RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

UTILIZAÇÃO DE REDES NEURAIS PARA VISUALIZAR E INTERPRETAR DADOS DE ELETROENCEFALOGRAMAS E ELETROOCULOGRAMAS

USING NEURAL NETWORKS TO VISUALIZE AND INTERPRET ELECTRONENCEPHALOGRAM AND ELECTROOCULOGRAM DATA

UTILIZACIÓN DE REDES NEURONALES PARA VISUALIZAR E INTERPRETAR DATOS DE ELECTROOCULOGRAMA Y ELECTROENCEFALOGRAMA

Matheus de Souza Perches[1], João Henrique Gião Borges[2], Fabiana Florian[3]

e361578
https://doi.org/10.47820/recima21.v3i6.1578

PUBLICADO: 06/2022

RESUMO

As interfaces cérebro máquina tem como objetivo integrar os humanos com as máquinas de forma mais direta e íntima. Este trabalho tem como objetivo demonstrar que a leitura de ondas cerebrais é possível através da comunicação entre a linguagem *Python* e o *hardware* responsável por obter os dados. O projeto aborda sobre o *hardware* das interfaces cérebro computador relevantes para os fins deste trabalho, dando um conhecimento de alto nível ao leitor sobre do que se trata a tecnologia e como ela funciona, bem como citando algumas aplicações, e então volta sua atenção para o lado do *software*, demonstrando como é possível aplicar o conhecimento com o suporte de tecnologias como o Google Collab para a hospedagem do projeto, a biblioteca MNE-Python para obtenção de leituras confiáveis, o Scikit-learn para realizar as computações de *machine learning*, entre outras ferramentas.

PALAVRAS-CHAVE: Interfaces Cérebro Máquina. Interfaces Cérebro Computador. Eletroencefalografia. MNE Python. Scikit-learn. SciPy. Eletrooculogramas.

ABSTRACT

Brain-machine interfaces aim to integrate humans with machines in a more direct and intimate way. This work aims to demonstrate that the reading of brain waves is possible through communication between the Python language and the hardware responsible for obtaining the data. The project addresses the hardware of the relevant brain computer interfaces, giving the reader a high level understanding of what the technology is and how it works, as well as citing applications, and then turns its attention to the software side, demonstrating how we can apply this knowledge with the support of technologies such as Google Collab for the project hosting, the MNE-Python library for reliable readings Scikit-learn to perform machine learning computations, and other tools.

KEYWORDS: *Brain Machine Interfaces. Brain Computer Interfaces. Electroencephalogram. MNE Python. Scikit-learn. SciPy. Electrooculogram.*

RESUMEN

Las interfaces cerebro-máquina tienen como objetivo integrar a los humanos con las máquinas de manera más directa e íntima. Este trabajo tiene como objetivo demostrar que la lectura de ondas cerebrales es posible a través de la comunicación entre el lenguaje Python y el hardware responsable de la obtención de los datos. El proyecto aborda el hardware de las interfaces cerebrales de la computadora relevantes para los propósitos de este trabajo, brindando un conocimiento de alto nivel al lector sobre de qué se trata la tecnología y cómo funciona, además de citar algunas aplicaciones, y luego dirige su atención al lado del software, demostrando cómo es posible aplicar el conocimiento con el apoyo de tecnologías como Google Collab para el alojamiento de proyectos, la biblioteca MNE-

[1] Universidade de Araraquara - UNIARA
[2] Universidade de Araraquara - UNIARA
[3] Universidade de Araraquara - UNIARA

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Python para lecturas confiables, Scikit-learn para realizar cálculos de aprendizaje automático, entre otras herramientas.

PALABRAS CLAVE: *Interfaces Cerebro-Máquina. Interfaces cerebro-computadora. Electroencefalograma. MNE Python. Scikit-learn. SciPy. Electrooculograma.*

INTRODUÇÃO

Azar e Hassanien (2014, p. 4) mencionam que desde o aparecimento dos primeiros computadores, os UNIVAC em 1951, a interação entre humanos e computadores tem se tornado cada vez mais fluida. Os computadores modernos podem ser móveis, como é o caso dos *notebooks, smartphones* e *tablets.* Através de um *smartphone,* por exemplo, podemos obter qualquer tipo de informação disponível na internet em qualquer lugar, desde que uma conexão via satélite esteja disponível.

Para Wolpaw E e Wolpaw J (2012, p. 3) Uma Interface Cérebro Computador (BCI, do inglês *Brain Computer Interfaces*) é um sistema que mede a atividade do sistema nervoso central (SNC), obtendo os impulsos como entrada de dados, e convertendo-os para uma saída artificial, substituindo, restaurando, realçando, suplementando ou aprimorando o SNC natural, por sua vez alterando a relação entre ele e o ambiente interno ou externo.

As aplicações médicas são um dos destaques da tecnologia. Wolpaw E e Wolpaw J (2012, p. 3) dissertam que uma pessoa que sofreu lesão na medula espinhal causando a paralisação dos movimentos das mãos e pernas, poderia utilizar um sistema de BCI para restaurar essas habilidades motoras, estimulando os músculos paralisados através de eletrodos, assim recuperando os movimentos.

Uma pessoa que perdeu o controle de sua bexiga por conta de esclerose múltipla poderia ter um sistema de BCI implementado que estimularia os nervos periféricos, permitindo que ela voltasse a urinar.

É mencionado também que um sistema de BCI também pode ser utilizado para aprimorar a saída natural do SNC, em aplicações em que é necessário atenção contínua, como dirigir. O sistema detectaria a atividade cerebral que precede um lapso de atenção, e então fornece algum tipo de estímulo (um som, por exemplo), retornando a pessoa para o estado de foco.

Wolpaw E e Wolpaw J (2012, p. 3) também citam que a recente explosão de interesse para pesquisas em Interfaces Cérebro Máquina faz parte de um surto em um espectro amplo de novas tecnologias e terapias que prometem um entendimento sem precedentes sobre o funcionamento do cérebro e como tratar suas doenças.

Por conta da complexidade multidisciplinar de um sistema de interface cérebro-máquina, este artigo pretende permanecer no escopo da camada de *software*. Portanto, para este estudo de caso foi introduzido um problema simplificado e uma rede neural simples para resolvê-lo. A biblioteca MNE-

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Python trará o conjunto de dados necessário, que contém 226 pontos de dados por amostra de EEG, onde cada amostra representa uma única tentativa descrita no problema.

