Revista Brasileira de Computação Aplicada, Abril, 2022

DOI: 10.5335/rbca.v14i1.13070
Vol. 14, Nº 1, pp. 55–69
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ARTIGO ORIGINAL

Arquiteturas de redes neurais e suas aplicabilidades para classificação de sinais EEG para BCI

Neural Networks architectures and its applications in EEG signal classification for BCI

Rafael Luís Savenhago[1], Paulo Muniz de Ávila[1], Rodrigo Lício Ortolan [1]

[1]Instituto Federal do Sul de Minas Gerais

*rafael.savenhago@gmail.com; paulo.avila@ifsuldeminas.edu.br;rodrigo.ortolan@ifsuldeminas.edu.br · · ·

Recebido: 19/10/2021. **Revisado:** 11/04/2022. **Aceito:** 25/04/2022.

Resumo

Muitas pessoas no mundo sofrem com algum tipo de doença motora que atrapalha sua vida cotidiana. Uma das formas de melhorar a vida dessas pessoas é através da chamada Interface Cérebro Computador. No entanto, esse método até o momento deixa a desejar quanto a taxa de acerto de suas classificações. Este artigo visa explorar e comparar arquiteturas de redes neurais para classificação de sinais de Eletroencefalograma para Interface Cérebro Computador utilizando diversas arquiteturas diferentes, inclusive as pouco exploradas Redes de Valores Complexos, e testar novas possibilidades de funções de ativação. A metodologia de execução deste trabalho envolve o pré-processamento de dados de sinal EEG já rotulados, divisão em bandas de sinal com base nas faixas de frequência características do cérebro definidas por delta (0.5-4HZ), theta (4-8HZ), alpha (8-13HZ), e beta (acima de 13HZ). Os frames de tempo gerados pela separação em bandas são utilizados para alimentar as diversas arquiteturas que serão avaliadas.

Palavras-Chave: Eletroencefalograma; Funções de ativação; Interface Cérebro Computador; Redes Neurais Artificiais

Abstract

A lot of people around the world suffer from some kind of motor illness that interferes with their daily lifes. One of the ways to help to improve the life of such people is the so called Brain Computer Interface. However, this method has some room to improve in its accuracy. This Article seeks to explore and to compare the large amount of Neural Networks Architectures for classification of EEG signals for Brain Computer Interfaces, using even the less-known Complex-Valued Networks, and try with new activation functions . The methodology of this work involves preprocessing of labeled-EEG signals, splitting the frequency components of the signal in bands of frequency based on the brainwaves frequencies, defined as delta (0.5-4HZ), theta (4-8HZ), alpha (8-13HZ), and beta (above 13HZ) The resulting time frames will then be used to feed the several evaluated-to-be architectures.

Keywords: Activation Functions; Artificial Neural Networks; Brain Computer Interface; EEG;

1 Introdução

Existem no mundo muitas pessoas que portam deficiências motoras. Garcia and Maia (2014) afirmam com base em dados do Censo do Instituto Brasileiro de Geografia e Estatística(IBGE) de 2010, que, apenas no Brasil existem mais de 45 milhões de pessoas com algum tipo de deficiencia fisica. Para essas, a vida cotidiana tende a ser mais

complexa do que para outras pessoas. Em casos extremos as pessoas não conseguem movimentar nenhum músculo, como em alguns casos de distrofia muscular. Na tentativa de melhorar a vida dessas pessoas alternativas são propostas. Uma que apresenta uma melhoria acentuada é a chamada Interface Cérebro Computador, como diz Ochoa (2002).

Para esses sistemas de Interface Cérebro Computador é importante que haja um método de classificar os sinais provenientes do cérebro. Para isso utiliza-se redes neurais artificiais. No entanto, existem dezenas de arquiteturas e funções de ativação para elas e cada uma afeta diretamente o quão boa a capacidade de classificação (generalização) de uma determinada representação (modelo) vai ser.

Tendo como base as pesquisas na área como as descritas por Schirrmeister et al. (2017), Yuksel and Olmez (2015), Fahimi et al. (2019), e outras mais, percebe-se como a escolha de uma arquitetura pode ser uma tarefa complicada. Mesmo as arquiteturas mais utilizadas podem ser ineficientes para determinados problemas, e não existe nenhum guia para padronização dos parâmetros utilizados. Também deve-se considerar os problemas de algoritmos de aprendizado utilizados.

Atualmente, os modelos utilizados para BCI são baseados em redes convolutivas profundas, como o utilizado por Shi et al. (2019). Apesar de apresentarem potencial para resolver tais problemas, muitos não são difundidos ou não tem suporte pelos *frameworks* utilizados.

O objetivo deste trabalho é desenvolver aplicação que realiza análise de sinais de eletroencefalograma (EEG) de *Motor Imagery* (MI) e os classifica em ações com redes neurais para Interface Cérebro-Computador (BCI). E também propor e comparar os resultados para diferentes funções de ativação e parâmetros das redes, tendo como diferencial a tentativa de abranger diversas tipologias de redes neurais muito utilizadas nos contextos de processamento de sinais (inclusive algumas menos conhecidas, como Redes de Valores Complexos).

2 Referencial Teórico

2.1 Interface Cérebro Computador (Brain Computer Interface, ou BCI)

Interface Cérebro Computador, segundo Forslund (2003), é um sistema técnico que permite alguém a controlar e influenciar o mundo ao redor sem necessitar de fazer uso dos músculos.

Normalmente, sistemas BCI são formados por 3 estágios: coleta de sinal, pré-processamento, e classificação. A seguir são descritos brevemente esses estágios segundo a definição de Forslund (2003).

A coleta de sinal geralmente é realizada por aparelhos de eletroencefalograma(EEG) que captam o potencial elétrico de determinados pontos do cérebro. Embora existam outros tipos de aparelhos, EEG é o mais utilizado devido ao custo reduzido e a não necessidade de operações invasivas.

Na etapa de pré-processamento o sinal coletado sofre transformações para facilitar o processo de classificação. As transformações mais comuns são a filtragem de ruído e a aplicação da transformada de Fourier para obter as componentes do sinal no domínio da frequência.

Quanto ao estágio de classificação, este se encarrega de interpretar o sinal pré-processado e determinar qual ação o mesmo indica. Para tal, costuma-se usar redes neurais, SVMs, lógica difusa, e outros. Com o avanço das redes de aprendizado profundo, a etapa de pré processamento pode ser parcialmente ignorada e integrada ao classificador.

2.1.1 *Coleta de Sinais*

Conforme descrito por Forslund (2003), as aplicações *BCI* costumam fazer uso de equipamentos de EEG para realizar a coleta dos sinais. Isto é ocasionado principalmente pelo custo relativamente reduzido dos aparelhos de coleta.

Os aparelhos de coleta de EEG são formados por diversos sistemas conforme Teplan (2002), sendo estes: Eletrodos com meio condutor, amplificador com filtros, conversores A/D, e dispositivo de gravação.

Segundo a descrição de Teplan (2002), os eletrodos devem ser colocados no escalpo do paciente, em alguma configuração estabelecida, sendo a mais conhecida o chamado sistema de posicionamento de eletrodos 10-20. Essas configuração levam em conta a posição dos centros de atividade do cérebro para fazer medidas mais precisas e úteis. Os eletrodos em si costumam ser feitos de materiais como ouro ou prata cloreto de prata (Ag/AgCl), devido a suas propriedades condutoras.

O segundo estágio são os amplificadores com filtros. Segundo Teplan (2002), os sinais de EEG obtidos pelos eletrodos tem uma amplitude baixa, na ordem dos micro Volts(μV), e portanto devem ser amplificados para que seja possível converter os mesmos para sinais digitais com menores perdas de informação. Por consequencia da baixa magnitude dos sinais, os mesmos apresentam uma baixa relação sinal-ruído, o que indica que o sinal não é muito confiavél. Para tentar melhorar a relação sinal-ruído, utilizam-se de filtros analógicos juntamente com os amplificadores.

Para fazer uso dos sinais coletados, estes devem ser convertidos para sinais digitais de forma que um computador possa armazenas e trabalhar com eles. Para isto, o sinal passa por um conversor A/D, que converte o sinal para um valor binário. Teplan (2002) também cita que, deve-se considerar dois fatores na escolha de um conversor A/D. A taxa de amostragem e a resolução do conversor.

A taxa de amostragem diz respeito à quantas vezes o sinal vai ser medido por unidade de tempo. Segundo Teplan (2002), em sistemas de EEG, é desejável utilizar-se de taxas de amostragem maiores que 200 Hz, sendo 500 hz um bom valor recomendado. Para evitar problemas derivados da coletado discreta de dados, como aliasing, é importante que os filtros estejam em sintonia com os valores de amostragem.

A resolução do conversor indica quantos valores discretos diferentes um sinal pode ter. Por exemplo, se um conversor tem 1 bit de resolução, então independentemente do sinal, o resultado será apenas o conjunto de 2 números. Segundo Teplan (2002), é essencial que um conversor tenha no mínimo 12 bits, ou 4096 níveis distintos. Para bons resultados recomenda-se que os conversores consigam identificar e quantificar diferenças de até 0,5 μV.

