UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO
LABORATÓRIO DE PROCESSAMENTO DE SINAIS PARA COMUNICAÇÕES

"Interface Cérebro-Computador Baseada em P300"

RELATÓRIO FINAL DE ATIVIDADES

Voluntário: Lucas Costa e Lima (RA: 182411)
Orientador: Prof. Dr. Romis Attux.

Junho de 2020
Campinas, Brasil

1. Resumo das atividades

a. Descrição de uma Interface P300

Uma Interface Cérebro-Computador (*Brain-Computer Interface - BCI*) é um sistema para comunicação direta entre o cérebro e um computador [1]. A comunicação se estabelece através de comandos dados por padrões cerebrais capturados e processados através de algum paradigma específico e utilizados para a comunicação. O paradigma escolhido durante a Iniciação Científica em questão utiliza sinais cerebrais extraídos através de Eletroencefalografia (EEG), especificamente no Lobo Occipital, com amostragem temporal do sinal.

A interface escolhida para reprodução foi feita em 1988 por Farwell e Donchin [2], consistente de uma matriz 6x6 com letras a serem escolhidas uma a uma, após seguidos períodos de excitação aleatória de linhas e colunas da matriz (tendo em vista o ganho de tempo na seleção das letras).

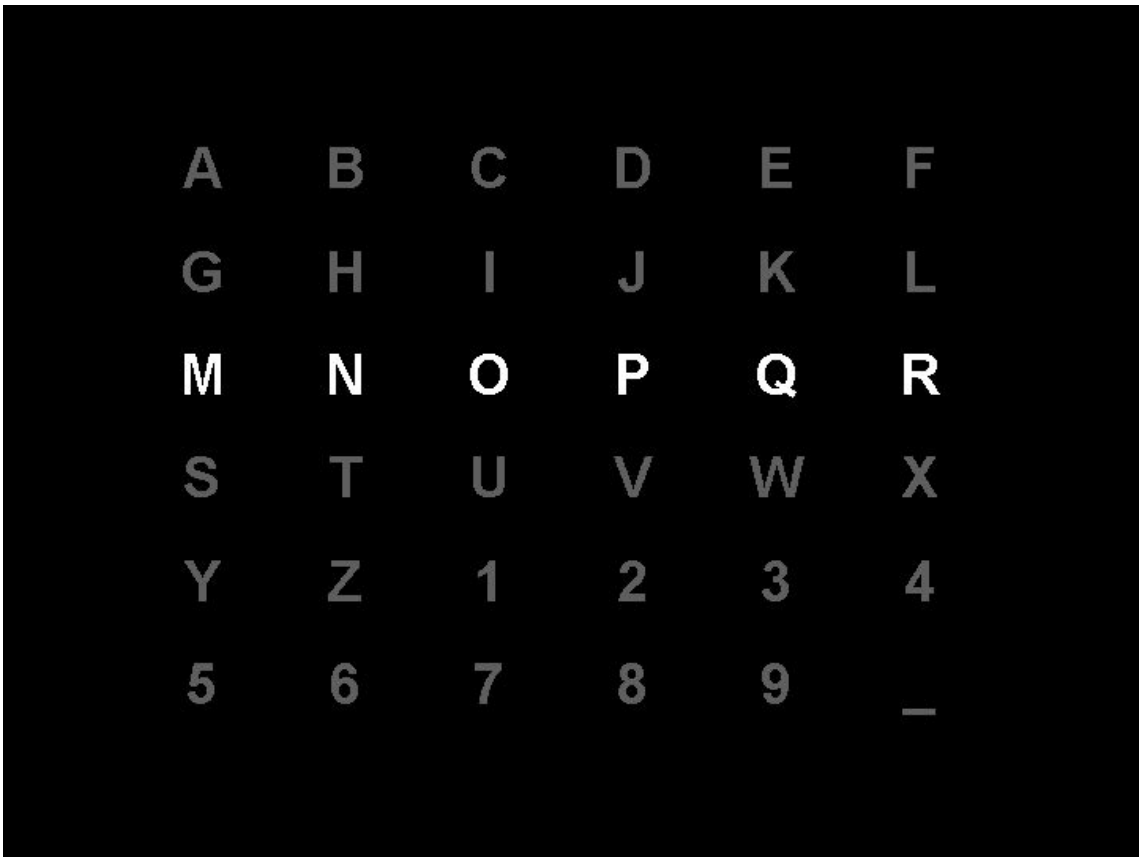


Figura 1: Representação gráfica da matriz a ser vista pelo sujeito.

O sinal a ser analisado na interface escolhida é o potencial P300. Tal potencial é evocado no cérebro cerca de 300ms após o seu estímulo. Este ocorre, na situação em questão, cada vez que a letra desejada é iluminada em sua linha ou coluna, então o estímulo visual gera o tal potencial, podendo ser classificado e utilizado para escrever.

Após a captura dos sinais para treino, utiliza-se um algoritmo de pré-processamento e subsequente classificação de sinais onde há ou não a presença de P300. Após um modelo de algoritmo treinado para o reconhecimento e classificação, a versão "*online*" pode ser utilizada, onde não mais se conhece a saída esperada para efetivar a comunicação.