Supondo que um paciente sentado à frente de uma tela seja apresentado a imagens sendo alternadas entre padrões de tabuleiro preto e branco, entre os olhos esquerdo e direito em um intervalo de 750 ms (milissegundos) entre os estímulos, tenha de forma aleatória um *emoji* sorridente ao centro do campo visual sendo exibido. Ao avistá-lo, o paciente deveria apertar um botão. O mesmo experimento poderia ser realizado com um padrão de cores, onde seria possível pedir para que o paciente pressionasse o botão assim que avistasse uma cor específica, ou a ausência dela.

Um experimento como este testaria a habilidade cognitiva e motora do paciente entre entender as imagens e apertar um botão, porém o objetivo dessa implementação de BCI é remover a entrada manual do usuário (apertar o botão), e deixar com que o sistema de rede neural reconheça que o usuário avistou a imagem que foi pedida, e registrar o evento de acordo. Através de dados EEG previamente coletados, podemos treinar uma rede neural para que entenda as amostras apresentadas e realize asserções corretas.

O objetivo é monitorar a resposta “surpresa” ou “antecipação” do paciente. Este tipo de sinal pode ser monitorado por meio do componente P300, uma resposta EEG já muito bem conhecida. Um sistema que detecta este componente por dentro dos dados EEG foi criado, sendo encontrada a solução para o problema proposto.

1 COMUNICANDO COM O CÉREBRO

Para Graimman, Allison e Pfurtscheller (2010, p. 7), o objetivo de uma BCI é medir a atividade cerebral, que produz sinais elétricos e atividade magnética, com a finalidade de entender a intenção do usuário. Portanto, diferentes sensores podem coletar diferentes mudanças elétricas ou magnéticas no cérebro, de acordo com o posicionamento desses instrumentos.

2. MÉTODOS DE COMUNICAÇÃO

Azar e Hassanien (2014, p. 35) citam que para entender de uma maneira geral como as BCI funcionam, é necessário entender como a atividade cerebral pode ser medida e quais sinais cerebrais podem ser utilizados. Existem métodos invasivos, parcialmente invasivos e não invasivos, pois a classificação depende da maneira que os sinais elétricos são obtidos dos neurônios no cérebro humano. A seguir, os principais tipos de BCI serão abordados.

2.1. INTERFACES CÉREBRO MÁQUINA INVASIVAS E PARCIALMENTE INVASIVAS

Os métodos invasivos normalmente permitem melhor precisão nas informações por conter menor quantidade de ruído (similar ao chiado em rádios) que atrapalham na leitura, consequentemente não necessitando tanto de processamento. Por outro lado, por serem invasivos apresentam um maior risco ao paciente e maior custo, por se tratar de um processo cirúrgico. Por exemplo, a taxa de disparo

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

de neurônios individuais é uma característica importante de BCIs invasivas, que utilizam gravações intracorticais (AZAR; HASSANIEN, 2014, p. 14).

2.1.1. ELETROCORTICOGRAFIA

A eletrocorticografia (ECoG, do inglês *electrocorticography*) é uma técnica de gravação de sinais cerebrais que envolve a colocação de eletrodos através da superfície do cérebro (massa encefálica). O procedimento requer uma incisão cirúrgica no crânio para o implante dos eletrodos na superfície do cérebro. O ECoG é tipicamente performado somente em pacientes em situações clínicas, como dentro de um hospital para realizar o monitoramento em pacientes que sofrem de epilepsia (RAO, 2013, p. 22).

Considerada parcialmente invasiva, a ECoG recebeu atenção da comunidade de BCI como um compromisso intermediário entre uma matriz de eletrodo invasiva e a não invasiva eletroencefalografia. Ao contrário das matrizes de multieletrodo, algumas formas de ECoG não penetram a barreira cérebro-sangue, portanto são mais seguras do que matrizes implantadas dentro do cérebro. Os eletrodos ECoG também são menos propensos a desgaste comparado a soluções que penetram o cérebro, por acumulação de células de glia e formação de tecido cicatricial (RAO, 2013, p. 23).

Porém ela não é perfeita, já que só pode ser utilizada em situações cirúrgicas, onde o usuário se encontraria no hospital, aumentando o risco. Além disso, somente as partes cirurgicamente relevantes do cérebro poderiam ter dados gravados, e ainda pode haver interferências causadas por medicamentos, ou condições próprias do paciente como convulsões (RAO, 2013, p. 24).

Outra desvantagem consiste no tamanho dos eletrodos, que é relativamente grande, segundo o autor (alguns mm em diâmetro). Esse problema em particular está sendo trabalhado por pesquisadores, utilizando eletrodos do tipo *micro ECoG.* Estes teriam somente uma fração do tamanho dos eletrodos comuns, diminuindo o espaçamento entre eletrodos necessários, aprimorando a detecção de atividade neural por conta da maior resolução. Isso possibilitaria a gravação de movimentos sutis, como o movimento de dedos individuais ou até mesmo a fala, sem realmente precisar penetrar ao cérebro (RAO, 2013, p. 24).

2.1.2 INTERFACES CÉREBRO MÁQUINA NÃO INVASIVAS

Os métodos não invasivos proporcionam baixo ou zero risco ao paciente que recebe o dispositivo, além de serem removíveis com uma facilidade muito maior e a qualquer momento, como é o caso da eletroencefalografia.

2.1.2.1 ELETROENCEFALOGRAFIA

Rao (2013, p. 28) diz que a Eletroencefalografia (EEG) é um bom método para se obter gravações de ondas de atividade cerebral, através de uma variedade de frequências. Determinadas ações possuem determinadas frequências, e são originadas a partir da sincronização de uma grande

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

população de neurônios, tendo também uma distribuição espacial característica, que estão frequentemente correlacionadas com diferentes estados funcionais do cérebro.

Por exemplo, atividades como mover os olhos, piscar, ou mover algum músculo do corpo causam registros nas gravações, tornando a filtragem desses dados extremamente necessária. Até mesmo ruído de outros equipamentos elétricos ou uma linha de energia ativa podem ser registradas erroneamente no relatório, segundo Graimman, Allison e Pfurtscheller (2010, p. 6).

