UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

MAURÍCIO MUSSATTO SCOPEL

RECONHECIMENTO DE IMAGÉTICA MOTORA UTILIZANDO ELETROENCEFALOGRAFIA

CAXIAS DO SUL

2020

MAURÍCIO MUSSATTO SCOPEL

RECONHECIMENTO DE IMAGÉTICA MOTORA UTILIZANDO ELETROENCEFALOGRAFIA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dr. André Gustavo Adami

CAXIAS DO SUL

2020

MAURÍCIO MUSSATTO SCOPEL

RECONHECIMENTO DE IMAGÉTICA MOTORA UTILIZANDO ELETROENCEFALOGRAFIA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 01/12/2020

BANCA EXAMINADORA

_____

Prof. Dr. André Gustavo Adami
Universidade de Caxias do Sul - UCS

_____

Prof. Dra. Adriana Miorelli Adami
Universidade de Caxias do Sul - UCS

_____

Prof. Dr. Guilherme Holsbach Costa
Universidade de Caxias do Sul - UCS

RESUMO

Pessoas acometidas de doenças degenerativas muito graves possuem atividade cerebral como qualquer outra pessoa. Entretanto, elas possuem obstruções nas vias normais de comunicação responsáveis pelo controle dos nervos e músculos. Nos casos mais extremos, essas obstruções impossibilitam completamente a realização de tarefas diárias. A imagética motora, que consiste na imaginação da movimentação de um membro, sem efetivamente executá-lo, vem sendo explorada para o desenvolvimento de interfaces cérebro-máquina que permitam a execução de algumas tarefas básicas do dia a dia. O principal componente de uma interface cérebro-máquina é o algoritmo para o reconhecimento de imagética motora a partir de sinais cerebrais. Este trabalho propõe um sistema para o reconhecimento de múltiplas classes de imagética motora através de sinais de EEG que utiliza uma Rede Neural Convolucional. Visando o cenário de classificação independente do sujeito, os sinais foram normalizados de forma a reduzir a variabilidade entre sinais provenientes de diferentes sujeitos, e de diferentes sessões de aquisição para um mesmo sujeito. Para a avaliação de desempenho foram utilizadas algumas bases de dados públicas. A principal delas é a disponibilizada pela *PhysioNet* uma vez que, para o melhor do nosso conhecimento, é a base de dados de sinais de EEG de imagética motora pública com o maior número de sujeitos disponível atualmente. Foram utilizadas também bases de dados da IV Competição de Interfaces Cérebro-Maquina, muito utilizadas em trabalhos de classificação de imagética motora presentes na literatura. Na base de dados da *PhysioNet*, o sistema obteve uma acurácia média de $82.11\%(\pm 3.65\%)$, no cenário de classificação das classes punho esquerdo e punho direito, independente do sujeito. Esse resultado se demonstra competitivo, uma vez que não apresenta diferença significativa com os melhores resultados encontrados na literatura. No cenário de classificação de 4 classes (olhos fechados, punho esquerdo, punho direito e pés), o sistema obteve uma acurácia média de $66.11\%(\pm 4.46\%)$, apresentando uma melhoria relativa de 1.6% em relação ao melhor resultado encontrado na literatura. Na base de dados II-B da IV Competição de interfaces cérebro-máquina, o sistema obteve uma acurácia média de $71.14\%(\pm 5.63\%)$, na classificação das classes mão esquerda e mão direita, independente do sujeito.

Palavras-chaves: Interface cérebro-máquina. Eletroencefalografia. Aprendizado de Máquina. Aprendizado Profundo. Redes Neurais Convolucionais. Redes Neurais Recorrentes.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	*Area Under the Curve*
CNN	*Convolutional Neural Networks*
CRNN	*Convolutional Recurrent Neural Network*
CSP	*Common Spatial Pattern*
CWT	*Continuous Wavelet Transform*
DWT	*Discrete Wavelet Transform*
ECoG	Eletrocorticografia
EEG	Eletroencefalografia
ELA	Esclerose Lateral Amiotrófica
EMD	*Empirical Mode Decomposition*
ERD	*Event-Related Desynchronization*
ERP	*Event-Related Potential*
ERS	*Event-Related Synchronization*
FC	*Fully Connected*
fMRI	*functional Magnetic Resonance Imaging*
fNIRS	*functional Near InfraRed Spectroscopy*
FBCSP	*Filter Bank Common Spatial Pattern*
FN	Falso Negativo
FP	Falso Positivo
Hz	*Hertz*
LDA	*Linear Discriminant Analysis*
LSTM	*Long Short-Term Memory*
MEG	Magnetoencefalografia
MLP	*Multilayer Perceptron*

NBPW	*Naïve Bayesian Parzen Window*
PSD	*Power Spectral Density*
SAE	*Stacked Autoencoders*
SBCSP	*Sub Band Common Spatial Pattern*
SCP	*Slow Cortical Potential*
SMR	*Sensorimotor Rhythm*
SSEP	*Steady-State Evoked Potential*
STFT	*Short-Time Fourier Transform*
SSVEP	*Steady-State Visual Evoked Potential*
SVM	*Support Vector Machine*
TSCNN	*Temporal-Spatial Convolutional Neural Network*
RNN	*Recurrent Neural Network*
ROC	*Receiver Operating Characteristic*
VEP	*Visual Evoked Potential*
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
WAMP	*Willison Amplitude*
WPD	*Wavelet Packet Decomposition*

SUMÁRIO

1 INTRODUÇÃO

Milhares de pessoas ao redor do mundo enfrentam dificuldades motoras devido ao acometimento de doenças degenerativas do sistema motor ou pelo sofrimento de algum acidente que tenha ocasionado a paralisia de membros. No Brasil, conforme o Censo 2012 realizado pelo IBGE, 2.33% da população brasileira apresentava alguma deficiência motora severa. Essas pessoas sofrem enormes dificuldades para a realização de suas tarefas diárias. Nos casos mais extremos, perdem completamente a capacidade de comunicação com o mundo externo. Tecnologias assistivas para o auxílio da execução de tarefas do dia a dia e recuperação dessas pessoas são de extrema importância. Interfaces cérebro-máquina são dispositivos que podem ser utilizados para este fim, uma vez que permitem a comunicação com o mundo externo através da análise dos sinais cerebrais dos sujeitos. A imagética motora é um tipo de sinal cerebral que pode ser utilizado para o desenvolvimento de interfaces cérebro-máquina. Consiste na imaginação da movimentação de um membro, sem efetivamente executá-lo. Interfaces cérebro-máquina baseadas em imagética motora permitem a comunicação com o mundo externo através da intenção de execução de determinados movimentos. Dessa forma, não requer quaisquer atividades nas vias normais de comunicação responsáveis pelo controle dos nervos e músculos, podendo ser utilizada em pacientes completamente inabilitados.

Neste trabalho serão discutidos dois aspectos fundamentais para o desenvolvimento de uma interface cérebro-máquina baseada em imagética motora: os métodos para o registro de sinais cerebrais e os algoritmos para o reconhecimento de imagética motora. O registro de sinais cerebrais pode ser realizado através de métodos invasivos ou não invasivos. Na década de 1920, o psiquiatra alemão Hans Berger descobriu que era possível registrar a atividade elétrica proveniente do cérebro humano através de eletrodos posicionados sobre o escalpo. Hans Berger foi o responsável pelo nascimento do termo eletroencefalografia (EEG), um método não invasivo para o registro de sinais cerebrais. A EEG é uma ferramenta essencial na neurociência e no contexto de interfaces cérebro-máquina é o método de registro de sinais cerebrais mais utilizado, devido principalmente a sua praticidade para utilização pelos sujeitos. Para este trabalho serão utilizados sinais de EEG de imagética motora de bases de dados públicas. Para o reconhecimento de imagética motora, algoritmos de aprendizado de máquina podem ser utilizados. Neste trabalho serão discutidos os algoritmos baseados em engenharia de características e em aprendizado profundo. Será realizada uma revisão da literatura de ambas abordagens para permitir a avaliação de seus pontos positivos e negativos, de forma a possibilitar a implementação de um sistema adequado para o reconhecimento de imagética motora independente do sujeito.

1.1 OBJETIVOS

O objetivo deste trabalho é implementar um sistema baseado em um algoritmo de aprendizado de máquina que reconheça, a partir de sinais de EEG, múltiplas classes de imagética motora independente do sujeito. Para atender o objetivo deste trabalho, as seguintes etapas devem ser realizadas:

1. Revisar literatura de algoritmos do estado da arte para o reconhecimento de imagética motora para embasar a proposta de algoritmo;
2. Selecionar bases de dados públicas de sinais de EEG de imagética motora;
3. Preparar os sinais de EEG das bases de dados selecionadas para utilização como dados de entrada do sistema de reconhecimento de imagética motora;
4. Implementar sistema de reconhecimento de imagética motora;
5. Gerar métricas para a avaliação de desempenho do sistema implementado nas bases de dados selecionadas.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta o referencial teórico, onde são apresentados os conceitos básicos da neurociência envolvidos na geração da atividade cerebral e os fatores envolvidos no projeto de uma interface cérebro-máquina. Por fim, são apresentadas diferentes abordagens para o desenvolvimento de sistemas de reconhecimento de imagética motora, e métricas comumente utilizadas para avaliação de desempenho destes sistemas.
- O Capítulo 3 apresenta como proposta de solução para o objetivo deste trabalho, a arquitetura de um sistema para o reconhecimento de múltiplas classes de imagética motora independente do sujeito. Três bases de dados públicas são apresentadas para a avaliação de desempenho do sistema. Os experimentos realizados, análises e resultados obtidos, são apresentados para as três bases de dados, juntamente com comparações com resultados encontrados na literatura.
- O Capítulo 4 apresenta as conclusões obtidas com o desenvolvimento do trabalho.

2 INTERFACES CÉREBRO-MÁQUINA

Neste capítulo serão apresentados os conceitos necessários para o desenvolvimento de uma interface cérebro-máquina baseada em imagética motora. A Seção 2.1 define interfaces cérebro-máquina. Na Seção 2.2 os conceitos básicos de neurociência são apresentados para dar a compreensão de como a atividade cerebral é gerada. Na Seção 2.3 são apresentadas as diferentes classificações de interfaces cérebro-máquina existentes. Na Seção 2.4 são apresentados os métodos disponíveis para o registro de sinais cerebrais, definindo suas vantagens e desvantagens para utilização em uma interface cérebro-máquina. Na Seção 2.5 são apresentados os diferentes tipos de sinais cerebrais utilizados para o desenvolvimento de interfaces cérebro-máquina. Na Seção 2.6 é apresentado o conceito de imagética motora, objeto de estudo deste trabalho. Na Seção 2.7 são apresentadas diferentes abordagens de algoritmos para o reconhecimento de imagética motora, visando a identificação da melhor abordagem disponível atualmente. Na Seção 2.8 são apresentadas as considerações finais deste capitulo.

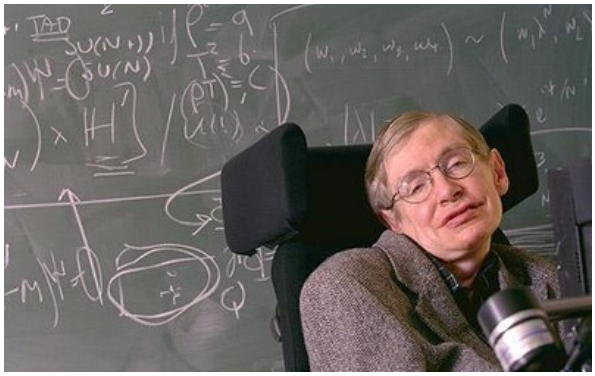
2.1 DEFINIÇÃO

O termo interface cérebro-máquina (do inglês, *Brain-Computer Interface*) foi cunhado por Vidal (1973). Ele definiu uma interface cérebro-máquina como a utilização de sinais cerebrais em um diálogo entre o homem e a máquina. Desde então houveram enormes avanços no poder computacional e no campo de aprendizado de máquina, que possibilitam hoje definir uma interface cérebro-máquina como um novo meio de comunicação em tempo real entre o homem e a máquina. Dessa forma é possível o desenvolvimento de tecnologias assistivas para o reestabelecimento de comunicação com o mundo externo para pacientes acometidos de doenças degenerativas do sistema nervoso. Um exemplo deste tipo de doença é a Esclerose Lateral Amiotrófica (ELA), a qual, de forma progressiva, degenera o sistema nervoso levando a paralisia motora irreversível. O físico teórico Stephen Hawking (Figura 1), falecido em 2018, sofria de ELA e no decorrer de sua vida foi perdendo de forma progressiva o controle motor até o ponto em que conseguia comunicar-se somente através de um sintetizador de fala. Interfaces cérebro-máquina também podem ser criadas como dispositivos de entretenimento para serem utilizadas, por exemplo, em jogos eletrônicos (COYLE *et al.*, 2011).

A estrutura básica de uma interface cérebro-máquina (Figura 2) é composta por um sistema de aquisição de sinais cerebrais, um algoritmo para a decodificação destes sinais em diferentes categorias e uma ponte para a comunicação deste algoritmo com algum dispositivo inteligente. O algoritmo para decodificação dos sinais cerebrais pode ser composto nos seguintes componentes: pré-processamento, engenharia de características e classificação. A partir desta

[1] Disponível em: <http://www.hawking.org.uk/images.html> Acesso em mar. 2020.

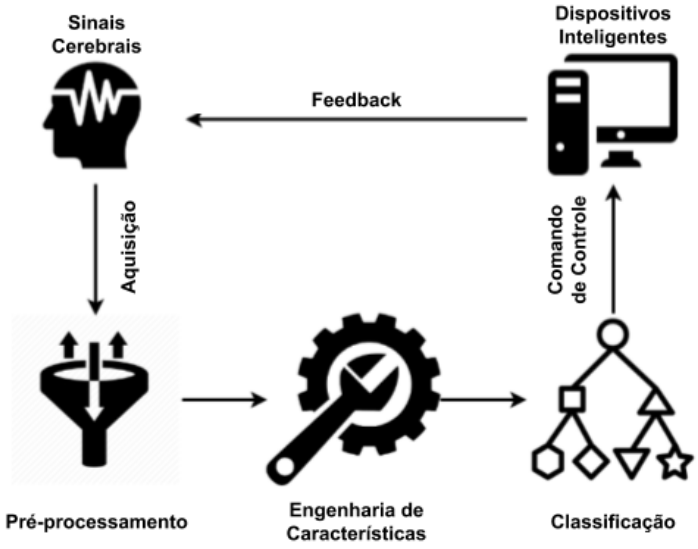
Figura 1 – Stephen Hawking.



Fonte: Site oficial do Stephen Hawking.[1]

classificação, um comando de controle é enviado a um dispositivo que irá executar diferentes ações de acordo com o comando de controle recebido.

Figura 2 – Estrutura básica de uma interface cérebro-máquina.



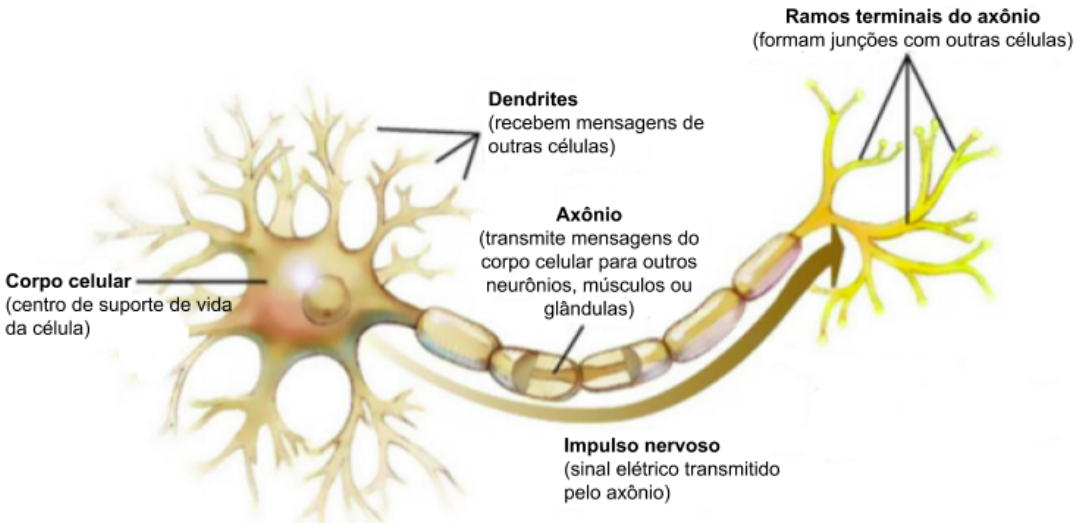
Fonte: Adaptado de Zhang *et al.* (2019).

2.2 NEUROCIÊNCIA BÁSICA

O sistema nervoso é composto por dois tipos de células, os neurônios e as glias. Os neurônios são responsáveis por todo o processamento das informações, enquanto as células gliais contribuem para a sustentação e nutrição dos neurônios mais próximos. Um encéfalo adulto possui aproximadamente o mesmo número de neurônios e de células gliais (cerca de 85 bilhões cada) (BEAR; CONNORS; PARADISO, 2017). Como ilustrado na Figura 3, um neurônio possui três componentes principais: o corpo celular, o axônio e o dendrite. O axônio funciona como um fio condutor pelo qual são transmitidas informações, podendo se estender de menos

de um milímetro para mais de um metro de comprimento, para realizar a comunicação com diferentes partes do sistema nervoso (BEAR; CONNORS; PARADISO, 2017). A terminação do axônio se dá no ponto de contato do axônio com a célula pós-sináptica, que pode ser um dendrite ou o corpo celular de outro neurônio, conhecido como neurônio pós-sináptico (BEAR; CONNORS; PARADISO, 2017). Neste ponto de contato é onde ocorre o processo de troca de informações de um neurônio para outro conhecido como sinapse. De forma geral, impulsos nervosos chegam às terminações axonais, liberando neurotransmissores, que são percebidos pelos receptores das células pós-sinápticas, desencadeando assim a geração de sinais elétricos ou químicos nestas células (BEAR; CONNORS; PARADISO, 2017).

Figura 3 – Estrutura básica de um neurônio.



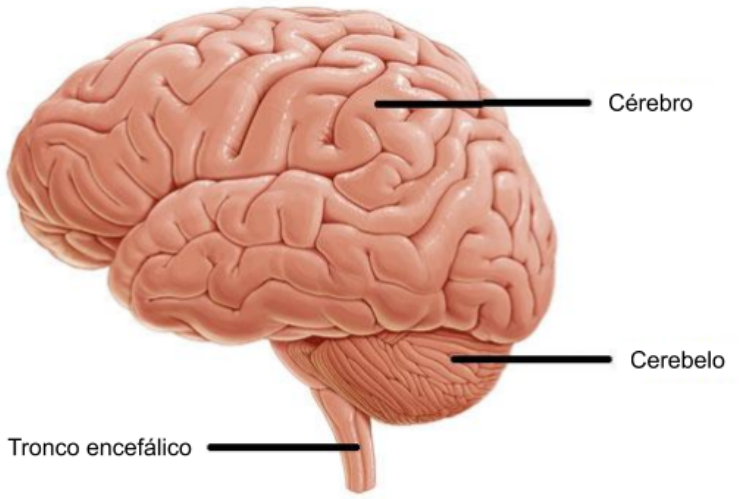
Fonte: Adaptado de Siuly, Li & Zhang (2016).

O sistema nervoso possui duas divisões: o sistema nervoso central e o sistema nervoso periférico. O sistema nervoso central é composto pelo encéfalo e pela medula espinhal. O sistema nervoso periférico consiste em todas as partes do sistema nervoso que não estão nem no encéfalo e nem na medula espinhal (BEAR; CONNORS; PARADISO, 2017). A partir dessa divisão, pode-se categorizar os axônios como aferentes e eferentes. Os aferentes são aqueles que trazem informações para o sistema nervoso central, enquanto que os eferentes são aqueles que emergem do sistema nervoso central para se comunicarem com músculos e glândulas (BEAR; CONNORS; PARADISO, 2017).

O encéfalo pode ser subdividido em três grandes partes: o cérebro, o cerebelo e o tronco encefálico (Figura 4). O cérebro é a maior área do encéfalo e está associado a funções relacionadas aos pensamentos, movimentos, emoções e funções motoras. A segunda maior parte do encéfalo, o cerebelo, está localizada na parte inferior traseira do encéfalo e possui extensas conexões com o cérebro e a medula espinhal, funcionando como uma área para o controle motor, a percepção sensorial e a coordenação. Embora o cerebelo seja menor que o cérebro, possui

aproximadamente a mesma quantidade de neurônios (BEAR; CONNORS; PARADISO, 2017). O tronco encefálico, localizado na parte inferior do encéfalo, funciona como uma ponte entre o corpo e o cérebro, retransmitindo informações do cérebro à medula espinhal e ao cerebelo, e vice-versa (BEAR; CONNORS; PARADISO, 2017). Ele possui um importante papel no controle das funções vitais do corpo humano como a respiração, a consciência e o controle da temperatura corporal (BEAR; CONNORS; PARADISO, 2017).

Figura 4 – Encéfalo.



Fonte: Adaptado da página do site myshepherdconnection.org.[2]

O cérebro é subdividido em dois hemisférios, esquerdo e direito, onde cada hemisfério é subdividido em quatro lobos: frontal, parietal, occipital e temporal (Figura 5). Cada lobo está associado a diferentes funções. O lobo frontal está envolvido com a personalidade, emoções, resolução de problemas, desenvolvimento motor, raciocínio, planejamento, partes do discurso e movimento (SIULY; LI; ZHANG, 2016). O lobo parietal é responsável pela sensação, compreensão sensorial, reconhecimento, percepção de estímulos, orientação e movimento (SIULY; LI; ZHANG, 2016). O lobo occipital é responsável pelo processamento visual (SIULY; LI; ZHANG, 2016). O lobo temporal está envolvido no tratamento do reconhecimento de estímulos auditivos, fala, percepção e memória (SIULY; LI; ZHANG, 2016).

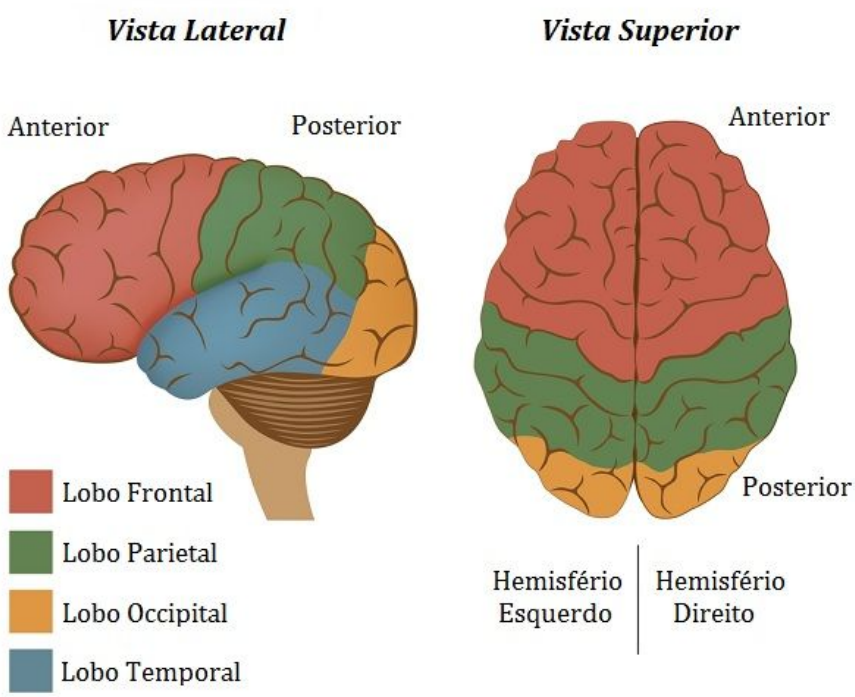
2.3 CLASSIFICAÇÕES DE INTERFACES CÉREBRO-MÁQUINA

De acordo com Ramadan & Vasilakos (2017), interfaces cérebro-máquina podem ser classificadas em termos de confiabilidade, grau de invasividade e sincronização (Figura 6):

[2] Disponível em: <https://www.myshepherdconnection.org/disorders-consciousness/Intro-disorders-of-consciousness> Acesso em mar. 2020.

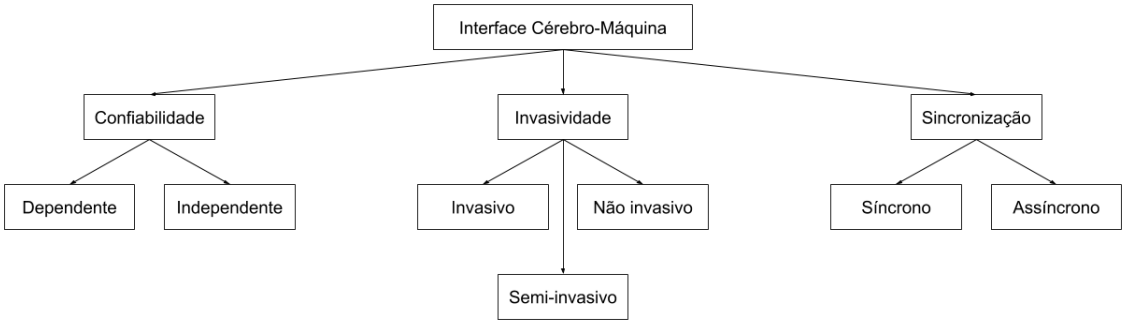
[3] Disponível em: <https://www.todamateria.com.br/cerebro> Acesso em mar. 2020.

Figura 5 – Lobos cerebrais.



Fonte: Página do site todamateria.com.br.[3]

Figura 6 – Classificações de uma interface cérebro-máquina.



Fonte: Adaptado de Ramadan & Vasilakos (2017).

- Confiabilidade: refere-se à necessidade ou não das vias normais de comunicação responsáveis pelo controle dos nervos e músculos, para a utilização da interface cérebro-máquina. Interfaces cérebro-máquina dependentes não utilizam as vias normais de comunicação para a decodificação e comunicação com o mundo externo. Porém atividades nestas vias são necessárias para que a atividade cerebral a ser decodificada seja gerada (WOLPAW *et al.*, 2002). Por outro lado, interfaces cérebro-máquina independentes não necessitam de qualquer atividade nas vias normais de comunicação. A atividade cerebral gerada depende

exclusivamente da intenção do sujeito em executar determinada ação (WOLPAW *et al.*, 2002). Isto torna-as muito mais atrativas visto que, para doenças degenerativas muito graves, as vias normais de comunicação podem estar completamente obstruídas, impedindo a utilização de interfaces cérebro-máquina dependentes;

- Invasividade: o grau de invasividade depende da tecnologia que será utilizada para o registro dos sinais cerebrais, uma etapa crucial para o desenvolvimento de uma interface cérebro-máquina;

- Sincronização: refere-se à forma pela qual os sinais cerebrais são recebidos e analisados. As interfaces cérebro-máquina síncronas analisam sinais cerebrais em janelas de tempo pré-definidas (NICOLAS-ALONSO; GOMEZ-GIL, 2012). Embora sejam de implementação muito mais simples por não fornecerem um meio natural para interação, não são muito atrativas. Por outro lado, as interfaces cérebro-máquina assíncronas dão muito mais liberdade ao sujeito, que pode decidir a qualquer momento executar ou não determinada ação. Sendo assim, os sinais cerebrais devem ser analisados continuamente, aumentando a complexidade do design e do poder computacional necessário (NICOLAS-ALONSO; GOMEZ-GIL, 2012).