Já os dispositivos de gravação servem apenas para armazenar os sinais convertidos. Teplan (2002) sugere que

os mesmos tenham uma grande capacidade de memória, uma vez que 1 hora de um sinal de 8 canais com frequencia de amostragem de 500 Hz e resolução de 14 bits consome em média 200 MB de armazenamento.

2.1.2 *Pré-Processamento de Sinais*

Conforme descrito por Forslund (2003), Teplan (2002), Ortolan et al. (2004), e Cunha et al. (2000), a etapa de pré-processamento é de muita importância para a análise de sinais temporais, principalmente se tratando de sinais coletados através de sensores de algum tipo.

Segundo Teplan (2002) e Ortolan et al. (2004),para EEG, pode-se dividir o pré-processamento em duas principais etapas: a filtragem, e a seleção de paramêtros.

Conforme cita Teplan (2002), o aparelho de EEG pode ter diversas fontes de ruídos, sendo elas: artefatos causados pelo movimento, sinais musculares, batimentos cardíacos, movimento dos olhos, suor, interferência da rede elétrica, entre outros.

Para resolver esses problemas, Teplan (2002) sugere o uso de filtros de *software* do tipo FIR(*Finite Impulse Response*) para evitar distorções de fase. Também salienta a importância de conhecer o sinal trabalhado para evitar filtrar informações importantes.

Teplan (2002) também define as faixas de frequência associadas ao funcionamento do cérebro da seguinte maneira:

- Beta (>13 Hz)
- Alpha (8-13 Hz)
- Theta (4-8 Hz)
- Delta (0.5-4 Hz)

Em sequência com a filtragem do sinal, Ortolan et al. (2004) descreve a extração de parâmetros do sinal. O mesmo divide os parâmetros em dois grupos, os parâmetros temporais, e os parâmetros espectrais.

Em sistemas de tempo real, Ortolan et al. (2004) afirma que os parâmetros temporais são mais viáveis devido à praticidade de não ter que realizar transformadas. Um exemplo de parâmetro temporal é o chamado valor RMS, que indica o valor eficaz do sinal, que está relacionado com a quantidade de energia contida no sinal, no qual N representa o tamanho da amostra, e X(i) representa a amplitude do sinal no momento i, conforme a Eq. (1):

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x(i)^2} \qquad (1)$$

Já os parâmetros espectrais requerem mais recursos computacionais, no entanto, segundo Ortolan et al. (2004), eles trazem preciosas informações. Um exemplo de parâmetro espectral é a densidade espectral de potência (DEP) de uma determinada faixa, que indica quais faixas de frequência possuem as maiores potências, no qual FFT(x)(i) é amplitude da transformada rápida de fourier do sinal X para uma frêquencia i, N é o tamanho da amostra no dominio da frequencia, e k é o ponto de inicio da faixa analizada, conforme a Eq. (2):

$$\mathrm{DEP(x,k,N)} = \sqrt{\frac{1}{N-k}\sum_{i=k}^{N} FFT(x)(i)^2} \qquad (2)$$

Para um maior número de opções, por exemplo,é possível combinar filtragem com a seleção de parâmetros para obter os valores paramétricos de apenas algumas faixas do espectro.

2.2 Redes Neurais de valores Reais

Segundo Ferneda (2006), redes neurais são parte de um conjunto de técnicas conhecidas na ciência da computação como Inteligência Artificial cujo objetivo é replicar o funcionamento do cérebro humano. Baseia-se modelos matemáticos fundamentados na unidade do Neurônio Artificial, conforme Fig. 1.

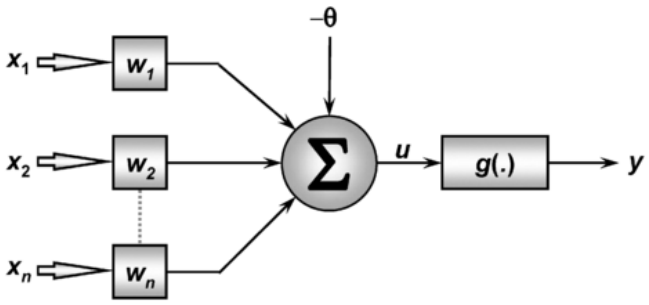

Figura 1: Neurônio Artificial(Perceptron)
fonte:da Silva et al. (2016)

Segundo da Silva et al. (2016), o neurônio artificial possui os seguintes componentes:

As entradas $x_1, x_2, \ldots, x_n$ são os diversos sinais advindos do meio externo, este podendo ser as saídas de outros neurônios ou aplicações externas. São representadas pelo conjunto $x_1, x_2, \ldots, x_n$ e são análogos aos impulsos elétricos recebidos em um neurônio biológico. Normalmente os valores são normalizados para obter melhores resultados durante o processo de treinamento.

Os pesos $W_1, W_2, \ldots, W_n$ são os valores para ponderar a importância de cada entrada para o disparo do neurônio. Indica a relevância de determinada entrada em relação a funcionalidade do neurônio.

A função somadora (Σ) é responsável por acumular o valor das entradas ponderadas pelos pesos (entrada multiplicada pelo seu respectivo peso) a fim de calcular o conhecido potencial de ativação **u**.

O bias de ativação(θ), representa o valor necessário para que o potencial de ativação cause o disparo do neurônio. O potencial de ativação é o resultado da diferença da função somadora com o limiar de ativação, representa o potencial excitatório ou inibitório do neurônio.

A função de ativação(g), indica o disparo do neurônio com base na função somadora. Existem várias funções de ativações que podem ser utilizadas para determinados pro-

blemas. Todas tem por objetivo limitar a saída do neurônio ao aplicar uma determinada função ao potencial de ativação.

2.2.1 *Funções de Ativação*

Conforme especificado por da Silva et al. (2016), as funções de ativação são importantes para o funcionamento do modelo do neuronio artificial. Com elas pode-se resolver problemas considerados não lineares. A seguir será discutido algumas das funções mais recorrentes.

2.2.1.1 *Função Degrau.*

Segundo da Silva et al. (2016), a função degrau representa um disparo quando o potencial de ativação do neurônio for maior ou igual a zero, caso contrário, a função permanece inibida. Normalmente utilizada em problemas de classificação.

$$g(u) = \begin{cases} 0, & u < 0 \\ 1, & u \geq 0 \end{cases} \tag{3}$$

2.2.1.2 *Função Rampa Simétrica.*

Conforme explicado por da Silva et al. (2016), esta apresenta um intervalo no qual o valor de saída é igual ao potencial de ativação, até chegar em um valor de saturação pré-determinado, este valor (a) normalmente é o número 1. Por ser simétrica, a função satura tanto para valores positivos (satura quando a entrada é maior que o valor a) quanto para negativos(satura negativamente quando a entrada é menor que o valor $-a$). Pode ser utilizada para problemas de classificação e regressão.

$$g(u) = \begin{cases} -a, & u < -a \\ u, & -a \leq u \leq a \\ a, & u > a \end{cases} \tag{4}$$

2.2.1.3 *Função Relu (retified linear unit).*

Segundo Nwankpa et al. (2018), essa função surgiu para resolver um dos maiores obstáculos no treinamento de redes neurais, conhecido como desaparecimento do gradiente (*vanishing gradient*) devido a sua característica de não saturar em ambos os lados. É a função mais utilizada para redes neurais profundas (deep neural networks).

A mesma apresenta a característica de ser extremamente eficiente para computar. Apenas realizando comparações, adições e multiplicações, também por ter somente dois valores distintos de derivadas e apenas um ponto cuja a derivada é inexistente, esta função é de simples treino.

$$g(u) = \begin{cases} 0, & u < 0 \\ u, & u \geq 0 \end{cases} \tag{5}$$

2.2.1.4 *Função Linear.*

Pela definição de da Silva et al. (2016) a função linear é na verdade apresentada pela ausência de um elemento função de ativação, pois o resultado de saída é exatamente o mesmo do potencial de ativação. Esta função costuma ser utilizada para problemas de regressão devido a capacidade de representar qualquer valor.

$$g(u) = u \tag{6}$$

2.2.1.5 *Função Gaussiana.*

Segundo o que da Silva et al. (2016) apresenta, esta é baseada na distribuição normal de estatística. Esta função apresenta o valor de 1 em um ponto central(c) e apresenta valores iguais para uma determinada distância do ponto central. O valor de decrescimento da função Gaussiana é relacionado ao chamado desvio padrão(σ) que indica o quão dispersa é a função em relação ao ponto central.

Esta função é utilizada principalmente em problemas de classificação, uma vez que os resultados já consideram o ruído de uma distribuição normal.

