
Gonçalo Vasconcelos Correia, nº 2019216331, pdcorreia@student.dei.uc.pt

João Geirinhas, nº 2019216397, uc2019216397@student.uc.pt

Arquitetura Computacional

# Prediction and detection of epileptic seizures

Trabalho Prático nº 2 - Turma PL 3

Coimbra, 09 de Novembro 2022

Coimbra, 09 de Novembro 2022

# Índice

## 1. Introdução

No âmbito da disciplina de Arquitetura computacional foi proposto desenvolver, implementar e treinar *multilayer NNs* capazes de prever e detetar um ataque de epilepsia. De facto, a epilepsia é uma das mais comuns doenças neurológicas que afeta cerca de 1% da população em Portugal. Porém, é conhecido que cerca de 30% das pessoas que sofrem desta doença têm que recorrer a tratamentos como uso de drogas medicinais ou através de cirurgia, no entanto, apesar destes tratamentos continua a não ser possível para um paciente saber quando vai surgir um ataque de epilepsia. Esta pode durar durantes alguns segundos ou minutos, trazendo à pessoa que sofreu o ataque diversos problemas de saúde tanto no momento como no futuro. Devido a isto, estes tipos de modelos e sistemas estão atualmente a ser bastante estudados.

Para a realização deste trabalho, tal como dito anteriormente, iremos implementar e treinar *multilayer NNs* com e sem *delays*, ou seja, dinâmicas. Estas serão separadas em duas categorias, uma para deteção e outra para treino. Para além disso, ao longo deste relatório iremos, também, abordar arquiteturas de *neural networks* distintas, tais como:

1. Feed Forward NN
2. Dynamic NN
3. CNN
4. Long Short Term Memory (LSTM) NN

Para a previsão e deteção de ataques de epilepsia iremos classificar o brain state em quatro estados: inter-ictal, pré-ictal, ictal e pos-ictal. De facto, o primeiro diz respeito ao normal brain state, o segundo indica que um ataque está prestes a acontecer, o terceiro corresponde a quando a seizure está a decorrer e, por fim, o quarto é a transição para o estado normal.

Por fim, de forma a podermos tirar os melhores resultados e conclusões possíveis abordámos e desenvolvemos este trabalho em diversas etapas:

1. Implementação e treino das *multilayers NNs*;
2. Treino dos *autoencoders* para reduzir o número de *features* para os *classifiers*;
3. Implementação e treino das *deep networks* (CNN e LSTM);
4. Results.

Para além disso, no que diz respeito ao ponto dos resultados iremos abordar e recorrer a três métricas de classificação: a *accuracy*, a *sensitivity* e a *specificity*.

## 2. Data set

No que diz respeito ao data set, este era constituído por dois pacientes, 1 e 2. Estes data sets são constituídos por diversas variáveis: FeatVectSel que são as Features matrix, onde cada coluna corresponde a uma feature; Trg, onde a coluna da matriz identifica as seizures, se o valor for 0 estamos numa non-ictal class (interictal, preictal ou postictal) e se o valor for 1 estamos numa ictal. Cada sequência é constituída por milhares de amostras que são compostas por 29 bandas de frequência. Para além disso, o vetor Trg foi alterado de forma a podermos identificar as 4 classes.

Na seguinte figura podemos analisar um EEG com um ataque epilético e as diferentes classes existentes.

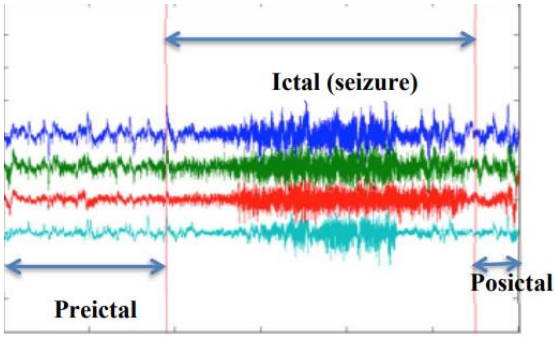


Figura 1: EEG com um ataque epilético e diferentes classes existentes

De seguida de forma a tentarmos obter os melhores resultados possíveis fizemos o balanceamento das classes e fomos testando diversos valores até alcançarmos aqueles que considerávamos os melhores. Paralelamente, de modo a obtermos melhores valores nas redes Dinâmicas e não Dinâmicas aplicámos *weight errors*.