b. *Dataset* utilizado e técnicas de classificação dos sinais

Devido a dificuldade da obtenção imediata dos dados e da inabilidade no manueseio da BCI disponível no laboratório, definiu-se que seria melhor utilizar um banco de dados disponível na internet para o desenvolvimento dos algoritmos de detecção dos sinais de P300. O banco de dados disponível em https://www.kaggle.com/rramele/p300samplingdataset se mostrou saftisfatoriamente explicado sobre a sua aquisição e amistoso ao trabalho.

Havia uma definição de utilizar o *software* MATLAB para o desenvolvimento dos algoritmos, uma vez que o OpenBCI [3] exporta dados na extensão .mat, arquivo nativo para MATLAB. Entretanto, durante o percurso da pesquisa, uma *framework open-source* para *Python* foi lançada para a integração do OpenBCI com algoritmos em *Python.* Desta forma, tanto pela praticidade e vastidão da comunidade *Python,* quanto pela habilidade do aluno em desenvolvimento de algoritmos neste *software*, adotar a mudança de *software* pareceu sensato para o orientador e para o aluno.

A abordagem com os dados se deu da seguinte forma:

i. Pré-processamento

Os dados já são filtrados utilizando um *notch* em 50Hz e um filtro passa-banda 0.1-30Hz. Entretanto, o sinal de P300 é muito menor, em amplitude, que a maior parte dos sinais basais cerebrais de maior frequência implementou-se uma média acerca dos estímulos de cada linha/coluna visando aumentar o sinal P300 e diminuir o ruído basal do cérebro. Além disso, um filtro *butterworth* passa-baixas foi implementado também na intenção de diminuir os ruídos de frequências que não interessam ao sinal em questão, como ondas gama.

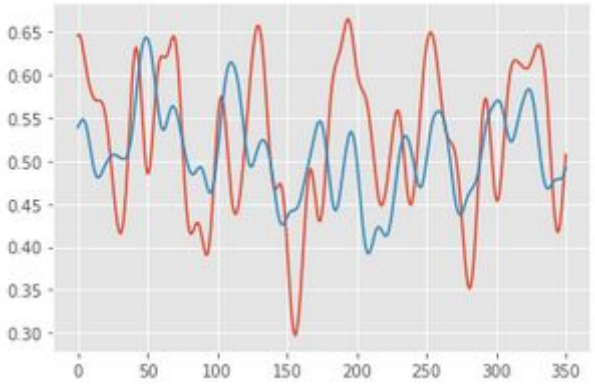


Figura 2: Em vermelho um sinal positivo pós-processamento, em azul um sinal negativo pós-processamento.