$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}} \tag{7}$$

2.2.1.6 *Função Logística.*

Para da Silva et al. (2016), esta é uma alternativa totalmente diferenciável para a função degrau, apresenta por característica um valor de inclinação(β) referente ao ponto de inflexão (valor da função quando o potencial é igual 0), quanto maior a inclinação mais esta função aproxima a função degrau. Assim como sua correspondente a mesma sempre retorna valores entre 0 e 1, e portanto é utilizada normalmente para classificação.

$$g(u) = \frac{1}{1 + e^{-\beta \cdot u}} \tag{8}$$

2.2.1.7 *Função Tangente Hiperbólica.*

Diferentemente da função logística, esta função apresenta valores entre -1 e 1, e tem seu ponto de inflexão no ponto cujo potencial de ativação é 0. Conforme varia-se a constante de inclinação a função tende a aproximar à conhecida função bipolar, segundo a explicação de da Silva et al. (2016).

$$g(u) = \frac{1 - e^{-\beta \cdot u}}{1 + e^{-\beta \cdot u}} \tag{9}$$

2.2.2 *Redes Multicamadas*

Segundo da Silva et al. (2016), redes multicamadas são redes que apresentam no mínimo duas camadas de neurônios interligadas normalmente de forma sequencial (arquitetura *feedforward*).

De acordo com Gao et al. (2021), ao utilizar os conceitos de aprendizado profundo (*Deep Learning*), que diz respeito ao uso de redes com várias camadas e é muito utilizado para identificar padrões complexos, e análise de sinais, pode-se obter modelos com capacidade de extração de características e classificação de sinais mais robustas.

2.2.2.1 *Perceptron multicamadas.*

Este tipo de arquitetura se caracteriza por diversas camadas de neurônios do tipo perceptron (neurônios simples) ligados de forma que a entrada de neurônios da camada posterior são as saídas da camada anterior, como pode-se ver na Fig. 2. Cada camada oculta i tem n_i neurônios, e a camada de saída tem m, sendo os valores de n_i e m estipulados durante a configuração.

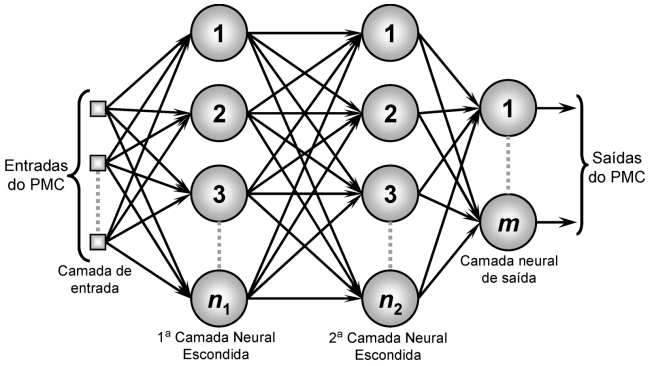

Figura 2: Rede Multicamadas com 2 camadas ocultas(Perceptron multicamadas)
fonte:da Silva et al. (2016)

da Silva et al. (2016) diz que em teoria ao adicionar mais camadas a rede MLP, ou *multilayer perceptron*, a rede deveria se tornar capaz de aprender padrões mais complexos, porém, devido ao problema do *vanishing gradient* do algoritmo de *backpropagation*, estas redes acabam tendo alguns problemas de convergências quando possuem muitas camadas.

2.2.2.2 *Redes recursivas.*

Segundo da Silva et al. (2016), este modelo apresenta como característica a realimentação de sinais de saída de neurônios nas camadas posteriores em camadas anteriores como um sinal de entrada qualquer, como pode ser visto pelo elemento "Realimentação"na Fig. 3. São muito úteis para trabalhar com entradas baseadas no tempo uma vez que a realimentação funciona como um mecanismo de memória de um estado anterior, desta forma, conseguem manter informações relevantes sobre o contexto em seu funcionamento. Como a realimentação pode ser interpretada como um laço de camadas esta arquitetura tende a sofrer mais com o problema do *vanishing gradient*.

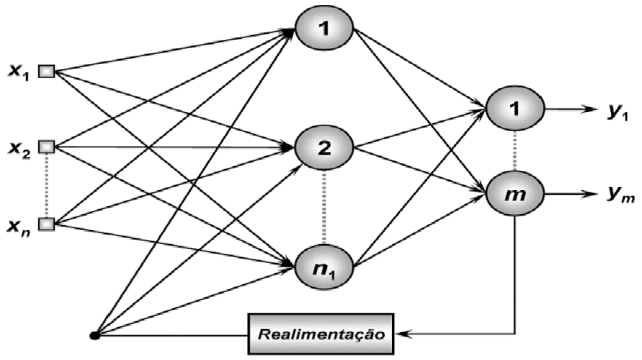

Figura 3: Rede Neural Realimentada(Recursiva)
fonte:da Silva et al. (2016)

2.2.2.3 *Long-short term memory (LSTM).*

Conforme a definição de Sak et al. (2014), o termo LSTM refere-se à um tipo de rede neural que contém blocos especiais de neurônios chamados de Blocos de Memória em suas chamadas recursivas. Estes blocos são organizados de forma que as informações são avaliadas, esquecidas e lembradas (exatamente nesta ordem).

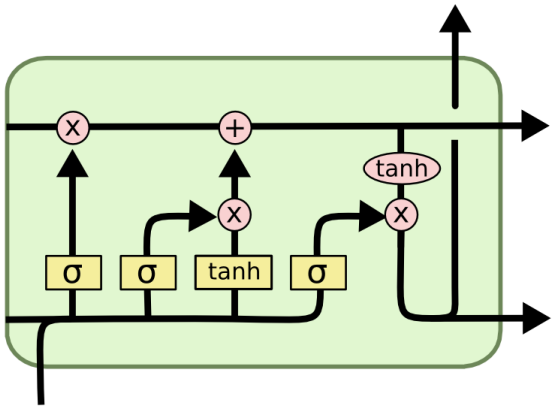

Figura 4: Módulo LSTM
fonte:https://colah.github.io/posts/2015-08-Understanding-LSTMs/

O funcionamento deste tipo de rede se dá por conta do uso de dois tipos de funções de ativação, assim como pode-se ver na Fig. 4, sendo estas a função Logística e a Função Tangente Hiperbólica. A primeira é utilizada para avaliar se a informação devido ao fato de seu retorno ser entre zero e um, o qual ao ser multiplicado pela entrada avaliada pode fazer a mesma ser parcialmente ou totalmente esquecida.

O fluxo de dados é realizado pelos *Gates*, que é o nome dado para as estruturas de controle de informação, representados pela função Logística que decidem quais informações são adicionadas ou removidas.

Primeiro a entrada e o valor realimentado de forma recursiva passam pelo *forget gate* que é responsável por manter ou não a memória da unidade (*cell state*). Isto é feito ao multiplicar o resultado do gate (entre zero e um) pelo valor salvo na memória. Caso o resultado do gate seja zero, então a memória é apagada.

Em seguida os mesmos dados passam pelo *input gate*, que serve para determinar quanto do estado computado pela camada será passado em frente na memória. Isso é feito ao realizar as operações comuns de um neurônio com a função de ativação Tangente Hiperbólica, passar essas informações pelo gate, multiplicar o resultado do gate com o da função de ativação, e somar com o valor do *cell state*.

Por fim, tem-se o *output gate*, que serve para computar quanto do *cell state* deve ser passado para a próxima camada. Isto é feito através do cálculo da Tangente Hiperbólica para o *cell state*, e da multiplicação deste resultado com o valor retornado pelo *gate*.

2.2.2.4 *Redes Convolucionais.*

Como descrito por Schirrmeister et al. (2017), redes convolucionais são tipos de redes neurais que conseguem aprender a identificar padrões locais através do uso de operações chamadas de convolução.

Segundo Schirrmeister et al. (2017), esse tipo de rede é utilizado em áreas como reconhecimento de imagens, reconhecimento de fala, e outras áreas como análise de séries temporais.

Como definido por Schirrmeister et al. (2017), redes convolucionais funcionam através de filtros, que "deslizam" sobre as variáveis, e são multiplicados pelas mesmas, assim como pode ser visto na Fig. 5. Cada filtro apresenta uma característica a ser extraída. Nas camadas iniciais, os filtros extraem informações básicas, no entanto, conforme progride-se pelas camadas os mesmos passam a identificar padrões complexos. Ao usar por exemplo o reconhecimento de imagem, nas primeiras camadas os filtros identificam arestas e contrastes, enquanto nas camadas mais profundas os filtros passam a identificar contornos de faces e outras características complexas.

Filtros, segundo Schirrmeister et al. (2017), são representados por matrizes de tamanho determinado que são multiplicados (produto interno ou dot product) por cada subgrupo de entradas com o mesmo tamanho. Eles são os pesos de uma rede convolutiva. No final de cada dot product o resultado é armazenado, e quando o filtro termina de "varrer" todos os subgrupos os resultados geram uma matriz de dados que representa sua saída.

Geralmente junto com camadas convolutivas utiliza-se camadas de *pooling*, que seleciona algum dos resultados da matriz de convolução para passar para a próxima camada, conforme descrito por Schirrmeister et al. (2017), assim garantindo algumas vantagens como tolerância à variação espacial de elementos. Também utiliza-se redes Perceptron Multicamadas em sequncia para fazer "inferências" sobre os padrões observados.