Para Graimman, Allison e Pfurtscheller (2010, p. 7) as ondas Alfa (8-13 Hz) são registradas quando um paciente está acordado e relaxado, ou com os olhos fechados, por exemplo. Um tipo particular de onda Alfa é o ritmo mu (8-12 Hz), e é encontrada ao redor de áreas sensório-motores na ausência de movimento, e diminui conforme o sujeito realiza um movimento ou se imagina realizando. Ondas Beta (13-30 Hz) são detectáveis sobre os lobos parietal e frontal em pessoas em estado de alerta ou ativamente em processo de concentração. Ondas Delta (0.5-4 Hz) são detectadas em bebês e em adultos durante sono profundo. Ondas Theta (4-8 Hz) são associadas com sonolência ou ociosidade em crianças e adultos. As ondas Gama (30-100 Hz) são relacionadas com memórias de curto prazo e integração multisensorial. Altas atividades Gama (mais de 70 Hz) estão relacionadas com atividades motoras e são utilizadas em BCIs de ECoG.

Além disso, Graimman, Allison e Pfurtscheller (2010, p. 7) explicam que apesar do EEG não ser tecnicamente exigente, o procedimento geral de preparação pode ser complicado. Para obter a qualidade ideal de sinal, as áreas da pele que receberão o contato com os eletrodos precisam ser preparadas com um gel abrasivo especial. Segundo o autor, a quantidade média requerida de eletrodos é por volta de 100, e a maioria das pessoas que utilizam esse equipamento tenta minimizar esse número para reduzir o tempo de preparação e dificuldade.

2.2 O PROCESSAMENTO DOS SINAIS

O processamento de uma BCI é feito em tempo real para detectar padrões, entendendo a intenção do usuário. Ele possui quatro estágios, constituídos de pré-processamento, extração de características, seleção de características e classificação (LEITE, 2016, p. 9).

2.2.1 PRÉ-PROCESSAMENTO

O objetivo do pré-processamento é simplificar as operações subsequentes de processamento, sem perder as informações relevantes. Uma de suas tarefas importantes é aprimorar a taxa de sinal para ruído (SNR em inglês). Uma SNR ruim ou baixa significa que os padrões cerebrais obtidos estariam soterrados por baixo desse ruído, tornando os padrões relevantes para a análise feita difíceis de detectar (AZAR; HASSANIEN, 2014, p. 15).

Por outro lado, uma boa ou alta SNR implicaria em um sinal limpo, simplificando a tarefa de detecção e classificação. Técnicas de transformação combinadas com filtragem são empregadas de forma frequente durante o pré-processamento em uma BCI. Além disso, os cientistas utilizam dessas

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

técnicas para transformar os sinais, de forma que os componentes indesejados dos sinais sejam eliminados ou reduzidos, dessa forma aprimorando a SNR (AZAR; HASSANIEN, 2014, p. 15).

2.2.2 EXTRAÇÃO DE CARACTERÍSTICAS

Azar e Hassanien (2014, p. 14) dizem que os padrões cerebrais utilizados em BCIs são caracterizados por certas características ou propriedades. Amplitudes e frequências são características essenciais dos ritmos sensório-motor e potenciais evocados visualmente em estado estacionário (SSVEP, do inglês *Steady-State Visually Evoked Potential*) onde estes são utilizados em sistemas de eletroencefalografia (EEG)

A definição de uma forma de representação dos dados de entrada é um dos principais desafios no projeto de um dispositivo de processamento. Para Leite (2016, p. 51) “Geralmente, usar o sinal de entrada diretamente para se obter os valores de saída é uma abordagem de baixo desempenho e alto custo computacional.” Além disso, “para a maioria das aplicações é conveniente mapear o sinal de entrada em outro espaço que permita a representação das informações relevantes de forma compacta e eficiente.”

Encontrar uma boa representação de dados em um domínio específico não é simples, pois é necessário reduzir o espaço de características para evitar que os dados sejam pouco representativos para a quantidade de material disponível, enquanto o descarte de informações é feito com cautela para não se perder informações úteis (LEITE, 2016, p. 51).

2.2.3 SELEÇÃO DE CARACTERÍSTICAS

Os problemas de aprendizado de máquina costumam envolvem muitas variáveis para descrever um determinado conjunto. Segundo Leite (2016, p. 65), “A etapa de seleção de características visa determinar as características mais informativas a serem usadas para se conceber o sistema de classificação.” Existe um seletor de atributos, onde três objetivos são determinados: melhorar o desempenho do classificador, proporcionar preditores mais rápidos e mais eficazes e permitir uma melhor compreensão do processo que gerou os dados.

2.2.4 CLASSIFICAÇÃO

Leite (2016, p. 72) explica que a função que mapeia os dados de entrada nas possíveis classes considerando a similaridade entre os objetos, pode ser vista como a tarefa de classificação.
Seguindo nessa linha de raciocínio, podemos construir uma função de mapeamento utilizando características extraídas da base de dados, dessa forma cada dado de entrada corresponderia a uma única classe. “O espaço de entradas é dividido em regiões de decisão, cujos limites são denominados de fronteiras ou superfícies de decisão”. A autora complementa, e diz que quando as regiões de decisão são definidas como hiperplanos, são classificadas como linearmente separáveis, e caso contrário, são um problema de classificação não linear.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

A etapa de detecção e classificação também pode ser simplificada quando o usuário se comunica com a BCI somente em intervalos regulares de tempo bem definidos. Tais intervalos são indicados pela BCI através de toques sonoros ou indicações visuais, como uma luz de led. Por exemplo, um toque sonoro indicaria que o usuário deve realizar uma ação mental específica, e a gravação duraria determinado tempo. Outro toque sonoro indicaria que a gravação terminou, então o usuário pode cessar a sua atividade, e a BCI tenta classificar os sinais cerebrais recebidos durante esse intervalo de tempo (AZAR; HASSANIEN, 2014, p.14).

Todavia, Azar e Hassanien (2014, p. 14) explicam que este método de detecção não considera que o usuário pode vir a querer se comunicar antes ou depois desse intervalo de tempo, ou até mesmo não querer se comunicar durante o intervalo de tempo. BCIs que utilizam esse modo de operação são chamadas de BCI síncronas ou BCI passo a passo. Apesar de serem relativamente fáceis de desenvolver, seu uso no mundo real não é prático, sendo similar a um teclado que só pudesse ser utilizado para digitação durante certos períodos.