ii. Escolha do algoritmo inteligente

A primeira abordagem foi feita sem um algoritmo inteligente propriamente dito. Utilizou-se um filtro casado na expectativa de perceber a correlação entre os dados. Para tal, estabeleceu-se um sinal "ideal", feito das médias de todos os sinais positivos de P300 disponíveis entre os 7 sujeitos e, em seguida, o filtro era aplicado entre o sinal ideal e o sinal a ser avaliado. Este processo sofreu algumas alterações de abordagem, como um filtro para cada indivíduo ao invés de utilizar os dados de todos, uma vez que o tempo de resposta é particularmente diferente para cada um. Entretanto não houveram resultados satisfatórios na classificação dos sinais, assim levando a abordagem com algoritmos inteligentes. O aluno era previamente capacitado na utilização de redes neurais multicamada para classificação (MLP) por ter feito um curso com o orientador na pós-graduação como aluno especial, assim, a abordagem escolhida foi utilizar um MLP construído no *Keras*[4] para a classificação dos dados.

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_11 (Conv1D)           (None, 351, 4)            24
_________________________________________________________________
global_average_pooling1d_7 ( (None, 4)                 0
_________________________________________________________________
dense_10 (Dense)             (None, 2)                 10
_________________________________________________________________
activation_7 (Activation)    (None, 2)                 0
=================================================================
Total params: 34
Trainable params: 34
Non-trainable params: 0
_________________________________________________________________
```

Figura 3: Arquitetura minimalista da MLP.

A arquitetura escolhida variou no número de camadas lineares e convolucionais na tentativa de otimizar a acurácia do modelo, mas observou-se uma falha de evadir um mínimo local da função de custo. Por causa disso um modelo minimalista com apenas uma camada convolucional com máscara de tamanho 5 e uma camada

linear para a classificação foi adotado na tentativa de não recair no caso do mínimo local em questão, que será abordado no próximo tópico.

Foi decidido utilizar ao menos uma camada convolucional uma vez que o sinal possui características, ainda que pequenas, no seu espectro que diferenciam o P300 de um sinal não-P300. A partir disso, também foi decidido utilizar os dados de forma categórica para melhorar o desempenho de classificação. O otimizador da rede é o Adaptative Momentum (Adam)[5]. O Adam tem se mostrado o otimizador mais robusto para redes neurais e sua escolha é praticamente inquestionável na maioria das soluções, exceto algumas muito específicas.

iii. Problemas com desempenho na classificação e prováveis soluções

Durante o treinamento, observamos uma rápida evolução para uma acurácia de ~83% apesar da função de custo continuar abaixando. Vemos também que no conjunto de validação do treino, a acurácia é exatamente a mesma do conjunto de treino. Este fato curioso se repetiu em várias arquiteturas diversas e sob diferentes processos para processar os sinais.

```
Train on 180 samples, validate on 240 samples
Epoch 1/10
180/180 [==============================] - 2s 10ms/step - loss: 0.6600 - acc: 0.6111 - val_loss: 0.5619 - val_acc: 0.8333
Epoch 2/10
180/180 [==============================] - 0s 3ms/step - loss: 0.5087 - acc: 0.8333 - val_loss: 0.4720 - val_acc: 0.8333
Epoch 3/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4638 - acc: 0.8333 - val_loss: 0.4534 - val_acc: 0.8333
Epoch 4/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4567 - acc: 0.8333 - val_loss: 0.4514 - val_acc: 0.8333
Epoch 5/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4551 - acc: 0.8333 - val_loss: 0.4503 - val_acc: 0.8333
Epoch 6/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4561 - acc: 0.8333 - val_loss: 0.4504 - val_acc: 0.8333
Epoch 7/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4548 - acc: 0.8333 - val_loss: 0.4501 - val_acc: 0.8333
Epoch 8/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4553 - acc: 0.8333 - val_loss: 0.4501 - val_acc: 0.8333
Epoch 9/10
180/180 [==============================] - 0s 2ms/step - loss: 0.4548 - acc: 0.8333 - val_loss: 0.4501 - val_acc: 0.8333
Epoch 10/10
180/180 [==============================] - 0s 3ms/step - loss: 0.4544 - acc: 0.8333 - val_loss: 0.4501 - val_acc: 0.8333
```

Figura 4: Dados do treinamento da MLP minimalista.

O desempenho da rede, ao visualizar apenas a sua métrica, parece extremamente satisfatório (Fig. 4). A alta acurácia na classificação se mostra animadora até que observemos as saídas da classificação (Fig. 5).