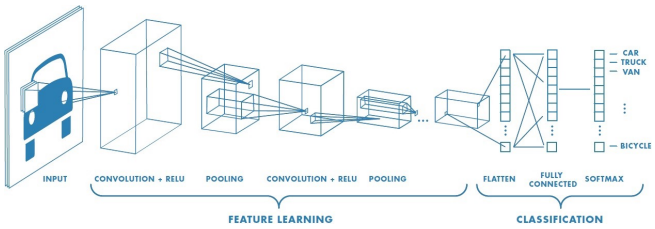

Figura 5: Rede Convolucional
fonte:https://medium.com/@Nikhil_Agarwal/cnn-convolutional-neural-network-a4cbef7b355f

2.2.2.5 *Redes Residuais.*

O conceito de redes residuais, conforme descrito por He et al. (2016), surgiu para tentar resolver um problema recorrente ao se tratar de redes neurais muito profundas. Em teoria, quanto maior o número de camadas, melhor é a generalização de funções complexas. No entanto, ao se tratar de funções um pouco mais simples, redes muito profundas apresentam um desempenho que deixa a desejar quando comparadas à redes mais simples devido ao fato de que o processo de treinamento atualiza todos os pesos em uma única iteração.

As redes residuais tentam resolver este problema através do que é chamado de bloco Residual. Como pode ser visto na Fig. 6, o bloco residual representa um "atalho" entre as camadas, que serve para que determinado valor pule algumas camadas e seja novamente colocado como entrada nas camadas posteriores.

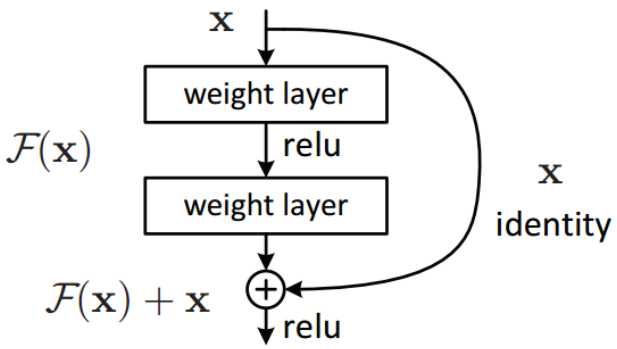

Figura 6: Bloco Residual
fonte:https://www.jeremyjordan.me/convnet-architectures/

Segundo He et al. (2016), este funcionamento parte do pressuposto de que é muito fácil para uma rede aprender a função Nula (função cujo retorno é zero independentemente das entradas), ou seja, em uma rede muito grande, algumas camadas aprendem a função enquanto as outras apenas propagam o valor devido à estrutura. Quando a função G(x) necessita ser aprendida pelo bloco, pode-se generalizar este aprendizado da camada como F(x)=G(x)-x, desta forma se obtém o resultado esperado.

2.3 Aprendizado por Gradiente Descendente

Com base na explicação de da Silva et al. (2016), o processo de treinamento mais comum de redes neurais é o chamado de gradiente descendente, este consiste em um processo de duas etapas repetidas diversas vezes.

A primeira das etapas é a chamada propagação adiante (*forward propagation*) e consiste em passar os dados a serem treinados pela rede e coletar o resultado de saída.

Em seguida, com o resultado coletado, é realizado um cálculo através da chamada função de custo, que representa o erro de uma rede em valores numéricos. Esta função é de extrema importância, pois todo o algoritmo se baseia em minimizá-la, ou em outras palavras minimizar o erro da rede. A função de custo muito utilizada é a chamada erro quadrático médio e é representada pela seguinte equação:

$$E = \frac{1}{p} \cdot \sum_{k=1}^{p} \frac{(d(k) - y(k))^2}{2} \tag{10}$$

Através de uma análise da Eq. (10) pode-se verificar que o erro médio quadrático é a média do quadrado da diferença entre a saída desejada (d) e a saída obtida(y) para cada neurônio(k) na camada de saída, onde p é o número de neuronios na última camada. Devido ao fato de possuir o quadrado da diferença, esta função apresenta como ponto mínimo o ponto em que todas as saídas esperadas são iguais as saídas obtidas, e este é o ponto em que deseja-se que a rede alcance.

A próxima etapa diz respeito ao processo de propagação reversa(*backpropagation*) na qual o erro calculado na função de custo é utilizado para minimizar a mesma através do método de gradiente descendente, que vem a ser um método iterativo para minimizar funções.

O algoritmo de gradiente descendente faz uma atualização de cada peso com base em sua influência no valor do erro (custo) com base na seguinte equação:

$$W_i = W_i - \eta \cdot \frac{\partial E}{\partial W_i} \tag{11}$$

Com base na Eq. (11) de atualização de pesos é possível entender que, o peso é atualizado de acordo com o valor antigo através de um movimento ponderado por um valor η na direção de maior declive da função de custo (pode ser inferido pelo gradiente negativo da função $-\eta \cdot \frac{\partial E}{\partial W_i}$), o que indica ir na direção de um mínimo da função, ou seja, um ponto em que minimiza-se função erro.

Após a atualização de todos os pesos para um determinado par de Entradas e Saídas, o mesmo procedimento é realizado para os outros pares. Este processo é repetido um determinado número de vezes, ou até que o erro se torne inexistente.

2.4 Redes de valores complexos

Com base nas afirmações de Trabelsi et al. (2017), redes de valores complexos são redes nas quais as entradas, pesos, funções de ativação, e saídas são compostos por numerais e operações complexas. Devido às características destas redes fazerem uso de elementos complexos existem muitas vantagens para o seu uso. Entre elas o potencial para permitir otimização mais facilmente, melhor generalização de funções, aprendizado em menos iterações, e melhor tolerância a ruídos.

Por se tratar de um conjunto de operações complexas este tipo de rede consegue processar dois valores simultaneamente por informação. Estes valores são geralmente valores de amplitude e fase, mas também podem ser tratados como dois valores independentes.

Segundo Trabelsi et al. (2017), o uso de numerais complexos tem muitas vantagens pelos pontos de vistas biológico, computacional e de processamento de sinais. Devido ao fato de cada uma das entradas e saídas serem complexas pode-se avaliar as mesmas como síncronas e assíncronas entre si com base nas informações de fase. Quando as fases são as mesmas os valores são adicionados normalmente, mas caso contrário, os valores interferem entre si, desta forma sendo muito útil para mecanismos de memória baseada em gates, uma vez que simplifica o processo ao substituir o uso da função Logística por operações de adição.

Apesar de suas diversas vantagens, redes de valores complexos não são muito difundidas, e não existe o estímulo para o uso das mesmas. Embora existam diversos artigos tais como o de Guberman (2016) que apresentam uma explicação aprofundada sobre o assunto, Trabelsi et al. (2017) fala sobre a ausência de frameworks para facilitar o acesso ao desenvolvimento, e portanto, não há muita difusão destas tecnologias.

2.4.1 *Funções de ativação complexas*

As funções de ativação complexas, assim como nas funções de valores reais, também podem ser classificadas em totalmente ou parcialmente diferenciáveis.Segundo Trabelsi et al. (2017), para ser totalmente diferenciáveis as funções devem satisfazer as equações de Cauchy-Riemann a seguir:

$$f(x + \imath \cdot y) = u(x, y) + \imath \cdot v(x, y) \tag{12}$$

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \tag{13}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{14}$$

Este tipo de função que satisfaz estas equações é chamado de função holomorfa. Estas são caracterizadas por serem totalmente diferenciáveis em todos os pontos do plano complexo (quatro dimensões divididas em duas para representar a variável de entrada e duas para o resultado).

Embora uma função holomorfa seja mais interessante para o funcionamento do algoritmo de gradiente descendente, as funções mais utilizadas pelas redes de valores complexos não são deste tipo. Devido à facilidade de uso(desempenho), as funções mais utilizadas são varia-

ções da função ReLU para valores complexos.

2.4.1.1 ModReLU.

A primeira das funções descrita por Trabelsi et al. (2017), é a ModReLU. Esta é uma das funções complexas mais utilizadas devido ao fato de não alterar a fase do número complexo e apenas modificar sua amplitude. Desta forma o encargo de modificar a fase dos valores de entrada pertence apenas aos pesos.

A função pode ser definida como:

$$modReLU(z) = ReLU(|z| + b) \cdot \frac{z}{|z|} \tag{15}$$

Devido ao módulo de um número complexo ser um número real positivo e a função ReLU apenas ser definida para valores reais, conclui-se que b deve ser um número real.

A variável b representa um valor de raio no qual, devido à função ReLU, os números complexos que estejam dentro do mesmo são descartados.

O valor da fase é preservado através do vetor unitário bidimensional obtido através da divisão do número z por seu módulo. Este vetor possui o mesmo valor de fase que o número z, porém o valor de amplitude é igual a um.

2.4.1.2 CReLU.

Outra função muito utilizada segundo Trabelsi et al. (2017) é a chamada CReLU. Esta função baseia-se na aplicação da função ReLU diretamente sobre as partes reais e imaginárias de forma isolada. Uma outra característica desta função é que, ela pode preservar a parte real ou imaginária do número de forma independente se for necessário. Isso pode ser notado ao observar a definição da função:

$$CReLU(z) = ReLU(\Re(z)) + \imath \cdot ReLU(\Im(z)) \tag{16}$$

Pode-se perceber que a função pode retornar valores diferentes de zero para quaisquer números cuja fase esteja entre -90 e 180º. No entanto, quando a fase se encontra entre -90º e 0º, o valor de fase retornado passa a ser 0º, e a amplitude se torna igual à parte real do número. Quando a fase se encontra entre 90° e 180°, a fase retornada é igual a 90º, e a amplitude é a mesma que a parte imaginária do número. Já quando o valor de fase se encontra entre 0º e 90º, o número retornado é o mesmo que o de entrada.