O inverso das BCIs síncronas são as BCIs assíncronas, realizando uma gravação contínua das intenções do usuário. Dessa forma, eles podem interagir com o sistema a qualquer momento, oferecendo uma forma mais natural e conveniente de interação, e são tecnicamente mais exigentes (AZAR; HASSANIEN, 2014, p.14).

2.3 MEDINDO A PERFORMANCE DE UMA INTERFACE CÉREBRO COMPUTADOR

A performance de um sistema de BCI pode ser medida de várias maneiras. Uma delas é a performance de classificação (podendo entrar como taxa de classificação ou precisão da classificação). É a frequência onde é contabilizado o número de classificações realizadas corretamente (tentativas de realizar atividades mentais que obtiveram êxito) versus o número total de classificações. Dessa forma, a taxa de erro torna-se fácil de calcular, já que contabiliza os números de classificações realizados incorretamente, versus o número total de classificações (AZAR; HASSANIEN, 2014, p. 15).

Essa medida de performance é válida, porém não permite que o sistema seja avaliado de forma objetiva no que diz respeito ao seu propósito real. Por exemplo, Azar e Hassanien (2014, p.15) citam que para uma aplicação de digitação mental, seria muito mais útil uma medida que obtivesse a taxa de palavras que o sistema conseguiu entender corretamente versus as palavras que não foram entendidas. A taxa de palavras entendidas poderia ser calculada, mas seria mais adequado saber quantas dessas palavras foram acertadas e quantas não.

Uma medida de *performance* mais geral é a taxa de transferência de informação (ITR, do inglês *Information Transfer Rate*). Ela depende do número dos diferentes padrões cerebrais utilizados, o tempo levado para a BCI classificá-los, e a precisão dessa classificação. Por conta do IRT ser dependente do número de padrões cerebrais que podemos ser confiavelmente detectados e classificados, a taxa de transferência depende da estratégia mental empregada (AZAR; HASSANIEN, 2014, p. 15).

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

De qualquer forma, a BCI implementa um caminho de comunicação alternativo, mas esse caminho é lento, dizem Azar e Hassanien. De acordo com ele, certamente não é uma via de interação de alta velocidade, e não pode ser comparado com as vias de comunicação naturais (como a fala ou escrita) ou interfaces humano-computador, como os celulares ou interfaces gráficas de aplicativos que estamos acostumados. Porém, as BCIs podem ser extremamente valiosas para pessoas com deficiências severas, com algumas aplicações emergentes para pessoas com deficiências menos severas, ou completamente saudáveis, complementa o autor.

3 METODOLOGIA DE PESQUISA

Para a metodologia de pesquisa, foram utilizadas bibliotecas gratuitas e bases de dados de eletroencefalograma (EEG) já existentes, como a biblioteca Python MNE que possui um número de amostras EEG com 226 pontos de dados por amostra. Isso permite que os resultados possam ser obtidos sem a necessidade de adquirir um equipamento, evitando eventuais despesas. O projeto nomeado “Interpretador de P300” para facilitar futuras menções e problema a ser apresentado a seguir foram baseados no exemplo fornecido pela Neurotech Edu.

Os pacotes de pré-requisito estão listados abaixo:

- Python: Linguagem de programação;
- MNE: Pacote de dados para EEG;
- NumPy: Computação Científica;
- SciPy: Computação Científica;
- MatPlotLib: Biblioteca de plotagem (geração dos gráficos);
- Scikit-Learn: Biblioteca para aprendizado de máquina
- PyTorch: Biblioteca para aprendizado de máquina

O projeto incluindo código fonte está publicamente acessível na plataforma Google Collab, permitindo que partes interessadas possam interagir e verificar os resultados por si mesmas.

3.1 O COMPONENTE P300

O componente P300 é uma grande forma de onda positiva, um pico, que pode ser extraída das medições em tempo real de um eletroencefalograma. Dessa forma, é possível utilizar dos conceitos do componente P300 para identificar picos de ondas cerebrais gerados pelo paciente quando observar a imagem que lhe foi indicada (DINTEREN *et al.*, 2014, p. 1).

3.1.1 O APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina é um termo que abrange uma grande variedade de algoritmos e ferramentas de modelagem para uma vasta gama de tarefas de processamento de dados. Para o problema apresentado, é importante saber que máquinas são muito boas em seguir parâmetros para

detectar padrões. Assim sendo, o algoritmo de aprendizado de máquina terá a tarefa de identificar e classificar esses padrões que não seriam detectáveis por humanos (GIUSEPPE *et al*, 2019, p. 1).

O autor Brandon Siebert explica que não há uma equação perfeita para o componente P300 para o interpretador desenvolvido. Além disso, esses eventos se manifestam de maneira diferente a depender de cada paciente e de onde os dados de EEG são obtidos. Portanto, é necessário um algoritmo capaz de se adaptar a tais situações, e identificar o componente P300 de forma correta. Dessa forma, o aprendizado de máquina será utilizado para se adaptar com as leituras e retornar os resultados corretos.

4. O INTERPRETADOR DE P300

O problema apresentado na introdução foi resolvido abaixo utilizando as ferramentas mencionadas na seção número 3. Vale reforçar que o código-fonte não é de autoria própria, sendo desenvolvido por Brandon Siebert e referenciado na seção Referências. Além disso, foi realizada uma alteração pontual explicada na seção 4.3 para que o treinamento da rede neural fosse realizado da maneira em que o autor original pretendia, produzindo os resultados esperados.

4.1 IMPORTAÇÃO DAS BIBLIOTECAS E CONFIGURAÇÃO DE AMBIENTE

A execução do projeto foi iniciada com a instalação dos módulos necessários, na máquina virtual do Google Collab:

```
[1]  1 # Run these from the console if following along locally
     2 !pip install mne
     3 !pip install sklearn
     4 !pip install matplotlib
     5 !pip install torch
     6 !pip install torchvision
     7 !pip install tensorboardX
```

Figura 1: Importação das bibliotecas. Fonte: Google Collab.

Na figura 2 é possível visualizar a saída do código, registrado quais requisitos já estavam satisfeitos e quais necessitavam algum tipo de alteração.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Figura 2: Saída de execução da instalação das bibliotecas. Fonte: Google Collab.

Em seguida, o autor explica que os avisos foram desativados para manter a limpeza da saída do código, já que são redundantes no contexto atual. Na linha número 2, a importação da biblioteca *warnings* é realizada, e na linha número 3 o módulo recebe o parâmetro para ignorar todos dentro de seu filtro. A instrução não possui retorno.