2.4 REGISTRO DE SINAIS CEREBRAIS

Os métodos de registro de sinais cerebrais podem ser classificados em três categorias (RAMADAN; VASILAKOS, 2017):

- Invasivos: o registro de sinais cerebrais pode ser via a medição intracortical da atividade das células nervosas, por meio do implante de eletrodos diretamente no córtex motor (KÜBLER, 2019). Embora este método permita o registro de sinais provenientes da atividade de grandes conjuntos de células nervosas com uma alta relação sinal-ruído e altas resoluções espacial e temporal, possui como desvantagem a necessidade de uma cirurgia cerebral e todos os seus riscos envolvidos. Além disso, com o tempo, devido à formação de tecidos cicatriciais ou à movimentação involuntária, o sinal irá deteriorar-se, podendo até mesmo ser perdido. Outra desvantagem é que uma vez implantado, para que seja possível realizar o registro de atividade em outras áreas do cérebro, é necessária nova intervenção cirúrgica (RAMADAN; VASILAKOS, 2017).

- Semi-invasivos: os eletrodos são implantados sobre o crânio e por estarem próximos à atividade neural, fornecem uma alta resolução espacial (RAO, 2013). Um dos métodos de registro é a Eletrocorticografia (ECoG), onde eletrodos especiais são aplicados diretamente sobre o córtex cerebral.

- Não-invasivos: embora possuam uma resolução espacial menor, são os mais comuns para o desenvolvimento de interfaces cérebro-máquina, devido principalmente a sua praticidade

para utilização pelos sujeitos. Os métodos não invasivos podem ainda ser classificados pelo tipo de medição:

- Direta: possuem uma resolução temporal tão alta quanto os métodos invasivos. Os métodos incluem a Eletroencefalografia (EEG), que registra a atividade elétrica gerada pelo cérebro através de eletrodos colocados sobre o escalpo (GRAIMANN; ALLISON; PFURTSCHELLER, 2010) e a Magnetoencefalografia (MEG), que mede os campos magnéticos gerados pela atividade cerebral (NAM; NIJHOLT; LOTTE, 2018);
- Não-direta: possuem baixas resoluções espacial e temporal. Os métodos incluem a Ressonância Magnética Funcional (do inglês, *functional Magnetic Resonance Imaging* - fMRI), que permite inferir a atividade cerebral através da medição do volume e fluxo sanguíneo das células nervosas ativas (NAM; NIJHOLT; LOTTE, 2018) e a Espectroscopia Funcional de Infravermelho Próximo (do inglês, *functional Near InfraRed Spectroscopy* - fNIRS), que detecta a variação de volume e oxigenação sanguínea gerada através da exposição à luz infravermelha (RAMADAN; VASILAKOS, 2017).

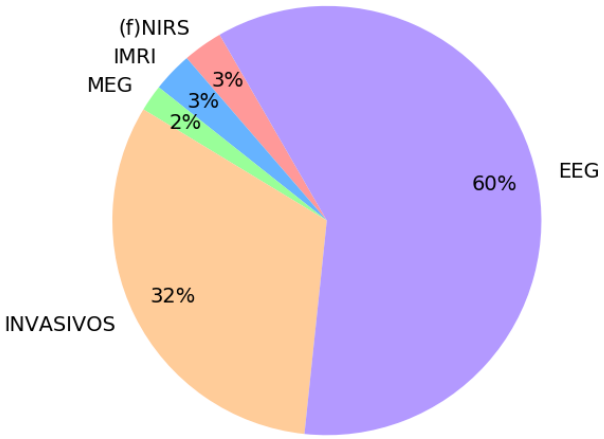
Em uma pesquisa bibliográfica sobre interfaces cérebro-máquina publicadas no período de 2007 a 2011 (Figura 7), realizada por Hwang *et al.* (2013), observou-se que EEG é o método de registro de sinais cerebrais mais utilizado. Em outra pesquisa bibliográfica dos trabalhos no período de 2013 a 2019, Zhang *et al.* (2019) confirmam que o sinal de EEG continua sendo o mais utilizado. A característica de não ser invasivo e o seu baixo custo são elementos cruciais para o acesso e utilização das interfaces cérebro-máquina pelos sujeitos. Com os devidos cuidados na etapa de registro dos sinais cerebrais, é possível obter como resultado uma base de dados de qualidade.

2.4.1 ELETROENCEFALOGRAFIA

Na década de 1920, o psiquiatra alemão Hans Berger descobriu que era possível registrar atividade elétrica proveniente do cérebro humano através de eletrodos posicionados sobre o escalpo, sendo responsável pelo nascimento do termo Eletroencefalografia (EEG). A EEG tem se provado uma ferramenta essencial na neurociência, sendo utilizada em diagnósticos de epilepsia, tumores cerebrais, distúrbios do sono, autismo, coma, entre outros. Sua utilização para o desenvolvimento de interfaces cérebro-máquina tem crescido significativamente nos últimos anos (SIULY; LI; ZHANG, 2016).

Embora a atividade elétrica gerada por um único neurônio seja muito pequena para ser registrada, a atividade elétrica gerada por grandes populações de neurônios em sincronia é suficiente para ser registrada pelas tecnologias EEG (SIULY; LI; ZHANG, 2016). O sinal de EEG é medido como uma diferença de potencial entre um eletrodo de registro e um eletrodo de referência (NICOLAS-ALONSO; GOMEZ-GIL, 2012). Um terceiro eletrodo de aterramento auxilia

Figura 7 – Distribuição dos métodos de registro de sinais encontrados nas publicações sobre interfaces cérebro-máquina publicadas no período de 2007 a 2011.



Fonte: Adaptado de Hwang *et al.* (2013).

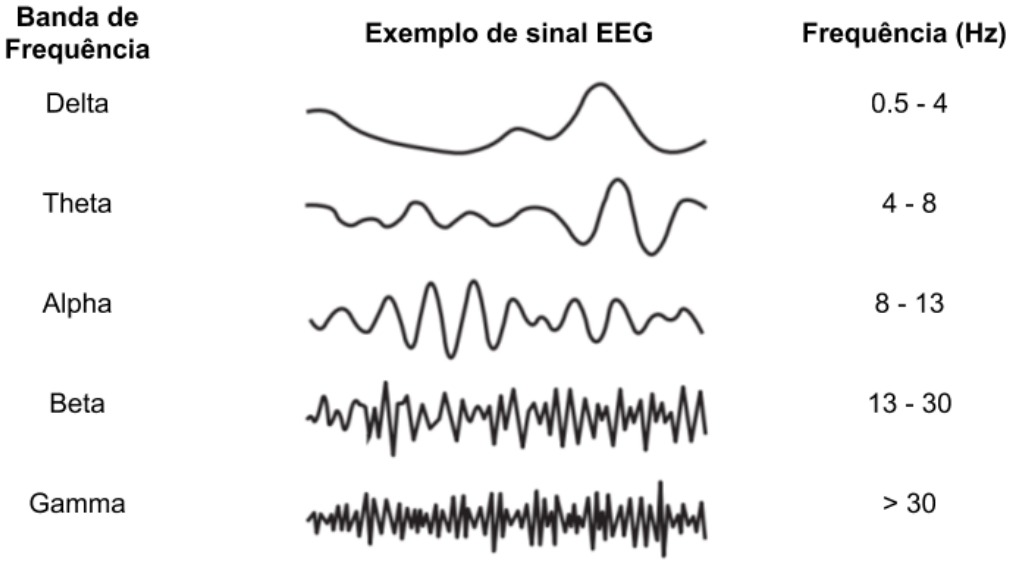
na eliminação de dados irrelevantes provenientes dos eletrodos de registro e de referência (NAM; NIJHOLT; LOTTE, 2018). O eletrodo de referência deve ser colocado em alguma área "inativa". Embora não exista nenhuma área 100% inativa, o lóbulo das orelhas é a opção mais comum.

Diferentes tipos de eletrodos podem ser utilizados para o registro dos sinais de EEG. Os eletrodos convencionais, conhecidos como *wet*, são construídos de prata e cloreto de prata (XIA; HU, 2019). Necessitam de um meio condutor como gel ou alguma solução salina para o preenchimento do espaço entre o eletrodo e o escalpo, construindo uma conexão condutora mais estável (XIA; HU, 2019). Embora estes eletrodos obtenham uma baixa impedância (< 10 kΩ), eles possuem várias limitações na aquisição dos sinais. O procedimento de preparação requer experiência e é demorado e muito desconfortável para os sujeitos. Além disso, o processo de limpeza do resíduo do meio condutor, tanto no sujeito quanto nos eletrodos, é complicado (XIA; HU, 2019). Para resolver essas complicações, eletrodos conhecidos como *dry* têm sido desenvolvidos (XIA; HU, 2019). Os eletrodos *dry* possuem uma estrutura que permite um contato direto com o escalpo, sem a necessidade de qualquer meio condutor. Devido a esta conexão com o escalpo não ser totalmente estável, uma impedância abaixo de 50 kΩ é considerada aceitável.

Uma importante propriedade do sinal de EEG é a frequência do sinal, que refere-se à atividade rítmica repetitiva expressa em *Hertz* (Hz), o número de ciclos por segundo (SIULY; LI; ZHANG, 2016). Faixas de frequências bem conhecidas têm sido classificadas em diferentes bandas de frequência, de acordo com sua relação com diferentes estados mentais: *delta* (δ, 0.5-4 Hz), *theta* (δ, 4-8 Hz), *alpha* (α, 8-13 Hz), *beta* (β, 13-30 Hz) e *gamma* (γ, > 30 Hz). Exemplos das bandas de frequência podem ser visualizados na Figura 8. A banda *delta* está associada ao estado de vigília e sono profundo ou a distúrbios cerebrais graves (SIULY; LI; ZHANG, 2016). A banda *theta* está associada à inatividade, à inconsciência, à meditação e à sonolência (RAMADAN;

VASILAKOS, 2017). A banda *alpha* está associada ao relaxamento e à concentração (RAMADAN; VASILAKOS, 2017). A banda *beta* está associada à forte concentração, ao foco e à resolução de problemas (SIULY; LI; ZHANG, 2016). A banda *gamma* está associada ao processamento somatossensorial[4] (RAMADAN; VASILAKOS, 2017).

Figura 8 – Bandas de frequências de sinais de EEG.

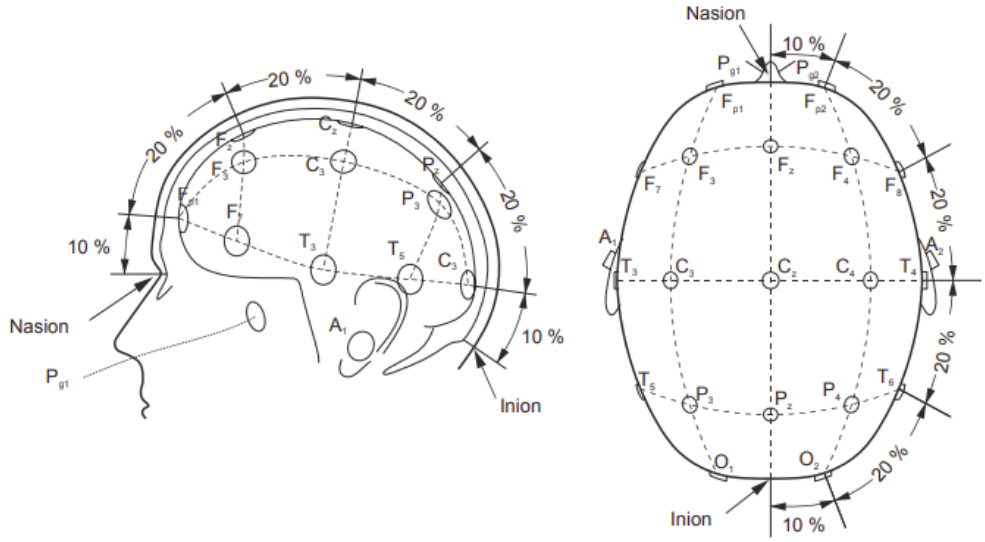


Fonte: Adaptado de Nam, Nijholt & Lotte (2018).

Para o registro da atividade elétrica cerebral utilizando EEG, os eletrodos devem ser dispostos sobre o escalpo de forma padronizada, visando minimizar ao máximo as diferenças entre sujeitos ou até mesmo entre diferentes sessões de aquisição para um mesmo sujeito. Para resolver este problema, os eletrodos geralmente são posicionados de acordo com o sistema internacional 10-20, padronizado pela Sociedade Americana de Eletroencefalografia (do inglês, *American Electroencephalographic Society*) (NICOLAS-ALONSO; GOMEZ-GIL, 2012). Conforme Nicolas-Alonso & Gomez-Gil (2012), este sistema utiliza dois pontos de referência: o *nasion*, localizado logo acima do nariz, no mesmo nível dos olhos e o *inion*, localizado na parte traseira do crânio, identificado por uma curvatura saliente. A partir destes dois pontos são realizadas divisões de trás para frente e da esquerda para direita, em intervalos de 10% e 20%, como pode ser visualizado na Figura 9. As letras correspondentes a cada eletrodo referem-se à uma área específica, sendo: A os lóbulos das orelhas; C a região central; Pg a região da nasofaringe; P a região parietal; F a região frontal; Fp a região frontal polar; e O a região occipital.

[4] O sistema somatossensorial é que permite ao ser humano experimentar sensações como tato, temperatura ou dor.

Figura 9 – Sistema internacional 10-20 para o posicionamento de eletrodos sobre o escalpo.

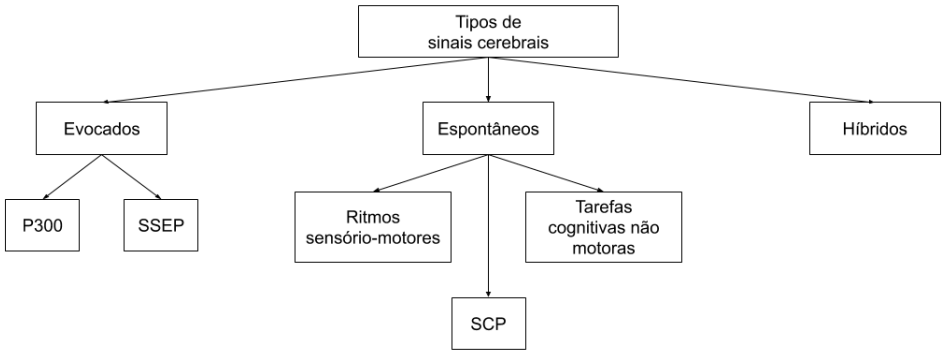


Fonte: Nicolas-Alonso & Gomez-Gil (2012)

2.5 TIPOS DE SINAIS CEREBRAIS

Interfaces cérebro-máquina baseiam-se na identificação de estados mentais através de sinais cerebrais. Enquanto alguns destes sinais são facilmente identificáveis, outros são mais complicados e requerem maior processamento (RAMADAN; VASILAKOS, 2017). Estes sinais podem ser categorizados em três categorias: os sinais evocados, espontâneos e híbridos (Figura 10).

Figura 10 – Tipos de sinais cerebrais.



Fonte: Adapatado de Ramadan & Vasilakos (2017).

2.5.1 SINAIS EVOCADOS

Sinais evocados são gerados involuntariamente pelo sujeito através do recebimento de estímulos externos (RAMADAN; VASILAKOS, 2017). Interfaces cérebro-máquina baseadas em

sinais evocados tendem a gerar um cansaço na sua utilização por um grande período de tempo, devido a exposição a um mesmo estímulo visual, auditório ou tátil, repetidamente. Entretanto, praticamente não requerem treinamento para novos sujeitos, uma vez que o sujeito não precisa aprender a controlar a geração deste tipo de sinal, basta ser exposto a determinados estímulos. Sinais evocados podem ser subdivididos em potenciais relacionados a eventos (do inglês, *Event-Related Potentials* - ERPs), gerados durante ou após um evento sensorial, motor ou psicológico (REZEIKA *et al.*, 2018), ou em potenciais evocados em estado estacionário, gerados em resposta a estímulos visuais, auditórios ou táteis.

O sinal P300 é um componente ERP do sinal de EEG que atinge um pico positivo máximo de tensão cerca de 300 ms após o início do estímulo (NAM; NIJHOLT; LOTTE, 2018), conforme ilustrado na Figura 11. O estímulo do sinal P300 é realizado através da aplicação do paradigma *oddball* (FARWELL; DONCHIN[5], 1988 apud REZEIKA *et al.*, 2018). Neste paradigma o sujeito é exposto a uma série de estímulos, sendo que cada estímulo pertence a um de dois eventos: o desejado ou o indesejado. Um destes eventos, conhecido como evento raro, é apresentado menos frequentemente ao sujeito (WOLPAW; WOLPAW[6], 2012 apud REZEIKA *et al.*, 2018). Para que o sinal P300 seja gerado, o sujeito precisa manter sua atenção no estímulo desejado. Geralmente é passada uma instrução para que o sujeito conte a quantidade de vezes em que o estímulo desejado ocorre.

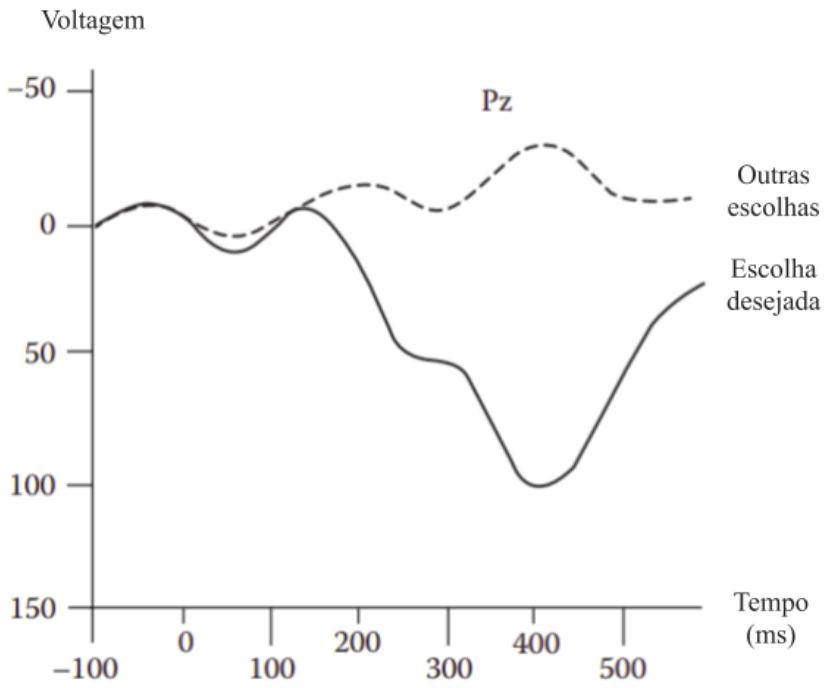
A aplicação mais comum para a utilização do sinal P300 é o *P300 speller*, uma interface cérebro-máquina dependente que consiste de uma matriz composta por letras, números e símbolos, como pode ser visualizado no exemplo da Figura 12. Conforme Rezeika *et al.* (2018), o *P300 speller* segue o paradigma *oddball*, onde as linhas e colunas da matriz piscam em uma intensidade aleatória. O sujeito deve manter sua atenção na letra, número ou símbolo desejado, contando o número de vezes em que o mesmo pisca. Desta forma, o sinal P300 pode ser correlacionado com a ordem de ocorrência do piscar das linhas e colunas, para obter a exata linha e coluna que induziu o sinal P300 no sujeito.

Potenciais Evocados em Estado Estacionário (do inglês, *Steady-State Evoked Potentials* - SSEPs) surgem como uma atividade rítmica na área cortical associada, visto que podem ser estímulos visuais, auditórios ou táteis (NAM; NIJHOLT; LOTTE, 2018). Segundo Ramadan & Vasilakos (2017), estes estímulos causam uma alteração na energia dos sinais de EEG, que deve aumentar até alcançar a frequência do estímulo. A sua variante mais utilizada é a SSVEP (do inglês, *Steady-State Visually Evoked Potential*). São potenciais evocados em estado estacionário através de estímulos visuais. Interfaces cérebro-máquina que a utilizam geralmente possuem luzes ou outros estímulos que piscam em diferentes frequências. Cada luz ou padrão está vinculado a uma opção de controle, como direção ou liga/desliga (NAM; NIJHOLT; LOTTE, 2018). A Figura

[5] FARWELL, L.; DONCHIN, E. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. **Electroencephalography and Clinical Neurophysiology**, 1988, 70, 510–523.

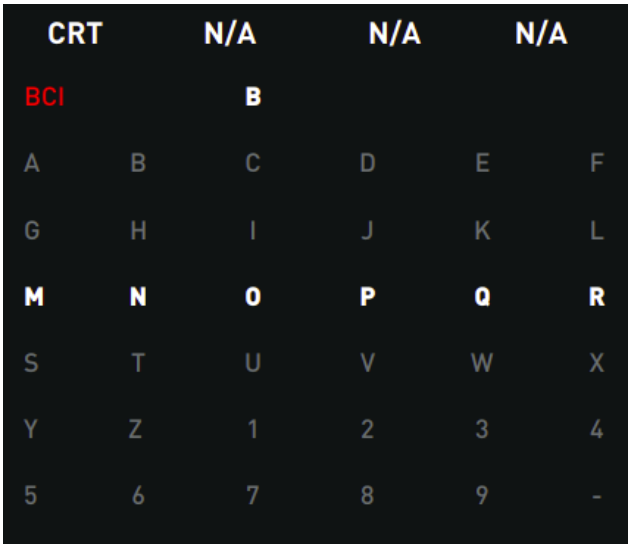
[6] WOLPAW, J.; WOLPAW, E. Brain-computer interfaces: Principles and practice. Oxford University Press: New York, NY, USA, 2012.

Figura 11 – Sinal P300.



Fonte: Adaptado de Nam, Nijholt & Lotte (2018).

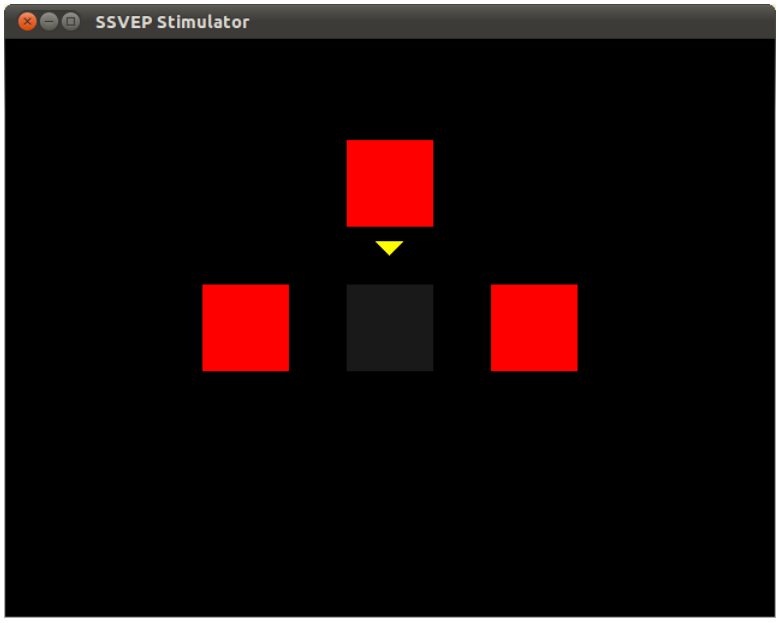
Figura 12 – P300 *speller*.



Fonte: Rezeika *et al.* (2018).

13 ilustra um exemplo de uma interface cérebro-máquina dependente que utiliza SSVEP para a escolha de múltiplas opções. Este tipo de interface cérebro-máquina pode ser utilizada para fins de comunicação com pacientes acometidos de alguma doença degenerativa ou mesmo para o controle de diversos tipos de dispositivos inteligentes.

Figura 13 – Interface cérebro-máquina baseada em SSVEP.



Fonte: Página do site openvibe.inria.fr.[7]

2.5.2 SINAIS ESPONTÂNEOS

Sinais espontâneos são gerados voluntariamente pelo sujeito sem a necessidade de qualquer estímulo externo (RAMADAN; VASILAKOS, 2017). Interfaces cérebro-máquina baseadas neste tipo de sinal fornecem uma interação mais natural com o sujeito, sem a necessidade da exposição à estímulos externos. Entretanto, é necessário o treinamento dos sujeitos para que consigam controlar a geração desses sinais. Os sinais espontâneos mais conhecidos são os potenciais corticais lentos e os ritmos sensório-motores.

Potenciais Corticais Lentos (do inglês, *Slow Cortical Potentials* - SCPs) são mudanças lentas de corrente contínua do sinal de EEG, originárias da camada cortical superior. Eles duram de 0,3 a vários segundos (STREHL *et al.*, 2006). É possível que o sujeito controle o aumento e diminuição destes sinais, mas devido à quantidade excessiva de treinamento necessário para obter este controle, muitos pesquisadores tem dado preferência aos ritmos sensório-motores (RAMADAN; VASILAKOS, 2017).

Ritmos Sensório-Motores (do inglês, *Sensorimotor Rhythms* - SMRs) são ritmos associados a movimentos motores provenientes do córtex motor com bandas de frequência localizadas em μ (8-13 Hz)[8] e β (13-30 Hz) (RAMADAN; VASILAKOS, 2017). Movimentos reais e imaginados geram oscilações nestas bandas de frequência que são conhecidas como Dessincronização Relacionada a Eventos (do inglês, *Event-Related Desynchronization* - ERD) e Sincronização Relacionada a Eventos (do inglês, *Event-Related Synchronization* - ERS) (NAM; NIJHOLT; LOTTE,

[7] Disponível em: <http://openvibe.inria.fr/motor-imagery-bci> Acesso em mar. 2020.

[8] A atividade alfa registrada nas áreas sensório-motoras também é conhecida como atividade μ (GRAIMANN; ALLISON; PFURTSCHELLER, 2010).

2018). Conforme Graimann, Allison & Pfurtscheller (2010), a dessincronização e sincronização relacionada a eventos correspondem ao decréscimo e acréscimo de atividade oscilatória em uma determinada banda de frequência, respectivamente. Estes ritmos podem ser controlados pelo sujeito através de condicionamento operante, que requer muito treinamento para que o sujeito possa controlar a amplitude destes ritmos, podendo levar semanas ou até mesmo meses. Outra forma de controle dos ritmos sensório-motores é através da imagética motora, objeto de estudo deste trabalho.

