22	65.86	83.34	98.36	7.63	67.19	93.13	99.90
			adam	31.94	81.90	71.34	96.59	21.70	83.23	79.36	99.64

CNN provided us with the best results of this assignment. Despite being really hard to have a network capable of performing a good detecting and performance we could achieve this goal with high accuracy results. The LSTM gave some good results detecting but not predicting. We can see that in terms of prediction the results tend to be a bit lower and if they are high they tend to have less specificity. Also this is an example of how having post processing (performance seizure by seizure) increases the performance, especially the specificity.

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	-	No

NN Type	Hidden layers/ Hidden Units	Transfer Function	Training Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
CNN	3/-	tanh/ softmax	sgdm	1.07	44.44	98	100	0	11.11	100	100
			rmsprop	1.07	55.56	99.09	99.48	0	11.11	100	99.48
			adam	2.15	44.44	97.29	100	0	11.11	99.09	100

LSTM	2/100	softmax	sgdm	0	20.61	100	99.57	0	20.23	100	99.90
			rmsprop	0.07	22.52	99.50	99.44	0	20.99	99.98	99.84
			adam	0	27.10	99.84	99.27	0	27.48	99.99	99.77

For the second patient we can see that the results are not good as a result of the dataset having less recorded seizures. However CNN could still provide some decent detection results

Configuration	Patient Number	Class balancing	Error Weights	Encoder
	54802	Yes	-	Yes -> 10 features

NN Type	Hidden layers/ Hidden Units	Transfer Function	Training Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
CNN	3/-	tanh/ softmax	sgdm	4.29	18.28	94.95	99.92	0.3	8.53	94.70	100
			rmsprop	0.15	43.90	97.73	99.55	0	34.14	96.82	99.85
			adam	0	69.51	99.62	96.94	0	80.48	100	96.94
LSTM	2/100	softmax	sgdm	0	0	100	100	0	0	100	100
			rmsprop	0.11	0	99.61	100	0	0	99.73	100
			adam	0	0	100	100	0	0	100	100

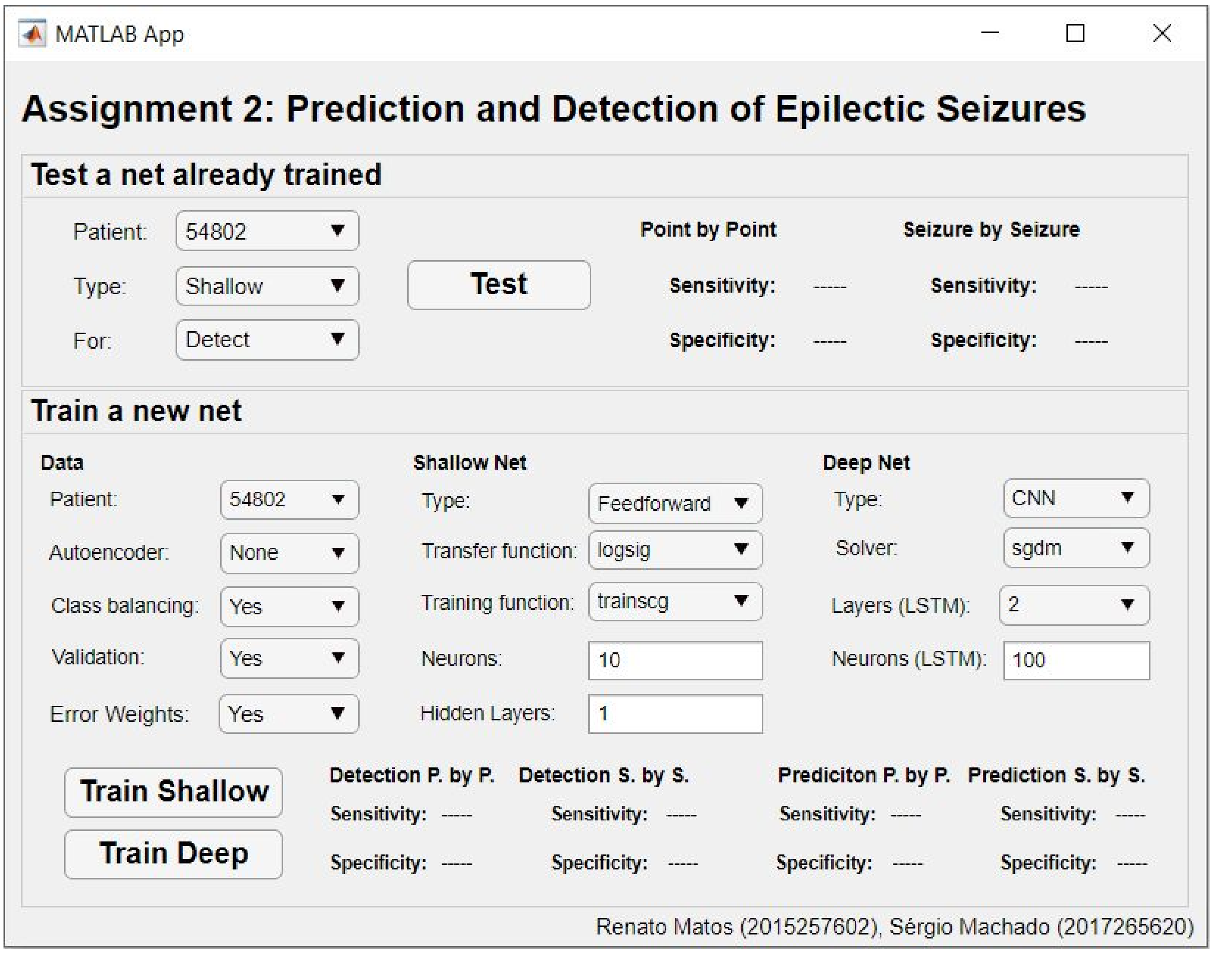
Configuration	Patient Number	Class balancing	Error Weights	Encoder
	112502	Yes	-	Yes -> 10 features