Apresentamos de seguida o balanceamento final alcançado para cada um dos pacientes e tipos de redes.

| | **Paciente 1** | | **Paciente 2** | |
|---|---|---|---|---|
| **Number** | **Train** | **Test** | **Train** | **Test** |
| InterIctalPoints | 24691 | 232 | 22961 | 935 |
| PreIctalPoints | 4961 | 232 | 2961 | 935 |
| IctalPoints | 2691 | 232 | 961 | 935 |
| PosIctalPoints | 4691 | 232 | 2961 | 935 |

Tabela 1- Dataset para redes NonDynamic e Dynamic

| | **Paciente 1** | | **Paciente 2** | |
|---|---|---|---|---|
| **Number** | **Train** | **Test** | **Train** | **Test** |
| InterIctalPoints | 4379 | 551 | 1363 | 348 |
| PreIctalPoints | 2929 | 551 | 1508 | 348 |
| ctalPoints | 2349 | 551 | 1363 | 348 |
| PosIctalPoints | 2929 | 551 | 1363 | 348 |

Tabela 2- Dataset para a rede CNN

| | **Paciente 1** | | **Paciente 2** | |
|---|---|---|---|---|
| **Number** | **Train** | **Test** | **Train** | **Test** |
| InterIctalPoints | 1914 | 551 | 1508 | 348 |
| PreIctalPoints | 2349 | 551 | 1508 | 348 |
| IctalPoints | 1914 | 551 | 1508 | 348 |
| PosIctalPoints | 1624 | 551 | 1508 | 348 |

Tabela 3- Dataset para a rede LSTM

## 3. Neural network architectures

Primeiramente, de forma a se compreender o problema desenvolvido ao longo deste projeto achámos que devíamos explicitar no que é que consistem cada uma das diferentes arquiteturas utilizadas.
De facto, no que diz respeito à Feed Forward Network, esta é um a artificial neural network onde as conexões entre os nós da sua rede neuronal não formam ciclos, pelo que a informação que corre na rede tem apenas uma direção, como pode ser analisado através da figura 1.

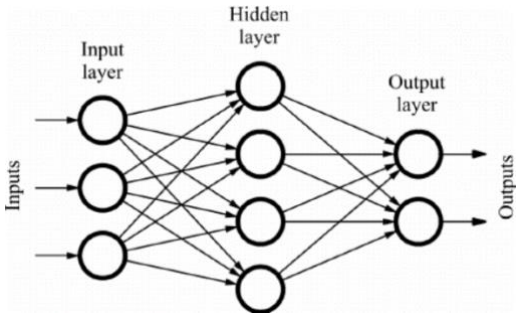


Figura 2: Esquema de uma Feed Forward Neural Network

Esta rede recebe uma série de inputs que cada vez que passam um *layer* são multiplicadas pelos *weights*. Caso a soma de todos os valores obtidos através de - valor de input x *wieghts* - for superior à *threshold* especificada o output que recebemos é 1, se se verificar o oposto o output será -1.

Posteriormente, abordando a *Dynamic Neural Network*, nestas adicionando mais algoritmos de decisão podemos fazer com que a rede neuronal consiga aprender dinamicamente através do input, gerando melhores resultados. Para além disso, a informação nestas não segue uma direção única, ao contrário das *Feed Forward*. Por fim, as *Dynamic NN* têm a capacidade de se poderem adaptar tendo em conta a situação.

De seguida iremos referir as CNN ou *convolutional neural network*, estas caracterizam-se por aprenderem diretamente através dos dados, eliminando a necessidade de extração manual de *features*. Para além disso, são bastante úteis no reconhecimento de imagens, caras ou cenas.

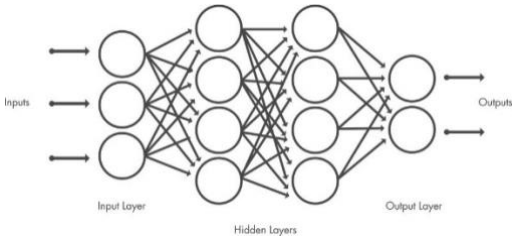


Figura 3: Esquema de uma CNN

Estas são compostas por diversos *layers* (*Convolution, Rectified linear unit* (ReLU) e *Pooling*), estes *layers* (e os restantes) realizam diversas operações de forma a alterar os dados com o objetivo de aprender features específicas para cada dado. O *layer convolution* coloca os inputs num set de *convolutional filters*, o ReLU permite que haja um treino mais rápido e eficiente, transformando os valores negativos em 0 e mantendo os positivos. Por fim, o *layer pooling* simplifica o output efetuando *nonlinear downsampling*, reduzindo o número de parâmetros que a rede necessita de aprender.

Por fim, as LSTM neural networks é uma rede neuronal que é capaz de aprender *long-term dependecies* em sequências. Para além de ser denominada por *long-term memory*, dado que apresenta um número elevado de serial *blocks* também é por *short term memory*, uma vez que cada bloco usa apenas o estado anterior. Estes blocos são constituídos por quatro *blocks* onde se realizam operações elementares (adição e *Hadamard multiplications*), sendo que estes são: Forget gate, update gate, output gate e input gate.