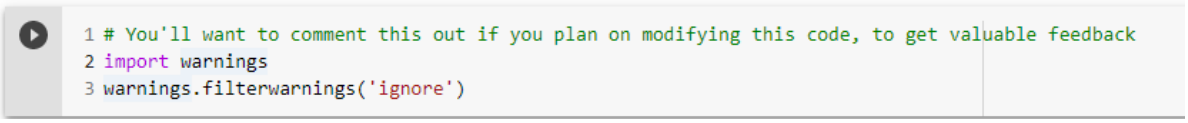

Figura 3: Desativação dos avisos. Fonte: Google Collab.

Na figura 4 são realizadas as importações de bibliotecas que foram utilizadas, após a instalação ter sido realizada na Figura 1. O MNE-Python que foi responsável por obter leituras de sistemas EEG está localizado na linha número 8 e o objeto *RobustScaler* da biblioteca SciKit está na linha número 9, sendo responsável por escalar as estatísticas geradas, de acordo com o autor.

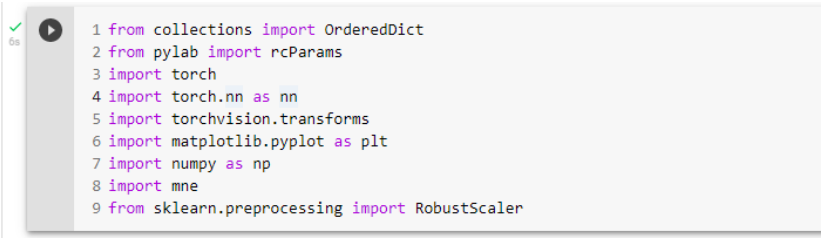

Figura 4: Importação de bibliotecas. Fonte: Google Collab.

Por fim, um randomizador foi inicializado com uma semente de valor 100 para certa consistência, sendo útil posteriormente:

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

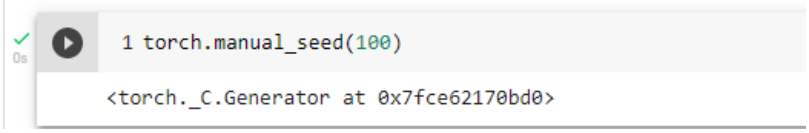

Figura 5: Inicialização do randomizador e seu retorno. Fonte: Google Collab.

4.2 INICIALIZANDO OS PARÂMETROS

O autor explica que o primeiro passo a ser tomado na inicialização do projeto foi informar ao sistema os parâmetros desejados. Na figura 6 é possível verificar a aplicação dos comandos:

- *eeg_sample_count:* Quantas amostras estão sendo utilizadas para treino da rede neural.
- *learning_rate:* O quão rápido a rede neural pode mudar os pesos, afeta a velocidade de aprendizado.
- *eeg_sample_length:* Número de pontos de dados por amostra.
- *number_of_classes:* Número de classes de saída. (O valor de 1 significa 100%, 0.0 significa 0%. É a certeza de que a amostra é um P300).
- *hidden1, 2 e 3:* Número de neurônios na primeira, segunda e terceira camada oculta.
- *output:* Número de neurônios na camada de saída.

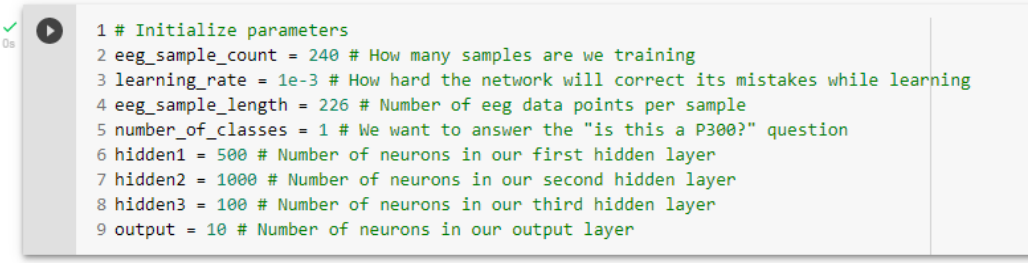

Figura 6: Inicializando parâmetros. Fonte: Google Collab.

4.3 CRIANDO A REDE NEURAL

Utilizando ajuda das bibliotecas já importadas, foi construído um gráfico que representa o aprendizado da rede neural (figura 7). O autor explica que primeiro é definida a entrada do gráfico da rede neural como um aglomerado de nodos lineares e de ativação. Os dados de entrada passam por esse gráfico, acumulando tendências escaladas por peso, sendo então transformados por funções de ativação, até que passe pelo nodo final de saída. Após isso, ele passa a uma função *sigmoide* (curva em formato de S) que coloca o número final em uma escala de 0 a 1. O número final é o preditor P300, crucial para o objetivo deste projeto.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218


```
## Define the network
tutorial_model = nn.Sequential()

# Input Layer (Size 226 -> 500)
tutorial_model.add_module('Input Linear', nn.Linear(eeg_sample_length, hidden1))
tutorial_model.add_module('Input Activation', nn.CELU())

# Hidden Layer (Size 500 -> 1000)
tutorial_model.add_module('Hidden Linear', nn.Linear(hidden1, hidden2))
tutorial_model.add_module('Hidden Activation', nn.ReLU())

# Hidden Layer (Size 1000 -> 100)
tutorial_model.add_module('Hidden Linear2', nn.Linear(hidden2, hidden3))
tutorial_model.add_module('Hidden Activation2', nn.ReLU())

# Hidden Layer (Size 100 -> 10)
tutorial_model.add_module('Hidden Linear3', nn.Linear(hidden3, 10))
tutorial_model.add_module('Hidden Activation3', nn.ReLU())

# Output Layer (Size 10 -> 1)
tutorial_model.add_module('Output Linear', nn.Linear(10, number_of_classes))
tutorial_model.add_module('Output Activation', nn.Sigmoid())
```

Figura 7: Parametrizando o gráfico. Fonte: Google Collab.