```
[[0.83340925 0.16659082]
 [0.8451361  0.15486386]
 [0.84853256 0.1514675 ]
 [0.83173007 0.1682699 ]
 [0.8530711  0.1469289 ]
 [0.8512113  0.14878874]
 [0.8377869  0.16221303]
 [0.83303463 0.16696532]
 [0.8638939  0.13610606]
 [0.83126783 0.16873214]
 [0.83261764 0.16738236]
 [0.84050864 0.15949139]]
```

Figura 5: Saídas da rede diante de 12 entradas contendo 2 sinais positivos.

É perceptível o que acontece com o modelo. O mínimo local onde a rede classifica todos os sinais como não-P300 é de fácil acesso pelo otimizador, uma vez que % dos sinais são negativos para P300. Assim, o outro mínimo da função de custo, onde a rede seria capaz de classificar os dados como positivos e negativos para P300 está tão mais distante que ela sempre converge para o mesmo mínimo.

Prováveis soluções seriam:

- Otimizar os parâmetros do otimizador e a função de custo utilizando um algoritmo genético;
- Abordar o problema com aprendizagem não-supervisionada através de clusterização dos dados com número fixo de centróides;
- Redução de dimensionalidade pela energia dos dados ou pela importância das características (PCA ou ICA);

c. Abordagens adotadas para a resolução dos problemas de classificação

Após diversas tentativas de convergência da rede para apenas um indivíduo fracassarem sem a eliminação de dados, isto é, abortada a hipótese de poucos dados para a convergência da rede, a abordagem foi diferente.

Já é conhecido o efeito de *datasets* desbalanceados em problemas de