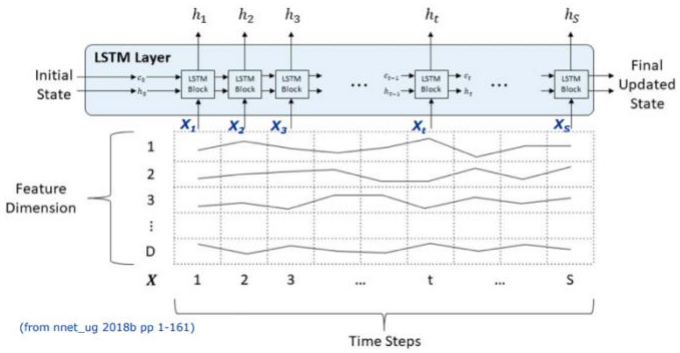


Figura 4: Esquema de uma LSTM Neural Network

Por fim, estas arquiteturas irão ser apresentadas posteriormente aplicadas ao tema do projeto, onde iremos apresentar os resultados que obtivemos para cada uma delas.

## 3.1 Feed Forward Neural Network

Neste ponto iremos abordar A arquitetura denominada por *Feed Forward NN*. De facto, para esta iremos apresentar certos gráficos correspondentes aos dois pacientes existentes no data set. Para além disso, estes gráficos irão conter os dados obtidos nas ações de *Detect* e *Predict* para cada combinação que realizámos, sendo que iremos apresentar os seus valores obtidos em três métricas de classificação, tal como dito anteriormente.
No que diz respeito às combinações experimentámos o uso de 1, 2 e 3 *layers*. Relativamente à com uma camada iremos apresentar os dados obtidos para 10 *hidden units*, para com duas camadas com 10 e 20 *hidden units* e, por fim, com três camadas para 10, 20 e 30 *hidden units*.

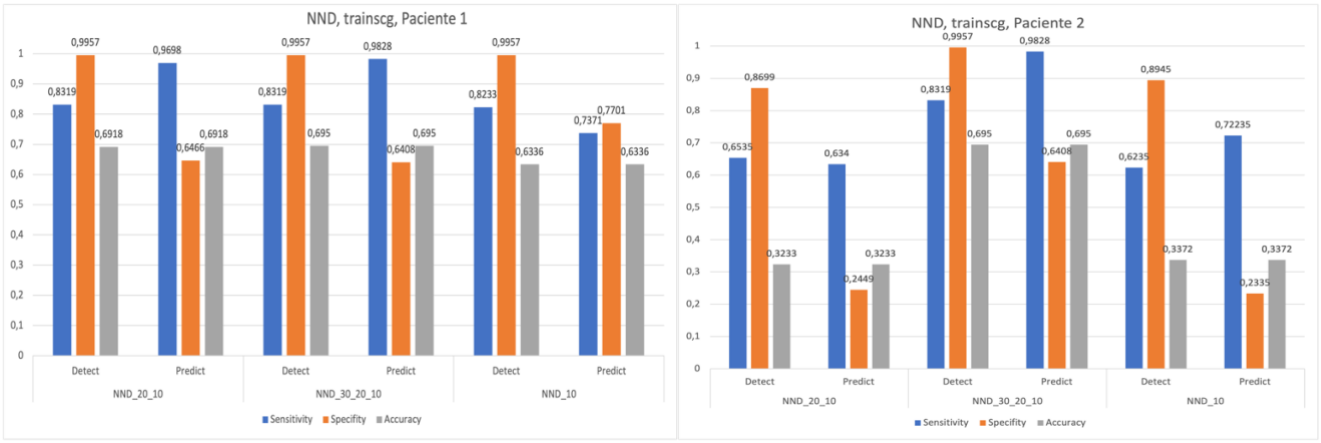


Figura 5: Resultados da arquitetura Feed Forward NN para o paciente 1 e 2 com o treino trainscg

Como pode ser analisado através dos gráficos, conseguimos alcançar os melhores resultados para o paciente 1. Este alcançou os resultados mais positivos para o Detect. Também é curioso analisar que para a ação de Predict a métrica que obteve melhores resultados foi a da Sensitivity enquanto que para o Detect foi a Specifiicity. Relativamente ao paciente 2 podemos também fazer esta análise. Para além disso, neste paciente os resultados foram pouco constantes.
Por fim, podemos verificar que para ambos os pacientes os melhores resultados foram obtidos quando se utilizaram as três camadas.