NN Type	Hidden layers/ Hidden Units	Transfer Function	Training Function	Performance Point by Point				P. Seizure by Seizure			
				Sensitivity		Specificity		Sensitivity		Specificity	
				P	D	P	D	P	D	P	D
CNN	3/-	tanh/ softmax	sgdm	0	3.85	99.37	100	0	0	100	100
			rmsprop	0	3.84	99.06	100	0	0	100	100
			adam	72.49	15.38	20.5	100	71.00	3.84	9.34	100
LSTM	2/100	softmax	sgdm	0	0	100	100	0	0	100	100
			rmsprop	0	0	100	100	0	0	100	100
			adam	0	0	100	100	0	0	100	100

This was an attempt to try using encoders before training the data with the deep neural networks however it didn't go well. Despite being able to detect some seizures with CNN we lose all capability of prediction.

Graphical User Interface (GUI)

Here is a print of the graphical user interface:



MATLAB App

Assignment 2: Prediction and Detection of Epilectic Seizures

Test a net already trained

Patient: 54802

Type: Shallow

For: Detect

Test

	Point by Point		Seizure by Seizure	
	Sensitivity:	-----	Sensitivity:	-----
	Specificity:	-----	Specificity:	-----

Train a new net

Data

Patient: 54802

Autoencoder: None

Class balancing: Yes

Validation: Yes

Error Weights: Yes

Shallow Net

Type: Feedforward

Transfer function: logsig

Training function: trainscg

Neurons: 10

Hidden Layers: 1

Deep Net

Type: CNN

Solver: sgdm

Layers (LSTM): 2

Neurons (LSTM): 100

Train Shallow

Train Deep

Detection P. by P.		Detection S. by S.		Prediciton P. by P.		Prediction S. by S.	
Sensitivity:	-----	Sensitivity:	-----	Sensitivity:	-----	Sensitivity:	-----
Specificity:	-----	Specificity:	-----	Specificity:	-----	Specificity:	-----

Renato Matos (2015257602), Sérgio Machado (2017265620)

Warning: In order for the GUI to work the user must put the patients dataset 54802.mat and 112502.mat in the Data folder!

The GUI is composed by two parts:

Test a net already trained

Here we can test the neural networks in which we got the best results regarding the patient (54802 or 112502), the type (Shallow or Deep) and for what it should be used (detection or prediction). After selecting the options and clicking on "Test", the data will be run in the NN to obtain the performance (Sensitivity and Specificity) Point by Point and Seizure by Seizure. It isn't shown, but the use of autoencoders, validation, class balancing, … is already considered in the pre-saved NN.