Esta função satisfaz as equações de Cauchy-Riemann quando a parte real e a imaginária compartilham o mesmo sinal.

2.4.1.3 ZReLU.

Outra função descrita por Trabelsi et al. (2017) é a conhecida por ZReLU. A mesma é definida como a seguir:

$$ZReLU(z) = \begin{cases} z, & 0 \leq \theta \leq \frac{\pi}{2} \\ 0 \end{cases} \tag{17}$$

Esta apresenta uma grande semelhança com a função CReLU, no entanto a mesma é holomorfa em quase todo seu domínio, com exceção apenas dos pontos onde:

$$\Re(z) = 0, \Im(z) > 0 \tag{18}$$

$$\Re(z) > 0, \Im(z) = 0 \tag{19}$$

3 Metodologia

Primeiramente foram utilizados os dados do *Dataset* I de EEG sobre o tema de *Motor Imagery* coletados no trabalho de Blankertz et al. (2007) e disponibilizados na edição IV da *BCI Competition* organizada por Blankertz et al. (2008).

Assim como descrito por Blankertz et al. (2007), o *dataset* utilizado é composto por sinais de EEG contínuos, sem interrupção entre as diferentes atividades, classificados em 2 atividades diferentes, e 1 classificação neutra. O sinal foi coletado com 59 canais de EEG, distribuídos principalmente nas áreas relacionadas ao movimento, obtidos através de eletrodos de Ag/AgCl, com uma frequência de amostragem de 1000Hz, 16 bits de resolução (precisão de 0.1uV), e filtrados com um filtro passa-faixa de 0.05Hz e 200Hz. Os dados são provenientes da coleta de sinais de 7 pacientes, seguindo instruções de uma tela, e que não receberam treinamento prévio . Conforme especificado, o paciente recebe um sinal para começar a realizar a atividade mental em um monitor e o sinal é coletado por um período de tempo. As atividades eram então alternadas, com um pequeno intervalo entre elas, referente a classificação neutra. O momento de início de cada atividade foi registrado, com exceção do período de alternância (classificação neutra). Foram utilizados apenas os dados obtidos do paciente A, para remover problemas provenientes das diferenças intrínsecas de cada indivíduo.

Em seguida, foi programado um *script* em python para extrair as características do sinal na forma de frames temporais. Foram utilizados os sinais obtidos de todos os canais. Cada *frame* contém informação referente à um período de 1s de um dos canais. As informações que cada *frame* contém são as densidades espectrais de potência de cada uma das faixas de frêquencia do cérebro, tendo como base as definidas por Teplan (2002). Os sinais referentes à cada faixa isolada foram obtidos através de filtros passa-faixa com as seguintes frequências de corte:

- delta (0.5 - 4HZ)
- theta (4 - 7.5HZ)
- alpha (7.5 - 14HZ)
- beta (14 - 40HZ)
- gamma (40 - 100HZ)

Os filtros passa-faixa implementados foram do tipo de Resposta à impulso finito(FIR), com janela do tipo Kaiser, através das funções da biblioteca em python scipy, para facilitar durante o desenvolvimento. O valor de atenuação utilizado foi de 40 db para banda de corte, e o valor de largura da região de transição de 1 Hz. O que indica que frequencias maiores que 1 Hz para fora da banda passante tiveram uma atenuação de no mínimo 40 db.

Os sinais de cada faixa isolada tiveram uma parte inicial removida referente ao atraso causado pelo processo de

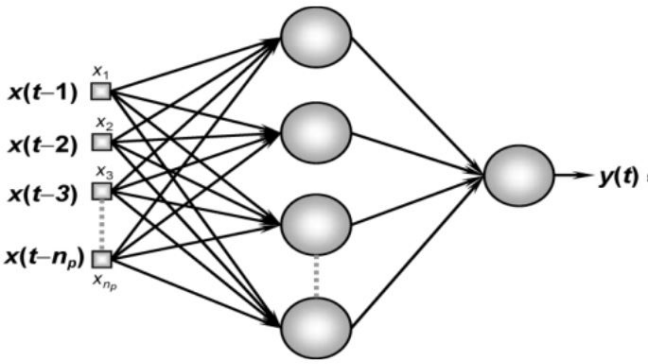

Figura 7: Entradas Atrasadas no tempo
fonte:da Silva et al. (2016)

filtragem, e foram utilizados para extrair os *frames* temporais. Já os momentos de início da atividade foram utilizados (após descontar atraso de filtragem) para obter os rótulos de cada *frame*.

Os *frames* extraídos foram apresentados na forma de vetores concatenados (5 bandas x 59 canais, ou 295 entradas), no formato entradas atrasadas no tempo. Desta forma cada *frame* é apresentado à diferentes variáveis de entrada, com base no instante de sua aquisição. Como mostra a Fig. 7. Após obtido o *dataset*, as entradas foram normalizadas utilizando a função ZScale da biblioteca SCIKIT.

Após a extração destes frames característicos dos dados, foi modificado o código de camadas de valores complexos para o framework Keras disponibilizado por Trabelsi et al. (2017), para torná-lo mais adaptável à diferentes funções de ativação. E foi programada uma função de ativação exponencial complexa baseada nos conceitos da fórmula de Euler e da Série de Fourier, como descrito na Eq. (20), onde C_n representa a amplitude de uma função senoídal referida por $e^{i\omega_n x}$ com frêquencia ω_n, devido ao fato de esta ser uma das equações mais famosas e ser um dos aproximadores de função mais utilizados. A função de ativação baseada na série de Fourier é representada pela Eq. (21), onde a representa a parte real da entrada complexa e a amplitude da função, e b representa a parte imaginária, e por consequência, a fase da senóide.

$$f(x) = \sum_{n=0}^{N} C_n e^{i\omega_n x} \tag{20}$$

$$f(a + bi) = ae^{ib} \tag{21}$$

As modificações ao código de Trabelsi et al. (2017) consistem em permitir o uso de bias n-dimensional (não necessariamente 2 dimensões como permitido pelo código) e alteração da chamada de funções de ativação, passando como parâmetros as partes reais, imaginárias e bias como entidades separadas, ao invés de valores pré-computados.

Para prosseguir, foi utilizado o Grid Search Cross Validation(gscv) da biblioteca SCIKIT para obter a melhor configuração de camadas (entre as variações já preparadas) e divisão de *dataset* para treino e teste na proporção 75/25% (por via de regra, os *datasets* deviam estar balanceados) usando como base 6 diferentes configurações de modelos Perceptron Multicamadas com ativação Relu para as camadas ocultas, e softmax para a camada de saída, variando o número de parâmetros treináveis e número de camadas. Os resultados desta etapa foram utilizados para configurar as diferentes topologias de redes neurais para teste.

Finalizando esta etapa, foram programados scripts contendo os modelos de diversas topologias de redes neurais, tais como redes convolucionais, residuais, LSTM, e MLP, e as mesmas foram treinadas. Também alternando entre as diversas funções de ativação. Um modelo pode pertencer à uma das categorias abaixo:

- Possuir a mesma função de ativação para todas as camadas.
- Possuir a mesma função de ativação para as camadas ocultas, e um outro tipo para a camada de saída.
- Possuir outro arranjo de funções de ativação não correspondente com as opções acima.

Para comparar os resultados de forma consistente, todas as topologias programadas deviam ter aproximadamente o mesmo número de parâmetros à serem aprendidos, ou seja, a somatória do número de pesos e de bias devem ser iguais entre as topologias. O número de camadas também deve ser o mesmo em todos os casos testados. Lembrando que redes de valores complexos possuem metade desta quantidade, uma vez que seus parâmetros são compostos por um valor real e um imaginário.

Em sequência, os dados de entrada para cada rede específica foram organizados da seguinte forma:

- Para as redes de valores complexos foram pareadas as bandas de dois canais, e um dos canais foi pareado com um canal vazio (DEP de todas as bandas igual a zero)
- Para redes convolutivas a entrada foi reordenada em forma matricial de 5 colunas e 59 linhas ao invés de um vetor de 295 colunas
- para redes LSTM a entrada foi preparada na forma de entradas atrasadas no tempo com 2 segundos (duas vezes a entrada comum da rede)

Para avaliar com maior precisão a influencia da função de ativação em cada modelo, foram treinados 100 cópias de cada um por uma quantidade de épocas que foi determinada com base nos resultados da *gscv*, utilizando-se da técnica de *Early Stop* (salvar o modelo na época em que a taxa de acerto de avaliação for a maior) para evitar *overfit* (modelo decorar os dados de treino e não conseguir generalizar). No final os dados de taxa de acerto foram comparados entre cada variação do modelo (avaliou-se cada modelo usando o *dataset* de teste).