classificação, que era o caso, numa proporção de **5:1** dos dados negativos em comparação aos positivos. Uma estratégia possível seria fazer K-Folding com as 5 partes de sinais negativos e a parte positiva [6], entretanto, a hipótese foi descartada por não querer permitir um *overfitting* da rede ao indivíduo. Então, arbitrariamente eliminou-se 80% dos dados negativos de P300 intentando retirar o mínimo local da função de custo, o que foi completamente positivo e os resultados convergiram de imediato.

d. Comparação de estratégias de *Deep Learning*

Após a resolução do pré-processamento, as estratégias para a classificação de P300 mais usuais eram comitês de máquinas de vetores suporte (SVM) e redução de dimensionalidade (PCA ou ICA) aplicados aos eletrodos, na tentativa de selecionar os melhores canais para a classificação. Na tentativa de abordar o problema de uma outra perspectiva foram considerados:

i. Dependência espacial dos dados:

Cada eletrodo se localiza em uma parte diferente do crânio, permitindo inferir informação espacial sobre o sinal de P300, suscitando fortemente a hipótese de alimentar a rede como uma rede de classificação de imagens;

ii. Dependência temporal dos dados:

Em BCI é comum analisar o espectro de frequências do sinal, entretanto, parte relevante do sinal precisa estar no domínio do tempo;

iii. *Transfer Learning*

Um conceito aplicado para um ajuste fino de redes deve ser fortemente considerado aqui, uma vez que podemos deixar uma arquitetura pré-treinada em dados de vários sujeitos por um tempo maior e, para o sujeito final, e.g. um consumidor, o tempo de treino seria bem menor e a rede convergiria bem para mais pessoas.

1. Considerações sobre *Transfer Learning*

   Primeiro achou-se por ser de grande valia a rede ser treinada de indivíduo em indivíduo, para analisar como a função de custo convergia, mas percebeu-se por bem avaliá-la em vários indivíduos primeiro e aplicar o ajuste fino apenas no sujeito final;