## 3.2 Dynamic Neural Network

De forma a podermos obter os melhores valores e resultados para esta arquitetura, *Dynamic NN*, utilizámos a função *narxnet* do *matlab*. Esta recebe como parâmetros dois tipos de *delay*, o *inputDelay* e o *feedbackDelay*. De facto, para esta arquitetura utilizámos os seguintes tipos de *delay*: 1:2 e 1:3, dado que após experimentarmos outros valores estes eram aqueles que nos davam melhores resultados e demoravam um tempo plausível. Para além disso, utilizámos as mesmas *layers* do ponto abordado anteriormente (1 *layer* com10 *hidden units*, 2 *layers* com 10 e 20 *hidden units* e 3 layers com 10, 20 e 30 *hidden units*), sendo que para cada uma das três aplicámos os *delays*.
De seguida, iremos apresentar os gráficos que desenvolvemos para podermos tirar as melhores conclusões possíveis. Como se pode ver utilizámos as mesmas métricas anteriormente especificadas e apresentamos os dados para cada um dos pacientes com os diferentes e *delays* e as diferentes *layers*.

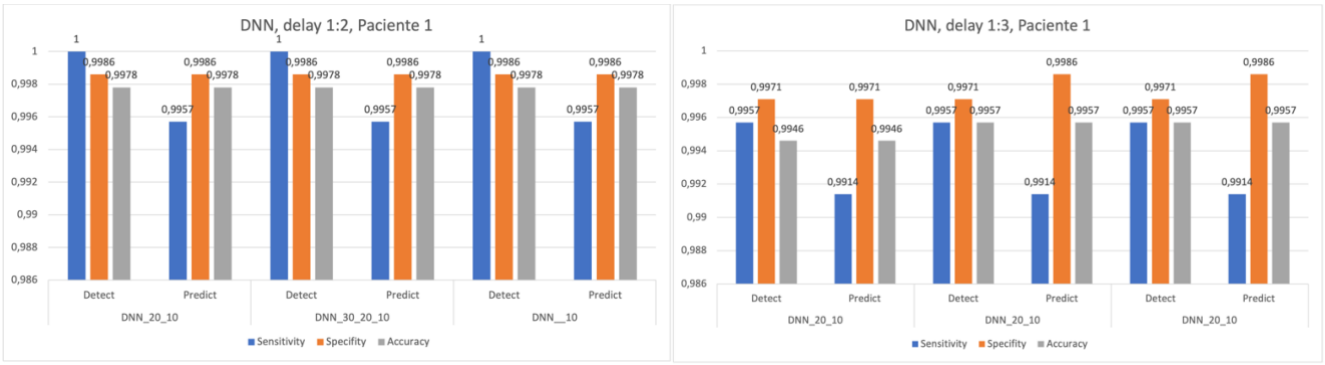


Figura 6: Resultados da arquitetura Dynamic NN para o paciente 1 e diferentes delays

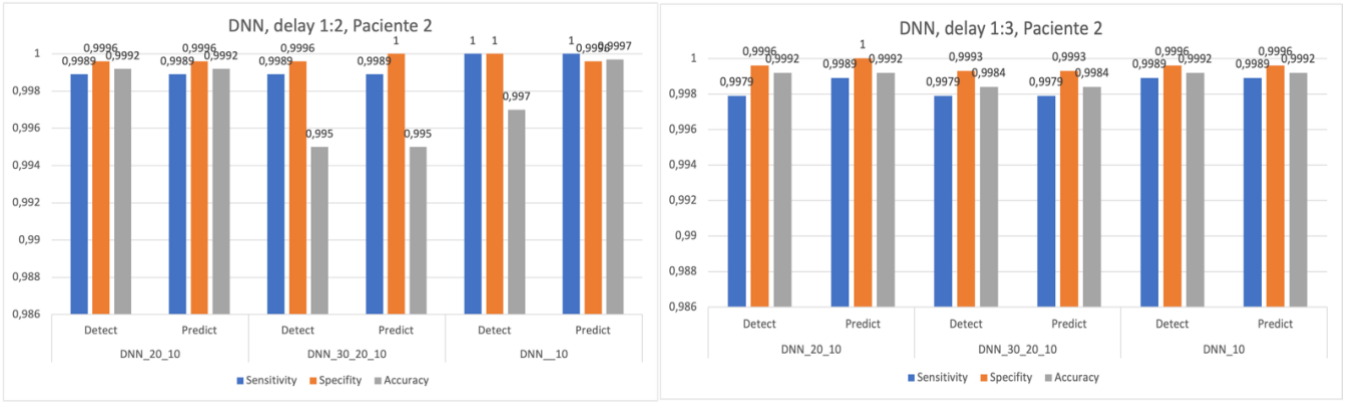


Figura 7: Resultados da arquitetura Dynamic NN para o paciente 2 e diferentes delays

Como pode ser analisado através dos gráficos, obtivemos resultados bastante semelhantes no que diz respeito ao Paciente 1 e 2 com delay de 1:2. Para além disso, podemos também reparar que os melhores valores foram alcançados na ação de Detect e quando se deu o uso do delay 1:2. Por fim, pode ser concluído também que quando analisamos o paciente 1, este obteve resultados mais constantes ao longo de toda a experiência, com qualquer uma das camadas. Já relativamente ao paciente 2 podemos verificar que com o delay 1:3 os resultados, apesar de menos satisfatórios, foram muito mais constantes do que com o outro delay.