Vale ressaltar que os pesos e bias dos modelos foram inicializados com um número aleatório entre -1 e 1, e cada cópia foi inicializada de forma diferente a fim de verificar quais modelos possuem mais tendencia de convergir independente das condições iniciais. Por via de regra, to-

Tabela 1: Resultado de avaliação de modelos para seleção de parâmetros

Topologias	Camadas ocultas	Neurônios por camada	Parâmetros treináveis	Avaliação do Modelo
1	1	80	23.842	55,30%
2	2	60,30	19.652	58,40%
3	3	30,20,10	9.732	65,70%
4	4	30,10,10,10	9.322	63,40%
5	4	20,10,10,10	6.372	59,80%
6	4	10,10,10,10	3.312	56,10%

dos os modelos foram desenvolvidos usando o *framework Keras*, com função de custo Erro Quadrático, utilizando o otimizador SGD(*Stochastic gradient descent*), com taxa de aprendizado de 0,007, corte de gradiente de norma 5, e possui camadas de *dropout* (camadas que aleatoriamente bloqueiam a passagem de uma certa porcentagem definidade de valores entre camadas consecutivas) com valor de 0.4 entre cada camada oculta, com o objetivo de reduzir a tendência ao *overfit*.

Os modelos foram avaliados quanto à sua precisão e capacidade de convergência através da distribuição estatística dos resultados. No final os mesmos foram analisados.

Para poder comparar estes resultados com os da IV Competição de BCI algumas modificações tiveram que ser realizadas devido à natureza da solução. Na competição, os modelos eram de regressão com os rótulos entre -1 e 1. Já neste trabalho, o modelo é de classificação usando *One-hot encoding*. O melhor modelo obtido passou pelo processo de avaliação conforme as normas da competição. Para isto, teve suas respectivas saídas (vetor referente ao *One-hot encoding*) multiplicadas por seus determinados rótulos (-1 e 1) e somadas, e o modelo foi avaliado usando a função de erro quadrático médio utilizando o *dataset* de avaliação disponibilizado pela competição.

4 Resultados e Discussão

Através da execução do *gscv* foi possível verificar qual configuração foi mais efetiva. Com base nos resultados da Tabela 1, pode-se perceber que a topologia com melhor desempenho foi a Topologia 3, com 3 camadas ocultas.

Com estes resultados sabe-se então que para realizar uma comparação válida e com bons resultados é necessário que os modelos comparados tenham aproximadamente 9732 parâmetros treináveis, e 4 camadas (contando a camada de saída, e ignorando as camadas de Dropout). Também é desejável que o número de neurônios por camada seja aproximadamente igual, no entanto para alguns modelos, como LSTM e Convolucional, isto seria difícil de alcançar.

Para entender melhor o funcionamento do processo de treinamento da Topologia 3, foram plotados os valores de precisão e loss de treino e validação durante cada uma das 4000 épocas ao qual o mesmo foi treinado. Os resultados podem ser vistos nas Figs. 8 e 9.

Ao realizar a análise dos resultados vistos nas Figs. 8 e 9, percebe-se que o modelo começa a apresentar sinais de *overfit* entre 1000 a 1500 épocas de treinamento. Por tanto, a fim de evitar que os demais modelos passassem por este caso, cada cópia do modelo foi treinada por apenas 1000 épocas. Também foram estimadas e montadas as arquiteturas das topologias com os parametros vistos na

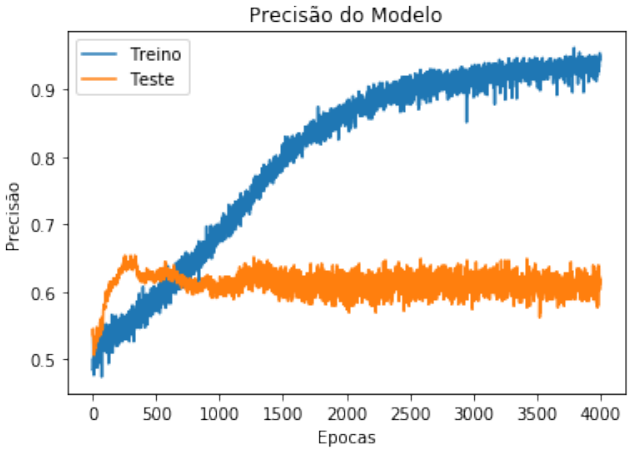

Figura 8: Comparação entre taxa de acerto de treinamento e de teste

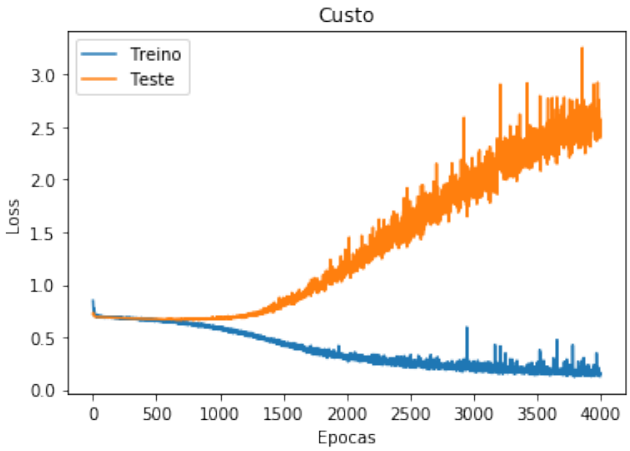

Figura 9: Comparação entre Loss de treinamento e de teste

Tabela 2, e com as funções de ativação da Tabela 3. Totalizando então 31 modelos testados que.

Com base nos resultados da Tabela 4, é possível verificar a eficácia dos modelos apresentados, assim como sua convergência. Para facilitar o processo de análise dos resultados dos modelos, foram plotados diagramas de caixa estreita dos modelos, divididos por topologia e função de ativação da última camada.

Tabela 2: Especificações das topologias de redes neurais e número de paramêtros

Topologia	Camada 1	Camada 2	Camada 3	Camada 4	Parâmetros
MLP	Densa 30	Densa 20	Densa 10	Densa 2	9732
MLP Complexa*	Complexa 15	Complexa 10	Complexa 5	Complexa 1	bias real:9762 bias complexo: 9824
LSTM 1	LSTM 7	Densa 36	Densa 25	Densa 2	9749
LSTM 2	LSTM 7	LSTM 13	Densa 10	Densa 2	9738
Convolucional	Filtro 50 [5:1]	Filtro 20 [1:5]	Densa 4	Densa 2	9734
Rede Residual**	Densa 27	Res-Densa 27	Res-Densa 27	Densa 2	9560

* MLP Complexa pode ter bias unidimensional (real) ou bidimensional (complexo).
** o bloco residual ocupa duas camadas (Res-Densa), e a realimentação é realizada na última camada.

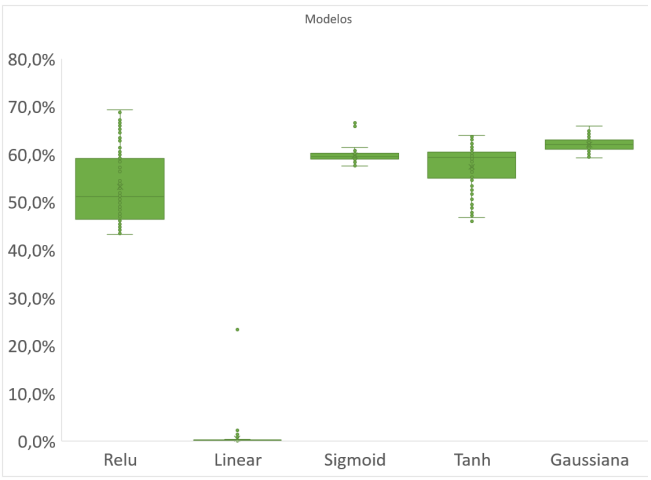

Figura 10: Distribuição de precisão de modelos MLP com apenas uma função de ativação

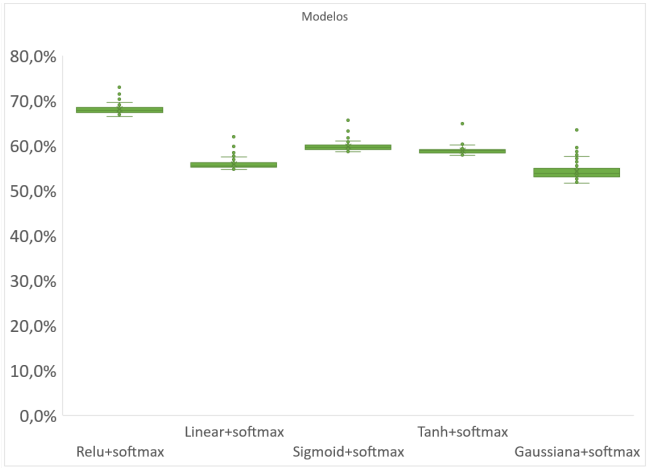

Figura 12: Distribuição de precisão de modelos MLP com função de ativação da camada de saída Softmax