e. Resultados das abordagens:

A seguir estão as matrizes de confusão de abordagens consideradas relevantes para a análise:

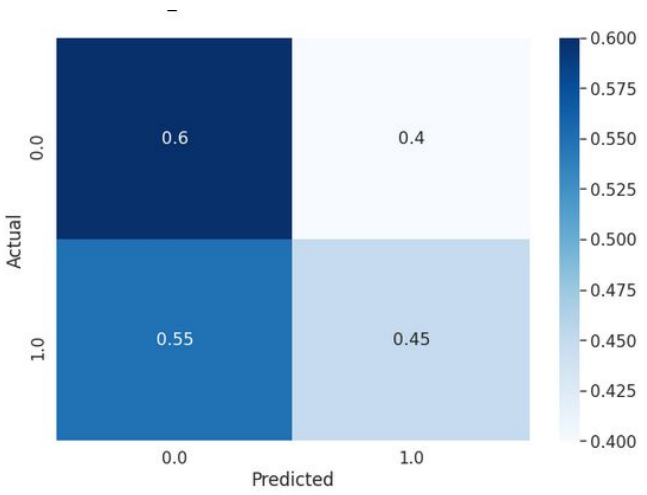


Figura 6: Matriz de Confusão - Modelo 1D com convoluções

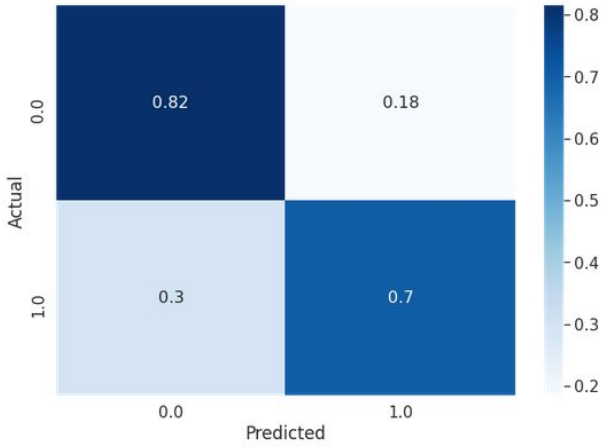


Figura 7: Matriz de confusão com *Transfer Learning* para sinais 2D

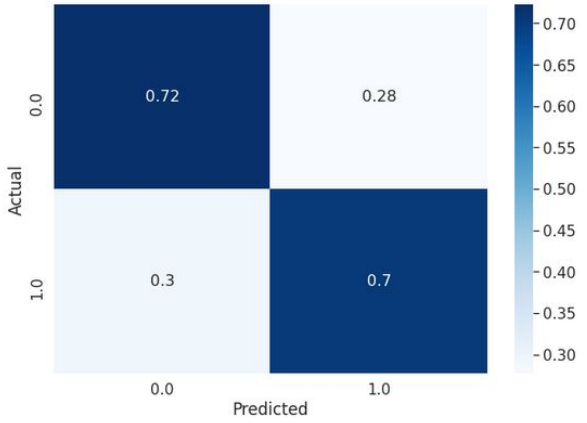


Figura 8: Matriz de Confusão com *Transfer Learning* apenas no último sujeito

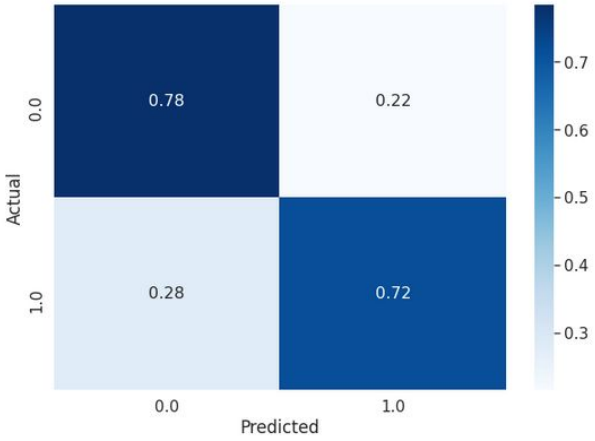


Figura 9: Melhor matriz de confusão apenas com ajuste fino para último indivíduo

Além dos resultados supracitados, houveram performances muito desbalanceadas quando:

- Aumentava-se o número de épocas de treino de forma desproporcional (*Overfitting*)
- Utilizava-se funções de ativação lineares nas camadas completamente conectadas no fim
- Inseriam-se camadas de normalização (*Batch Normalization*) ou redução de dados (*Dropout*)

A explicação para a queda de desempenho do *Overfitting,* principalmente em uma queda de desempenho nos dados que continham P300, provém da conformação do modelo às características basais dos sinais, comuns em ambos, que provavelmente reduziram a

sensibilidade do modelo ao sinal de P300. A queda de desempenho na utilização de camadas lineares (*ReLu*) na ativação no final da rede, após camadas convolucionais, escala as frequências de forma monotônica, assim, reduz o impacto das pequenas frequências, responsáveis pelo sinal de P300. A inserção de *Dropout* e *Batch Normalization* implicaram num empobrecimento de performance da rede uma vez que o *Dropout* age na intenção de fazer a rede evitar um *overfit* a dados muito iguais, o que é exatamente o contrário do que desejava-se, o *Batch Normalization,* pelo fato de operar nos *mini-batches* dos dados de treino, atuando em suas médias e no *convariance shift.