Em seguida a função de perda foi treinada. Estas são vitais para dar o *feedback* em quão bem a rede está sendo treinada. O autor explica que uma perda próxima a zero indica que a predição está sendo feita corretamente, feito com a ajuda do *PyTorch*. No código da figura 8 a função de perda foi definida, com um procedimento de treinamento simples que atualiza os pesos da rede neural e calcula a perda para cada iteração. O autor do código fonte complementa que ao final, o estado padrão da rede é salvo para que seja possível obtê-lo novamente no futuro. A linha número 18 foi modificada do código original pelo autor deste artigo com *loss.item()* para corrigir um erro de execução ao gerar o gráfico.

```
# Define a loss function
loss_function = torch.nn.MSELoss()

# Define a training procedure
def train_network(train_data, actual_class, iterations):

  # Keep track of loss at every training iteration
  loss_data = []

  # Begin training for a certain amount of iterations
  for i in range(iterations):

    # Begin with a classification
    classification = tutorial_model(train_data)

    # Find out how wrong the network was
    loss = loss_function(classification, actual_class)
    loss_data.append(loss.item())

    # Zero out the optimizer gradients every iteration
    optimizer.zero_grad()

    # Teach the network how to do better next time
    loss.backward()
    optimizer.step()

  # Plot a nice loss graph at the end of training
  rcParams['figure.figsize'] = 10, 5
  plt.title("Loss vs Iterations")
  plt.plot(list(range(0, len(loss_data))), loss_data)
  plt.show()

# Save the network's default state so we can retrain from the default weights
torch.save(tutorial_model, "/home/tutorial_model_default_state")
```

Figura 8: Definindo a função de treinamento e de perda. Fonte: Google Collab e Autor.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

4.4 OBTENDO O CONJUNTO DE DADOS MNE EEG

O autor comenta que a biblioteca MNE é um recurso valioso que se especializa em processamento de sinais cerebrais, e fornece acesso gratuito a bancos de dados com amostras relacionadas ao assunto. Para o projeto “Interpretador de P300”, foi utilizado o banco de dados com amostras de componentes P300 para treinar a rede. A instrução da figura 9 retorna o caminho do *dataset* no banco de dados do MNE-Python:

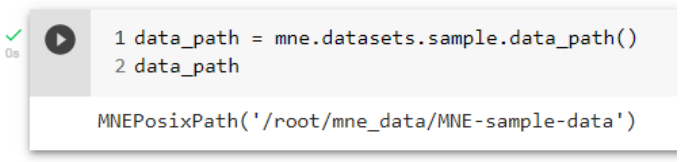

Figura 9: Iniciando a obtenção dos dados reais. Fonte: Google Collab.

Para obter os dados utilizando Python, o autor explicou que é necessário configurar o caminho para o conjunto de dados específico que se gostaria de acessar. Neste caso é uma base de dados são amostras de sinais audiovisuais onde as ondas cerebrais foram filtradas de 0 a 40 Hz (*Hertz*). São obtidos dados brutos, o que simplesmente significa que é uma coleção simplificada de vários canais EEG, ao invés de sinais mais focados em algum evento de interesse.

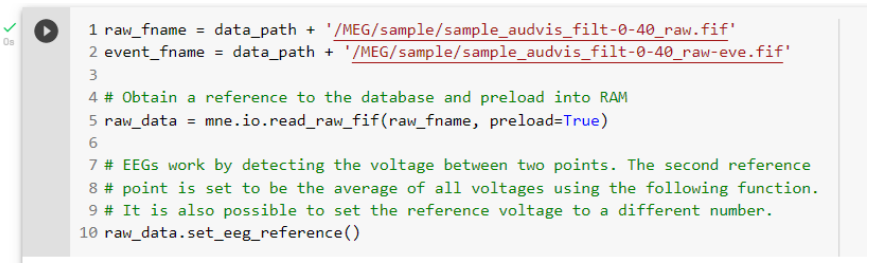

Figura 10: Definindo o caminho para obtenção dos dados de EEG. Fonte: Google Collab.

Feito isso, os dados foram carregados em uma variável chamada *raw_data*. O autor utilizou a função *pick* para que as fontes de interesse fossem escolhidas, para melhor filtrar a informação desejada. O comando é visualizado nas linhas 2 e 3 da figura 11. EEG é escolhido assim como eletrooculograma (EOG), já que são capturas dos mesmos eletrodos.

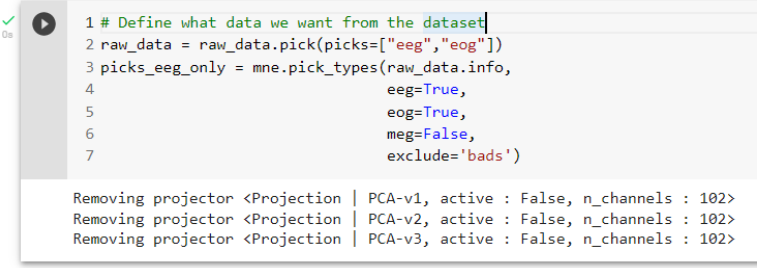

Figura 11: Especificando as fontes. Fonte: Google Collab.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Com o retorno do código visualizado na figura 12, podemos verificar que existem apenas 12 exemplos de eventos P300, o que é pouco, considerando que em média uma aplicação prática possui cerca de 100 eventos. Porém, a quantidade deve ser suficiente para o projeto atual, de acordo com o autor.

O autor complementa que o arquivo contém eventos que indicam quando algo aconteceu durante a gravação do EEG. Por exemplo, os eventos com o ID 5 representam pacientes que foram apresentados um *emoji* sorridente.

```
events = mne.read_events(event_fname)
event_id = 5
tmin = -0.5
tmax = 1
epochs = mne.Epochs(raw_data, events, event_id, tmin, tmax, proj=True,
                    picks=picks_eeg_only, baseline=(None, 0), preload=True,
                    reject=dict(eeg=100e-6, eog=150e-6), verbose = False)
print(epochs)
```

```
<Epochs |  12 events (all good), -0.499488 - 0.998976 sec, baseline -0.499488 – 0 sec, ~4.2 MB, data loaded,
 '5': 12>
```

Figura 12: Detalhes da fonte escolhida. Fonte: Google Collab.

O autor disserta que para os fins deste projeto é preferível um canal que possua a indicação mais alta de um evento P300, portanto o canal EEG 058 será especificado na figura 13:

```
# This is the channel used to monitor the P300 response
channel = "EEG 058"

# Display a graph of the sensor position we're using
sensor_position_figure = epochs.plot_sensors(show_names=[channel])
```

Figura 13: Especificando o canal para eventos P300. Fonte: Google Collab.