Tarefas cognitivas não motoras também são exploradas para o controle de interfaces cérebro-máquina. Vários tipos de atividades podem ser exploradas. Um bom exemplo é a base de dados coletada por Keirn & Aunon (1990). Esta base é composta por sinais coletados através de cinco atividades:

- Relaxamento: nesta atividade foi solicitado ao sujeito que apenas relaxasse e tentasse pensar em nada;
- Resolução de problemas complexos: nesta atividade o sujeito foi exposto a multiplicações não triviais e instruído a resolvê-las mentalmente;
- Rotação de figuras geométricas: nesta atividade foi exibido ao sujeito uma figura 3D e foi solicitado para que, após analisar a figura, ele imaginasse a rotação da mesma sobre algum dos eixos;
- Composição mental da escrita de uma carta: nesta atividade o sujeito foi instruído a mentalmente compor a escrita de uma carta para alguma pessoa próxima. Como a atividade foi repetida várias vezes, foi instruído para que em cada iteração continuasse de onde parou;
- Contagem visual: nesta atividade o sujeito foi instruído a visualizar a escrita de números sequenciais em um quadro negro, com o número anterior sendo apagado antes do próximo ser escrito. Novamente, foi solicitado que em cada iteração, o sujeito continuasse de onde parou na iteração anterior.

2.5.3 SINAIS HÍBRIDOS

Sinais híbridos são a composição de mais de um tipo de sinal cerebral para o controle de uma interface cérebro-máquina. A utilização de dois ou mais tipos de sinais cerebrais visa aumentar a confiabilidade da interface cérebro-máquina, procurando evitar as desvantagens inerentes de cada tipo de sinal cerebral (RAMADAN; VASILAKOS, 2017). Wang *et al.* (2019) demonstraram a viabilidade de uma interface cérebro-máquina híbrida para interação com jogos eletrônicos. O jogo escolhido para demonstrar a viabilidade foi o Tetris. Para a movimentação dos blocos para a esquerda ou para a direita, sinais espontâneos de imagética motora foram utilizados e para a rotação dos blocos, sinais evocados SSVEP foram utilizados.

2.6 IMAGÉTICA MOTORA

A imagética motora consiste na imaginação da movimentação de um membro, sem efetivamente executá-lo. A partir do conceito conhecido como Homúnculo[9], sabe-se que para cada parte do corpo humano existe uma respectiva região na área motora e somatossensorial do córtex cerebral (LEMM *et al.*, 2005). Logo, é possível realizar o reconhecimento de diferentes intenções motoras, através da identificação de oscilações ERD/ERS em sinais de imagética motora. Entretanto, para a geração de oscilações ERD/ERS discriminativas, as áreas corticais associadas devem ser suficientemente grandes, para que a atividade gerada possa ser discriminada do restante da atividade cerebral (GRAIMANN; ALLISON; PFURTSCHELLER, 2010). Como as áreas das mãos, pés e língua são grandes e topograficamente diferentes, estas são classes comumente utilizadas em interfaces cérebro-máquina baseadas em imagética motora (GRAIMANN; ALLISON; PFURTSCHELLER, 2010). Dada a proximidade das áreas corticais dos pés e dos dedos, torna-se inviável com EEG, através das oscilações ERD/ERS, a discriminação entre os dois pés, ou entre os dedos (GRAIMANN; ALLISON; PFURTSCHELLER, 2010). Já a diferenciação das mãos é viável, visto que a mão esquerda é representada lateralizada no hemisfério direito e a mão direita no hemisfério esquerdo (LEMM *et al.*, 2005). Desta forma, voltando-nos à Figura 9, podemos notar que os eletrodos C3 e C4 tornam-se muito importantes em interfaces cérebro-máquina baseadas em imagética motora que visam diferenciar as classes mão esquerda e mão direita.

Interfaces cérebro-máquina baseadas em imagética motora podem ser construídas como ferramentas assistivas para pessoas acometidas de doenças degenerativas do sistema motor. Além de melhorar o dia a dia destas pessoas, relatos médicos apontam que o *neurofeedback* baseado em imagética motora é efetivo no reestabelecimento dos movimentos motores imediatamente após um acidente vascular cerebral (MATTIA *et al.*[10]; PICHIORRI *et al.*[11], 2016, 2015 apud NAM; NIJHOLT; LOTTE, 2018). Exemplos de interfaces cérebro-máquina desenvolvidas como ferramentas assistivas são a adição de controle via imagética motora a dispositivos existentes, como uma cadeira de rodas elétrica (RESHMI; AMAL, 2013), um braço robótico (QUILES *et al.*, 2020), um robô (ALJALAL; DJEMAL; IBRAHIM, 2019), entre outros.

Para a aquisição de sinais de imagética motora um protocolo deve ser definido. A Figura 14 ilustra um protocolo experimental (CHO *et al.*, 2017) para a aquisição de sinais de imagética motora, para as classes mão esquerda e mão direita. Neste protocolo, o sujeito é posicionado em frente a um computador, utilizando alguma tecnologia para o registro dos sinais cerebrais e é instruído para, de acordo com o estímulo exibido na tela do computador, imaginar a respectiva ação motora. A expressão "imaginar uma ação motora" pode ocasionar que diferentes sujeitos a

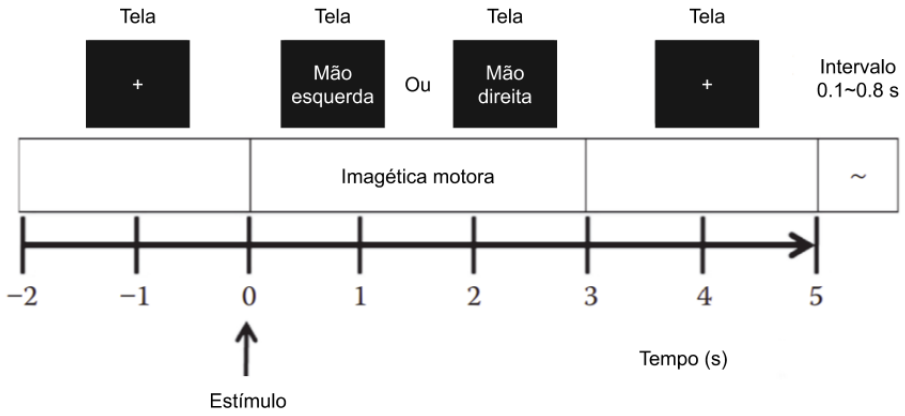
[9] O Homúnculo é uma representação do mapeamento da relação do cérebro para o corpo proposto pelo neurocirurgião Wilder Penfield, baseado em seus estudos em pacientes com epilepsia.

[10] MATTIA, D. et al. Interfacing brain and computer in neurorehabilitation. In: Brain–Computer Interface (BCI), 2016 4th International Winter Conference on. IEEE, 1–2.

[11] PICHIORRI, F. et al. Brain-computer interface boosts motor imagery practice during stroke recovery. Annals of neurology, 77(5), 851–865.

executem de diferentes maneiras. Por exemplo, o sujeito pode imaginar-se executando a ação motora ou pode imaginar a imagem dele ou de outra pessoa executando a ação. O estudo de Neuper *et al.* (2005) demonstrou que a primeira forma de imaginação, a qual envolve experiências cinestésicas, produzem melhores resultados de classificação no reconhecimento de imagética motora, permitindo uma clara visualização das oscilações ERD/ERS nos sinais de EEG.

Figura 14 – Protocolo experimental para aquisição de sinais de imagética motora, para as classes mão esquerda e mão direita.



Fonte: Adaptado de Cho *et al.* (2017).

As oscilações ERD/ERS (Seção 2.5.2) consistem em decréscimos ou acréscimos de energia em determinadas bandas de frequência. Conforme Pfurtscheller & Silva (1999), uma característica básica destas oscilações é que a energia identificada na banda de frequência de interesse é exibida como um percentual de decréscimo ou acréscimo em relação a um período de referência (alguns segundos antes do evento de imagética motora ocorrer). A ocorrência destas oscilações nos sinais de EEG pode indicar se possuem as características esperadas para classificação de imagética motora. Conforme Pfurtscheller & Silva (1999), o método clássico para o cálculo da trajetória temporal destas oscilações é composto das seguintes etapas:

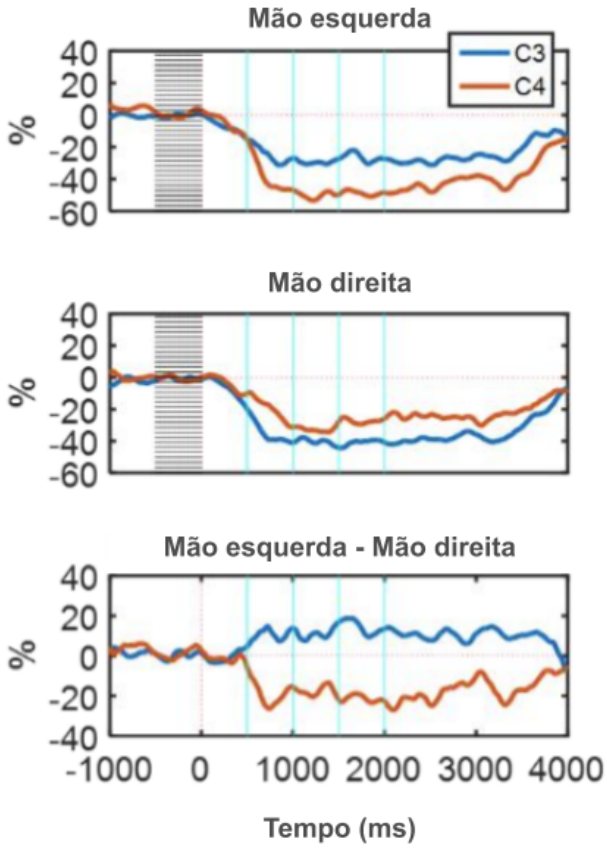
1. Aplicação de um filtro passa-banda em todos os ensaios;
2. Cálculo da energia das amostras elevando as amostras ao quadrado;
3. Cálculo da média das trajetórias de energias de todos os ensaios;
4. Cálculo da média das amostras ao longo do tempo para suavizar os dados e reduzir a variabilidade;
5. Cálculo dos percentuais das oscilações ERD/ERS em relação ao período de referência:

$$ERD\% = \frac{A - R}{R} \times 100, \tag{2.1}$$

onde A corresponde a energia dentro da banda de frequência de interesse durante o evento de imagética motora, enquanto R corresponde a energia no período de referência.

A Figura 15 mostra a trajetória temporal das oscilações ERD para sinais de imagética motora da movimentação da mão esquerda e da mão direita em dois eletrodos (ocorrido após o tempo 0). Os dois primeiros gráficos mostram as trajetórias do ERD dos eletrodos C3 e C4 para a movimentação imagética das mãos esquerda e direita, respectivamente. O terceiro gráfico (Mão esquerda - Mão Direita) mostra a diferença entre os ERDs das movimentações imagéticas em cada eletrodo.

Figura 15 – Trajetória temporal das oscilações ERD em sinais de imagética motora.



Fonte: Adaptado de Cho *et al.* (2017).

2.7 RECONHECIMENTO DE IMAGÉTICA MOTORA ATRAVÉS DE SINAIS DE EEG

A tarefa de reconhecimento de imagética motora tem por objetivo realizar o reconhecimento da classe de imagética motora através de sinais espontâneos de EEG. Para realizar

tal tarefa, inúmeras técnicas de aprendizado de máquina têm sido empregadas. Esta seção tem por objetivo apresentar os principais algoritmos utilizados para o reconhecimento de imagética motora, bem como métodos para avaliação de desempenho destes algoritmos.

2.7.1 AVALIAÇÃO DE DESEMPENHO DOS ALGORITMOS

Uma ampla gama de métodos estatísticos podem ser empregados para a avaliação de desempenho de algoritmos de aprendizado de máquina (JAMES *et al.*, 2013). Como não há almoço grátis em estatística, nenhum método domina todos os outros em todas as possíveis bases de dados (JAMES *et al.*, 2013). Baseado em fatores como a quantidade de classes do problema e a quantidade de amostras disponíveis para cada classe, pode-se definir um conjunto de métricas e ferramentas de visualização que permitam avaliar os resultados.

É necessário de alguma forma quantificar até que ponto as predições do algoritmo correspondem aos dados observados (JAMES *et al.*, 2013). Para algoritmos de classificação é utilizada a taxa de erro, que corresponde à proporção de amostras classificadas incorretamente:

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \tag{2.2}$$

onde $\hat{y}_i$ é a predição para a amostra i e $I(y_i \neq \hat{y}_i)$ resulta em 1 se a predição foi realizada corretamente e 0 caso contrário. A taxa de erro de treinamento pode ser calculada facilmente pois o conjunto de dados de treinamento está disponível no momento da avaliação de desempenho. Entretanto, o que realmente importa é se o algoritmo será capaz de classificar corretamente novas amostras, as quais não tenham sido expostas durante o treinamento. Nesse sentido, é necessário calcular a taxa de erro de teste. Como geralmente o que se tem é apenas uma amostragem do conjunto de dados de teste, é necessário realizar uma estimativa dessa taxa de erro.

Para bases de dados pequenas (cenário muito comum), a divisão em conjuntos de teste e treino de tamanhos fixos é um problema, uma vez que um conjunto de teste pequeno implica em incerteza estatística em torno da estimativa do erro médio de teste, dificultando a comparação de desempenho entre algoritmos (GOODFELLOW; BENGIO; COURVILLE, 2016). Para enfrentar este problema, técnicas visando a utilização de toda a base de dados para a estimativa do erro médio de teste são empregadas. Dentre estas técnicas, a mais comum é a de validação cruzada *K-fold* (GOODFELLOW; BENGIO; COURVILLE, 2016). Esta técnica permite aproveitar todos os dados disponíveis para o treinamento do algoritmo. Isso é realizado através da divisão do conjunto de dados disponíveis em K partições. O classificador é construído com os dados de $K-1$ partições e a partição deixada de fora é utilizada para a avaliação de desempenho do algoritmo. Este processo é repetido K vezes, utilizando em cada iteração uma partição diferente como conjunto de dados de teste. Por fim, são obtidas as médias e desvios-padrão das K estimativas da taxa de erro de teste.

Para a visualização do desempenho dos algoritmos pode-se utilizar a matriz de confusão (Tabela 1). Suas linhas e colunas correspondem às classes reais das amostras e as preditas pelo algoritmo, respectivamente. Os valores de cada posição correspondem ao: número de verdadeiros positivos (VP), número de falsos negativos (FN), número de falsos positivos (FP) e o número de verdadeiros negativos (VN)[12]. A matriz de confusão de um problema de n classes é denotada como $\mathbf{M} \in \mathbb{N}^{n \times n}$. Os elementos da diagonal principal ($\mathbf{M}_{ii}$) correspondem as amostras classificadas corretamente, enquanto que os elementos fora da diagonal principal representam as amostras classificadas incorretamente (RAO, 2013). Por exemplo, o elemento $\mathbf{M}_{ij}$ (onde $i \neq j$), representa quantas amostras da classe i foram incorretamente classificadas como sendo da classe j.

Tabela 1 – Exemplo de uma matriz de confusão para um algoritmo de classificação binária.

		Predição	
		Positivo	Negativo
Atual	Positivo	VP	FN
	Negativo	FP	VN

Fonte: O Autor (2020).

Conforme Subasi (2019), para problemas binários, a partir da matriz de confusão outras métricas úteis podem ser obtidas como:

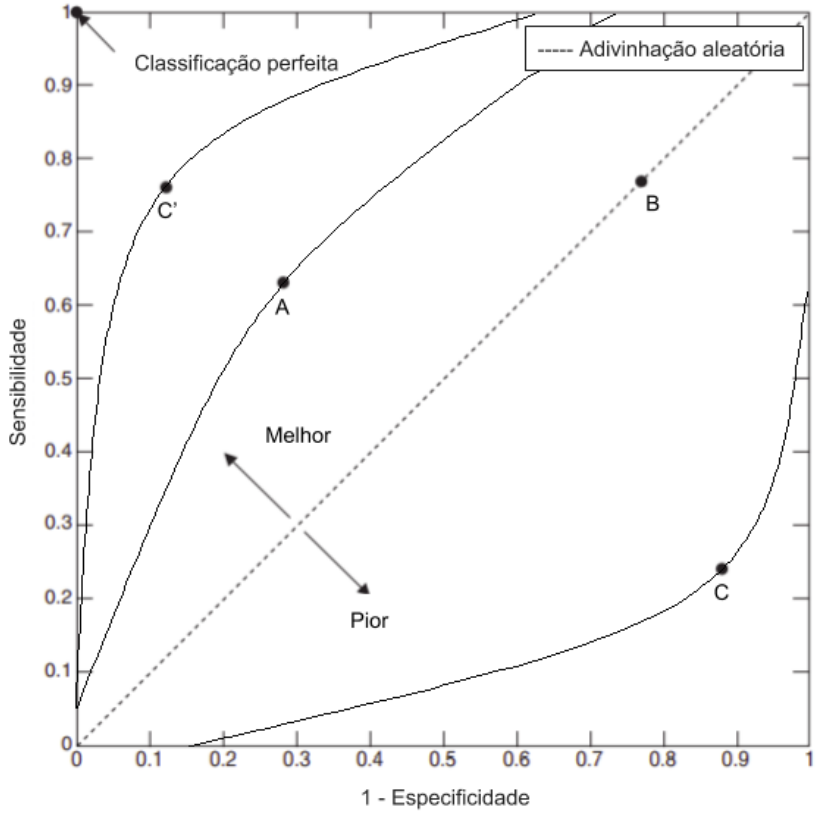
- **Acurácia**: é a proporção das amostras classificadas corretamente em relação a todas as amostras:
$$\frac{VP + VN}{VP + VN + FP + FN}. \tag{2.3}$$
- **Sensibilidade**: é a taxa de verdadeiros positivos, calculada como a proporção das amostras positivas classificadas corretamente como positivas em relação a todas as amostras positivas:
$$\frac{VP}{VP + FN}. \tag{2.4}$$
- **Especificidade**: é a taxa de verdadeiros negativos, calculada como a proporção das amostras negativas classificadas corretamente como negativas em relação a todas as amostras negativas:
$$\frac{VN}{VN + FP}. \tag{2.5}$$

As métricas de sensibilidade e especificidade tornam-se particularmente importantes em bases de dados desbalanceadas, onde o número de amostras representando uma classe é significativamente menor do que o número de amostras representando outra classe. Geralmente não é possível obter a máxima sensibilidade e especificidade simultaneamente, ou seja, com

[12] A terminologia positivo e negativo referencia problemas de classificação binária, onde positivo corresponde ao evento de interesse.

uma única configuração de parâmetros do algoritmo (TU, 2019). Uma ferramenta para auxiliar neste *trade-off* é a curva ROC (do inglês, *Receiver Operating Characteristic*). No gráfico da curva ROC, a sensibilidade é exibida como uma função da taxa de falsos positivos (1 - Especificidade), para diferentes valores de um determinado parâmetro do algoritmo. A Figura 16 ilustra aonde diferentes algoritmos podem aparecer no espaço da curva ROC. **A** e **C'** são algoritmos que tem um desempenho superior à adivinhação aleatória, enquanto que **B** é um algoritmo de adivinhação aleatória. Já o algoritmo **C** tem um desempenho significativamente inferior à adivinhação aleatória. O algoritmo ideal estaria posicionado no canto superior esquerdo e, quanto mais próximo a esta região, melhor é o desempenho do algoritmo. A área sobre a curva ROC (do inglês, *Area Under the Curve* - AUC) é uma métrica que representa quão bem um determinado algoritmo é capaz de realizar a distinção entre duas classes (TU, 2019).

Figura 16 – Curva ROC.



Fonte: Adaptado de Rao (2013).

Dentre as métricas citadas, a acurácia é comumente utilizada para a avaliação de desempenho de algoritmos de reconhecimento de imagética motora. É uma métrica informativa quando aplicada sobre bases de dados balanceadas entre todas as classes e quando o problema não possui um viés, ou seja, as decisões para todas as classes têm igual probabilidade de ocorrência a priori (THOMAS; DYSON; CLERC, 2013). No entanto, a acurácia possui algumas limitações. Ela não considera o desbalanceamento entre as classes, logo, uma acurácia de 90% em uma base de

dados onde 90% das amostras são de uma única classe não diz muita coisa sobre o sistema. Além disso, a acurácia não considera a distribuição das classificações incorretas (i.e., os elementos fora da diagonal principal na matriz de confusão), que torna-se importante no cenário de classificação de múltiplas classes. Uma métrica que corrige estas limitações da acurácia é o coeficiente *kappa*. Ele considera a distribuição das classificações incorretas e o desbalanceamento entre as classes normalizando a frequência de ocorrência para cada classe (Dornhege *et al.*, 2007).

Conforme Thomas, Dyson & Clerc (2013), dependendo do protocolo de aquisição de dados utilizado, diferentes estratégias para a avaliação de desempenho do algoritmo podem ser definidas:

- *Offline*: resultados são obtidos sem a geração de nenhum *feedback* para o sujeito;
- *Online*: *feedback* é gerado em tempo real para o sujeito;
- *Online* simulado: resultados são obtidos através da aplicação de um novo algoritmo em dados previamente coletados com *feedback*.

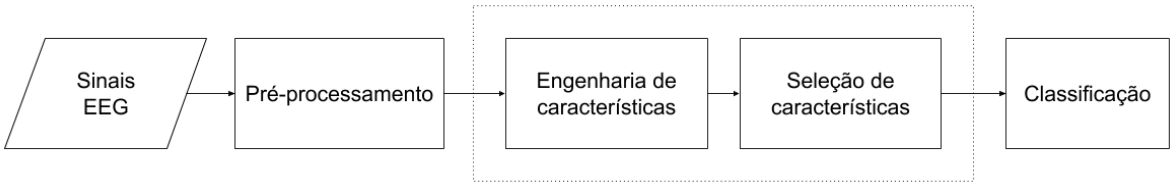
A melhor estratégia é a *online*, visto que representa o melhor cenário visando a aplicabilidade do algoritmo em uma interface cérebro-máquina. Além disso, o *feedback* pode permitir que o sujeito consiga melhorar sua capacidade de modulação dos sinais de imagética motora. O maior empecilho desta estratégia é a necessidade de uma implementação em tempo real durante a etapa de aquisição dos dados. Além disso, somente um algoritmo com uma configuração específica de parâmetros pode ser avaliado por experimento. Dessa forma, neste trabalho será considerada a estratégia de avaliação de desempenho *offline*, visto que permite a comparação do algoritmo a ser proposto com outros algoritmos já avaliados em diferentes bases de dados públicas, possibilitando analisar quais algoritmos têm um maior potencial no reconhecimento de imagética motora. Logo, os algoritmos explorados neste trabalho serão mais apropriados para aplicabilidade direta em interfaces cérebro-máquina síncronas, não impedindo tentativas de aplicabilidade em interfaces cérebro-máquina assíncronas. Entretanto, é mais provável que o desempenho obtido na implantação em uma interface cérebro-máquina síncrona seja muito mais próximo ao obtido no processo de avaliação de desempenho do algoritmo, do que quando implantado em interfaces cérebro-máquina assíncronas.

2.7.2 ABORDAGENS BASEADAS EM ENGENHARIA DE CARACTERÍSTICAS

Um sistema de reconhecimento de imagética motora baseado em engenharia de características pode ser subdividido em cinco etapas: aquisição dos sinais cerebrais, pré-processamento, engenharia e seleção de características e classificação (Figura 17). A aquisição dos sinais cerebrais é realizada via EEG através de um protocolo experimental para aquisição de sinais de imagética motora. De forma geral, os sinais de EEG são pré-processados para aumentar sua

relação sinal-ruído. Um vetor de características é produzido a partir dos sinais pré-processados a fim de discriminar diferentes classes de imagética motora. Opcionalmente, pode ser aplicado um algoritmo de seleção de características para reduzir a dimensionalidade do vetor de características e, consequentemente, reduzir a complexidade do classificador (o que pode melhorar o desempenho do algoritmo) (THEODORIDIS; KOUTROUMBAS, 2009). O vetor de características obtido é utilizado para alimentar um classificador que retorna as probabilidades de pertencimento da amostra a cada classe de imagética motora.

Figura 17 – Estrutura de um sistema de reconhecimento de imagética motora baseado em engenharia de características.



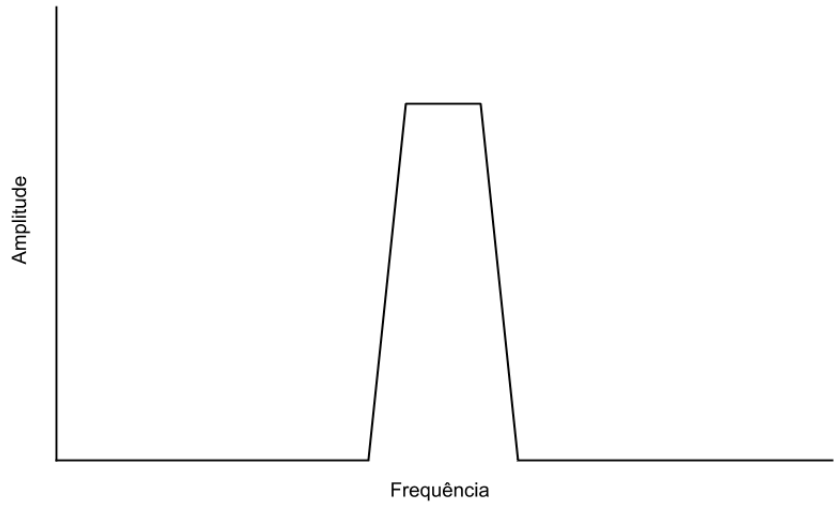
Fonte: Adaptado de Padfield *et al.* (2019).

Pré-Processamento

O sinal de EEG possui uma baixa relação sinal-ruído por ser muito suscetível a artefatos, os quais podem ser categorizados como fisiológicos ou não-fisiológicos (CHEN; ZHANG; WU, 2019). Os artefatos fisiológicos são gerados pelo próprio sujeito através do piscar de olhos ou movimentos musculares. Instruções durante a etapa de aquisição de sinais podem ajudar a reduzir estes artefatos, porém outros, como a frequência cardíaca por exemplo, são inevitáveis. Os artefatos não-fisiológicos são provenientes principalmente da interferência da rede elétrica, mas também podem surgir do mal posicionamento dos eletrodos sobre o escalpo. Chen, Zhang & Wu (2019) demonstraram que sinais de interfaces cérebro-máquina baseadas em imagética motora são afetados principalmente por artefatos provenientes de movimentos musculares e por interferências da rede elétrica. Artefatos fora do intervalo de interesse da banda de imagética podem ser facilmente removidos através de técnicas de filtragem de sinais.

Filtros digitais, parte muito importante do processamento digital de sinais, podem ser utilizados para a separação de sinais que tenham sido combinados, ou para a restauração de sinais que tenham sido distorcidos de alguma forma (SMITH, 1999). Filtros passa-banda (Figura 18) são utilizados para manter nos sinais somente as frequências dentro de um intervalo de interesse. Um único filtro cobrindo todo o intervalo de interesse da imagética motora tem sido largamente utilizado (CHO *et al.*, 2017; JOADDER *et al.*, 2019a; JOADDER *et al.*, 2019b; SREEJA *et al.*, 2017; TAN *et al.*, 2017; WANG *et al.*, 2018). Outros algoritmos, com o intuito de explorar a variabilidade da frequência de interesse para diferentes sujeitos, utilizam um banco de filtros passa-banda (ANG *et al.*, 2008; KUMAR; SHARMA; TSUNODA, 2017; NOVI *et al.*, 2007; TANG *et al.*, 2019b).

Figura 18 – Exemplo da resposta de frequência para um filtro passa-banda.