## 3.3. CNN

Neste tópico com o intuito de interpretarmos e entendermos melhor a *Convolutional Neural Network* iremos apresentar alguns gráficos correspondentes às experiências que desenvolvemos. De facto, iremos apresentar gráficos para ambos os pacientes (Paciente 1 e 2) e os treinos que utilizámos para alcançar os melhores resultados possíveis. Para além disso, nestes gráficos iremos apresentar três métricas de classificação de resultados, tal como explicado anteriormente. Com estes dados iremos comparar o paciente 1 com o paciente 2 e o *Predict* com o *Detect* tentando chegar a uma conclusão e a uma boa análise dos gráficos.

Inicialmente, como podemos analisar pela Figura 8 apresentamos a CNN com o treino *Adam* e com o treino *Sigmoid*. Para além disso, usámos diversas combinações através do comando convolution2dLayer com 20 filtros 5x5, outra com 32 filtros 5x5 e, por fim, duas combinações com 32 e 16 filtros ambas 6x4.

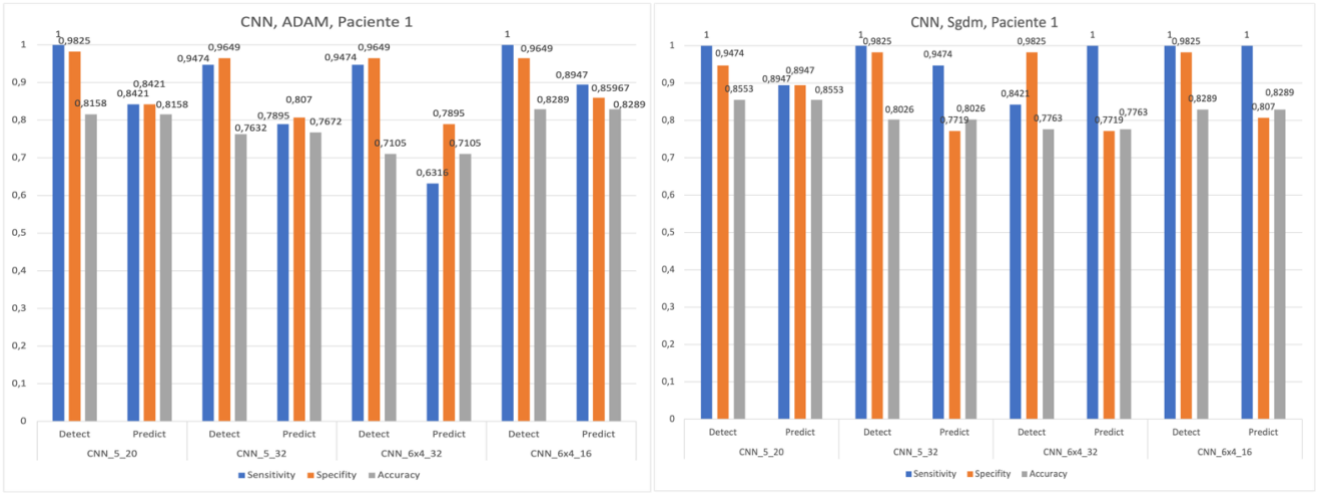


Figura 8: Resultados da arquitetura CNN para o paciente 1 e com os treinos Adam e Sigmoid