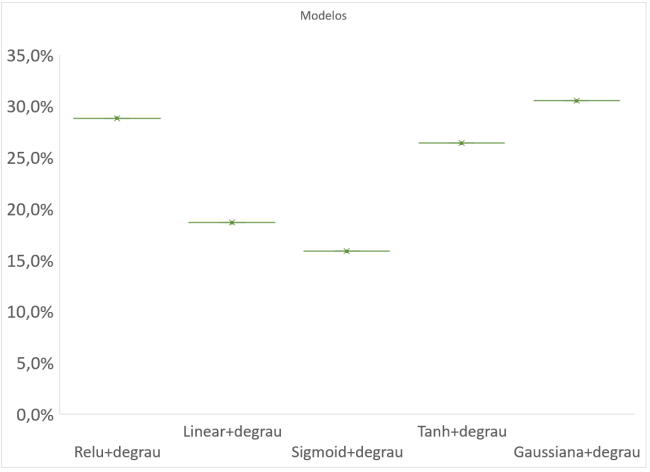

Figura 11: Distribuição de precisão de modelos MLP com função de ativação da camada de saída Degrau

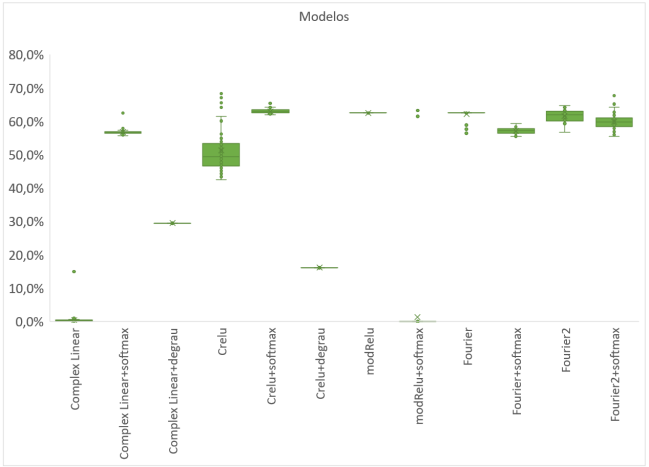

Figura 13: Distribuição de precisão de modelos MLP Complexos

Tabela 3: Função de ativação por camada dos modelos

Topologia	Identificação	Camada 1	Camada 2	Camada 3	Camada 4
Convolucional	Convolucional	Relu	Relu	Relu	Softmax
MLP	Gaussiana	Gaussiana	Gaussiana	Gaussiana	Gaussiana
	Gaussiana+degrau	Gaussiana	Gaussiana	Gaussiana	Degrau
	Gaussiana+softmax	Gaussiana	Gaussiana	Gaussiana	Softmax
	Linear	Linear	Linear	Linear	Linear
	Linear+degrau	Linear	Linear	Linear	Degrau
	Linear+softmax	Linear	Linear	Linear	Softmax
	Relu	Relu	Relu	Relu	Relu
	Relu+degrau	Relu	Relu	Relu	Degrau
	Relu+softmax	Relu	Relu	Relu	Softmax
	Sigmoid	Sigmoid	Sigmoid	Sigmoid	Sigmoid
	Sigmoid+degrau	Sigmoid	Sigmoid	Sigmoid	Degrau
	Sigmoid+softmax	Sigmoid	Sigmoid	Sigmoid	Softmax
	Tanh	Tanh	Tanh	Tanh	Tanh
	Tanh+degrau	Tanh	Tanh	Tanh	Degrau
	Tanh+softmax	Tanh	Tanh	Tanh	Softmax
MLP Complexa	Crelu	Crelu	Crelu	Crelu	Crelu
	Crelu+degrau	Crelu	Crelu	Crelu	Degrau
	Crelu+softmax	Crelu	Crelu	Crelu	Softmax
	Fourier	Fourier	Fourier	Fourier	Fourier
	Fourier+softmax	Fourier	Fourier	Fourier	Softmax
	Fourier2	Fourier	Crelu	Crelu	Crelu
	Fourier2+softmax	Fourier	Crelu	Crelu	Softmax
	Complex Linear	Complex Linear	Complex Linear	Complex Linear	Complex Linear
	Complex Linear+degrau	Complex Linear	Complex Linear	Complex Linear	Degrau
	Complex Linear+softmax	Complex Linear	Complex Linear	Complex Linear	Softmax
	modRelu	modRelu	modRelu	modRelu	modRelu
	modRelu+softmax	modRelu	modRelu	modRelu	Softmax
Residual Network	ResNet	Relu	Relu	Relu	Softmax
Recursiva	LSTM1	LSTM	Relu	Relu	Softmax
	LSTM2	LSTM	LSTM	Relu	Softmax

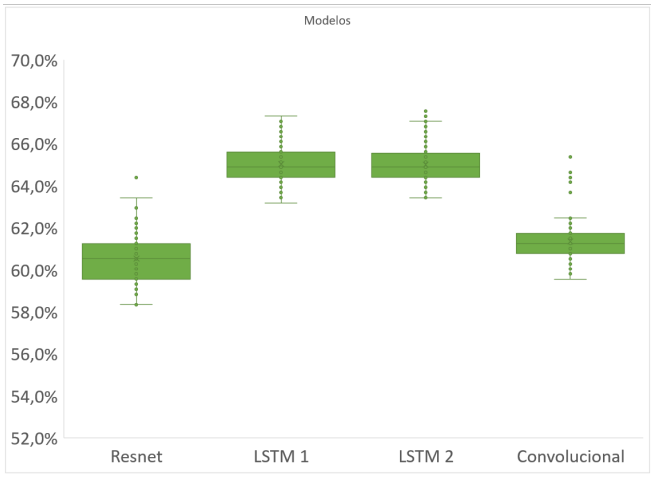

Figura 14: Distribuição de precisão de modelos das outras Topologias

Através da observação das Figs. 10 a 14, assim como a Tabela 4, pode-se perceber que a topologia que apresentou os melhores resultados para o problema de classificação de sinais EEG para aplicações BCI, foi a topologia MLP, com o modelo Relu+Softmax, tendo esta configuração alcançado o maior valor de precisão mínimo, médio, e máximo. Conforme pode ser visto nas Tabelas 5 e 6.

Tabela 5: Modelos com melhor taxa de acerto média

Topologia	Identificação	Média
MLP	Relu+softmax	68,1%
Recursiva	LSTM 1	65,0%
Recursiva	LSTM 2	65,0%
MLP Complexa	Crelu+softmax	63,0%
MLP Complexa	modRelu	62,5%
MLP Complexa	Fourier	62,2%

Tabela 6: Modelos com melhor taxa de acerto máxima

Topologia	Identificação	Máxima
MLP	Relu+softmax	72,9%
MLP	Relu	69,4%
MLP Complexa	Crelu	68,6%
MLP Complexa	Fourier2+softmax	68,3%
Recursiva	LSTM 2	67,6%
Recursiva	LSTM 1	67,3%

Tabela 4: Distribuição de valores de precisão dos modelos treinados.

Topologia	Identificação	Mínimo	Média	Máxima	Desvio Padrão
Convolucional	Convolucional	59,6%	61,4%	65,4%	1,0%
MLP	Gaussiana	59,3%	62,1%	66,0%	1,5%
	Gaussiana+degrau	30,5%	30,5%	30,5%	0,0%
	Gaussiana+softmax	51,6%	54,2%	63,4%	1,8%
	Linear	0,1%	0,5%	23,4%	2,3%
	Linear+degrau	18,6%	18,6%	18,6%	0,0%
	Linear+softmax	54,7%	55,8%	62,0%	1,1%
	Relu	43,3%	53,3%	69,4%	7,6%
	Relu+degrau	28,8%	28,8%	28,8%	0,0%
	Relu+softmax	66,4%	68,1%	72,9%	1,0%
	Sigmoid	57,6%	59,8%	66,6%	1,2%
	Sigmoid+degrau	15,9%	15,9%	15,9%	0,0%
	Sigmoid+softmax	58,6%	59,7%	65,6%	0,9%
	Tanh	46,0%	57,4%	64,0%	4,6%
	Tanh+degrau	26,4%	26,4%	26,4%	0,0%
	Tanh+Softmax	57,9%	58,8%	64,9%	0,8%
MLP Complexa	Crelu	42,5%	51,3%	68,6%	6,5%
	Crelu+degrau	16,1%	16,1%	16,1%	0,0%
	Crelu+softmax	62,0%	63,0%	65,4%	0,6%
	Fourier	56,4%	62,2%	62,5%	1,1%
	Fourier+Softmax	55,4%	57,1%	59,3%	0,9%
	Fourier2	56,7%	61,6%	64,6%	1,7%
	Fourier2+softmax	55,4%	59,9%	68,3%	2,3%
	Complex Linear	0,1%	0,4%	14,9%	1,5%
	Complex Linear+degrau	29,4%	29,4%	29,4%	0,0%
	Complex Linear+softmax	55,7%	56,8%	62,5%	0,8%
	modRelu	62,5%	62,5%	62,5%	0,0%
	modRelu+softmax*	0,0%	1,2%	63,2%	8,8%
Residual Network	ResNet	58,4%	60,5%	64,4%	1,2%
Recursiva	LSTM 1	63,2%	65,0%	67,3%	0,8%
	LSTM 2	63,4%	65,0%	67,6%	0,9%

* O modelo da topologia MLP Complexa com identificação modRelu+softmax sofreu de estouro de gradiente em 98 das 100 instâncias e portanto não fez parte da avaliação.