Fonte: Adaptado de Smith (1999).

Para bases de dados com a disponibilidade de um grande número de eletrodos, uma técnica de pré-processamento comum é a seleção de eletrodos para redução de dimensionalidade e melhoria de desempenho do algoritmo. Geralmente utiliza-se uma técnica simples de seleção manual de eletrodos, mantendo somente aqueles sobre o córtex motor, que são de maior interesse para a imagética motora (ALI *et al.*, 2019; SREEJA *et al.*, 2017; TAN *et al.*, 2017). Outra técnica de pré-processamento para aumentar a qualidade das bases de dados consiste na utilização de métricas para a remoção de ensaios de baixa qualidade. Cho *et al.* (2017) remove ensaios que, após a aplicação de um filtro passa-banda no intervalo de 8-30 Hz, contenham alguma amplitude acima de $100\mu V$ dentro de um intervalo específico logo após a instrução de imagética motora ao sujeito.

Para melhorar o desempenho do algoritmo, geralmente descarta-se o início e o fim de cada ensaio, a fim de manter somente o momento de maior engajamento do sujeito na tarefa de imagética motora. Em praticamente todos os estudos obtidos da literatura utiliza-se um intervalo de tempo fixo com algo em torno de 0,5 a 2,5 segundos (considerando como tempo 0 a exibição do estímulo de imagética motora ao sujeito). Entretanto, alguns estudos como o de Wang *et al.* (2018) têm explorado a utilização de algoritmos para a seleção do segmento de tempo mais efetivo por sujeito, para melhorar o desempenho do algoritmo.

Engenharia de Características

A etapa de engenharia de características é crucial para o bom desempenho do algoritmo. O sinal de EEG é transformado em um vetor de características que realça partes importantes dos sinais e elimina redundâncias (KHORSHIDTALAB; SALAMI; HAMEDI, 2013). Conforme Duda, Hart & Stork (2000), um processo de engenharia de características ideal produziria uma representação que torna trivial o trabalho do classificador. A engenharia de características é uma tarefa muito

mais dependente do domínio do que a classificação e, portanto, requer conhecimento do domínio (DUDA; HART; STORK, 2000).

O vetor de características de um problema EEG pode ser construído através de características no domínio do tempo, frequência ou domínios espaciais (TU, 2019). As características no domínio do tempo são calculadas com bases nas amplitudes dos sinais e desta forma não requerem transformações ou cálculos complexos (KHORSHIDTALAB; SALAMI; HAMEDI, 2013). Alguns estudos utilizaram características no domínio do tempo para a classificação de múltiplas classes de imagética motora (HAMEDI *et al.*, 2014; KHORSHIDTALAB; SALAMI; HAMEDI, 2012; KHORSHIDTALAB; SALAMI; HAMEDI, 2013). Os três estudos identificaram que, dentre as características avaliadas, a *Willison Amplitude* (WAMP) obteve, em termos de acurácia, o melhor desempenho na etapa de classificação. A característica WAMP é o número de vezes em que o valor absoluto da diferença entre a amplitude do sinal de EEG de duas amostras consecutivas, excede um valor limiar predeterminado (KHORSHIDTALAB; SALAMI; HAMEDI, 2013).

Para a engenharia de características no domínio da frequência torna-se necessária a aplicação de técnicas de estimação espectral para a transformação dos sinais de EEG do domínio do tempo para o domínio da frequência. A base destas técnicas é a transformada de Fourier, uma importante ferramenta no processamento digital de sinais e análise de séries temporais (COHEN, 2014). Ela produz uma representação em três dimensões da série temporal, nesse caso, o sinal de EEG, onde as três dimensões são a frequência, a energia e a fase. Baseado na transformada de Fourier, é possível calcular a Densidade Espectral de Energia (do inglês, *Power Spectral Density* - PSD), que descreve como a energia do sinal é distribuída nas diferentes frequências (ZHANG, 2019). Dentre os inúmeros algoritmos para estimação espectral amplamente utilizados em sinais de EEG estão o periodograma, o método de *Welch*, o método *multitaper* e modelos auto-regressivos baseados em estimativas espectrais (ZHANG, 2019). Com o sinal no domínio da frequência, características como a energia em determinadas bandas de frequências ou estatísticas do espectro da frequência como a média, a variância e a entropia podem ser extraídas (ZHANG, 2019).

O grande problema da análise no domínio da frequência é que ela pressupõe que os sinais a serem analisados são estacionários, o que significa que suas propriedades estatísticas, incluindo a média, a variância e a estrutura de frequência não variam com o tempo (ou seja, os sinais são bem comportados) (COHEN, 2014). Sinais de EEG violam essa suposição pois são não estacionários. Sua estrutura de frequência em atividades neurofisiológicas variam com o tempo, tanto em tarefas relacionadas a eventos (como no caso da imagética motora), quanto devido a processos endógenos (COHEN, 2014). Para que seja possível capturar essas características, técnicas para a análise no domínio tempo-frequência são mais adequadas. Uma técnica é a distribuição de energia no domínio tempo-frequência. Ela permite obter a quantidade de energia em um determinado ponto no domínio tempo-frequência. Algoritmos de janelamento como o *Short-Time Fourier Transform* (STFT) (ROY *et al.*, 2020; TABAR; HALICI, 2016) e o

Continuous Wavelet Transform (CWT) (LEE; CHOI, 2018; LI *et al.*, 2020) são comumente utilizados. Nestes algoritmos, o sinal é dividido em pequenos segmentos nos quais pode-se assumir a estacionariedade. Logo, em cada segmento, técnicas de estimação espectral podem ser aplicadas. Através da concatenação destas estimativas espectrais, pode-se obter uma distribuição espectral no domínio tempo-frequência (ZHANG, 2019). Outra técnica consiste na decomposição de sinais em um conjunto de componentes, de acordo com suas propriedades distintas no domínio tempo-frequência. Alguns algoritmos populares são a Transformada Discreta de *Wavelet* (do inglês, *Discrete Wavelet Transform* - DWT) e a Decomposição do Modo Empírico (do inglês, *Empirical Mode Decomposition* - EMD) (ZHANG, 2019).

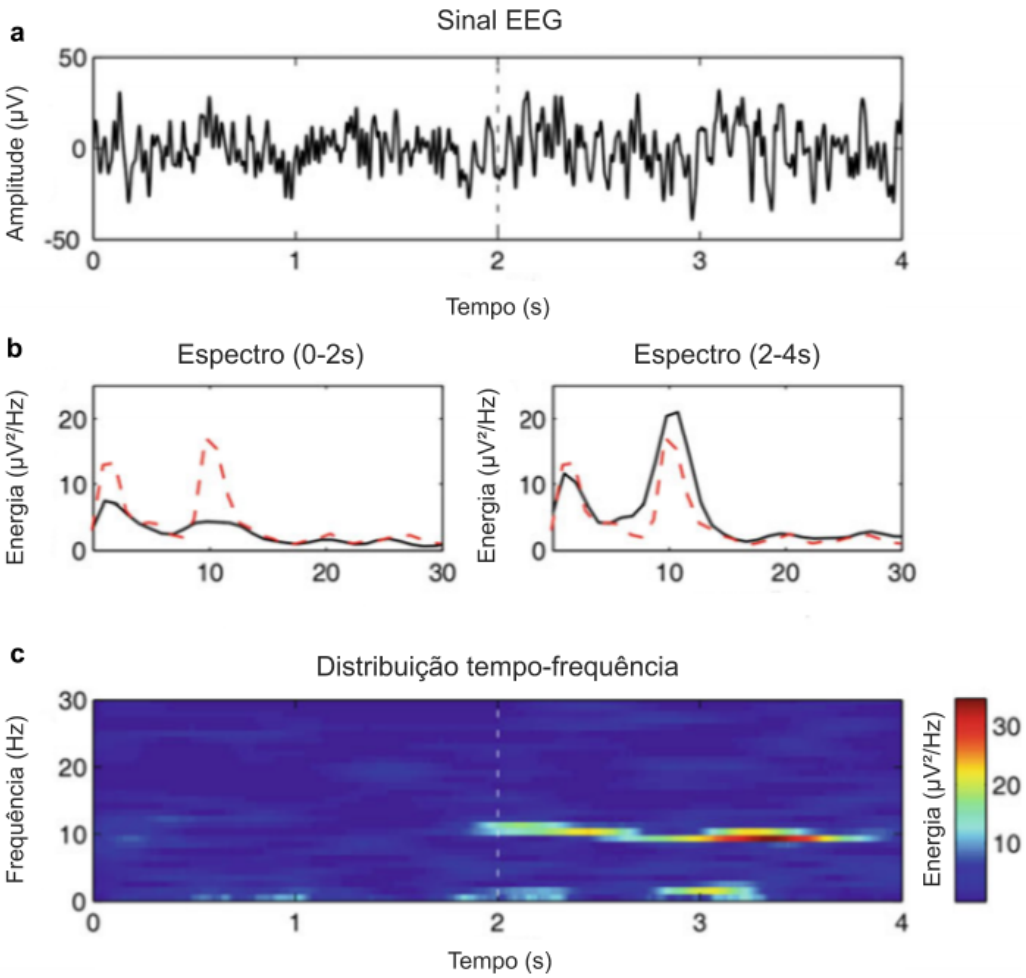
A análise realizada por Zhang (2019) fornece uma clara visualização de como a análise no domínio tempo-frequência consegue capturar características que variam com o tempo. A Figura 19-a ilustra um sinal de EEG obtido durante duas condições, olhos abertos (de 0 a 2 segundos) e olhos fechados (de 2 a 4 segundos). A Figura 19-b ilustra a estimação espectral realizada através da aplicação do método de *Welch*. As linhas tracejadas em vermelho correspondem ao espectro do sinal no período completo e as linhas sólidas em preto aos espectros dos períodos correspondentes às duas condições, separadamente. Pode-se notar uma diferença significativa entre as estimações espectrais dos diferentes períodos. A Figura 19-c ilustra uma análise no domínio tempo-frequência utilizando o algoritmo STFT. Pode-se observar um aumento de frequência a partir do início do período da condição olhos fechados, característica esta que não é possível identificar na análise no domínio da frequência da Figura 19-b.

Pattnaik, Dash & Sabut (2016) utilizam a DWT para decompor o sinal em sub-bandas de frequência. A banda beta (β) é selecionada para criar um vetor composto de características estatísticas e o pico de energia para ser utilizado na classificação de 2 classes de imagética motora. Shi *et al.* (2018) demonstram um aumento no desempenho da classificação de imagética motora, em termos de acurácia, ao utilizar características extraídas no domínio tempo-frequência, a partir de uma extensão da DWT, a Decomposição de Pacotes *Wavelet* (do inglês, *Wavelet Packet Decomposition* - WPD), em comparação com características extraídas somente no domínio da frequência através do algoritmo PSD. O algoritmo EMD também vem sendo utilizado para a classificação de imagética motora (BASHAR; BHUIYAN, 2016; GAUR *et al.*, 2015).

Características no domínio espacial são muito exploradas no desenvolvimento de algoritmos para o reconhecimento de imagética motora, principalmente no cenário da movimentação das mãos esquerda e direita, uma vez que essas 2 classes de imagética motora geram atividade cerebral nos hemisférios direito e esquerdo, respectivamente. Dessa forma, um algoritmo para a discriminação destas 2 classes pode levar em consideração oscilações ERD/ERS nos mesmos hemisférios. Uma técnica de processamento de sinais cerebrais para resolver este problema é a filtragem espacial:

> As técnicas de filtragem espacial tomam como entrada os sinais cerebrais adquiridos de vários locais diferentes (ou "eletrodos") e os transformam em

Figura 19 – Análise de sinal de EEG nos domínios da frequência e tempo-frequência.



Fonte: Adaptado de Zhang (2019).

> uma de várias maneiras. Os objetivos possíveis incluem aprimorar a atividade local, reduzir o ruído comum entre eletrodos, diminuir a dimensionalidade dos dados, identificar fontes ocultas ou encontrar projeções que maximizem a discriminabilidade entre diferentes classes (RAO, 2013, p. 54-55, tradução nossa).

O algoritmo *Common Spatial Pattern* (CSP) é uma técnica de filtragem espacial que tem obtido muito sucesso na identificação de oscilações ERD/ERS (KOLES; SOONG[13], 1998 apud LEMM *et al.*, 2005). É um algoritmo de aprendizado supervisionado que busca filtros espaciais para projetar os sinais, de tal forma que a variância de uma classe é maximizada, enquanto a variância da outra classe é minimizada (RAO, 2013). A partir dos sinais projetados pelos filtros, pode-se construir um vetor de características composto pelo logaritmo da variância, melhorando assim a discriminabilidade entre as classes.

[13] KOLES, Z. J.; SOONG, A. C. Eeg source localization: implementing the spatio-temporal decomposition approach. Elsevier, 1998.

Outros estudos têm explorado a combinação de diferentes tipos de características para o reconhecimento de imagética motora, com o objetivo de utilizar as informações complementares das características. Wang *et al.* (2018) propuseram a combinação de características espaciais e temporais, extraídas através do algoritmo CSP e de um modelo auto-regressivo, respectivamente. Tan *et al.* (2017) propuseram a combinação de características espaciais e espectrais, extraídas através dos algoritmo CSP e DWT, respectivamente. Joadder *et al.* (2019a) propuseram a combinação de quatro diferentes características, compreendendo características temporais e espectrais. Kim *et al.* (2018) propuseram um método de engenharia de características que possam refletir tempo, frequência e espaço.

Seleção de Características

A tarefa de seleção de características consiste em, a partir do vetor de características, selecionar aquelas mais importantes de forma a reduzir sua dimensão e, ao mesmo tempo, reter o máximo possível das informações discriminatórias entre as classes (THEODORIDIS; KOUTROUMBAS, 2009). Dado que um grande número de características se traduz em um grande número de parâmetros do classificador, a seleção de um número mínimo de características discriminatórias entre as classes reduz a complexidade do classificador (THEODORIDIS; KOUTROUMBAS, 2009). Além disso, a redução de complexidade tende a gerar um classificador com maior capacidade de generalização.

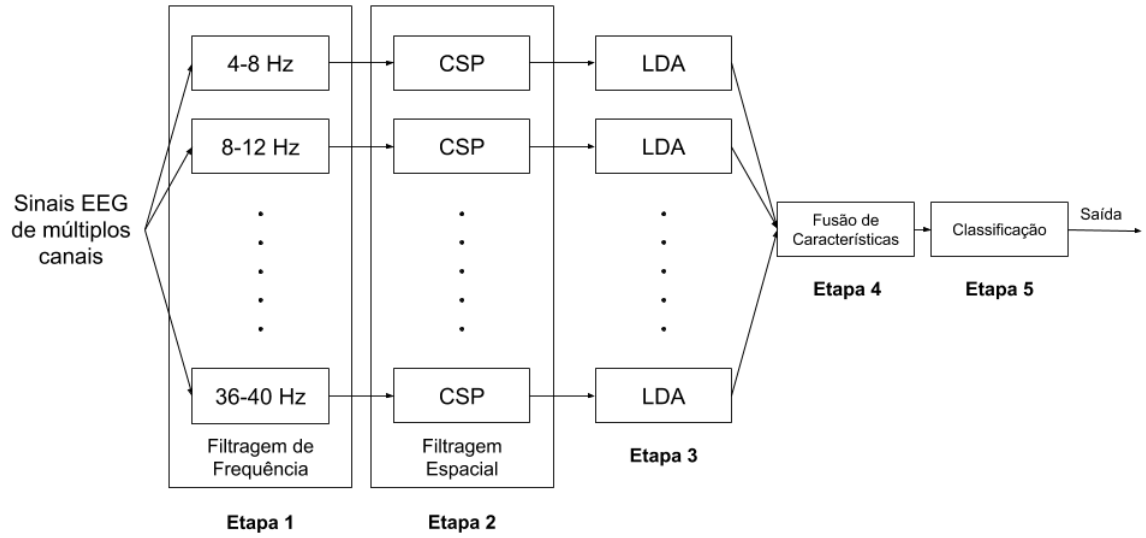
A seleção de características pode ser subdividida em duas técnicas: *filter* e *wrapper* (THEODORIDIS; KOUTROUMBAS, 2009). A técnica *filter* é independente do classificador. Uma pontuação é atribuída a cada característica individualmente e, baseado em alguma métrica de avaliação, um subconjunto das características com a maior pontuação é selecionado, descartando as restantes sem a aplicação de qualquer tipo de classificador (WANG *et al.*, 2018). A técnica *wrapper* é dependente do classificador. Para cada combinação do vetor de características, a probabilidade de erro do classificador deve ser estimada e a combinação que resulta na probabilidade mínima de erro é selecionada (THEODORIDIS; KOUTROUMBAS, 2009). Devido à eficiência das técnicas *filter* e sua simplicidade, elas são as mais comumente utilizadas (WANG *et al.*, 2018).

A partir de características extraídas no domínio do tempo, no domínio da frequência, no domínio tempo-frequência e no domínio espacial, alguns estudos têm utilizado algoritmos de seleção de características para selecionar aquelas mais discriminatórias para o problema de reconhecimento de imagética motora (JOADDER *et al.*, 2019b; SREEJA *et al.*, 2017). Entretanto, a principal utilização da seleção de características no reconhecimento de imagética motora é para a obtenção das bandas de frequência mais informativas por sujeito. De forma geral, algoritmos como o CSP filtram os sinais de EEG no intervalo de interesse da imagética motora e geram características baseadas na variância destes sinais. Porém, dado que a variância dos sinais de EEG filtrados em uma determinada banda de frequência corresponde à energia nessa banda (RAO, 2013) e que, para diferentes sujeitos, as oscilações ERD/ERS devem ocorrer em diferentes bandas

de frequência (TANG *et al.*, 2019b), a utilização de toda a banda de interesse da imagética motora para todos os sujeitos pode resultar em uma diminuição de desempenho do algoritmo. Visando aumentar o desempenho através da seleção automática de bandas de frequência específicas por sujeito, variações do algoritmo CSP têm sido desenvolvidas. De forma geral, estes algoritmos utilizam bancos de filtros para decompor os sinais de EEG em múltiplas sub-bandas de frequência e selecionam as características relacionadas com as bandas de frequência mais informativas para cada sujeito.

O algoritmo *Sub Band Common Spatial Pattern* (SBCSP), proposto por Novi *et al.* (2007), é composto de cinco etapas (Figura 20). A Etapa 1 consiste na decomposição dos sinais de EEG em múltiplas sub-bandas de frequência através da aplicação de um banco de filtros. A Etapa 2 consiste na engenharia de características específicas por banda de frequência através da aplicação do algoritmo CSP. A Etapa 3 consiste na redução de dimensionalidade dos vetores de características de cada sub-banda, transformando-os em vetores unidimensionais através da aplicação do algoritmo de Análise de Discriminantes Lineares (do inglês, *Linear Discriminant Analysis* - LDA). A Etapa 4 consiste na fusão das características e seleção daquelas relacionadas com as bandas de frequência mais informativas. A Etapa 5 consiste na classificação das características.

Figura 20 – Arquitetura do algoritmo SBCSP.

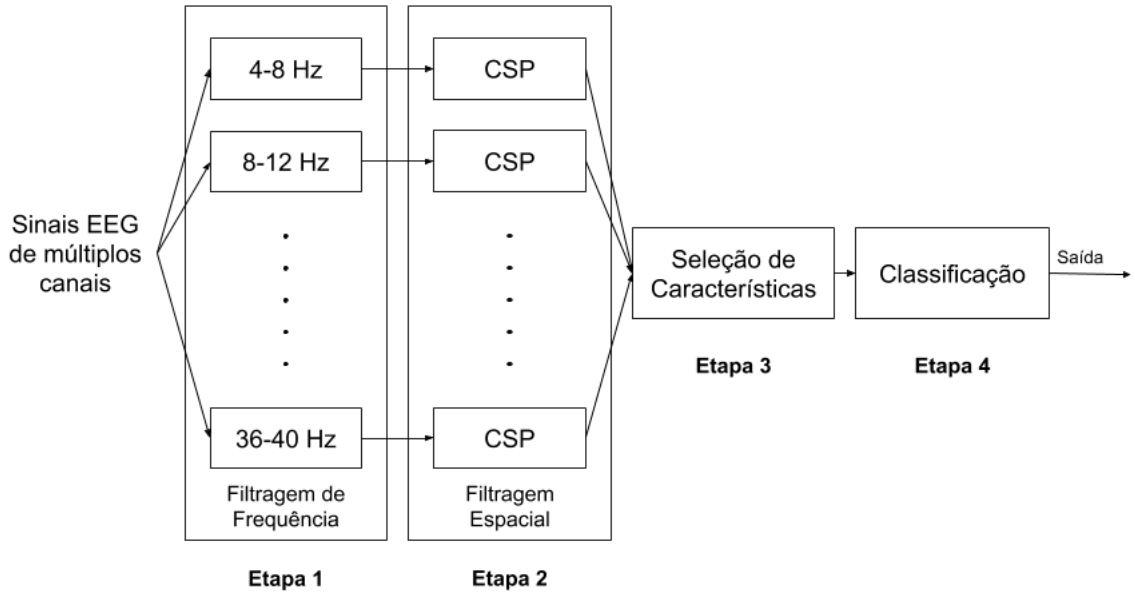


Fonte: Adaptado de Ang *et al.* (2008).

Uma modificação no algoritmo SBCSP foi proposta por Ang *et al.* (2008), produzindo o algoritmo *Filter Bank Common Spatial Pattern* (FBCSP), vencedor da IV Competição de Interfaces Cérebro-Máquina para as bases de dados II-A e II-B (TANGERMANN *et al.*, 2012). O algoritmo FBCSP substitui as Etapas 3 e 4 do algoritmo SBCSP por uma única etapa de seleção de características (Figura 21). Essa modificação torna o algoritmo FBCSP, em relação ao

algoritmo SBCSP, mais genérico, visto que nas Etapas 3 e 4 quaisquer algoritmos de seleção de características e classificação da literatura de aprendizado de máquina e inteligência computacional podem ser empregados (ANG *et al.*, 2008). No estudo para a avaliação de desempenho, em termos de acurácia, é apresentada uma comparação de diversos algoritmos de seleção de características, incluindo algoritmos baseados em técnicas *filter* e *wrapper*.

Figura 21 – Arquitetura do algoritmo FBCSP.



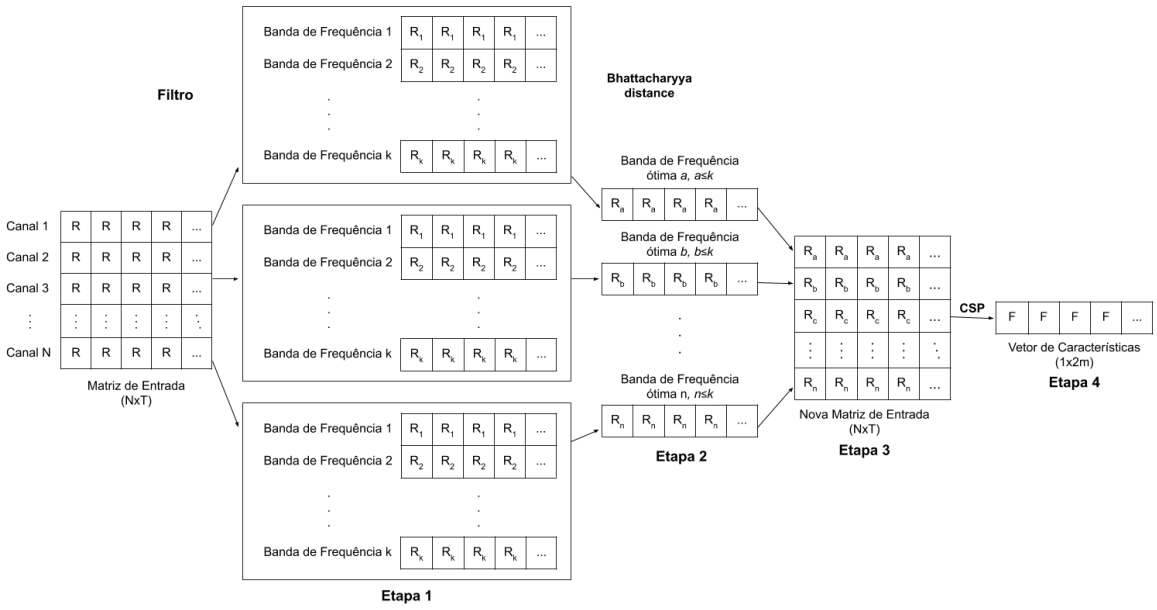
Fonte: Adaptado de Ang *et al.* (2008).

O algoritmo B-CSP (TANG *et al.*, 2019b) propõe a seleção da banda de frequência mais informativa para cada canal dos sinais de EEG (Figura 22). Na Etapa 1, os sinais de cada canal são decompostos em k bandas de frequências. Na Etapa 2, a banda de frequência mais informativa para cada canal é selecionada através do medida de distância de *Bhattacharyya*[14]. Um maior valor da distância de *Bhattacharyya* em uma banda de frequência significa que os sinais de EEG dessa banda de frequência podem resultar no melhor desempenho de classificação do algoritmo (BAI *et al.*[15], 2007 apud TANG *et al.*, 2019b). Na Etapa 3, os sinais de cada canal são filtrados de acordo com a banda de frequência selecionada, gerando uma nova matriz de entrada. Na Etapa 4, um vetor de características é construído utilizando as variâncias dos primeiros e últimos m filtros espaciais, obtidos através do algoritmo CSP.

[14] A medida de distância de *Bhattacharyya* é utilizada como uma medida de separabilidade entre classes (THEODORIDIS; KOUTROUMBAS, 2009).

[15] BAI, O. et al. A high performance sensorimotor beta rhythm-based brain-computer interface associated with human natural motor behavior. IOP Publishing Ltd, 2007.

Figura 22 – Arquitetura do algoritmo B-CSP.



Fonte: Adaptado de Tang *et al.* (2019b).

Classificação

A tarefa de classificação de um padrão desconhecido $\mathbf{x}$ em uma de C possíveis classes consiste na busca da classe $\hat{c}$ que possui a maior probabilidade de ter gerado o padrão $\mathbf{x}$, isto é,

$$\hat{c} = \arg\max_{c} P(\mathbf{x}|c), \tag{2.6}$$

assumindo que todas as classes possuem a mesma probabilidade de ocorrer (THEODORIDIS; KOUTROUMBAS, 2009). Assim, o classificador é estimado determinando os parâmetros das funções de densidade probabilidade $P(\mathbf{x}|c)$ para todas as classes. Este processo, chamado de treinamento, é realizado a partir de um conjunto de dados composto de N amostras e representado por dois vetores, o vetor de amostras $\mathbf{x} \in \mathbb{R}^N$ e o vetor dos respectivos rótulos de classe $\mathbf{y} \in \mathbb{N}^N$ (ANG *et al.*, 2008). Após o treinamento, pode-se utilizar o classificador para estimar as probabilidades de pertencimento a cada classe para novas amostras. Algoritmos para o reconhecimento de imagética motora baseados em engenharia de características geralmente são aplicados em cenários de classificação binária. Embora algumas técnicas possam ser aplicadas na classificação de múltiplas classes, a utilização de algoritmos baseados em aprendizado profundo é mais comum.