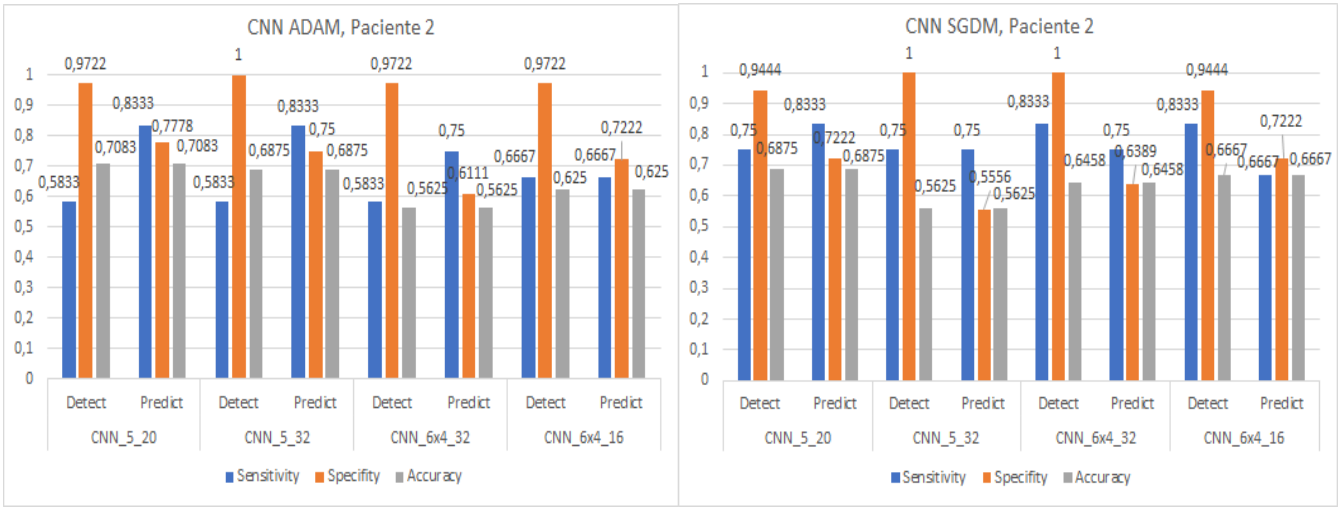


Figura 9: Restultados da arquitetura CNN para o paciente 2 e com os treinos Adam e Sigmoid

Como pode ser analisado através dos gráficos anteriores conseguimos obter valores bastantes positivos para o paciente 1, sendo que os melhores resultados foram alcançados no Detect. Apesar disto, os valores obtidos para o Predict também foram bastantes positivos.
Paralelamente, no que diz respeito ao paciente 2 os resultados alcançados apesar de não terem sido tão bons quanto os do outro paciente, conseguimos reparar que o Detect continua a ser razoavelmente melhor.
Para além disso, analisando os treinos entre si podemos concluir que para o paciente 1 o Sigmoid foi aquele que nos deu melhores valores, dado que, como podemos analisar atinge o máximo em certas experiências no Detect e acaba por ser mais constante ao longo de todos os casos de teste.

## 3.4. LSTM

No que diz respeito à LSTM de forma a podermos tirar as melhores conclusões realizámos, também, algumas combinações relativas às *number of hidden units*. De facto, para cada paciente fizemos experiências com os seguintes valores 29, 50, 100, 125, 150 e 200. Para além disso, neste tópico utilizámos dois tipos de treino o *Rmsprop* e o *Adam*.

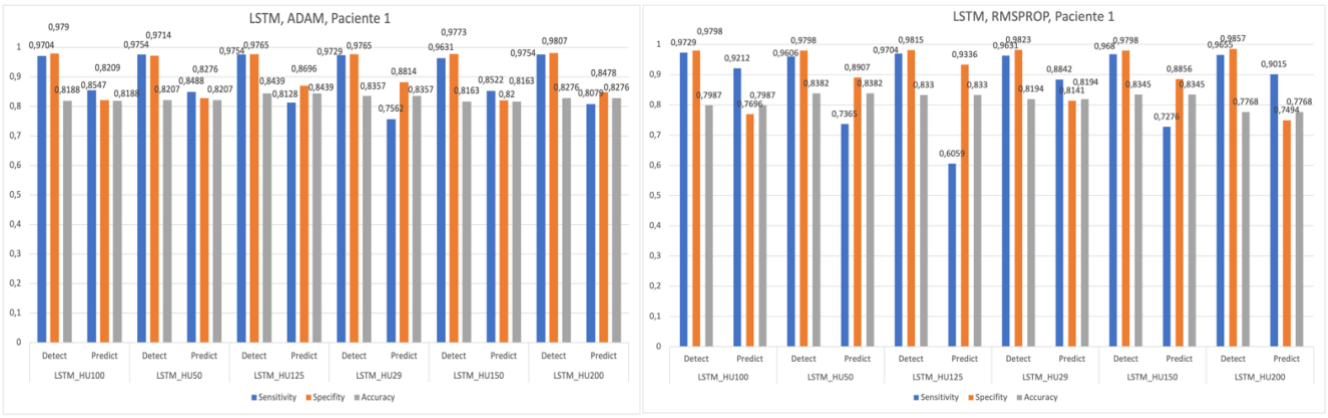


Figura 10: Resultados da arquitetura LSTM para o paciente 1 e com os treinos Rmsprop e Adam