O autor explica que com os eventos obtidos, é necessário adicionar contra exemplos para realizar um contraste de dados, permitindo que a rede neural analise e filtre conforme foi treinada. Foram obtidos todos os outros eventos contidos no conjunto de dados.

```
event_id=[1,2,3,4]
epochsNoP300 = mne.Epochs(raw_data, events, event_id, tmin, tmax, proj=True,
                    picks=picks_eeg_only, baseline=(None, 0), preload=True,
                    reject=dict(eeg=100e-6, eog=150e-6), verbose = False)
print(epochsNoP300)
```

```
<Epochs |  208 events (all good), -0.499488 - 0.998976 sec, baseline -0.499488 – 0 sec, ~24.5 MB, data loaded,
 '1': 47
 '2': 56
 '3': 57
 '4': 48>
```

Figura 14: Adicionando todos os outros eventos. Fonte: Google Collab.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

Ainda é necessário realizar alguns passos para que a rede neural possa entender os dados e classificá-los conforme esperado. O autor explica que é necessário:

- Escalar os dados tendo em mente possíveis discrepâncias. Se os dados forem escalados somente utilizando valores mínimos e máximos, há um risco de os dados serem definidos por discrepâncias e os valores regulares serem espremidos no centro. Para evitar isso, serão utilizados valores estatísticos para escalar os dados.
- Criar uma variável que nomeia as amostras positivas de P300 como 1 e negativas como 0.
- Combinar os dados em uma estrutura de dados única.
- Realizar conversão de vários tipos de dados.

O código da figura 15 realiza este trabalho:

```
eeg_data_scaler = RobustScaler()

# We have 12 p300 samples
p300s = np.squeeze(epochs.get_data(picks=channel))

# We have 208 non-p300 samples
others = np.squeeze(epochsNoP300.get_data(picks=channel))

# Scale the p300 data using the RobustScaler
p300s = p300s.transpose()
p300s = eeg_data_scaler.fit_transform(p300s)
p300s = p300s.transpose()

# Scale the non-p300 data using the RobustScaler
others = others.transpose()
others = eeg_data_scaler.fit_transform(others)
others = others.transpose()

## Prepare the train and test tensors
# Specify Positive P300 train and test samples
p300s_train = p300s[0:9]
p300s_test = p300s[9:12]
p300s_test = torch.tensor(p300s_test).float()

# Specify Negative P300 train and test samples
others_train = others[30:39]
others_test = others[39:42]
others_test = torch.tensor(others_test).float()

# Combine everything into their final structures
training_data = torch.tensor(np.concatenate((p300s_train, others_train), axis = 0)).float()
positive_testing_data = torch.tensor(p300s_test).float()
negative_testing_data = torch.tensor(others_test).float()

# Print the size of each of our data structures
print("training data count: " + str(training_data.shape[0]))
print("positive testing data count: " + str(positive_testing_data.shape[0]))
print("negative testing data count: " + str(negative_testing_data.shape[0]))

# Generate training labels
labels = torch.tensor(np.zeros((training_data.shape[0],1))).float()
labels[0:10] = 1.0
print("training labels count: " + str(labels.shape[0]))

training data count: 18
positive testing data count: 3
negative testing data count: 3
training labels count: 18
```

Figura 15: Preparação dos dados para a rede neural. Fonte: Google Collab.

4.5 CLASSIFICANDO O CONJUNTO DE DADOS COM A REDE NEURAL

Utilizando as definições criadas na seção 4.3, a rede neural foi treinada com dados reais de EEG.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218


```
# Make sure we're starting from untrained every time
tutorial_model = torch.load("/home/tutorial_model_default_state")

# Define a learning function, needs to be reinitialized every load
optimizer = torch.optim.Adam(tutorial_model.parameters(), lr = learning_rate)

# Use our training procedure with the sample data
print("Below is the loss graph for dataset training session")
train_network(training_data, labels, iterations = 50)
```

Figura 16: Obtendo os dados de EEG para treinar a rede neural. Fonte: Google Collab.

Na figura 17 é possível verificar o gráfico que representa o treinamento da rede neural:

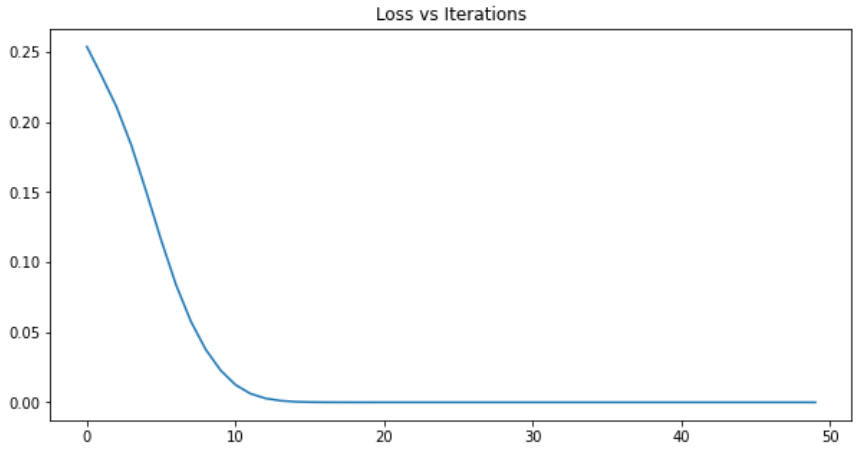

Figura 17: Treinamento da rede neural, perda x iteração. Fonte: Google Collab.

Ao criar, testar e treinar a rede neural com os dados necessários, a classificação foi realizada para concluir a resolução do problema:

```
# Classify our positive test dataset and print the results
classification_1 = tutorial_model(positive_testing_data)
for index, value in enumerate(classification_1.data.tolist()):
  print("P300 Positive Classification {1}: {0:.2f}%".format(value[0] * 100, index + 1))

print()

# Classify our negative test dataset and print the results
classification_2 = tutorial_model(negative_testing_data)
for index, value in enumerate(classification_2.data.tolist()):
  print("P300 Negative Classification {1}: {0:.2f}%".format(value[0] * 100, index + 1))

P300 Positive Classification 1: 100.00%
P300 Positive Classification 2: 99.94%
P300 Positive Classification 3: 100.00%

P300 Negative Classification 1: 99.92%
P300 Negative Classification 2: 0.04%
P300 Negative Classification 3: 0.00%
```

Figura 18: Instruções para classificação dos dados e pontuação. Fonte: Google Collab.