Os classificadores mais utilizados no problema de reconhecimento de imagética motora incluem:

- Máquina de Vetores de Suporte (do inglês, *Support Vector Machine* - SVM): baseia-se na

projeção dos dados em uma dimensão suficientemente grande, de tal forma que dados de duas classes possam sempre ser linearmente separáveis (DUDA; HART; STORK, 2000). Tem sido amplamente utilizado para o reconhecimento de imagética motora, demonstrando altas taxas de acurácia (JOADDER *et al.*, 2019b; KHORSHIDTALAB; SALAMI; HAMEDI, 2012; KHORSHIDTALAB; SALAMI; HAMEDI, 2013; KUMAR; SHARMA, 2018; KUMAR; SHARMA; TSUNODA, 2017; KUMAR; SHARMA; TSUNODA, 2019; NOVI *et al.*, 2007; THOMAS* *et al.*, 2009);

- Análise de Discriminantes Lineares: projeta os dados em um hiperplano, de tal forma que minimize a variância dos dados de uma mesma classe, enquanto que maximiza a distância entre as classes (ALI *et al.*, 2019). Geralmente é utilizado em problemas binários, mas vários hiperplanos podem ser utilizados para resolver problemas de múltiplas classes. Tem sido utilizado para o reconhecimento de imagética motora devido ao seu alto desempenho e baixa demanda computacional na classificação de duas classes de imagética motora (ALI *et al.*, 2019; JOADDER *et al.*, 2019a).

Tang *et al.* (2019b) utilizaram uma rede neural para classificar as características produzidas pelos algoritmos CSP e B-CSP em 2 classes de imagética motora. Na avaliação de desempenho foi utilizada a técnica de validação cruzada *10-fold* para a estimativa da acurácia. Em uma base de dados pública com sinais de EEG de imagética motora das classes mão esquerda e mão direita, provenientes de 5 sujeitos, foi obtida uma acurácia de $80.0\%(\pm 5.30\%)$ para o algoritmo CSP e $84.50\%(\pm 5.42\%)$ para o algoritmo B-CSP. Em uma base de dados pública com sinais de EEG de imagética motora das classes mão esquerda e pés, provenientes de 2 sujeitos, foi obtida uma acurácia de $87.50\%(\pm 3.54\%)$ para o algoritmo CSP e $91.25\%(\pm 1.77\%)$ para o algoritmo B-CSP. Em uma base de dados experimental com sinais de EEG de imagética motora das classes mão esquerda e pés, provenientes de 3 sujeitos, foi obtida uma acurácia de $86.17\%(\pm 2.13\%)$ para o algoritmo CSP e $90.43\%(\pm 4.26\%)$ para o algoritmo B-CSP.

Novi *et al.* (2007) utilizaram o classificador SVM para a classificação de 2 classes de imagética motora. Na avaliação de desempenho foi utilizada a técnica de validação cruzada *10-fold* para a estimativa da acurácia. Em uma base de dados pública com sinais de EEG provenientes de 5 sujeitos, foi obtida uma acurácia média de 90.0%.

Ang *et al.* (2008) utilizaram diversos classificadores para avaliar o desempenho das características extraídas pelo algoritmo FBCSP para a classificação de 2 classes de imagética motora. Na avaliação de desempenho foi utilizada a técnica de validação cruzada *10x10-fold* para a estimativa da acurácia. A avaliação foi realizada em uma base de dados com sinais de EEG provenientes de 5 sujeitos. Dentre os classificadores utilizados, aqueles que obtiveram as melhores acurácias foram o NBPW (do inglês, *Naïve Bayesian Parzen Window*), SVM e LDA com acurácias de $90.3\%(\pm 0.7\%)$, $90.0\%(\pm 0.8\%)$ e $89.9\%(\pm 0.9\%)$, respectivamente.

Os algoritmos baseados em engenharia de características até aqui citados são todos

dependentes do sujeito. Dessa forma, necessitam de aquisição de dados e treinamento para todo novo sujeito. Além de ser um processo tedioso, pode tornar inviável sua aplicabilidade em interfaces cérebro-máquina nos cenários de doenças degenerativas muito graves, os quais são o maior foco de desenvolvimento e pesquisa no campo atualmente. Na tentativa de resolver este problema, a ideia de algoritmos independentes do sujeito têm surgido. Como um primeiro passo, Lotte, Guan & Ang (2009) decidiram realizar comparações com algoritmos presentes na literatura da época (i.e., dependentes do sujeito) entre os cenários de classificação dependente e independente do sujeito. Dentre os algoritmos considerados nessa pesquisa estão o CSP e o FBCSP. No cenário de classificação dependente do sujeito, o treinamento e a avaliação de desempenho foram realizados utilizando os dados de um único sujeito. No cenário de classificação independente do sujeito, o treinamento e a avaliação de desempenho foram realizados utilizando dados de diferentes sujeitos. Considerando uma base de dados com sinais de EEG de N sujeitos (dados de $N-1$ sujeitos são utilizados para o treinamento e dados de 1 sujeito são utilizados para a avaliação de desempenho do algoritmo). Seguindo essa mesma abordagem de treinamento e avaliação de desempenho, alguns outros algoritmos têm sido propostos (JOADDER *et al.*, 2019a; JOADDER *et al.*, 2019b). Outra técnica que tem sido explorada para reduzir a variabilidade entre diferentes sujeitos e diferentes sessões de aquisição de sinais cerebrais é a normalização de características através da aplicação de métodos como o *Z-score* (SREEJA *et al.*, 2017; ZHANG *et al.*, 2017; ZHANG *et al.*, 2018). Zhang *et al.* (2017) realizou uma comparação entre diferentes métodos de normalização de características, para ser utilizado como etapa de pré-processamento em um algoritmo de reconhecimento de múltiplas classes de imagética motora independente do sujeito, obtendo os melhores resultados com o método *Z-score*. Este método coloca todas as características em escalas similares através da equação:

$$Z = \frac{X - \mu}{\sigma}, \tag{2.7}$$

onde μ e σ são a média e o desvio padrão estimados a partir do conjunto de vetores de características denotado por X.

Abordagens baseadas em engenharia de características demandam um alto conhecimento do domínio para que se possa obter características discriminativas e classificá-las com sucesso. As suposições que definem a escolha das técnicas e algoritmos nem sempre capturam as características mais discriminativas para o problema em questão. O problema pode ser agravado quando informações discriminativas são descartadas no processo de engenharia de características. Os algoritmos, quando expostos a cenários de múltiplas classes ou múltiplos sujeitos, sofrem uma queda de desempenho.

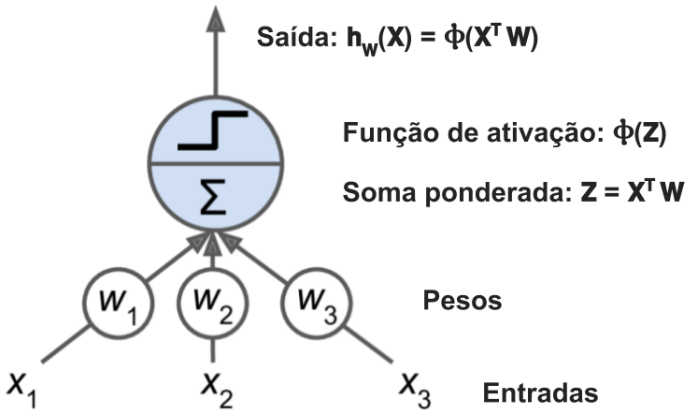
2.7.3 ABORDAGENS BASEADAS EM APRENDIZADO PROFUNDO

O Aprendizado Profundo (*Deep Learning*), subárea do aprendizado de máquina, tem sido explorado para o reconhecimento de imagética motora. Ele utiliza técnicas baseadas em

redes neurais artificiais que possuem múltiplas camadas ocultas, estrutura que permite capturar características representativas de alto nível e dependências latentes, através das estruturas profundas (ZHANG *et al.*, 2019). O conceito de redes neurais artificiais não é nada recente. Em 1943, McCulloch & Pitts (1943) propuseram um modelo matemático baseado em lógica proposicional para simular o comportamento de uma rede neural biológica (considerada primeira arquitetura de redes neurais artificiais). Desde então inúmeras arquiteturas têm sido propostas, algumas bastante inspiradas nas redes neurais biológicas, outras nem tanto. De qualquer modo, para evitar confusão, conforme Goodfellow, Bengio & Courville (2016), é melhor pensar nas redes neurais do ponto de vista matemático como um aproximador de funções, projetado para obter generalização estatística utilizando alguns conceitos básicos da neurociência para a sua arquitetura, ao invés de modelos projetados para representar a atividade cerebral. A generalização estatística consiste na inferência de resultados de uma população utilizando dados obtidos de uma amostra representativa desta população selecionada de forma aleatória (MILLS; DUREPOS; WIEBE, 2010).

A menor unidade da rede neural é o neurônio artificial (Figura 23). Ele é responsável pelo processamento das informações. A comunicação entre diferentes neurônios se dá por meio de conexões que podem ser fortes ou fracas, determinando assim como a informação deve ser processada. O estado interno de cada neurônio é representado por todas as conexões de entrada originadas de outros neurônios. A saída de cada neurônio é calculada através de uma função de ativação aplicada sobre seu estado interno (VASILEV *et al.*, 2019).

Figura 23 – Neurônio artificial.



Fonte: Adaptado de Géron (2019).

De forma geral, redes neurais são compostas de uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. A camada de entrada geralmente possui a mesma dimensão dos dados de entrada (PATTERSON, 2017). As camadas ocultas permitem que a rede neural realize o aprendizado de representações através dos dados brutos. Este aprendizado é codificado através de pesos existentes nas conexões entre as camadas. A camada de saída da rede neural pode retornar um valor (para problemas de regressão) ou um conjunto de probabilidades

(para problemas de classificação) (PATTERSON, 2017). As camadas de uma rede neural são conectadas umas as outras através das conexões entre seus neurônios. Uma camada é dita totalmente conectada quando todos seus neurônios estão conectados a todos os neurônios da camada anterior (GéRON, 2019).

Uma rede neural sem a adição de não-linearidade entre as camadas ocultas resume-se a um encadeamento de múltiplas transformações lineares. Este encadeamento resulta em uma única transformação linear, impossibilitando que a rede neural consiga modelar funções não-lineares mais complexas. Este problema é resolvido através da aplicação de funções de ativação não-lineares nas saídas dos neurônios das camadas ocultas. Dessa forma, redes neurais com funções de ativação não-lineares e um número suficientemente grande de camadas ocultas podem teoricamente aproximar qualquer função contínua (GéRON, 2019). Dentre as funções de ativações mais populares para as camadas ocultas estão a função de Unidade Linear Retificada (do inglês, *Rectified Linear Unit* - ReLU) com suas variantes e a função de Unidade Linear Exponencial (do inglês, *Exponential Linear Unit* - ELU):

- **ReLU**: é a função recomendada como padrão para utilização com a maioria das redes neurais (GOODFELLOW; BENGIO; COURVILLE, 2016). É uma função contínua, porém não diferenciável para $x = 0$, onde x é o valor de entrada dos neurônios da rede neural:

$$ReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}. \tag{2.8}$$

- **Leaky ReLU**: durante o treinamento, a função ReLU pode acabar inutilizando alguns neurônios produzindo como saída sempre o valor 0. A função *Leaky ReLU* introduz o hiper-parâmetro α para definir o declive da função para $x < 0$, normalmente fixado em um valor pequeno como 0.01, de forma a garantir que os neurônios nunca tornem-se inutilizáveis (DING; QIAN; ZHOU, 2018):

$$LeakyReLU_{\alpha}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}. \tag{2.9}$$

- **Randomized Leaky ReLU (RReLU)**: utiliza valores aleatórios para o parâmetro α da Função 2.9. Durante o treinamento, α é amostrado aleatoriamente a partir de uma distribuição uniforme e no teste é fixado para o valor médio da distribuição (DING; QIAN; ZHOU, 2018).

- **Parametric Leaky ReLU (PReLU)**: nesta função, o parâmetro α da Função 2.9 passa ser um parâmetro treinável da rede neural (DING; QIAN; ZHOU, 2018).

- **ELU**: a função de Unidade Linear Exponencial (do inglês, *Exponential Linear Unit*) possui uma saída média próxima de zero, o que contribui para uma convergência mais rápida

(DING; QIAN; ZHOU, 2018):

$$ELU_\alpha(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}, \text{onde } \alpha > 0. \tag{2.10}$$

Além de introduzir a não-linearidade, funções de ativação podem ser utilizadas nas camadas de saída para modelar a saída da rede neural de diferentes maneiras. Para o problema da classificação de um vetor de características $\mathbf{x}$, pode-se utilizar a função de ativação Sigmoide no cenário de classificação binária:

$$f(\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{x})}, \tag{2.11}$$

onde $f(\mathbf{x})$ retorna valores entre 0 e 1 possibilitando a interpretação da saída da rede como a probabilidade de pertencimento do vetor de características $\mathbf{x}$ à classe positiva. No cenário de classificação de múltiplas classes pode-se utilizar uma generalização da função Sigmoide, a função de ativação *Softmax*:

$$f(\mathbf{x}) = \frac{exp(\mathbf{x})}{\sum_{j=1}^{n} exp(\mathbf{x}_j)}, \tag{2.12}$$

onde $f(\mathbf{x})$ retorna uma distribuição de probabilidades das classes.

As redes neurais baseadas em aprendizado profundo são redes neurais de múltiplas camadas totalmente conectadas, conhecidas como redes neurais profundas. A arquitetura de uma rede neural profunda pode ser composta por uma ou mais redes neurais menores. Arquiteturas compostas por mais de um tipo de rede neural são conhecidas como arquiteturas híbridas. Dentre as redes neurais utilizadas nas arquiteturas propostas para o reconhecimento de imagética motora estão: *Autoencoders*, Redes Neurais Convolucionais (do inglês, *Convolutional Neural Networks* - CNN) e as Redes Neurais Recorrentes (do inglês, *Recurrent Neural Networks* - RNN).

O treinamento de redes neurais profundas utiliza até hoje o algoritmo de retro-propagação introduzido por Rumelhart, Hinton & Williams (1986):

> O procedimento ajusta repetidamente os pesos das conexões na rede, a fim de minimizar uma medida da diferença entre o vetor de saída real da rede e o vetor de saída desejado. Como resultado dos ajustes de peso, as unidades internas "ocultas" que não fazem parte da entrada ou da saída, representam características importantes do domínio da tarefa e as regularidades na tarefa são capturadas pelas interações dessas unidades (RUMELHART; HINTON; WILLIAMS, 1986, p. 533, tradução nossa).

O grande desafio do aprendizado de máquina é a construção de modelos que tenham um bom desempenho não somente nos dados de treinamento, mas principalmente em dados nunca antes vistos pelo modelo. Conforme Goodfellow, Bengio & Courville (2016), a habilidade de ter um bom desempenho em dados nunca antes vistos é chamada de generalização. Quanto maior for a complexidade do modelo, maiores são as chances de que o modelo detecte padrões sutis presentes nos dados. O grande problema é que todo tipo de dado inerentemente possui algum

tipo de ruído e a detecção de padrões originados pelo ruído ocasiona o problema conhecido como sobreajuste (do inglês, *overfiting*), onde o modelo se ajusta quase perfeitamente aos dados de treinamento e no entanto não generaliza bem em novos dados. Redes neurais profundas tipicamente possuem milhares de parâmetros, o que as torna altamente propensas à *overfitting*. De forma geral, para que algoritmos de aprendizado de máquina tenham um bom desempenho é necessário equilibrar a complexidade do modelo com a complexidade da tarefa e com a quantidade de dados de treinamento disponível (GOODFELLOW; BENGIO; COURVILLE, 2016).

A técnica para a redução de *overfitting* é chamada de regularização. Goodfellow, Bengio & Courville (2016) definem regularização como qualquer modificação feita em um algoritmo de aprendizado com o objetivo de reduzir seu erro de generalização, mas não seu erro de treinamento. Uma abordagem para a regularização consiste na restrição dos valores dos pesos da rede neural. Baseada nesta abordagem, uma técnica muito utilizada é a *Max Norm* (DOSE *et al.*, 2018; WANG *et al.*, 2020). Ela garante que a norma L2 dos pesos seja restringida a um valor menor que o hiper-parâmetro r. Esta restrição é realizada apenas reescalando os pesos retornados pela função objetivo, não sendo necessário alterá-la (GéRON, 2019). Uma das técnicas de regularização mais populares em redes neurais profundas é o *Dropout*. De forma geral, todo neurônio de entrada tem uma probabilidade p de ser completamente ignorado em toda etapa do treinamento, onde p é um hiper-parâmetro conhecido como taxa de *dropout* (GéRON, 2019).

As redes neurais profundas podem ser subdivididas em três classes: as discriminativas, as generativas e as representativas. As redes neurais discriminativas são aquelas utilizadas para a classificação dos dados de entrada em classes conhecidas através do aprendizado adaptativo de características discriminativas (ZHANG *et al.*, 2019). As redes neurais generativas são utilizadas para a reconstrução ou geração de novos dados, podendo ser empregadas para o aumento das bases de dados de treinamento dos algoritmos (ZHANG *et al.*, 2019). As redes neurais representativas são utilizadas para o aprendizado de características representativas para um determinado conjunto de dados (ZHANG *et al.*, 2019).

Muitos algoritmos para o reconhecimento de imagética motora baseados em aprendizado profundo realizam etapas de pré-processamento computacionalmente complexas para a geração dos dados de entrada dos algoritmos. Alguns estudos têm explorado a transformação dos sinais de EEG em imagens de representações dos sinais no domínio tempo-frequência, para serem então utilizadas como dados de entrada para CNNs (LEE; CHOI, 2018; LI *et al.*, 2020; TABAR; HALICI, 2016). Outros estudos têm explorado a aplicação de técnicas oriundas da engenharia de características para produzir os dados de entrada das redes neurais. Como por exemplo, Kwon *et al.* (2019), que extraem uma matriz espacial-espectral para utilizar como dados de entrada para uma CNN, através de um banco de filtros e do algoritmo CSP. Neste trabalho serão considerados os algoritmos que recebem os dados brutos e automaticamente realizam o aprendizado e classificação. No máximo, serão considerados alguns algoritmos que realizam um pré-processamento mínimo nos sinais para a redução de ruídos ou para a reorganização do

formato dos dados.

Redes Neurais Convolucionais

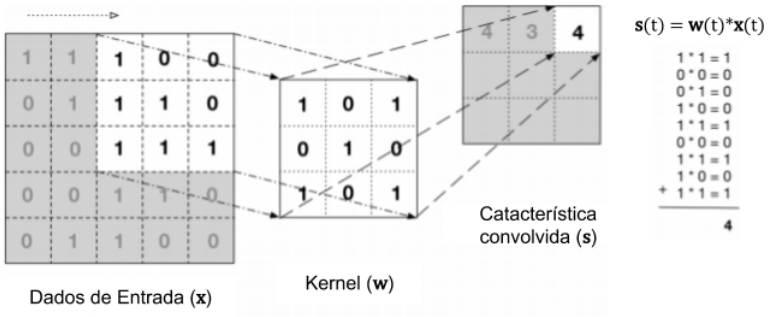
A CNN é uma rede neural composta por uma camada de entrada, múltiplos pares de camadas de convolução e *pooling*, e uma camada de saída totalmente conectada (TABAR; HALICI, 2016). Possuem uma arquitetura especializada para trabalharem com qualquer tipo de dado que possa ser representado por múltiplos vetores. As CNNs têm obtido resultados extraordinários em tarefas visuais complexas, permitindo o desenvolvimento de serviços de busca por imagem, carros autônomos, classificação automática de vídeos, entre outros (GéRON, 2019).

O termo "convolucional" vem do fato de que estas redes neurais, em pelo menos uma de suas camadas, utilizam a operação matemática conhecida como convolução (GOODFELLOW; BENGIO; COURVILLE, 2016). A operação de convolução (Figura 24) constrói um mapa de características operando sobre dois parâmetros, conforme

$$\mathbf{s}(t) = \mathbf{x}(t) * \mathbf{w}(t), \tag{2.13}$$

onde $\mathbf{x}(t)$ é um vetor dos dados de entrada da camada em que a operação de convolução é aplicada e $\mathbf{w}(t)$, conhecido também como *kernel*, é um vetor de parâmetros, os quais são adaptados automaticamente durante o processo de aprendizado da rede neural (GOODFELLOW; BENGIO; COURVILLE, 2016). Visando a redução do custo computacional, um terceiro parâmetro pode ser utilizado para definir um número de posições para incrementar t em cada etapa, gerando assim uma subamostragem do resultado da operação completa de convolução (GOODFELLOW; BENGIO; COURVILLE, 2016). A quantidade de mapas de características gerados pelas operações convolucionais é definida por um hiper-parâmetro comumente chamado de filtros ou mesmo de mapas de características.

Figura 24 – Operação Convolucional.



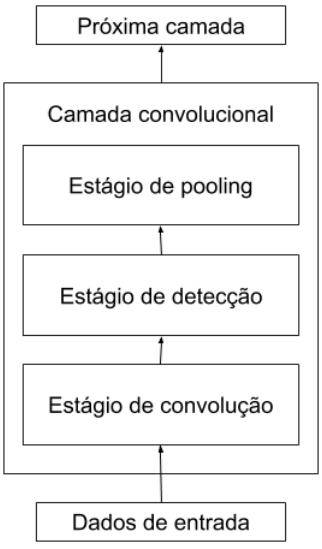
Fonte: Adaptado de Patterson (2017).

Uma camada típica de uma rede neural convolucional consiste basicamente de três estágios (Figura 25) (GOODFELLOW; BENGIO; COURVILLE, 2016). No primeiro estágio é realizada a aplicação de várias convoluções sobre os dados de entrada da camada, produzindo um conjunto

de ativações lineares. No segundo estágio, uma função de ativação é aplicada para introduzir a não-linearidade. No terceiro estágio, uma função de *pooling* é aplicada sobre os dados, a qual substitui a saída da camada em um ponto específico com um resumo estatístico dos pontos próximos (GOODFELLOW; BENGIO; COURVILLE, 2016). A aplicação de uma função de *pooling* auxilia na geração de uma representação mais invariante a pequenas translações, muito útil para cenários onde é uma preocupação maior para o modelo identificar a presença de determinadas características, do que a sua localização exata (GOODFELLOW; BENGIO; COURVILLE, 2016). Dentre os tipos mais comuns de funções de *pooling* estão:

- *Average Pooling*: realiza a subamostragem dos dados de entrada retornando o valor médio sobre a janela (definida pelo parâmetro do tamanho do *pooling*);
- *Max Pooling*: realiza a subamostragem dos dados de entrada retornando o valor máximo sobre a janela (definida pelo parâmetro do tamanho do *pooling*);
- *Global Average/Max Pooling*: realiza a subamostragem dos dados de entrada retornando o valor médio ou máximo (i.e., de acordo com a função) sobre a dimensão temporal.

Figura 25 – Arquitetura de uma camada típica de uma rede neural convolucional.



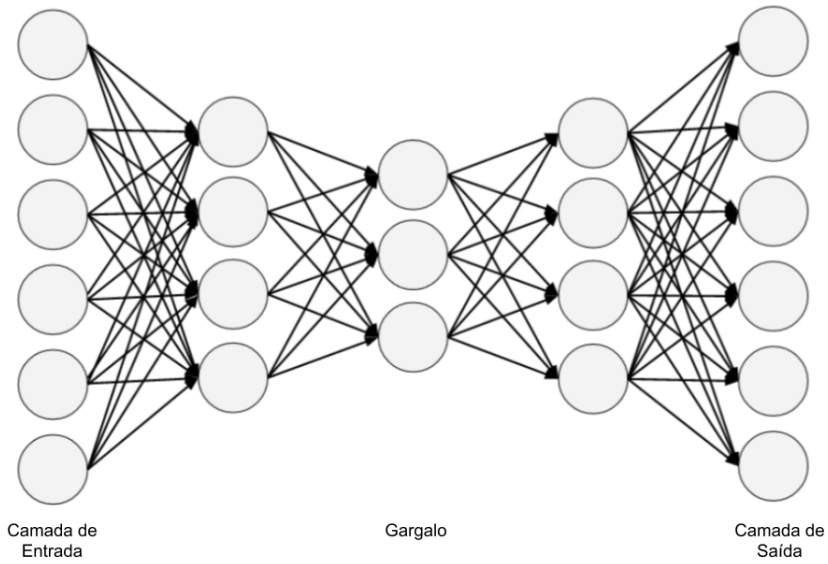
Fonte: Adaptado de Goodfellow, Bengio & Courville (2016).

Autoencoders

Um *Autoencoder* (Figura 26) é uma rede neural geralmente composta por três camadas e é capaz de aprender de forma não supervisionada representações densas dos dados de entrada (GéRON, 2019). Pode ser utilizada para redução de dimensionalidade, treinamento não supervisionado de redes neurais profundas e geração de novos dados similares aos dados de

treinamento existentes (GéRON, 2019). Um *Autoencoder* reconstrói os dados de entrada com algumas restrições que o impedem de aprender a simplesmente realizar uma cópia dos dados de entrada. As camadas ocultas de um *Autoencoder* são projetadas com dimensões menores que as camadas de entrada e saída, de forma a atuarem como um gargalo. Essa estrutura permite o aprendizado das características mais importantes dos dados de entrada, descartando as características menos importantes (GéRON, 2019). Um *Autoencoder* Empilhado é um *Autoencoder* composto por múltiplas camadas ocultas, o que permite o aprendizado de representações mais complexas dos dados de entrada (GéRON, 2019).

Figura 26 – Arquitetura de uma rede neural *Autoencoder*.



Fonte: Adaptado de Patterson (2017).

Redes Neurais Recorrentes

A RNN possui como diferencial uma arquitetura especializada para trabalhar com dados de entrada e/ou saída que possuam relações temporais entre si. Característica esta não considerada por classificadores como o SVM, ou mesmo outras redes neurais *feedforward*, que assumem independência temporal entre as amostras (PATTERSON, 2017). Pela sua capacidade de trabalhar com séries temporais é extremamente útil em aplicações de processamento de linguagem natural, como tradução automática de texto ou mesmo a conversão de fala para texto, onde sentenças, documentos inteiros de texto ou sinais de áudio podem ser recebidos como dados de entrada da rede neural (GéRON, 2019).