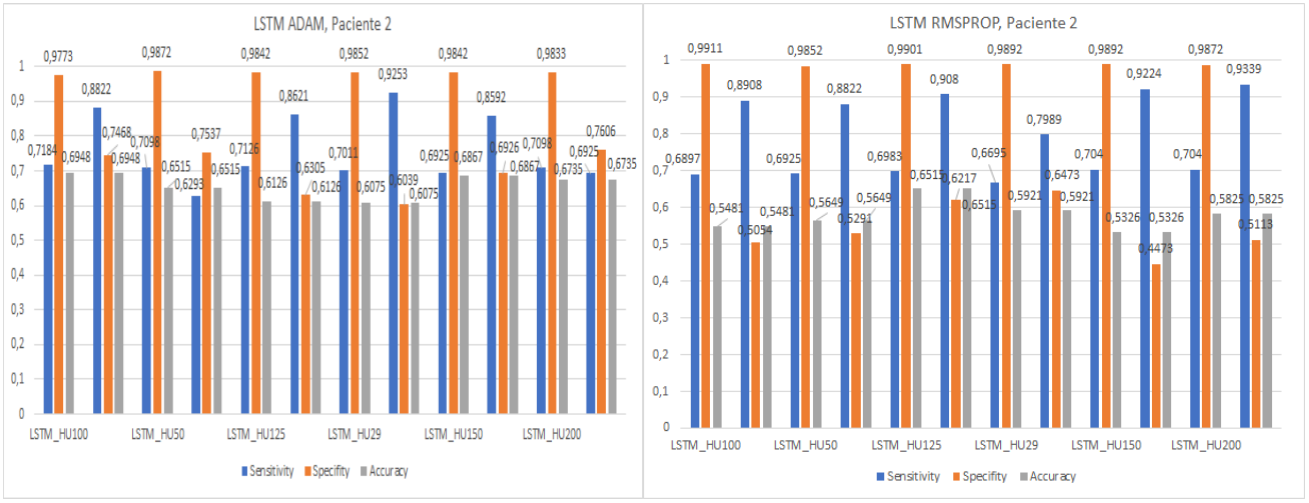


Figura 11: Resultados da arquitetura LSTM para o paciente 2 e com os treinos Rmsprop e Adam

Como podemos analisar, obtivemos melhores resultados para o paciente 1, principalmente, na *Detection*. Para o paciente 2 também obtivemos bons resultados para esta ação, no entanto, relativamente à *Prediction* os valores não foram tão constantes nem tão positivos. Por fim, comparando os dos treinos podemos verificar que o Adam foi aquele que nos deu melhores dados.

## 4. Autoencoder

Nesta secção do trabalho iremos apresentar a redução de *features* com os *autoencoders*. Iremos reduzir as 29 *features* a apenas 15, 10 e 3. Os testes realizados forma feitos com a rede LSTM com 100 *HiddenUnits*.

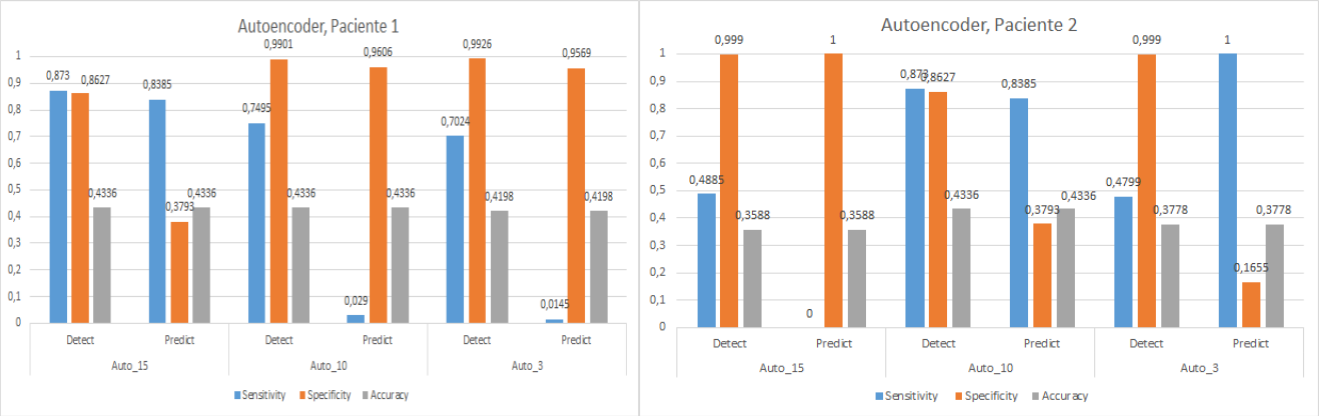


Figura 12- Gráficos com resultados da redução de *features* com os *autoencoders*

Como é possível observar pela figura 12, o primeiro paciente é o que tem melhores resultados. Para além disso, a deteção apresenta melhores resultados que a previsão. Paralelamente, quanto menos são as *features,* mais a rede perde a capacidade de classificação principalmente na previsão.