This were the chosen NN's (the same presented on the excel file):

- 54802, Feedforward:
  - HiddenLayers: 1, Neurons: 10, Training Fcn: trainscg, Transfer Fcn: logsig, With class balancing, error weights and validation
- 54802, Recurrent:
  - HiddenLayers: 1, Neurons: 10, Training Fcn: trainlm, Transfer Fcn: purelin, With class balancing, error weights and validation
- 54802, CNN:
  - 3 average convolutional 2d layers, followed by batch normalization, tanh layer, and average poolingLayers, sgdm, with class balancing
- 54802, LSTM:
  - Hidden Layers: 2, Hidden Units: 100, Transfer Fcn: softmax, Solver: adam, With class balancing
- 112502, Feedforward:
  - HiddenLayers: 1, Neurons: 10, Training Fcn: trainscg, Transfer Fcn: trainsig, With class balancing and error weights
- 112502, Recurrent:
  - HiddenLayers: 1, Neurons: 10, Training Fcn: trainscg, Transfer Fcn: logsig, With class balancing
- 112502, CNN:
  - 3 average convolutional 2d layers, followed by batch normalization, tanh layer, and average poolingLayers, sgdm, with class balancing
- 112502, LSTM:
  - Hidden Layers: 2, Hidden Units: 100, Transfer Fcn: softmax, Solver: adam, With class balancing

Train a new neural network

In this part of the GUI we choose how to train a new net, considering how to preprocess the data and the characteristics of the NN chosen. After selecting the parameters and clicking in "Train Shallow" or "Train Deep", the NN will be trained and the performance results will be obtained. Both detection and prediction results are shown at the same time, both considering performance Point by Point and Seizure by Seizure.

Conclusions

After testing the different dynamic neural networks presented before we were able to reach some good results, being able to detect around **80%** of seizures with a specificity of **99%** with CNN in patient 54802. We were also able to predict close to **70%** of pre ictal instants with a specificity of **85%** also with CNN.

The convolutional neural network with the 3 layer configuration as explained before was our best network in both detecting and predicting seizures. However we had other networks with some good performances, with the feedforward showing decent detecting performances and the layer recurrent network having positive results predicting seizures. Our more disappointing experiment was the LSTM, that despite showing some good results in detecting the seizures was bad at predicting. We believe that this was caused by a possible bad configuration of the network because given that with the concept of LSTM RNN it should be good for predicting since it was long short term memory.

Also we were able to observe that the difference between the data corresponding to each patient seems to have a big impact on the trained networks performances. Our best results were achieved by training the neural networks with the data of patient 54802, which had 31 recorded seizures. However, the patient number 112502 had overall worst results and a proportional high difficulty predicting its seizures since its data set had only 14 recorded seizures.

In terms of training performances we were able to conclude that just by using class balancing the shallow neural networks were only able to give some good results detecting the seizures. However by using inversionally proportional error weights the performance increased a lot and the neural networks were able to predict and detect seizures with good accuracy.

The autoencoder seemed to be really good for shallow neural networks, besides decreasing the features number and consequently the training time, were able to get the same results as with 29 features. In fact we were surprised that the trained NN's with 3 features data were able to show better results than with 10. We thought that we would have worse results by reducing the number of features. However, using the autoencoder with the deep neural networks did not show good results.

In terms of performance measurement, having post processing as explained earlier was better than just measuring the results point by point, resulting in the improvement of the results of specificity values, counting less false positive values.

Overall, we could see that it is difficult to have a neural network good at both detecting and predicting seizures. To achieve excellent results it's better to train a network specifically for one task and result in having good sensitivity and specificity. Also, predicting seizures seems to be truly difficult. In order to have a good sensitivity (true positives), it's more likely that we also have worse specificity (false positives).

References

[1] A. Dourado, Dynamic Networks and Deep Learning

[2] Matlab’s documentation: Choose a Multilayer Neural Network Training Function

[3] Beale et. al, Deep Learning ToolboxT User's Guide

[4] Beale et. al, Deep Learning ToolboxTM Reference