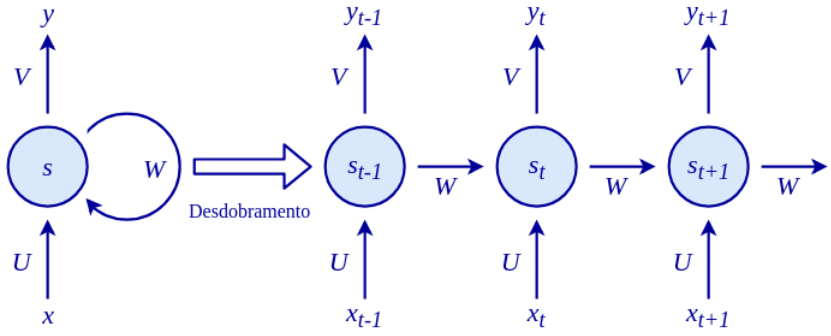
O termo “recorrente” referencia a forma pela qual os neurônios artificiais da rede neural são projetados de forma a modelar a dependência temporal dos dados de entrada. Isso é realizado através da aplicação recorrente da mesma função sobre a sequência de dados:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{x}_t), \tag{2.14}$$

onde f é uma função de ativação, $\mathbf{s}_t$ é um vetor do estado interno da rede no instante de tempo t e $\mathbf{x}_t$ é um vetor dos dados de entrada da rede no instante de tempo t (VASILEV *et al.*, 2019). Ao contrário de outras redes neurais, nas quais o estado interno do neurônio artificial depende somente do seu valor de entrada e seus pesos associados, na RNN, o estado interno no instante de tempo t depende da entrada corrente e de todos os estados internos $\mathbf{s}_{t'}$ anteriores (para $t' \leq t$). Essa dependência temporal é ilustrada na Figura 27, onde $\mathbf{U}$, $\mathbf{W}$ e $\mathbf{V}$ são os três conjuntos de pesos de uma RNN, os quais, conforme Vasilev *et al.* (2019), são utilizados para:

- $\mathbf{U}$: transformar a entrada $\mathbf{x}_t$ no estado interno $\mathbf{s}_t$;
- $\mathbf{W}$: transformar o estado interno anterior $\mathbf{s}_{t-1}$ no estado interno corrente $\mathbf{s}_t$;
- $\mathbf{V}$: transformar o estado interno $\mathbf{s}_t$ na saída $\mathbf{y}_t$.

Figura 27 – Desdobramento de um neurônio artificial de uma RNN.



Fonte: Adaptado de Vasilev *et al.* (2019).

Essas transformações são realizadas através de alguma transformação linear dos pesos $\mathbf{U}$, $\mathbf{W}$ e $\mathbf{V}$ sobre suas respectivas entradas (VASILEV *et al.*, 2019). Utilizando como transformação linear a soma ponderada, o estado interno e a saída podem ser definidos da seguinte forma:

$$\begin{aligned} \mathbf{s}_t &= f(\mathbf{s}_{t-1} * \mathbf{W} + \mathbf{x}_t * \mathbf{U}), \\ \mathbf{y}_t &= f(\mathbf{s}_t * \mathbf{V}). \end{aligned} \tag{2.15}$$

Redes neurais de uma única camada são capazes somente de classificar classes linearmente separáveis (VASILEV *et al.*, 2019). Por isso é importante estender a definição da RNN para que seja possível o empilhamento de camadas. Logo, o estado interno da RNN na camada l para o instante de tempo t pode ser definido como:

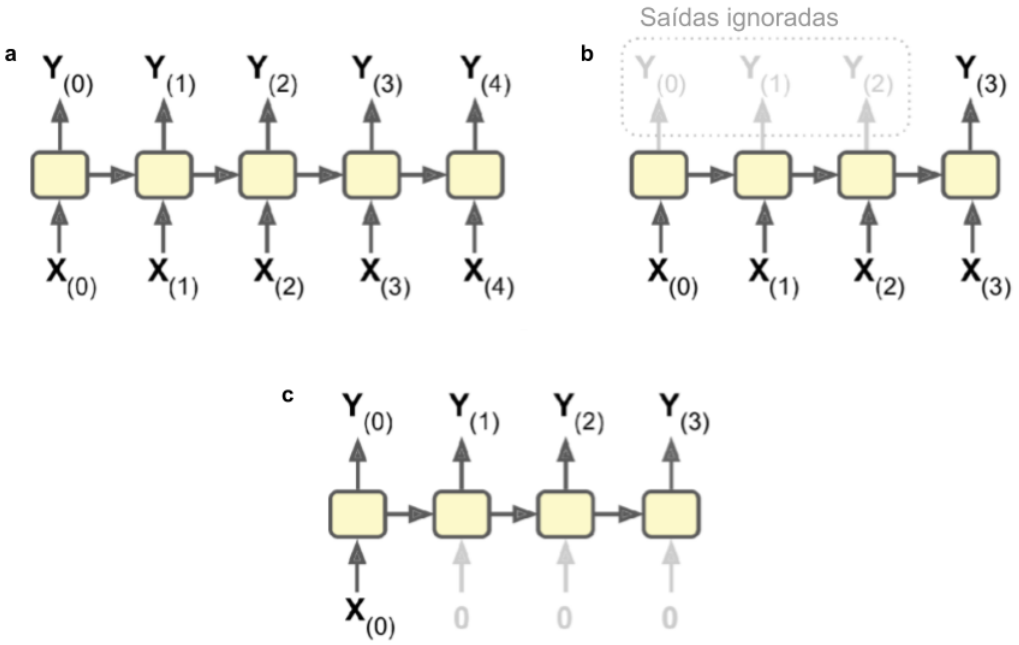
$$\mathbf{s}_t^l = f(\mathbf{s}_{t-1}^l, \mathbf{y}_t^{l-1}), \tag{2.16}$$

onde $\mathbf{s}_{t-1}^l$ é o estado interno anterior da camada corrente e $\mathbf{y}_t^{l-1}$ é a saída para o mesmo instante de tempo na camada anterior (VASILEV *et al.*, 2019).

Diferentemente de outras arquiteturas de redes neurais que possuem uma estrutura bem definida dos dados de entrada e saída, a RNN pode operar sobre múltiplos vetores de características. A Figura 28 ilustra três cenários comuns de dados de entrada e saída de uma RNN:

- Vetores-para-vetores: a RNN recebe múltiplos vetores como entradas e produz múltiplos vetores como saídas, simultaneamente (Figura 28-a);
- Vetores-para-vetor: a RNN ignora todas as saídas, exceto a última (Figura 28-b);
- Vetor-para-vetores: a RNN recebe um único vetor que é repassado entre todas as unidades temporais da RNN, produzindo múltiplos vetores como saídas (Figura 28-c).

Figura 28 – Cenários comuns de dados de entrada e saída de uma RNN.

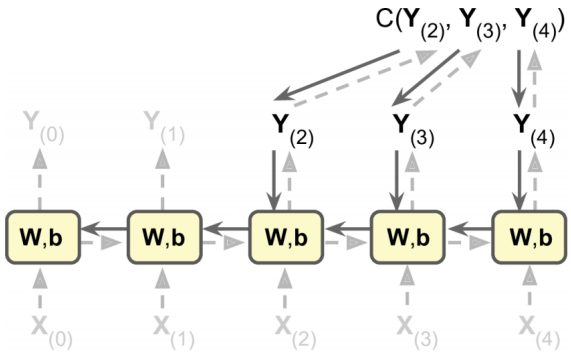


Fonte: Adaptado de Géron (2019).

O treinamento da RNN se dá pelo algoritmo padrão de retro-propagação aplicado através do tempo (Figura 29). Os gradientes são calculados com base em uma função de custo C e retro-propagados por todas as saídas consideradas pela função de custo. Por fim, os pesos $\mathbf{W}$ e o viés b associados a cada instante no tempo são atualizados com base nos gradientes calculados. Como ilustrado na Figura 29, algumas saídas podem ser desconsideradas pela função de custo e pelo fluxo de retro-propagação dos gradientes. Isto ilustra o cenário de arquiteturas que desconsideram algumas saídas como, por exemplo, o cenário ilustrado na Figura 28-b.

Teoricamente, é possível para uma RNN recuperar informações de qualquer instante no tempo, uma vez que os estados internos para cada instante no tempo são armazenados. Entretanto,

Figura 29 – Aplicação do algoritmo padrão de retro-propagação através do tempo.



Fonte: Géron (2019).

na prática, devido a problemas ocasionados no treinamento de uma RNN para sequências de dados muito longas, somente é possível recuperar informações para um pequeno intervalo de tempo. Durante o processo de retro-propagação dos gradientes, para cada ponto no tempo, este irá crescer ou diminuir ocasionando em longas sequências, a explosão ou o desaparecimento do gradiente (LECUN; BENGIO; HINTON, 2015).

Redes de Memória de Longo Prazo

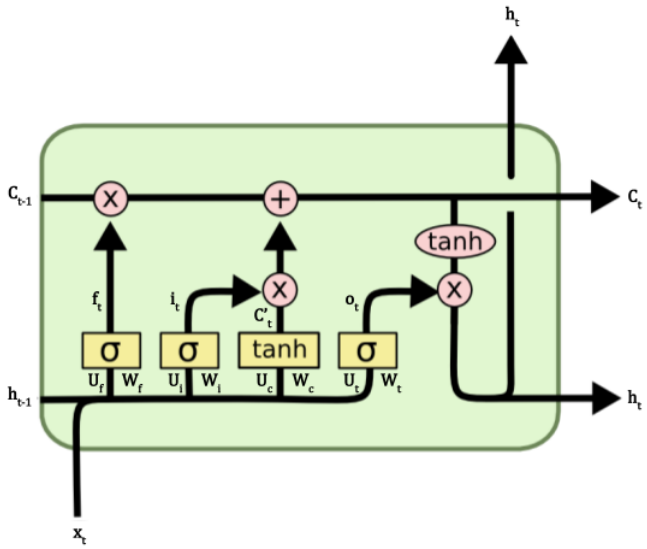
Hochreiter & Schmidhuber (1997) estudaram os problemas enfrentados pela RNN e como resultado de seus estudos propuseram a Rede de Memória de Longo Prazo (do inglês, *Long Short-Term Memory* - LSTM). O objetivo da LSTM é evitar o problema de dependência de longo prazo possibilitando o treinamento da rede em sequências de dados mais longas do que é possível na RNN (VASILEV *et al.*, 2019).

O ponto chave da LSTM está na arquitetura da sua célula de memória (Figura 30). Em adição ao estado interno armazenado na célula de memória da RNN é adicionado um novo estado, o estado da célula. A adição e remoção de informações neste novo estado é regulada por meio de estruturas conhecidas como *gates* (OLAH, 2015). A célula LSTM recebe como dados de entrada o estado interno anterior $\mathbf{h}_{t-1}$, o estado da célula anterior $\mathbf{c}_{t-1}$ e o dado de entrada corrente $\mathbf{x}_t$. Estes dados são vetores que fluem da esquerda para a direita interagindo com os *gates* para produzir o novo estado interno $\mathbf{h}_t$ e o novo estado da célula $\mathbf{c}_t$.

Os *gates* (Figura 31) são estruturas compostas por uma rede neural de uma camada Sigmoide e uma operação de multiplicação elemento a elemento. A função Sigmoide retorna valores entre 0 e 1, representando quanto de cada elemento deve ser transmitido pela operação de multiplicação (OLAH, 2015). Um valor de zero representa que nenhuma informação será transmitida e um valor de 1 representa que toda informação será transmitida.

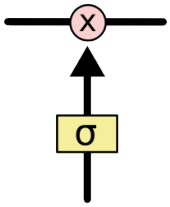
A arquitetura da célula LSTM é composta por uma série de equações (VASILEV *et al.*, 2019). Inicialmente, baseado no estado interno anterior e no dado de entrada, a estrutura *forget*

Figura 30 – Arquitetura da célula LSTM.



Fonte: Adaptado de Olah (2015).

Figura 31 – Gate LSTM.



Fonte: Olah (2015).

gate define quais informações devem ser apagadas do estado da célula:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1}). \tag{2.17}$$

Na sequência, baseado no estado interno anterior e no dado de entrada, a estrutura *input gate* define quais informações devem ser adicionadas ao estado da célula:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1}). \tag{2.18}$$

Os valores candidatos a serem adicionados ao estado da célula são criados através da aplicação da função $tanh$:

$$\mathbf{c}'_t = tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1}). \tag{2.19}$$

Agora, o novo estado da célula pode ser calculado, com as estruturas *forget gate* e *input gate* definindo quais informações do estado anterior e do estado atual incluir:

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} \oplus \mathbf{i}_t * \mathbf{c}'_t. \tag{2.20}$$

A estrutura *output gate* permite a LSTM definir quais informações retornar do seu estado interno:

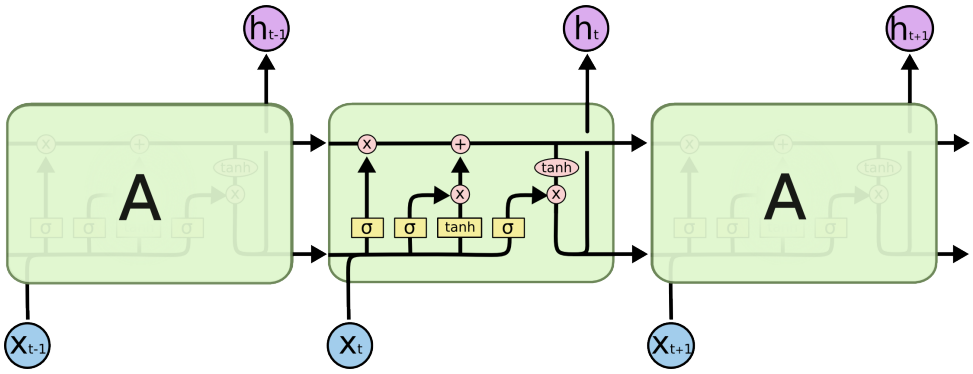
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1}). \tag{2.21}$$

Por fim, o estado interno é transformado pela função $tanh$ e regulado pela estrutura *output gate*:

$$\mathbf{h}_t = \mathbf{o}_t * tanh(\mathbf{c}_t). \tag{2.22}$$

Como todas essas equações são deriváveis pode-se encadear múltiplas células e realizar o treinamento pelo algoritmo de retro-propagação através do tempo, assim como na RNN (Figura 32) (VASILEV *et al.*, 2019). A célula LSTM protege a rede do desaparecimento dos gradientes uma vez que o estado da célula somente pode ser apagado pela estrutura *forget gate*, podendo permanecer inalterado por um longo período de tempo (VASILEV *et al.*, 2019). Já o problema de explosão dos gradientes é evitado através da aplicação da função *tanh* ao estado da célula, distribuindo melhor os gradientes (VASILEV *et al.*, 2019).

Figura 32 – Encadeamento de células em uma LSTM.



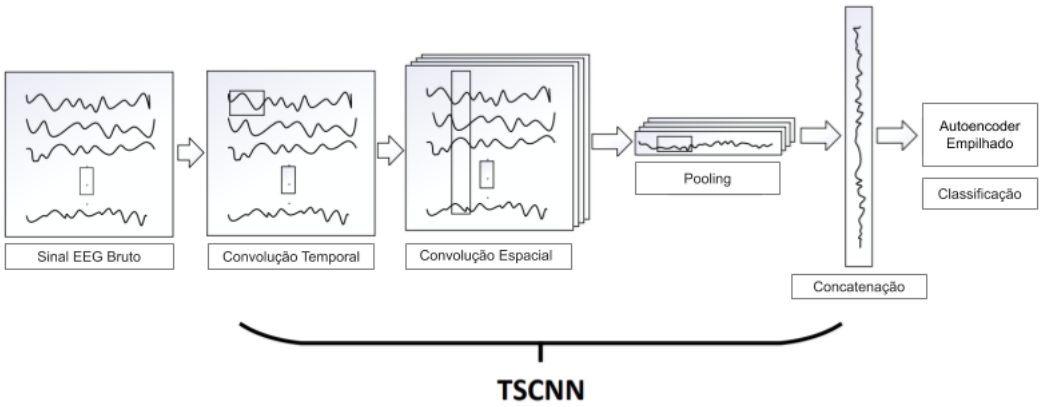
Fonte: Olah (2015).

Arquiteturas

Tang *et al.* (2019a) propuseram, para a classificação de 2 classes de imagética motora, uma arquitetura híbrida composta por uma Rede Neural Convolucional Temporal-Espacial (do inglês, *Temporal-Spatial Convolutional Neural Network* - TSCNN) e um *Autoencoder* Empilhado. A TSCNN (Figura 33), consiste de duas camadas convolucionais e uma camada de *pooling*. A primeira camada convolucional aplica as convoluções somente no domínio temporal, com o intuito de que a rede neural possa aprender filtros que permitam a separação dos sinais em importantes bandas de frequência, bem como a atenuação da atividade naquelas bandas menos importantes. A segunda camada convolucional consiste na redução de dimensionalidade através do aprendizado de um filtro espacial, aplicando às convoluções no domínio espacial. Na sequência, a camada de *pooling* reduz ainda mais a dimensionalidade dos dados. Por fim, a camada de saída transforma os dados em um vetor unidimensional para ser utilizado como dados

de entrada para o *Autoencoder* Empilhado. Este tem por objetivo realizar a classificação a partir da representação espacial-temporal dos sinais de EEG aprendida pela TSCNN. O desempenho da arquitetura foi avaliado de forma dependente do sujeito em duas bases de dados utilizando a técnica de validação cruzada *10-fold*. Foi obtida uma acurácia média de 79.9% em uma base de dados de 9 sujeitos. Em outra base de dados de um único sujeito, foi obtida uma acurácia média de 90.9%.

Figura 33 – Arquitetura de uma rede neural convolucional temporal-espacial.

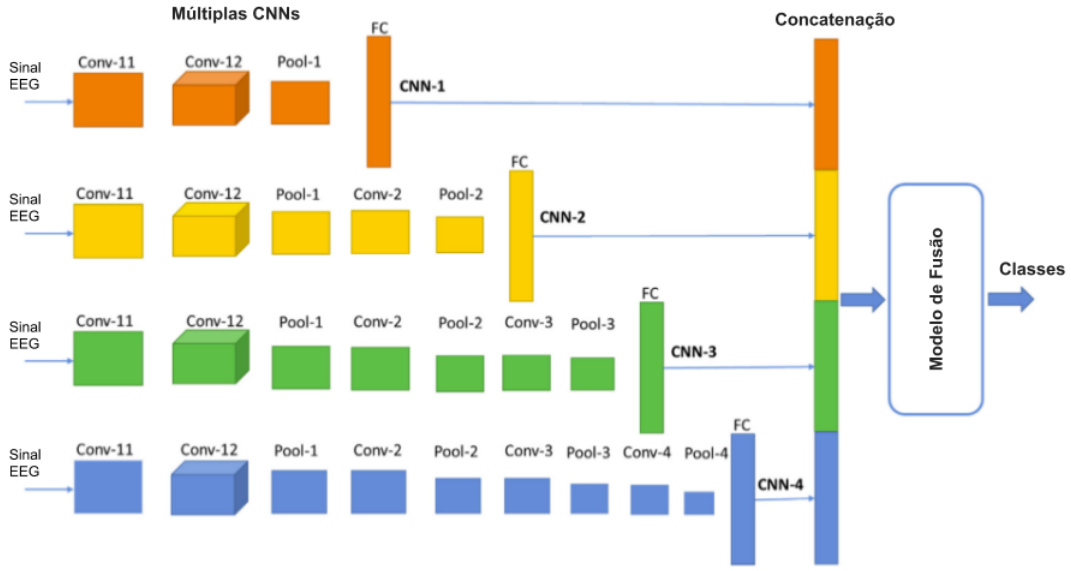


Fonte: Adaptado de Tang *et al.* (2019a).

Amin *et al.* (2019) propuseram, para a classificação de 4 classes de imagética motora, uma arquitetura para a fusão e classificação de características aprendidas a partir de múltiplas CNNs (Figura 34), baseados na hipótese de que diferentes arquiteturas de CNNs são boas para a engenharia de diferentes tipos de características. Inicialmente, um pré-processamento é realizado para a remoção de ruído dos sinais de EEG. Então, múltiplas CNNs realizam a engenharia de características temporais e espaciais. As características aprendidas são concatenadas e passadas como dados de entrada para uma Perceptron de Múltiplas Camadas (do inglês, *Multilayer Perceptron* - MLP) ou um *Autoencoder*, os quais têm por objetivo realizar a fusão das características e classificação. O *Autoencoder*, a partir do vetor de características gerado pelas múltiplas CNNs, extrai características robustas que são estáveis para diferentes sujeitos e ensaios, de forma a gerar um modelo que possa ser utilizado independentemente do sujeito. Isto é realizado forçando o *Autoencoder* a gerar um conjunto de características pertencentes a mesma classe, mas de diferentes sujeitos, agindo como um gargalo aprendendo características importantes pertencentes a cada classe e removendo automaticamente as informações menos relevantes (AMIN *et al.*, 2019). Por fim, tanto a MLP, quanto o *Autoencoder*, possuem uma camada de saída *Softmax*, a qual recebe a representação obtida por cada rede neural e retorna as probabilidades de predição para cada uma das 4 classes de imagética motora. A avaliação de desempenho foi realizada tanto para o cenário de classificação dependente do sujeito, quanto independente. O método que utilizou a MLP obteve os melhores resultados no cenário dependente do sujeito, e o método que utilizou o *Autoencoder* obteve os melhores resultados no cenário independente do sujeito. Em uma base de

dados de 20 sujeitos, foi obtida uma acurácia média de 95.4% para o cenário de classificação dependente do sujeito e uma acurácia média de 79.2% para o cenário de classificação independente do sujeito. Esta base de dados também foi utilizada para pré-treinar os algoritmos, de modo a realizar a avaliação de desempenho em uma base de dados menor, com dados provenientes de 9 sujeitos. Foi obtida uma acurácia média de 75.7% para o cenário de classificação dependente do sujeito e uma acurácia média de 55.3% para o cenário de classificação independente do sujeito.

Figura 34 – Arquitetura de um modelo para a fusão de características extraídas por múltiplas CNNs.



Fonte: Adaptado de Amin *et al.* (2019).

Zhang *et al.* (2018) propuseram, para a classificação de 5 classes de imagética motora independente do sujeito, uma arquitetura híbrida composta por uma LSTM e uma CNN, chamada de Rede Neural Recorrente Convolucional. Visando considerar as características espaciais dos sinais de EEG, estes são reorganizados em uma matriz de duas dimensões preservando suas posições espaciais no sistema de aquisição. Duas variações da arquitetura foram implementadas: em cascata e em paralelo. A arquitetura em cascata recebe como dados de entrada a representação em duas dimensões dos sinais de EEG, e realiza a engenharia das características espaciais e temporais em cascata. Inicialmente, a partir dos dados de entrada, uma CNN realiza a engenharia de características espaciais. Na sequência, a partir das características espaciais geradas pela CNN, uma LSTM realiza a engenharia de características temporais. Já a arquitetura em paralelo realiza a engenharia das características espaciais e temporais em dois fluxos separados. Para a engenharia das características espaciais utiliza-se uma CNN, que recebe como dados de entrada a representação em duas dimensões dos sinais de EEG. Para a engenharia das características temporais utiliza-se uma LSTM, que recebe como dados de entrada os sinais na sua forma original. Por fim, é realizada a fusão de ambos os vetores de características extraídas, gerando

uma representação espacial-temporal dos sinais de EEG. A saída de ambas arquiteturas possuem uma camada *Softmax* que recebe as características espaciais-temporais extraídas e retorna a probabilidade de predição para cada uma das 5 classes de imagética motora. A avaliação de desempenho foi realizada em uma base de dados pública com sinais de EEG provenientes de 108 sujeitos e em uma base de dados experimental coletada pelos autores, com sinais de EEG provenientes de 9 sujeitos. Na base de dados pública, as arquiteturas em cascata e em paralelo obtiveram acurácias médias de 98.31% e 98.28%, respectivamente. Na base de dados experimental, as arquiteturas em cascata e em paralelo obtiveram acurácias médias de 91.2% e 93.0%, respectivamente.

Pode-se notar um grande potencial nos algoritmos baseados em aprendizado profundo para o reconhecimento de imagética motora independente do sujeito. Ainda assim, há um amplo espaço para exploração de novas arquiteturas e técnicas para o desenvolvimento de um modelo robusto, que possa tornar viável sua aplicabilidade em interfaces cérebro-máquina comercializáveis.

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou diferentes métodos para o registro de sinais cerebrais. Dentre os métodos disponíveis, a eletroencefalografia (EEG) mostrou-se o mais apropriado para utilização em interfaces cérebro-máquina, devido a sua praticidade a alta resolução temporal.

Diferentes abordagens para o problema de reconhecimento de imagética motora foram apresentadas. Inúmeras melhorias realizadas no decorrer dos últimos anos em algoritmos baseados em engenharia de características possibilitaram a obtenção de uma taxa de acurácia média em torno de 90.0% para o cenário dependente do sujeito. Diversos trabalhos utilizaram algoritmos baseados em engenharia de características para o cenário independente do sujeito foram realizadas. Entretanto, as taxas de acurácia obtidas são muito baixas.

Devido a melhoria dos resultados do estado da arte através da utilização de algoritmos de aprendizado profundo em problemas de diversos domínios (por exemplo, visão computacional, processamento de linguagem natural e processamento de fala), diversas arquiteturas baseadas em aprendizado profundo têm sido exploradas. Estas arquitetura têm sido aplicadas com sucesso tanto nos cenários dependente e independente do sujeito. Mesmo com a ausência de grandes bases de sinais de EEG, resultados expressivos tem sido obtidos em ambos os cenários, incentivando pesquisadores a explorarem novas arquiteturas e técnicas para o desenvolvimento de modelos robustos.

Outro ponto crucial para possibilitar a aplicabilidade em tempo real, além da confiabilidade de predição do reconhecimento, é que estes modelos possam realizar as predições em um tempo de resposta adequado. Dessa forma, algoritmos baseados em aprendizado profundo sem um pré-processamento oneroso dos sinais de EEG têm sido explorados. Dentre as arquiteturas

apresentadas, destaca-se a proposta por Zhang *et al.* (2018), que no problema de classificação de múltiplas classes de imagética motora independente do sujeito obteve taxas de acurácia média em torno de 93% e 98.3% para duas bases de sinais de EEG. Resultados como este produzem forte indício da possibilidade do desenvolvimento de modelos robustos para o reconhecimento de imagética motora em tempo real, independente do sujeito, tornando possível o desenvolvimento de interfaces cérebro-máquina comercializáveis.

3 SISTEMA DE RECONHECIMENTO DE IMAGÉTICA MOTORA BASEADO EM APRENDIZADO PROFUNDO

Este capítulo apresenta uma proposta de desenvolvimento e avaliação de um sistema de reconhecimento de múltiplas classes de imagética motora. Utilizando uma abordagem em aprendizado profundo, o sistema realizará o reconhecimento de forma independente do sujeito (i.e., não demandará treinamento com dados do sujeito). A Seção 3.1 descreve a arquitetura do sistema. A Seção 3.2 apresenta as bases de dados que são utilizadas para avaliar o sistema desenvolvido. A Seção 3.3 apresenta os experimentos e resultados obtidos na base de dados da *PhysioNet*. As Seções 3.4 e 3.5 apresentam os resultados obtidos nas base de dados II-A e II-B da IV Competição de Interfaces Cérebro-Máquina, respectivamente.