## 5. GUI

Com o objetivo de permitir a qualquer utilizador usar o nosso projeto e poder analisar o comportamento das nossas *neural networks*, desenvolvemos uma interface através da funcionalidade Matlab Design App, que permite ao *user* escolher qual paciente pretende analisar, qual arquitetura deseja analisar, se quer realizar o treino ou o teste e o treino e, por fim, são apresentados os valores da *specificty* e da *sensivity* para os pontos escolhidos por ele.

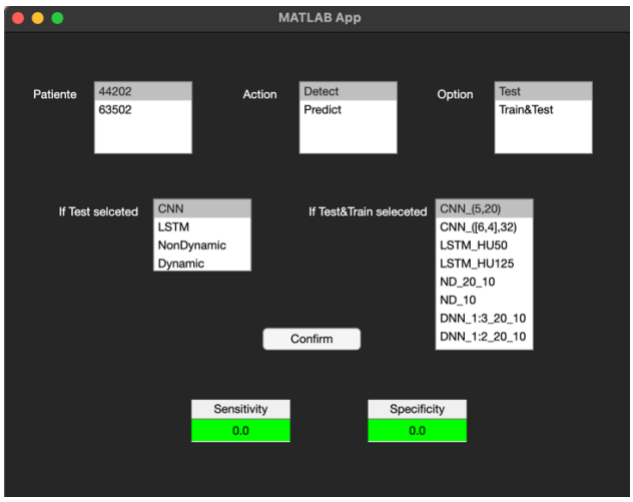


Fig 13- Interface

De modo a interagir com a interface pode escolher o dataset que pretende treinar e/ou testar (paciente um ou dois). De seguida, pode escolher se pretende visualizar os valores respetivos à deteção ou previsão de *seizures.* Para além disso, tem ainda a opção de querer apenas testar uma rede já treinada, ou treinar uma rede e testá-la. Ao escolher a opção de testar, irá escolher que rede pretende testar. Para selecionar a rede irá ter de usar as opões do lado esquerdo (CNN, LSTM, NonDynamic e Dynamic.). Caso pretende treinar e testar irá escolher dentro das opções do lado direito. Tendo em conta opção que escolha, treinar ou testar, as opções do lado esquerdo ou direito não terão efeito. As redes da opção ("If test selected") correspondem às seguintes newtorks:

- CNN
  - Paciente 1: CNN_net_SGDM_6x4_16
  - Paciente 2: CNN_net_SGDM_5_20
- LSTM
  - Paciente 1: LSTM_net_ADAM_HU100
  - Paciente 2: LSTM_net_rmsprop_HU125
- NonDynamic
  - Paciente 1: ND_20_10
  - Paciente 2: ND_10
- Dynamic
  - Paciente 1: DN_20_10
  - Paciente 2: DN_20_10

Para iniciar a interface basta escrever "app1" no terminal da diretoria "source" e, de seguida interagir fazendo as ações que pretende.

## 6. Conclusão

Em jeito de conclusão, como pode ser analisado ao longo de todo este projeto, consideramos que conseguimos obter valores bastante positivos e que correspondem com a expectativa. Tendo em conta o facto de ser muito difícil conseguir ou detetar um ataque de epilepsia com a total precisão, o facto de termos conseguido obter dados em que atingimos o valor máximo (1) leva-nos a considerar que explorámos e desenvolvemos bem o trabalho. Para além disso, no início do projeto percebemos que podia ser muito difícil obter valores bastantes bons nas métricas selecionadas para ambos os pacientes. No entanto, como se pode verificar pelos gráficos conseguimos obter dados bastante positivos, apesar de o paciente 1 ter sido mais constante e ter alcançado valores melhores. No que diz respeito às diferentes arquiteturas pudemos verificar que aquelas que conseguimos obter melhores resultados foram a CNN, a LSTM e a Dynamic, sendo que a Feed Forward foi a pior. Analisando estas individualmente também conseguimos analisar que dentro da Feed Forward NN conseguimos os melhores dados quando utilizámos três camadas e na ação de Detect. No que diz respeito à Dynamic, os valores mais positivos foram com o uso do delay 1:2 para duas ou três camadas e também na ação de Detect. Relativamente à CNN, foram durante o Detect e com o uso do treino Sigmoid e o uso de 32 filtros ambas 6x4 ou 32 filtros 5x5. Por fim, abordando a LSTM, podemos analisar que na Detection os resultados foram melhores e quando se utilizou o treino Adam com 200 hidden units (apesar de os valores serem todos bastante positivos).

## 7. Nota

Como só era permitido enviar ficheiros até 500MB, separámos em três entregas. De modo a juntar tudo, basta colocar a pasta "Paciente1" e "Paciente2" dentro da pasta "source" que se encontra dentro da pasta da submissão principal.

## 8. Referência

[1] https://stackoverflow.com/questions/42013665/predicting-time-series-y-t1-with-neural-networks-in-matlab