* Isso deveria fazer a rede ser capaz de generalizar bem, classificar mais rápido (ganho de 4x no tempo de classificação: 4ms - 1ms) e suavizar a função de custo, o que de fato fez (Imagens abaixo), todavia, por visar a generalização através do deslocamento da covariância, o modelo se torna menos capaz de classificar bem os dados de um único sujeito.

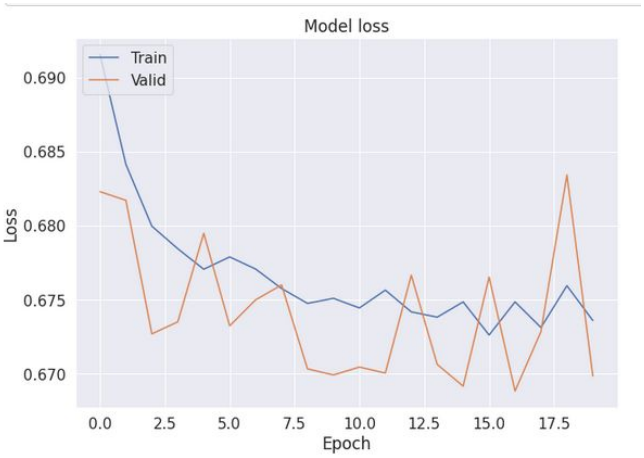


Figura 10: Função de custo no treino sem *Batch Normalization*

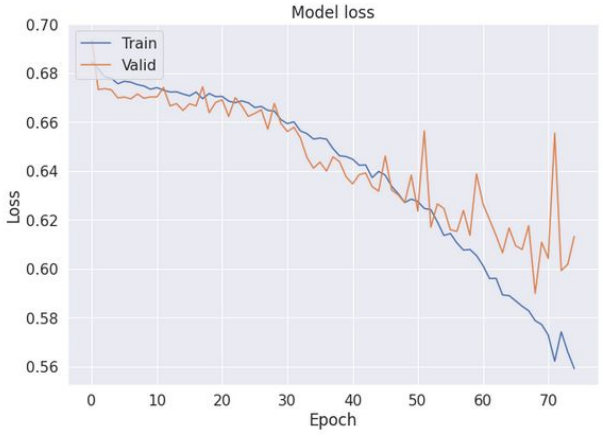
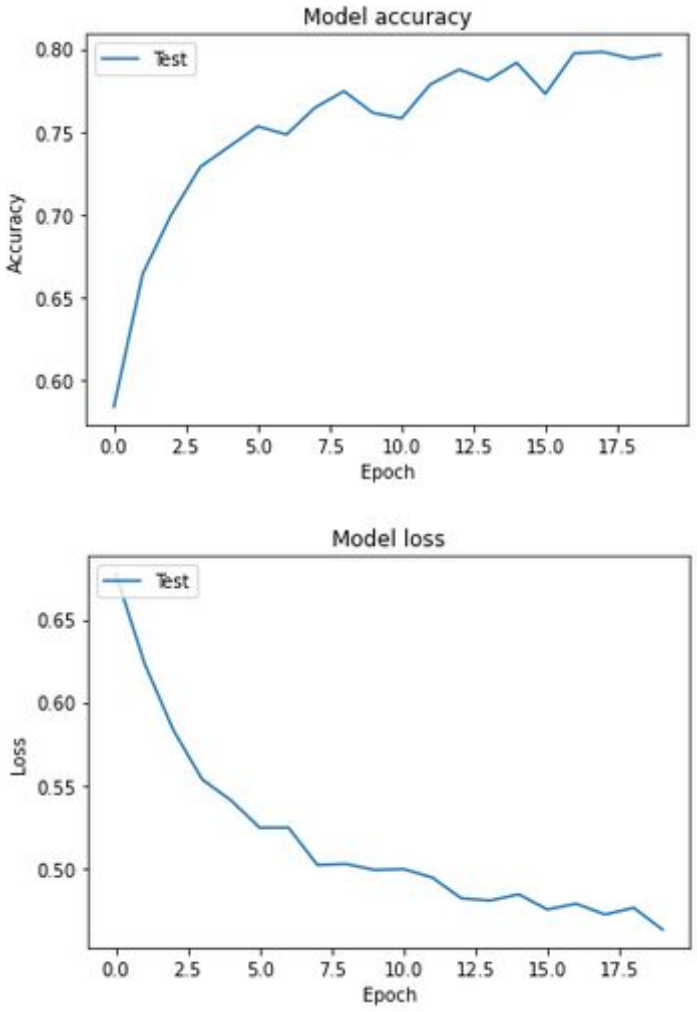


Figura 11: Função de custo durante o treino com *Batch Normalization*

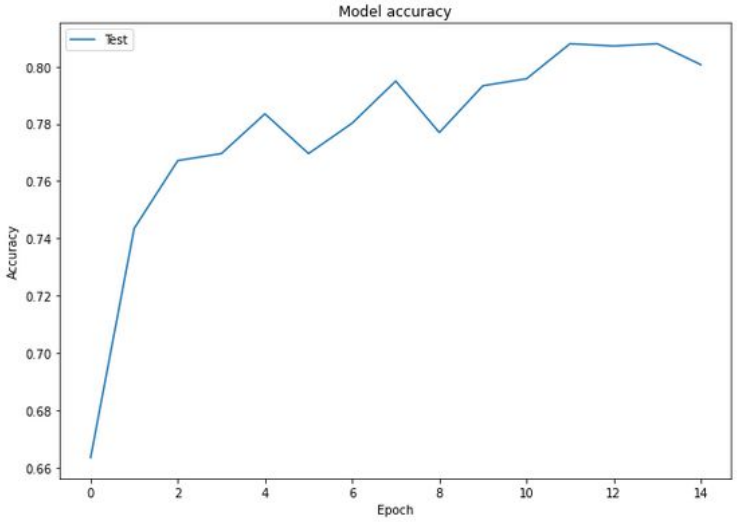
Uma última discussão válida é a aplicação do *Transfer Learning* na fase de treinos ou não. Como descrito em seções anteriores, duas abordagens foram feitas:

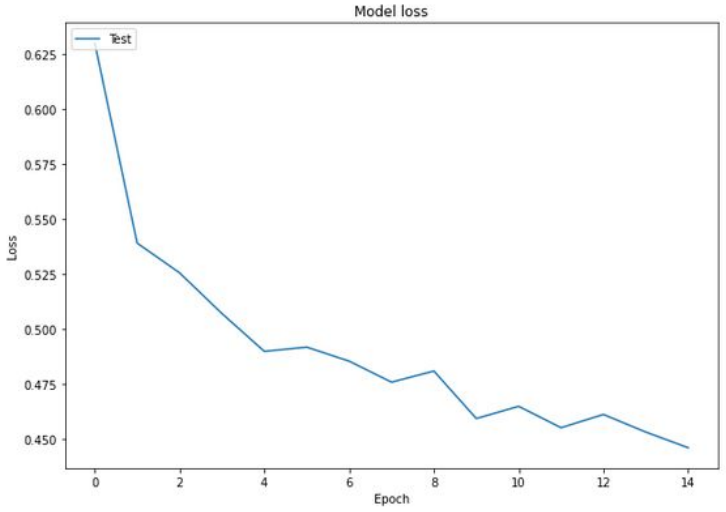
- 1 - *Transfer learning* de indivíduo para indivíduo durante o treino e de novo no ajuste fino;
- 2 - *Transfer learning* apenas no ajuste fino, misturando os dados dos indivíduos na fase de treino;

A seguir veremos que a diferença na velocidade de convergência no ajuste fino é mínima, respectivamente as situações 1 e 2:



Figuras 12 e 13: Acurácia e Custo para o Caso 1





Figuras 14 e 15: Acurácia e custo para o Caso 2

Sendo possível perceber que não há diferença prática de convergência no ajuste fino entre os casos.

2. Perspectivas técnicas e aplicações