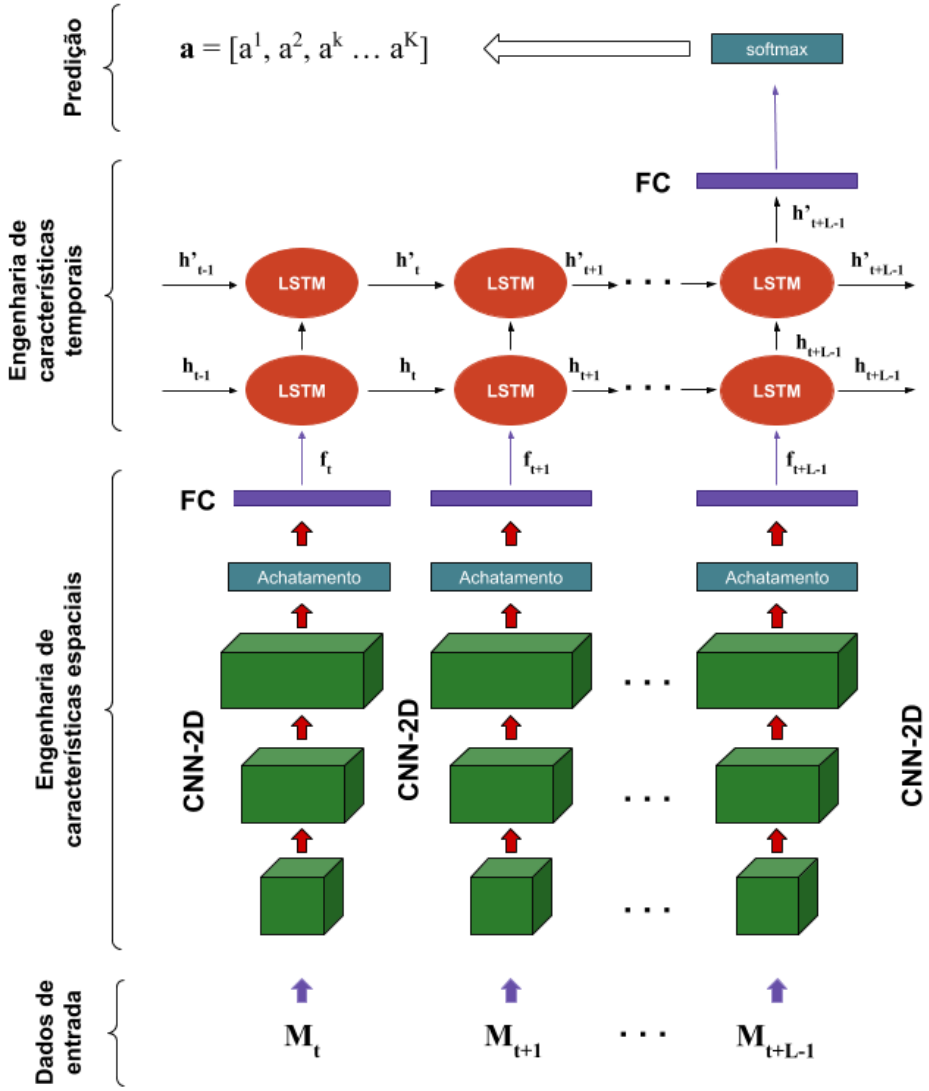
3.1 ARQUITETURA DO SISTEMA DE RECONHECIMENTO

O sistema de reconhecimento utilizará uma abordagem baseada em aprendizado profundo. A Rede Neural Recorrente Convolucional (do inglês, *Convolutional Recurrent Neural Network* - CRNN), proposta por Zhang *et al.* (2018), será utilizada devido aos seus excelentes resultados neste tipo de aplicação. A CRNN é uma arquitetura híbrida composta por uma CNN para a engenharia de características espaciais, uma LSTM para a engenharia de características temporais e uma camada *Softmax* para a obtenção de um conjunto de probabilidades. Para que a representação espacial-temporal criada pelas redes neurais CNN e LSTM seja robusta, no topo de cada uma é adicionada uma camada totalmente conectada (do inglês, *Fully Connected* - FC) composta de 1024 neurônios. De maneira geral, para um problema de K classes de imagética motora, a CRNN irá produzir para cada sinal de EEG um vetor de probabilidades $\mathbf{a} = [a^1, a^2, ..., a^K]$, a partir do qual é possível prever qual é a classe de imagética motora mais provável de pertencimento do sinal de EEG. Entretanto, para alimentar a CRNN, os sinais de EEG deverão passar por um processo de preparação de dados que consiste das etapas de reorganização, normalização e segmentação.

Para avaliar o sistema proposto, serão utilizadas mais de uma base de dados. Como cada base de dados possui diferentes número de classes e condições, diferentes cenários serão utilizados para a avaliação de desempenho. A CRNN será implementada utilizando a linguagem de programação Python[1]. Bibliotecas de aprendizado profundo como Keras[2] ou TensorFlow[3] serão utilizadas para os algoritmos básicos.

[1] Site oficial da linguagem de programação Python <https://www.python.org/>.
[2] Site oficial da biblioteca de aprendizado profundo Keras <https://keras.io/>.
[3] Site oficial da biblioteca de aprendizado profundo TensorFlow <https://www.tensorflow.org/>.

Figura 35 – Arquitetura em cascata de uma rede neural recorrente convolucional.



Fonte: Adaptado de Zhang *et al.* (2018).

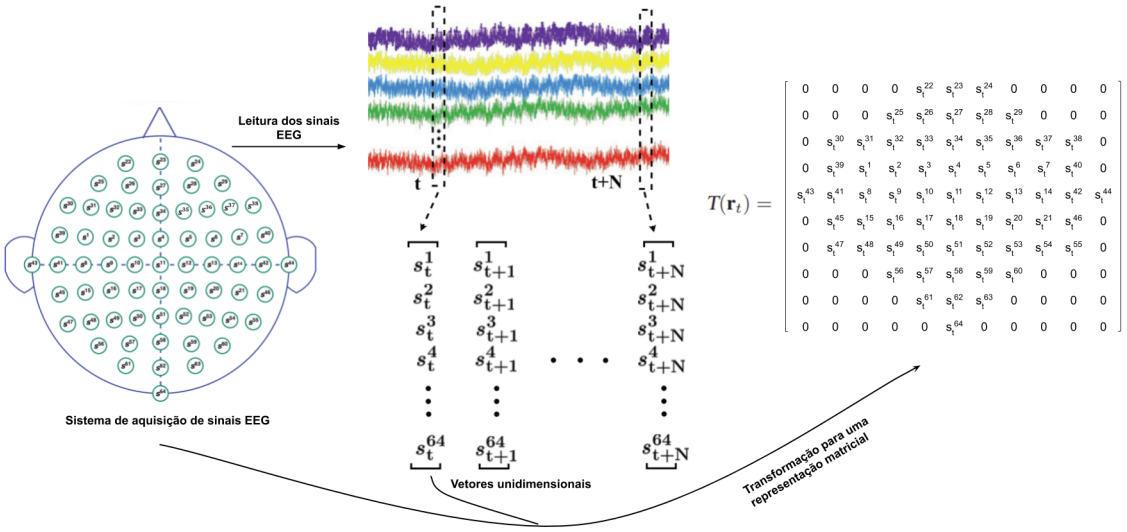
3.1.1 PREPARAÇÃO DOS DADOS DE ENTRADA

Os sinais de EEG devem ser reorganizados para um formato a partir do qual a CRNN possa considerar as informações espaciais inerentes do sinal de EEG. Geralmente, o sinal de EEG em um determinado instante de tempo t é representado por um vetor unidimensional $\mathbf{r}_t = [s_t^1, s_t^2, s_t^i \ldots s_t^C]$, onde s_t^i, $1 \leq i \leq C$, corresponde ao i-ésimo potencial adquirido no canal do eletrodo i no instante de tempo t e C corresponde à quantidade de eletrodos. Logo, o sinal de EEG em um determinado período de observação $[t, t+N]$ é representado por $N+1$ vetores de dimensão C. A reorganização é realizada pela aplicação da seguinte transformação em cada vetor $\mathbf{r}_t$:

$$\mathbf{M}_t = T(\mathbf{r}_t), \mathbf{M}_t \in \mathbb{R}^{b \times c} \tag{3.1}$$

onde $\mathbf{M}_t$ é uma matriz do sinal de EEG no instante de tempo t que espelha o sistema de aquisição. O número de linhas e colunas de $\mathbf{M}_t$ correspondem a quantidade máxima de eletrodos nas posições vertical e horizontal no sistema de aquisição, respectivamente. O valor de cada eletrodo em $\mathbf{r}_t$ é mapeado para uma posição específica em $\mathbf{M}_t$, de forma a espelhar o sistema de aquisição. Para posições sem eletrodo correspondente é utilizado o valor zero. Um exemplo deste mapeamento é ilustrado na Figura 36 para um sistema de aquisição composto por 64 eletrodos.

Figura 36 – Exemplo da reorganização dos sinais de EEG para um sistema de aquisição composto por 64 eletrodos.



Fonte: Adaptado de Zhang *et al.* (2018).

Para um sistema de reconhecimento independente do sujeito, é importante reduzir ao máximo a variabilidade entre sinais provenientes de diferentes sujeitos e de diferentes sessões de aquisição para um mesmo sujeito. Visto que o sinal de EEG tem alta variabilidade nesses cenários, o método *Z-score* (Equação 2.7) será aplicado entre os elementos não zerados das matrizes $\mathbf{M}_t$ para diminuir esta variabilidade.

Por fim, os sinais de EEG devem ser segmentados para que a CRNN possa prever para cada segmento a classe de imagética motora correspondente. A quantidade de segmentos irá variar de acordo com cada base de dados. Para bases de dados com ensaios de curta duração, o segmento poderá compreender todo o período de observação, gerando assim um único segmento por ensaio. Para bases de dados com ensaios de maior duração, um ensaio poderá ser dividido em múltiplos segmentos, aumentando assim a quantidade de dados de treinamento. Considerando como período de observação de um ensaio $[t, t+N]$, L como sendo o comprimento de cada segmento deste período e d como sendo o comprimento do deslocamento aplicado sobre t entre cada segmento do período, a segmentação pode ser definida como:

$$\mathbf{S}_j = [\mathbf{M}_{t+d}, \mathbf{M}_{t+d+1} \dots \mathbf{M}_{t+d+L-1}], j = 1 \dots q, \tag{3.2}$$

onde d varia de 0 a N com incremento de d amostras por segmento e q é a quantidade de segmentos gerados por ensaio.

3.1.2 ENGENHARIA DE CARACTERÍSTICAS

A engenharia de características é realizada em duas etapas. Inicialmente é realizada a engenharia de características espaciais por meio de uma CNN. Na sequência, a representação espacial gerada pela CNN é utilizada como dados de entrada para uma LSTM, que realiza a engenharia de características temporais.

A arquitetura da CNN proposta por Zhang *et al.* (2018) é composta por três camadas convolucionais. Como a responsabilidade da CNN é a engenharia de características espaciais, as convoluções são aplicadas somente na dimensão espacial dos dados. Os *kernels* das camadas convolucionais possuem dimensão 3×3. A técnica *zero-padding* é aplicada em todas as convoluções para que os mapas de características criados pela rede contenham a mesma dimensão dos dados de entrada. A primeira camada convolucional gera 32 mapas de características, dobrando este valor em cada camada, resultando em um total de 128 mapas de características ao final da terceira camada convolucional. A dimensão da representação gerada como saída da terceira camada convolucional é $L \times b \times c \times 128$. Para conectar esta representação a uma camada totalmente conectada de forma a obter uma representação mais robusta dos dados, a dimensão espacial (i.e., $b \times c \times 128$) deve passar por um processo de achatamento produzindo como saída uma representação de dimensão $L \times w$, onde $w = b \times c \times 128$. Esta representação achatada permite a conexão de cada um dos w neurônios de saída a cada neurônio de entrada de uma camada totalmente conectada. A quantidade de neurônios da camada totalmente conectada é 1024. A representação final pode ser denotada como $\mathbf{F} \in \mathbb{R}^{L \times 1024}$, mantendo a dimensão temporal (L) inalterada para posterior processamento pela LSTM. Logo, a representação espacial em um determinado instante de tempo t pode ser denotada como $\mathbf{f}_t$, onde t varia de 0 a $L - 1$.

A arquitetura da LSTM é composta por duas camadas empilhadas, cada qual composta por L células de memória correspondentes a cada instante de tempo de um segmento. Uma célula de memória em uma LSTM armazena seu estado interno para que a informação em cada instante de tempo possa influenciar a saída final. Os estados internos no instante de tempo t da primeira e segunda camada podem ser denotados como $\mathbf{h}_t$ e $\mathbf{h}'_t$, respectivamente. A primeira camada recebe como dados de entrada a representação espacial gerada pela CNN e a saída da célula de memória imediatamente anterior. Logo, para o instante de tempo t, a célula de memória correspondente recebe como dados de entrada a representação espacial $\mathbf{f}_t$ e o estado interno da célula de memória imediatamente anterior $\mathbf{h}_{t-1}$. A segunda camada recebe como dados de entrada o estado interno produzido pela camada anterior e a saída da célula de memória imediatamente anterior. Logo, para o instante de tempo t, a célula de memória correspondente recebe como dados de entrada os estados internos $\mathbf{h}_t$ e $\mathbf{h}'_{t-1}$.

Para a engenharia de características de um determinado segmento $\mathbf{S}_j$, as L matrizes $\mathbf{M}_t$

do segmento serão fornecidas como dados de entrada para a CNN individualmente, produzindo para cada matriz sua respectiva representação espacial:

$$\mathbf{f}_t = CNN(\mathbf{M}_t), \mathbf{f}_t \in \mathbb{R}^{1024}. \tag{3.3}$$

Logo, a representação espacial $\mathbf{F}_j$ do segmento $\mathbf{S}_j$ pode ser denotada como:

$$\mathbf{F}_j = [\mathbf{f}_t \dots \mathbf{f}_{t+L-1}] \in \mathbb{R}^{L \times 1024}. \tag{3.4}$$

A representação espacial $\mathbf{F}_j$ é utilizada como dados de entrada para a LSTM que realiza a engenharia de características temporais. A representação temporal gerada pela LSTM pode ser denotada da seguinte forma:

$$\mathbf{h}^{'}_{t+L-1} = LSTM(\mathbf{F}_j), \mathbf{h}^{'}_{t+L-1} \in \mathbb{R}^{u}, \tag{3.5}$$

onde u é a dimensão do estado interno definido para cada célula de memória da LSTM.

3.1.3 CLASSIFICAÇÃO

Para a classificação das características extraídas é utilizada uma camada *Softmax*:

$$\mathbf{p}_j = Softmax(\mathbf{h}^{'}_{t+L-1}), \mathbf{p}_j \in \mathbb{R}^{K}, \tag{3.6}$$

onde $\mathbf{p}_j$ é um conjunto de probabilidades de pertencimento a cada uma das classes de $\mathbf{a}$ para o segmento $\mathbf{S}_j$. Logo, a partir de $\mathbf{p}_j$, pode-se prever a qual das classes de imagética motora é mais provável que o segmento do sinal de EEG pertença.

3.2 BASES DE DADOS

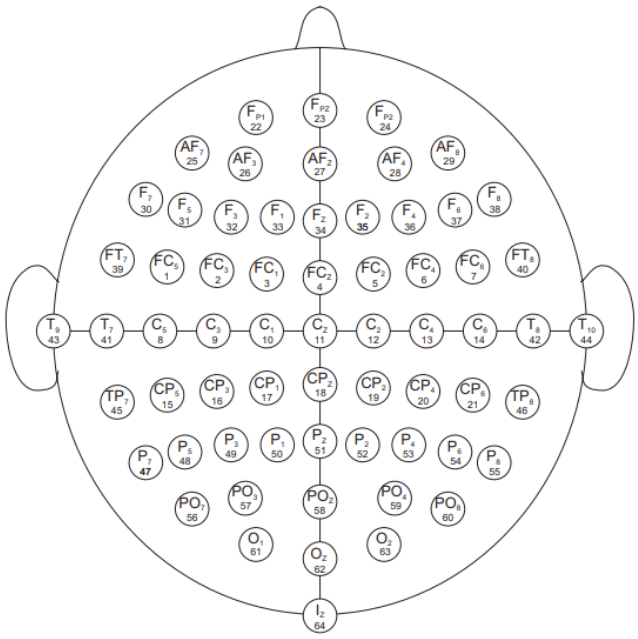
Serão utilizadas três bases de dados para que se possa avaliar o desempenho do algoritmo nos cenários de classificação binária e classificação de múltiplas classes. Além disso, graças a uma base de dados com sinais de EEG adquiridos com e sem *feedback*, será possível avaliar a diferença de desempenho entre as estratégias de avaliação *offline* e *online* simulado.

3.2.1 BASE DE DADOS DA PHYSIONET

A base de dados de imagética motora e movimento motor da *PhysioNet*[4] é, para o melhor do nosso conhecimento, a maior base de dados pública de imagética motora disponível atualmente. Compreende cerca de 1500 ensaios provenientes de 109 sujeitos. Para a aquisição dos sinais de EEG foi utilizado um sistema composto por 64 eletrodos com uma taxa de amostragem de 160 Hz. A disposição dos eletrodos sobre o escalpo foi realizada com base no sistema internacional 10-10 (Figura 37).

[4] Disponível em: <https://physionet.org/content/eegmmidb/> Acesso em jun. 2020.

Figura 37 – Sistema para a aquisição dos sinais de EEG da base de imagética motora e movimento motor da *PhysioNet*.



Fonte: Página do site da *PhysioNet*.[5]

Cada sujeito realizou 14 execuções experimentais: 2 execuções de 1 minuto para descanso (uma de olhos abertos, uma de olhos fechados) e 3 execuções de 2 minutos para cada uma das quatro atividades a seguir:

- Atividade 1: um estímulo é exibido no lado esquerdo ou direito da tela. O sujeito é instruído para abrir e fechar o punho correspondente até que o estímulo desapareça e então relaxar;
- Atividade 2: um estímulo é exibido no lado esquerdo ou direito da tela. O sujeito é instruído para imaginar a abertura e fechamento do punho correspondente até que o estímulo desapareça e então relaxar;
- Atividade 3: um estímulo é exibido na parte superior ou inferior da tela. O sujeito é instruído para abrir e fechar os dois punhos (quando o estímulo estiver na parte superior), ou os dois pés (quando o estímulo estiver na parte inferior), até que o estímulo desapareça e então relaxar;
- Atividade 4: um estímulo é exibido na parte superior ou inferior da tela. O sujeito é instruído para imaginar a abertura e fechamento dos dois punhos (quando o estímulo estiver na parte superior), ou os dois pés (quando o estímulo estiver na parte inferior), até que o estímulo desapareça e então relaxar.

[5] Disponível em: <https://physionet.org/content/eegmmidb/1.0.0/64_channel_sharbrough.pdf> Acesso em jun. 2020.

As 14 execuções foram realizadas na seguinte ordem: olhos abertos, olhos fechados e as atividades 1, 2, 3, e 4, repetidas 3 vezes em ordem.

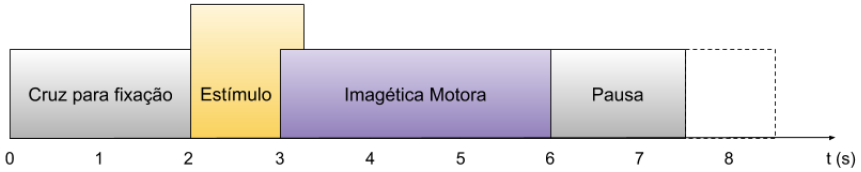
Para este trabalho serão utilizados apenas os ensaios correspondentes às execuções de descanso (olhos fechados) e às execuções das 4 classes de imagética motora (punho esquerdo, punho direito, punhos e pés). Além disso, os dados de 5 sujeitos foram descartados da base por apresentarem problemas (reduzindo a um total de 104 sujeitos). Zhang *et al.* (2018) removeram o sujeito 89, pois os rótulos do mesmo estão incompletos. Wang *et al.* (2020) removeram os sujeitos 88, 92, 100 e 104 devido à variabilidade apresentada no número de ensaios.

3.2.2 BASE DE DADOS II-A DA IV COMPETIÇÃO DE INTERFACES CÉREBRO-MÁQUINA

A base de dados II-A da IV competição de interfaces cérebro-máquina[6] possui sinais de EEG provenientes de 9 sujeitos. O protocolo experimental compreende 4 classes de imagética motora (mão esquerda, mão direita, os dois pés e a língua). Para cada sujeito, foram realizadas duas sessões de aquisição, em diferentes dias. Em cada sessão foram realizadas seis execuções separadas por pequenas pausas. Cada execução produziu 48 ensaios (12 para cada uma das 4 classes de imagética motora), produzindo um total de 288 ensaios por sessão.

O protocolo experimental utilizado é ilustrado na Figura 38. Os sujeitos sentaram-se em uma poltrona confortável em frente à tela de um computador. No início do ensaio (t = 0 s), uma cruz para fixação é exibida no centro da tela juntamente com a execução de um curto tom de aviso sonoro. Alguns segundos depois (t = 2 s), um estímulo na forma de uma seta apontando para a esquerda ou para a direita ou para cima ou para baixo (correspondendo a cada uma das 4 classes de imagética motora) é exibido sobre a cruz, por um período de 1,25 s. Os sujeitos foram instruídos para, de acordo com o estímulo exibido, efetuar a imagética motora correspondente, até que a cruz para fixação desaparecesse da tela em t = 6 s. Antes do fim do ensaio, a tela fica preta por um curto período de tempo para uma pausa entre diferentes ensaios. Nenhum tipo de *feedback* foi fornecido aos sujeitos durante os ensaios.

Figura 38 – Protocolo experimental para a aquisição de sinais de EEG da base de dados II-A da IV competição de interfaces cérebro-máquina.

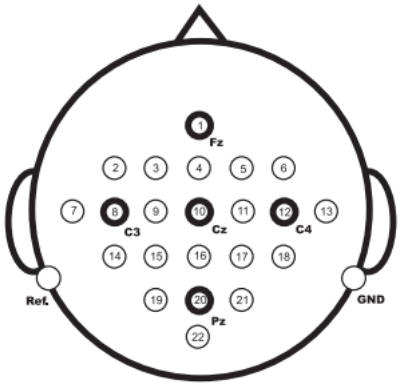


Fonte: Adaptado da página do site da BBCI.[6]

[6] Disponível em: <http://www.bbci.de/competition/iv/desc_2a.pdf> Acesso em jun. 2020.

Para o registro dos sinais cerebrais foram utilizados 22 eletrodos (conforme Figura 39) com uma taxa de amostragem de 250 Hz. Dois filtros digitais foram aplicados aos sinais para redução de ruído: filtro passa-banda de 0.5-100 Hz e um filtro *notch* de 50 Hz. Por inspeção visual realizada por especialistas, amostras contendo artefatos foram marcadas como rejeitadas.

Figura 39 – Sistema para a aquisição dos sinais de EEG da base de dados II-A da IV competição de interfaces cérebro-máquina.



Fonte: Adaptado da página do site da BBCI.[6]

3.2.3 BASE DE DADOS II-B DA IV COMPETIÇÃO DE INTERFACES CÉREBRO-MÁQUINA

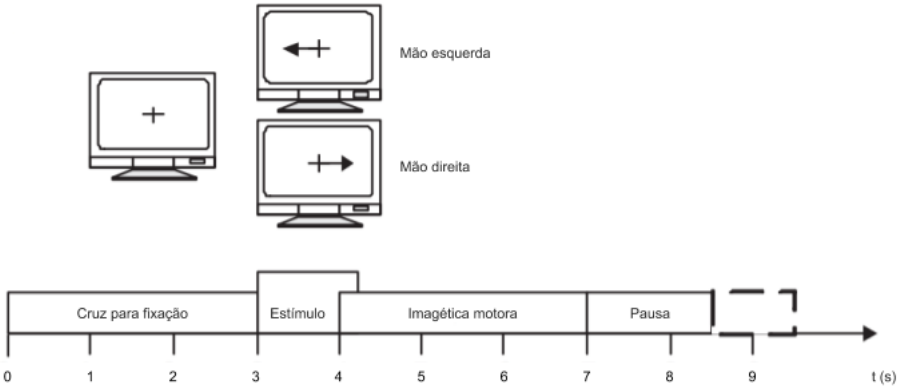
A base de dados II-B da IV competição de interfaces cérebro-máquina[7] possui sinais de EEG provenientes de 9 sujeitos. O protocolo experimental compreende 2 classes de imagética motora (mão esquerda e mão direita). Para cada sujeito, foram realizadas cinco sessões de aquisição, sendo que as três últimas sessões forneceram *feedback* aos sujeitos. Um total de 120 ensaios para cada classe de imagética motora sem *feedback* e 240 ensaios para cada classe de imagética motora com *feedback* foram coletados para cada sujeito.

O protocolo experimental utilizado nas sessões sem *feedback* é ilustrado na Figura 40. No início do ensaio (t = 0 s), uma cruz para fixação é exibida no centro da tela juntamente com a execução de um curto tom de aviso sonoro. Alguns segundos depois (t = 3 s), um estímulo na forma de uma seta apontando para a esquerda ou para a direita (correspondendo a uma das 2 classes de imagética motora) é exibido sobre a cruz, por um período de 1,25 s. Os sujeitos foram instruídos para, de acordo com o estímulo exibido, efetuar a imagética motora correspondente, até que a cruz para fixação desaparecesse da tela em t = 7 s. Antes do fim do ensaio, a tela fica branca por um curto período de tempo para uma pausa entre diferentes ensaios.

O protocolo experimental utilizado nas sessões com *feedback* é ilustrado na Figura 41. No início do ensaio (t = 0 s), um sorriso cinza é exibido no centro da tela. Alguns segundos depois

[7] Disponível em: <http://www.bbci.de/competition/iv/desc_2b.pdf> Acesso em jun. 2020.

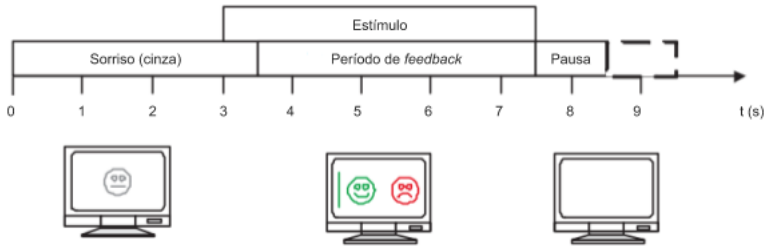
Figura 40 – Protocolo experimental para a aquisição de sinais de EEG sem *feedback* da base de dados II-B da IV competição de interfaces cérebro-máquina.



Fonte: Adaptado da página do site da BBCI.[7]

(t = 2 s), um curto tom de aviso sonoro é executado e em t = 3 s um estímulo correspondente a uma das 2 classes de imagética motora é exibido, por um período de 4,5 s. Os sujeitos foram instruídos para, de acordo com o estímulo exibido. Durante o período de *feedback*, conforme o desempenho do sujeito em executar a respectiva imagética motora, a cor e a curvatura do sorriso são atualizadas para indicar ao sujeito o seu desempenho. Antes do fim do ensaio, a tela fica branca por um curto período de tempo para uma pausa entre diferentes ensaios. Os sujeitos foram instruídos para manter o sorriso na direção correta pelo maior tempo possível.

Figura 41 – Protocolo experimental para a aquisição de sinais de EEG com *feedback* da base de dados II-B da IV competição de interfaces cérebro-máquina.



Fonte: Adaptado da página do site da BBCI.[7]

Para o registros dos sinais de EEG foram utilizados apenas três eletrodos (C3, Cz e C4) com uma taxa de amostragem de 250 Hz. Dois filtros digitais foram aplicados aos sinais para redução de ruído. Um filtro passa-banda de 0.5-100 Hz e um filtro *notch* de 50 Hz. Por inspeção visual realizada por especialistas, amostras contendo artefatos foram marcadas como rejeitadas.