Tabela 7: Configurações com menor desvio padrão entre os resultados de precisão(excluindo configurações com função degrau)

Topologia	Identificação	Desvio Padrão
MLP Complexa	modRelu	0,0%
MLP Complexa	Crelu+softmax	0,6%
MLP Complexa	Complex Linear+softmax	0,8%
MLP	Tanh+Softmax	0,8%
Recursiva	LSTM 1	0,8%
MLP Complexa	Fourier+Softmax	0,9%

Ainda com base nas Tabelas 5 a 7, percebe-se que os modelos recursivos (LSTM), possuem um bom desempenho médio e máximo, aparencendo duas vezes entre os 6 melhores em ambas as tabelas e aparecendo entre os 6 com maior convergência uma vez. Os modelos da Topologia MLP complexa também apresentaram resultados bons, aparecendo três vezes entre os 6 com melhor acerto médio, duas vezes entre os de acerto máximo, e quatro vezes entre os de maior convergência. Incluindo o modelo com função modRelu, que convergiu para os mesmos resultados em todos os testes. Quanto aos modelos propostos de função de ativação Fourier, os mesmos aparecem em sexto lugar em acerto médio (modelo Identificado como Fourier) quarto lugar em acerto máximo (modelo Identificado como Fourier2+softmax).

Conforme a Fig. 11 e a Tabela 4, todos os modelos com função degrau apresentaram desvio padrão 0 e taxas de acerto inferiores à 50%, portanto, foram excluídos da comparação.

O modelo relu+softmax passou pelo processo de avaliação utilizado por Blankertz et al. (2008) na BCI Competition IV e obteve resultado para o paciente A equivalente à **0,29**. Ao comparar com o modelo da competição que obteve o menor valor de erro para este paciente, que diz respeito ao segundo colocado da competição para o *dataset* I, Dieter Devlaminck, obteve o resultado de **0,35**.

5 Conclusão

No geral os resultados dos experimentos foram excelentes em comparação à um modelo de resposta aleatória, que teria uma taxa de acerto de aproximadamente 33,3%, o que implica que os modelos foram capazes de identificar algumas características do sinal. Através dos resultados foi possível verificar o funcionamento de diversas topologias de redes nesta aplicação específica, e ainda obter um resultado melhor que os anunciados por Blankertz et al.

(2008) na BCI Competition IV para os dados trabalhados.

Observou-se nos resultados que os modelos que utilizaram a função degrau sempre convergiram para modelos com taxa de acerto idênticas, o que também ocorreu para o modelo modRelu. Isto provavelmente se deu por conta da ausência de gradientes em alguns pontos (devido à forma que o *Tensorflow* usa para calcular os gradientes).

Também pode-se ver que, os modelos com função de ativação de camada saída *softmax* tiveram maior convergência de resultados (ignorando modelos com função degrau), e estes tenderam a ter maior taxa de acerto.

Quanto às redes de valores complexos, estas apresentaram resultados bons, com taxa de convergência e acerto relativamente altas em comparação aos modelos avaliados, e poderiam ser estudadas mais profundamente.

Com base nas análises, observou-se que o modelo *Relu+softmax* apresentou os melhores resultados no geral. Isto pode ser devido ao experimento da Tabela 1 ter sido realizado com esse mesmo tipo de modelo, podendo ter sido encontrada a melhor configuração para modelos do tipo Relu+softmax, ao invés de ter sido encontrada a melhor configuração no geral. No entanto, isto não passa de especulação, não havendo dados que comprovem esta hipótese. Para resolver esta dúvida seria necessário realizar esta etapa novamente com os outros modelos também, o que aumentaria a complexidade do experimento, e com os recursos disponíveis não seria viável.

Os principais fatores limitantes no experimento foram a quantidade de dados (referente à aproximadamente 30 minutos de sinais) e poder de processamento para processar modelos mais complexos e profundos. Para tentar obter melhores resultados poderiam ser aplicadas técnicas de PCA (Análise de componentes principais), diminuição do período dos frames (para evitar *overlap* de rótulos), aumento da janela de entrada das redes, aplicação de CSP nas amostras, escolha de parâmetros usando algoritmos de meta-heurística e algoritmo genético, ou qualquer outro método para melhoria da etapa de pré-processamento

Agradecimentos

Agradeçemos primeiramente a Deus pela vida, aos familiares por todo o apoio e esforço, aos professores pelos ensinamentos, e aos amigos pela amizade e apoio.

Referências

Blankertz, B., Dornhege, G., Krauledat, M., Müller, K.-R. and Curio, G. (2007). The non-invasive berlin brain–computer interface: Fast acquisition of effective performance in untrained subjects, *NeuroImage* **37**(2): 539–550. https://doi.org/10.1016/j.neuroimage.2007.01.051.

Blankertz, B., Vidaurre, C., Tangermann, M., Müller, K.-R., Brunner, C., Leeb, R., Müller-Putz, G., Schlögl, A., Pfurtscheller, G., Waldert, S., Mehring, C., Aertsen, A., Birbaumer, N., Miller, K. J. and Schalk, G. (2008). *BCI Competition IV*. Available at http://www.bbci.de/competition/iv/.

Cunha, F. L., Franca, J. E. M., Ortolan, R. L. and Junior, A. C. (2000). O uso de redes neurais artificiais para o reconhecimento de padrões em uma prótese mioelétrica de mão, pp. 339–342.

da Silva, I. N., Spatti, D. H. and Flauzino, R. A. (2016). *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas*, 2 edn, Artliber Editora Ltda, São Paulo, SP, Brasil.

Fahimi, F., Zhang, Z., Goh, W. B., Lee, T.-S., Ang, K. K. and Guan, C. (2019). Inter-subject transfer learning with an end-to-end deep convolutional neural network for EEG-based BCI, *Journal of Neural Engineering* **16**(2): 026007. https://doi.org/10.1088/1741-2552/aaf3f6.

Ferneda, E. (2006). Redes neurais e sua aplicação em sistemas de recuperação de informação, *Ciência da Informação* **35**: 25 – 30. Available at: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652006000100003&nrm=iso.

Forslund, P. (2003). *A neural network based brain-computer interface for classification of movement related eeg*, Master of science in applied physics and electrical engineering, Linköping University. Available at http://liu.diva-portal.org/smash/get/diva2:21837/FULLTEXT01.pdf.

Gao, Z., Dang, W., Wang, X., Hong, X., Hou, L., Ma, K. and Perc, M. (2021). Complex networks and deep learning for eeg signal analysis, *Cognitive Neurodynamics* **15**(3): 369–388. https://doi.org/10.1007/s11571-020-09626-1.

Garcia, V. G. and Maia, A. G. (2014). Características da participapação das pessoas com deficiência e/ou limitação funcional no mercado de trabalho brasileiro, *Revista Brasileira de Estudos de População* **31**: 395 – 418. https://doi.org/10.1590/S0102-30982014000200008.

Guberman, N. (2016). *On complex valued convolutional neural networks*, Master of science, School of Computer Science and Engineering. https://doi.org/10.48550/arXiv.1602.09046.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Las Vegas, Nevada, USA. https://doi.org/10.1109/CVPR.2016.90.

Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning, *CoRR* **abs/1811.03378**. https://doi.org/10.48550/arXiv.1811.03378.

Ochoa, J. B. (2002). Eeg signal classification for brain computer interface applications, *Ecole Polytechnique Federale De Lausanne* **7**: 1–72. Available at https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.6148&rep=rep1&type=pdf.

Ortolan, R. L., Itiki, C. and Junior, A. C. (2004). Análise de parâmetros temporais e espectrais do EMG para classificação de diferentes padrões de contrações, *Proceedings of the International Federation for Medical and Biological*

Eng., Vol. 5, IFMBE, João Pessoa, Paraíba, Brasil, pp. 1151–1154.

Sak, H., Senior, A. W. and Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, *CoRR* **abs/1402.1128**. https://doi.org/10.48550/arXiv.1402.1128.

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W. and Ball, T. (2017). Deep learning with convolutional neural networks for eeg decoding and visualization, *Human Brain Mapping* **38**(11): 5391–5420. https://doi.org/10.1002/hbm.23730.

Shi, T., Cui, W. and Ren, L. (2019). Multimedia remote interactive operations based on eeg signals constructed bci with convolutional neural network, *Multimedia Tools and Applications* . https://doi.org/10.1007/s11042-019-7338-5.

Teplan, M. (2002). Fundementals of eeg measurement, *Measurement Science Review* **2**: 1–11.

Trabelsi, C., Bilaniuk, O., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y. and Pal, C. J. (2017). Deep complex networks, *CoRR* **abs/1705.09792**. https://doi.org/10.48550/arXiv.1705.09792.

Yuksel, A. and Olmez, T. (2015). A neural network-based optimal spatial filter design method for motor imagery classification, *PLOS ONE* **10**(5): e0125039. https://doi.org/10.1371/journal.pone.0125039.