Do ponto de vista técnico, este trabalho exerce uma função fundamental, aplicando conceitos atuais e completamente relevantes de inteligência artificial, se tratando de uma classificação utilizando Redes Neurais, Camadas Convolucionais, *Transfer Learning*, tratamento de dados e adequação de um modelo a partir da sua utilização.

Apesar de outros trabalhos partirem para uma perspectiva de que o modelo ideal numa BCI P300 é o que melhor generaliza os dados para quaisquer indivíduos, este trabalho mostra que uma rede que se adequa, a partir de um treinamento prévio relativamente generalista, aos padrões

cerebrais de uma pessoa pode performar muito melhor e em um tempo de treinamento ainda menor.

Tal aspecto é relevante do ponto de vista comercial, uma vez que a mesma arquitetura pré-treinada poderia ser aplicada para a venda em larga escala de uma interface que seria treinada ao sujeito usuário em poucos passos e logo alcançaria performances aceitáveis.

Além disso, a arquitetura foi treinada nos sinais de P300 sem médias, ou seja, a acurácia média de 75% é para cada sinal individualmente, implicando que, numa BCI comum, onde faz-se várias épocas para a seleção de um alvo do desejo do usuário, a acurácia será ainda maior. Além disso, o tempo atingido de classificação girou em torno de 1ms para cada amostra de dados, o que implica numa classificação extremamente confortável para o usuário.

Referências

1 - H. Serby , E. Yom-Tov - "An Improved P300-Based Brain-Computer Interface"

2 - L. A. Farwell and E. Donchin, "Talking off the top of your head: A mental prosthesis utilizing event-related brain potentials," Electroencephalogr. Clin. Neurophysiol., vol. 70, pp. 510–523, 1988.

3 - https://openbci.com/

4 - https://keras.io/

5 - https://arxiv.org/pdf/1412.6980.pdf