Conforme as porcentagens indicam, os eventos P300 foram classificados corretamente, porém um deles foi classificado incorretamente como um evento P300. De acordo com o autor, isso é em

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

parte consequência de poucos dados para treinamento, portanto a rede foi incapaz de diferenciar a o evento negativo de P300 número 1 do positivo.

Podemos instruir ao sistema que nos exiba em forma de gráfico as amostras analisadas com o seguinte código:

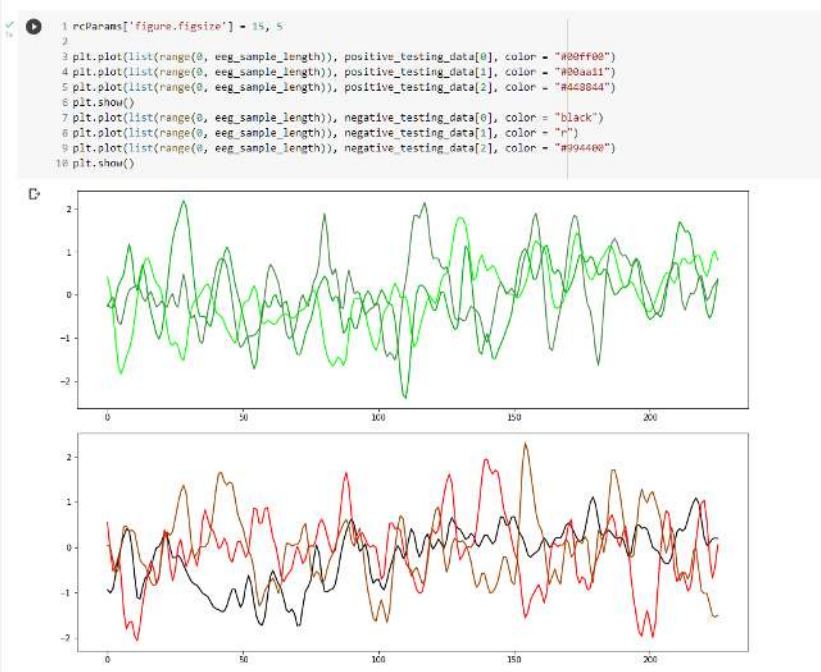

Figura 19: Amostras classificadas pela rede neural. Fonte: Google Collab.

O autor explica que a linha em cor preta no segundo gráfico representa a classificação de evento negativa realizada por engano pela rede neural. Visualmente ela é parecida com as amostras positivas, facilmente causando um engano ao ser classificado. Um problema como esse só pode ser corrigido com mais amostras para que a diferenciação seja feita.

5 CONSIDERAÇÕES FINAIS

Conforme mencionado na seção 4.4, o recomendado é possuir ao menos 100 amostras, e neste caso a rede neural possui somente 9. O autor menciona que não é o suficiente para uma aplicação verdadeira, mas o suficiente para uma prova de conceito. Ao longo deste projeto foi possível provar que a classificação de dados cerebrais por reconhecimento de redes neurais é possível e viável, requerendo apenas uma quantidade maior de dados para uma aplicação real.

Além disso, foi possível constatar que a rede neural artificial aprendeu a classificar dados com certa precisão, e o conceito do componente P300 foi utilizado para resolver o problema proposto.

RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

REFERÊNCIAS

AZAR, A.; HASSANIEN, A. **Brain-Computer Interfaces:** Current Trends and Applications. Berlin Heidelberg: Springer International Publishing. 2014. Disponível em: https://link.springer.com/book/10.1007/978-3-319-10978-7?utm_medium=referral&utm_source=google_books&utm_campaign=3_pier05_buy_print&utm_content=en_08082017 Acesso em: 31 maio 2022.

DINTEREN, R.; ARNS, M.; JONGSMA, M.; KESSELS, R. **P300 Development across the Lifespan:** A Systematic Review and Meta-Analysis. USA: National Library of Medicine, 2014. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3923761/ Acesso em: 31 maio 2022.

GIUSEPPE, C.; CIRAC, I.; CRANMER, K.; DAUDET, L.; SCHULD, M.; TISHBY, N.; VOGTMARANTO, L.; ZDEBOROVÁ, L. Machine learning and the physical sciences, **Rev. Mod. Phys.,** v. 91, 6 December 2019. Disponível em: https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.91.045002 Acesso em: 31 maio 2022.

GRAIMMANN, B.; ALLISON, B.; PFURTSCHELLER, G. **Brain-Computer Interfaces:** Revolutionizing Human-Computer Interaction. Berlin Heidelberg: Springer, 2010. Disponível em: https://link.springer.com/book/10.1007/978-3-642-02091-9?utm_medium=referral&utm_source=google_books&utm_campaign=3_pier05_buy_print&utm_content=en_08082017 Acesso em: 31 maio 2022.

LEITE, S. **Contribuições ao Desenvolvimento de Interfaces Cérebro-Computador Baseadas em Potenciais Evocados Visualmente em Regime Estacionário**. 2016. Tese (Doutorado) -Universidade Estadual de Campinas - UNICAMP, Campinas, 2016. Disponível em: http://repositorio.unicamp.br/acervo/detalhe/970748?guid=1654047707563&returnUrl=%2fresultado%2flistar%3fguid%3d1654047707563%26quantidadePaginas%3d1%26codigoRegistro%3d970748%23970748&i=4 Acesso em: 31 maio 2022

RAO, R. **Brain-Computer Interfacing:** An Introduction. Cambridge: Cambridge University Press, 2013. Disponível em: https://www.cambridge.org/br/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/brain-computer-interfacing-introduction?format=HB&isbn=9780521769419 Acesso em: 31 maio 2022.

SIEBERT, Brandon. **Interpretador de P300, com o código fonte com autoria de Brandon Siebert**. [S. l.]: Neurotechedu, s. d. Disponível em: http://learn.neurotechedu.com/machinelearning/ e https://colab.research.google.com/drive/1ZQt8RCkkmTEYXRDmFbj1kpcXpUXD2Da0 Acesso em: 26 maio 2022.

WOLPAW, E.; WOLPAW, J. **Brain-Computer Interfaces:** Principles and Practice, USA: Oxford University Press, 2012. Disponível em: https://global.oup.com/academic/product/brain-computer-interfaces-9780195388855?cc=br&lang=en& Acesso em: 31 maio 2022.