3.3 EXPERIMENTOS E RESULTADOS NA BASE DA PHYSIONET

Como a *PhysioNet* possui o maior número de sujeitos entre todas as bases, a maioria dos experimentos e melhorias no sistema foram realizados com esta base. A quantidade de sujeitos é importante para a obtenção de um sistema que seja independente do sujeito, conforme definido pelo objetivo deste trabalho.

O teste T de *Student* pareado para validação cruzada, de acordo com as definições formalizadas por Dietterich (1998), é utilizado para verificar se a diferença de desempenho dos sistemas é estatisticamente significante. A menos que seja especificado, o nível de significância é definido em 5% (i.e., $\alpha = 0.05$).

O código-fonte pode ser encontrado nos anexos do presente trabalho. Mais detalhes para utilização e execução dos modelos podem ser encontrados no repositório do projeto no Github[8].

3.3.1 SISTEMA DE REFERÊNCIA PARA O PROBLEMA DE 5 CLASSES

Para validar a implementação da arquitetura CRNN proposta por Zhang *et al.* (2018) utilizando a biblioteca de aprendizado profundo *Keras*, o mesmo experimento (5 classes: olhos fechados, punho esquerdo, punho direito, punhos e pés) realizado por Zhang foi realizado. As amostras para treinamento e teste foram geradas, conforme as etapas descritas na Seção 3.1.1, utilizando uma janela e deslocamento de 10 e 5 amostras, respectivamente (i.e., $L = 10$ e $d = 5$).

A seleção dos dados para treinamento e teste seguiram a mesma metodologia utilizada no artigo. Como a seleção dos dados não está clara no artigo, buscou-se maiores informações no repositório Github[9] do autor. Conforme o código-fonte, foi possível verificar que foram utilizados os dados das execuções 2, 4 e 6, as quais contemplam as 5 classes. As demais execuções foram descartadas do experimento. As amostras foram embaralhadas e selecionadas aleatoriamente dividindo-as nos subconjuntos de treinamento (75% das amostras) e teste (25% das amostras). A Tabela 2 apresenta as quantidades de treinamento e teste para cada classe. Com este processo de separação de dados, é possível que os subconjuntos de treinamento e teste possuam amostras do mesmo sujeito (i.e., não é um sistema independente do sujeito).

O sistema de reconhecimento implementado obteve uma acurácia de 97.70% nos dados de teste. Como o desempenho apresentado por Zhang *et al.* (2018) é de 98.31%, pode-se afirmar que a implementação realizada é muito similar (uma diferença relativa de 0.6%) ao proposto por Zhang. Esta diferença no resultado pode ser atribuída às características do processo iterativo de otimização das redes neurais artificiais e pelos dados dos sujeitos que foram removidos.

Como o objetivo do presente trabalho é desenvolver um sistema independente do sujeito,

[8] https://github.com/mauricio-ms/motor-imagery-deep-learning
[9] https://github.com/dalinzhang/Cascade-Parallel

Tabela 2 – Quantidade de amostras geradas (com $L = 10$ e $d = 5$) para os subconjuntos de treinamento (75%) e teste (25%) para a base de dados da *PhysioNet*.

Classe	Treinamento	Teste
Olhos Fechados	149956	50213
Punho Esquerdo	76874	25805
Punho Direito	75157	24934
Punhos	76559	25346
Pés	75807	25058

Fonte: O Autor (2020).

a técnica de validação cruzada *K-fold* sobre os sujeitos é utilizada para avaliação do sistema (WANG *et al.*, 2020; DOSE *et al.*, 2018). O conjunto de dados foi dividido em 10 partições (*10-fold*) com base nos sujeitos. Isto é, cada partição de teste recebeu as amostras de 1/10 dos sujeitos. Como são 104 sujeitos, 4 partições de teste possuem 11 sujeitos e as demais possuem 10 sujeitos cada. Portanto, são gerados 10 modelos e a acurácia é estimada em cada partição.

As etapas de treinamento e avaliação de desempenho do sistema são realizadas de forma independente. O processo de validação cruzada *10-fold* é realizado de forma determinística, garantindo que os mesmos sujeitos sejam atribuídos aos subconjuntos de treinamento e teste em cada partição.

Para o treinamento da rede neural, os dados de treinamento são embaralhados, pois devido à natureza estocástica do algoritmo de aprendizagem, fornecer amostras inesperadas acelera o processo de aprendizagem (LECUN *et al.*, 2012). O número de épocas de treinamento foi limitado em 50 por partição, e ao final do treinamento de cada partição os pesos da rede neural são salvos.

Para a avaliação de desempenho são criados K modelos já inicializados com os pesos salvos na etapa de treinamento. A acurácia de cada modelo é estimada nos dados de teste correspondentes. Por fim, a estimativa da acurácia final do sistema é calculada pela média das K acurácias obtidas.

O sistema independente do sujeito obteve uma acurácia média de $33.88\%(\pm 2.72\%)$ (uma redução relativa de 65.32%). Como esperado, a redução no desempenho mostra que a informação *a priori* dos sujeitos no treinamento produz um melhor resultado nos dados de teste.

3.3.2 SISTEMA DE REFERÊNCIA PARA O PROBLEMA DE 2 CLASSES

Com a implementação do sistema validada com o resultado de Zhang *et al.* (2018) para 5 classes e o resultado do sistema independente do sujeito aquém do esperado, verificou-se a necessidade da realização de alterações no sistema a fim de melhorar o seu desempenho.

Como o processo de treinamento das redes neurais profundas demanda considerável tempo de processamento, a classificação binária foi utilizada para viabilizar um número maior de experimentações. O problema de classificação binária na base *PhysioNet* consiste em identificar se, o sinal de EEG da amostra é um sinal espontâneo de imagética motora da movimentação do punho esquerdo ou do punho direito. Uma vez que as redes neurais baseadas em aprendizado profundo necessitam de uma grande quantidade de dados para o aprendizado, as execuções 8, 10, 12 e 14 foram incluídas nos experimentos.

O sistema de classificação binária independente do sujeito foi avaliado utilizando a técnica de validação cruzada *10-fold*. A Tabela 3 apresenta as quantidades de treinamento e teste para cada classe por partição. O sistema obteve uma acurácia média de $54.03\%(\pm 1.67\%)$. Por se tratar de um problema mais simples que o de 5 classes, era esperado que o desempenho seria melhor. Entretanto, a acurácia é próxima de um palpite aleatório para um problema de 2 classes (50%), o que demanda uma reavaliação dos parâmetros do sistema.

Tabela 3 – Quantidade de amostras geradas (com $L = 10$ e $d = 5$) por partição (k) no processo de validação cruzada *10-fold* para a base de dados da *PhysioNet*.

	Punho Esquerdo		Punho Direito	
k	Treinamento	Teste	Treinamento	Teste
1	273834	32890	270096	31460
2	274484	32240	269446	32110
3	274484	32240	269446	32110
4	274634	32090	269332	32224
5	276706	30018	273086	28470
6	276832	29892	272960	28596
7	277238	29486	272578	28978
8	277612	29112	272180	29376
9	277474	29250	272306	29250
10	277218	29506	272574	28982
Total	2760516	306724	2714004	301556

Fonte: O Autor (2020).

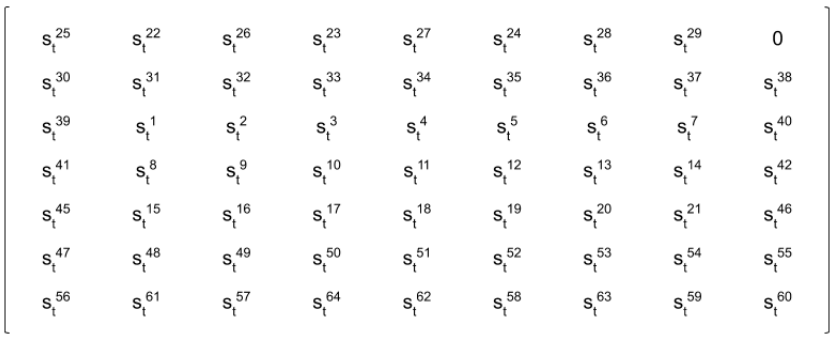
3.3.3 FORMATO DOS DADOS DE ENTRADA

Com o objetivo de representar a vizinhança dos sinais de EEG, a representação proposta por Zhang *et al.* (2018) introduz diversos valores zerados nos dados de entrada. A matriz de dados (Figura 36) para 64 eletrodos possui 46 valores zerados. Isto é, de um total de 110 valores, 41.82% deles são valores iguais a 0.

Visando a redução do número de parâmetros do modelo, propõe-se uma nova representação dos dados que mantenha a característica da vizinhança, mas com uma quantidade reduzida de zeros (Figura 42). Esta representação necessita somente de um valor zero ao custo da remoção

dos eletrodos 43 e 44. Avaliando o sistema com esta nova representação, obteve-se uma acurácia de $53.54\%(\pm 1.57\%)$. Apesar de que não houve um aumento no desempenho, a quantidade de parâmetros do modelo reduziu em 41.37%. Entretanto, fica a questão se a informação da vizinhança traz alguma informação adicional para o sistema.

Figura 42 – Representação otimizada dos sinais de EEG considerando a vizinhança do sistema de aquisição da base de dados da *PhysioNet*.



Fonte: O Autor (2020).

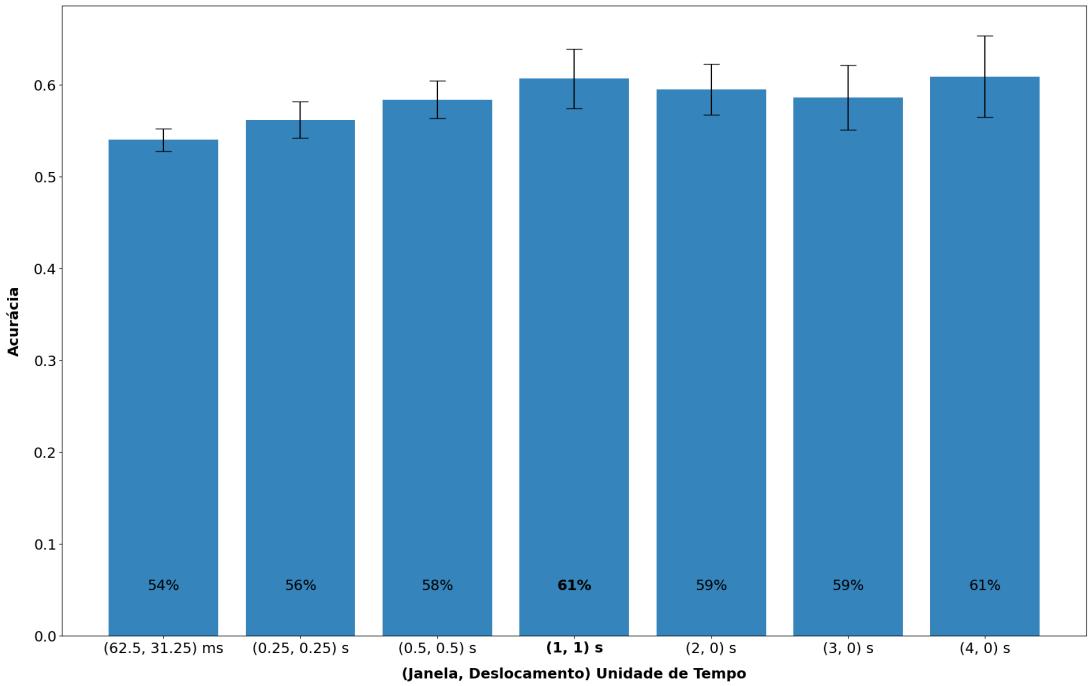
Para avaliar se a vizinhança auxilia no reconhecimento, um experimento foi realizado alterando a entrada de dados para uma representação unidimensional (os valores dos 64 eletrodos representados em um vetor em vez da forma matricial). A única alteração realizada no modelo foi nas camadas convolucionais para que aplicassem as convoluções em apenas 1 dimensão (i.e., a dimensão espacial). O resultado deste experimento foi uma acurácia média de $54\%(\pm 1.25\%)$. Assim, o resultado mostra que para este modelo, a informação da vizinhança (representação matricial dos eletrodos) não agrega nenhuma informação adicional ao modelo.

Baseado na hipótese de que um maior contexto temporal pode auxiliar no desempenho do sistema, os parâmetros de segmentação correspondentes aos comprimentos da janela (L) e do deslocamento (d) serão alterados, de forma a fornecer como dados de entrada para o modelo janelas com um maior contexto da informação temporal produzida durante os ensaios. Diversos experimentos foram realizados aumentando o comprimento da janela com deslocamentos de 100% (nos cenários aplicáveis). Como pode ser observado nos resultados (Figura 43), janelas de tempo maiores melhoram o desempenho do sistema. Entretanto, uma janela de tempo muito longa é indesejável pois prejudica diretamente o tempo de resposta do sistema, reduzindo o escopo de sua aplicabilidade em soluções de interfaces cérebro-máquina. Como a diferença entre os sistemas que utilizaram janelas de 1 e 4 segundos não é estatisticamente significante, a escolha foi pela janela de 1 segundo (L = 1s) e deslocamento de 1 segundo (d = 1s).

3.3.4 ARQUITETURA DA REDE NEURAL

Uma vez definido que a vizinhança não está trazendo nenhuma informação complementar, o foco das experimentações voltou-se para a rede neural profunda. É importante frisar que a

Figura 43 – Resultados da CRNN com diferentes comprimentos de janela.



Fonte: O Autor (2020).

dimensionalidade dos dados de entrada da rede foi reduzida drasticamente permitindo a utilização de diferentes arquiteturas.

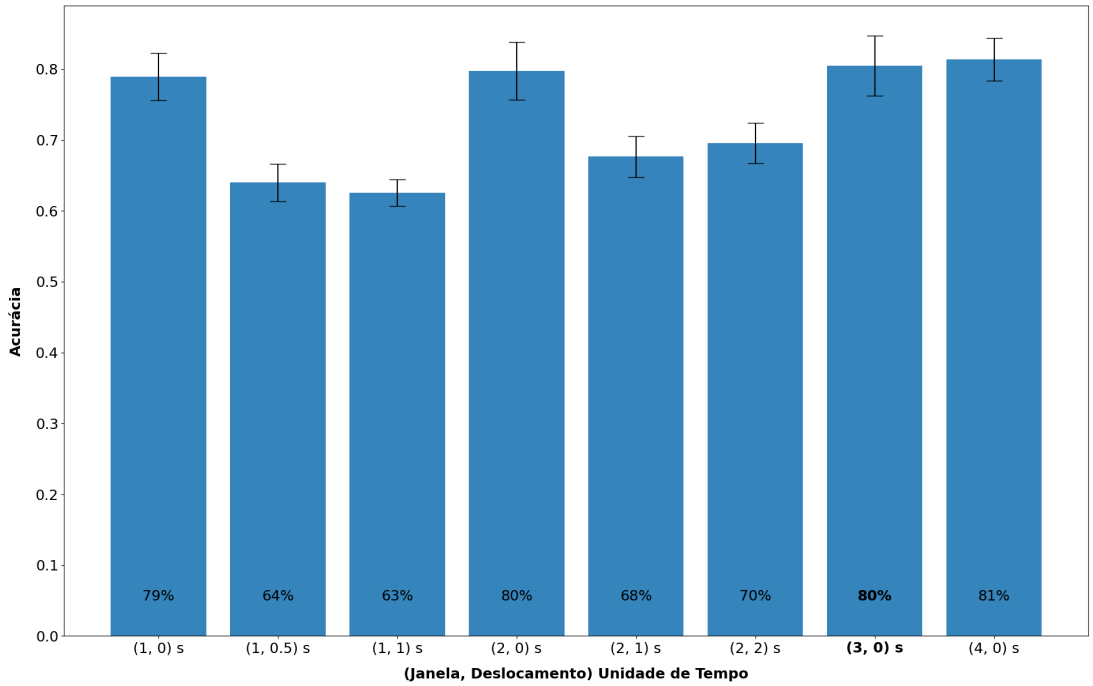
Existem indícios de que realizar a engenharia de características temporais com Redes Neurais Recorrentes (i.e., a LSTM na arquitetura CRNN) não auxilia no desempenho de sistemas para o reconhecimento de imagética motora. Os trabalhos de Dose *et al.* (2018) e Wang *et al.* (2020) utilizam somente camadas convolucionais e de *pooling*, seguidas por camadas totalmente conectadas para a classificação de imagética motora. Em ambos os trabalhos, é reportada uma acurácia de cerca de 80% no cenário de classificação das classes punho esquerdo e punho direito independente do sujeito. Para avaliar o desempenho do sistema sem a engenharia de características temporais pela LSTM, pode-se removê-la juntamente com a camada totalmente conectada alimentada por ela. Assim, a responsabilidade da engenharia de características temporais ficará à cargo da CNN que deverá passar a aplicar as convoluções em duas dimensões: espacial e temporal. No entanto, as configurações de hiper-parâmetros atuais para os mapas de características das camadas convolucionais são muito altas. Isso inviabiliza o treinamento por falta de memória disponível para realizar a alocação dos mapas de características, que devem ser mantidos em memória durante todo o treinamento. Além disso, estes mapas de características são utilizados como dados de entrada para uma camada totalmente conectada. Isso produziria uma quantidade excessiva de parâmetros que tornaria o modelo muito complexo e propenso a sofrer do efeito de *overfitting*, devido a ausência de dados disponíveis para treinar toda essa gama de parâmetros. Para resolver este problema propõe-se três métodos. No primeiro método, a

redução do número de parâmetros é realizada pela redução dos hiper-parâmetros dos mapas de características das camadas convolucionais e dos neurônios da camada totalmente conectada. Após alguns experimentos realizados chegou-se ao número de 8, 16 e 32 mapas de características na primeira, segunda e terceira camadas convolucionais, respectivamente, e o número de 256 neurônios na camada totalmente conectada. Com este método, obteve-se uma acurácia média de $61.37\%(\pm 2.02\%)$. No segundo método, as configurações de hiper-parâmetros são mantidas inalteradas e a redução de parâmetros é realizada pela técnica de *Pooling* (muito comum em arquiteturas de Redes Neurais Convolucionais). Após cada camada convolucional, é adicionada uma camada de *Average Pooling* de dimensão 3×1. Com este método, obteve-se uma acurácia média de $60.03\%(\pm 1.81\%)$. No terceiro método, as alterações realizadas nos dois primeiros métodos são aplicadas e obteve-se uma acurácia média de $62.56\%(\pm 1.86\%)$. O terceiro método possui a menor quantidade de parâmetros e apresentou o melhor desempenho. Contudo, a diferença dos resultados do sistema realizando a extração de características temporais com e sem a LSTM não é estatisticamente significante.

Como o resultado obtido foi muito aquém do esperado, decidiu-se pela reavaliação dos parâmetros utilizados na segmentação, uma vez que os trabalhos de Dose *et al.* (2018) e Wang *et al.* (2020) utilizaram uma janela de 3 segundos (duas vezes maior que a janela utilizada nos experimentos anteriores). Para avaliar janelas maiores, foram realizados experimentos variando os comprimentos da janela (L) e do deslocamento (d), partindo com base na configuração utilizada nos experimentos anteriores (i.e., $L = 160$ e $d = 160$). Os experimentos foram realizados para janelas de 1 segundo ($L = 160$), 2 segundos ($L = 320$), 3 segundos ($L = 480$) e 4 segundos ($L = 640$), sem deslocamento, e nos cenários aplicáveis, também com deslocamentos de 50% e 100% do comprimento da janela. Os resultados destes experimentos (Figura 44) confirmam os resultados reportados por Dose *et al.* (2018) e Wang *et al.* (2020). Pode-se notar que aumentar a quantidade de janelas de dados pelo deslocamento dentro dos ensaios degrada o desempenho em todos os cenários em que foi aplicado. Esse resultado demonstra que existem informações importantes para o modelo no início dos ensaios (logo após o sujeito ser exposto ao estímulo de imagética motora). Além disso, pode-se concluir que a engenharia de características temporais realizada pela CNN é mais efetiva do que a realizada pela LSTM. Para os três maiores comprimentos de janela (i.e., 2, 3 e 4 segundos) a CNN conseguiu gerar uma representação robusta dos sinais de EEG, chegando a uma acurácia média de $80.49\%(\pm 4.23\%)$ para uma janela de 3 segundos (número de amostras de treinamento e teste para cada classe são apresentadas na Tabela 4). Enquanto que a CRNN com LSTM não apresentou a mesma robustez ao receber um maior contexto da informação temporal dos sinais de EEG, ficando com acurácias entre 59% e 61% para o mesmo experimento.

A representação matricial é a representação natural dos dados de entrada de Redes Neurais Convolucionais de 2 dimensões. Estas redes são comumente utilizadas em problemas de visão computacional, onde os dados de entrada são imagens representadas por matrizes compostas por pixels de altura x pixels de largura. Para a aplicação destas redes em outros

Figura 44 – Resultados da CNN-2D com diferentes comprimentos de janela.



Fonte: O Autor (2020).

Tabela 4 – Quantidade de amostras geradas (com $L = 480$ e $d = 0$) por partição (k) no processo de validação cruzada *10-fold* para a base de dados da *PhysioNet*.

	Punho Esquerdo		Punho Direito	
k	Treinamento	Teste	Treinamento	Teste
1	2107	253	2078	242
2	2112	248	2073	247
3	2112	248	2073	247
4	2113	247	2072	248
5	2129	231	2101	219
6	2130	230	2100	220
7	2133	227	2097	223
8	2136	224	2094	226
9	2135	225	2095	225
10	2133	227	2097	223
Total	21240	2360	20880	2320

Fonte: O Autor (2020).

tipos de dados são realizadas transformações para adequar o formato dos dados de entrada, como é o caso em Zhang *et al.* (2018), utilizado como referência para a proposta do presente trabalho. Entretanto, como foi verificado nos experimentos realizados, utilizar uma representação matricial como formato dos dados de entrada não auxilia na melhoria do desempenho do sistema. Os experimentos realizados até aqui sugerem que os sinais de EEG não contêm informações

espaciais relevantes para o modelo. Para confirmar essa suposição, as camadas convolucionais foram alteradas para realizarem as convoluções apenas no domínio temporal. Isso apresentou uma melhoria no desempenho do sistema chegando a uma acurácia média de $82.11\%(\pm 3.65\%)$. Esta nova arquitetura será chamada de Rede Neural Convolucional de 1 dimensão (CNN-1D).

3.3.5 RESULTADOS NO CENÁRIO DE CLASSIFICAÇÃO BINÁRIA

O problema de classificação binária (punho esquerdo e punho direito) na *PhysioNet* independente do sujeito foi abordado em Zhang *et al.* (2020), Dose *et al.* (2018) e Wang *et al.* (2020). Zhang *et al.* (2020) propuseram um modelo recorrente convolucional baseado em mecanismos de atenção que não são contemplados no presente trabalho. O processo de construção dos dados de entrada realiza a segmentação pela técnica de janelamento utilizando uma janela de 62.5 ms com deslocamento de 31.25 ms gerando múltiplas amostras por ensaio. A arquitetura do modelo considera as informações espaciais e temporais contidas nos sinais de EEG. A acurácia é estimada pela média obtida por 9 experimentos realizados de forma independente do sujeito. Em cada experimento, o subconjunto de teste é composto pelas amostras de 10 sujeitos selecionados aleatoriamente e o subconjunto de treinamento é composto pelas amostras dos sujeitos restantes.

Dose *et al.* (2018) e Wang *et al.* (2020) realizaram o processo de avaliação de desempenho muito próximo ao realizado no presente trabalho. A única diferença é que utilizaram a validação cruzada *5-fold*. Dose *et al.* (2018) propuseram uma CNN composta por camadas responsáveis pela geração de uma representação espacial-temporal dos sinais de EEG e por camadas totalmente conectadas responsáveis pela classificação da representação gerada nas classes do problema. Wang *et al.* (2020) propuseram uma arquitetura muito parecida visando a redução em grande escala do número de parâmetros pela utilização de camadas convolucionais simplificadas.

Os três trabalhos propõe modelos de aprendizado profundo ponta a ponta para a classificação de imagética motora independente do sujeito. Todos consideram as informações espaciais contidas nos sinais de EEG, característica esta que pelas experimentações realizadas neste trabalho não apresentou melhoria no desempenho do sistema proposto.

A Tabela 5 mostra os resultados para os referidos trabalhos e o sistema proposto. O resultado obtido por Zhang *et al.* (2020), reforça ainda mais o fato de que aumentar a quantidade de dados de treinamento pelo deslocamento dentro dos ensaios, não auxilia na obtenção de resultados competitivos no cenário de classificação de imagética motora independente do sujeito. O resultado reportado por Wang *et al.* (2020) não possui diferença significativa ao resultado obtido pelo sistema proposto, no entanto é o modelo que possui o menor número de parâmetros.

Tabela 5 – Resultados da classificação das classes punho esquerdo e punho direito.

Trabalho	Janela	Acurácia (%)	Parâmetros
Zhang *et al.* (2020)	2.5 s (deslocamento de 62.5 ms)	$74.71(\pm 4.19)$	420356
Dose *et al.* (2018)	3 s	80.10	203122
Wang *et al.* (2020)	3 s	$\mathbf{82.43}$	2918
CNN-1D (Sistema proposto)	3 s	$82.11(\pm 3.65)$	143546

Fonte: O Autor (2020).

3.3.6 RESULTADOS NO CENÁRIO DE CLASSIFICAÇÃO DE 5 CLASSES

Para a avaliação de desempenho e comparação de resultados considerando um número maior de classes, por não haver um padrão nas classes utilizadas nos trabalhos encontrados na literatura, são realizados dois experimentos. Em um primeiro experimento foram consideradas as mesmas classes utilizadas para a reprodução dos resultados da implementação da arquitetura CRNN (olhos fechados, punho esquerdo, punho direito, punhos e pés). A Tabela 6 apresenta as quantidades de treinamento e teste para cada partição. O resultado deste experimento possibilitará avaliar se as modificações realizadas durante as experimentações no cenário de classificação binária produziram alguma melhoria de desempenho no cenário de classificação de 5 classes independente do sujeito.

Tabela 6 – Quantidade de amostras geradas (com $L = 480$ e $d = 0$) por partição (k) no processo de validação cruzada *10-fold* para o cenário de 5 classes da base de dados da *PhysioNet*.

	Olhos Fechados		Punho Esq.		Punho Dir.		Punhos		Pés	
k	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste
1	1859	220	2107	253	2078	242	2096	246	2089	249
2	1859	220	2112	248	2073	247	2092	250	2093	245
3	1859	220	2112	248	2073	247	2094	248	2091	247
4	1859	220	2113	247	2072	248	2091	251	2094	244
5	1879	200	2129	231	2101	219	2115	227	2115	223
6	1879	200	2130	230	2100	220	2117	225	2113	225
7	1879	200	2133	227	2097	223	2117	225	2113	225
8	1879	200	2136	224	2094	226	2119	223	2111	227
9	1880	199	2135	225	2095	225	2120	222	2110	228
10	1879	200	2133	227	2097	223	2117	225	2113	225
Total	18711	2079	21240	2360	20880	2320	21078	2342	21042	2338

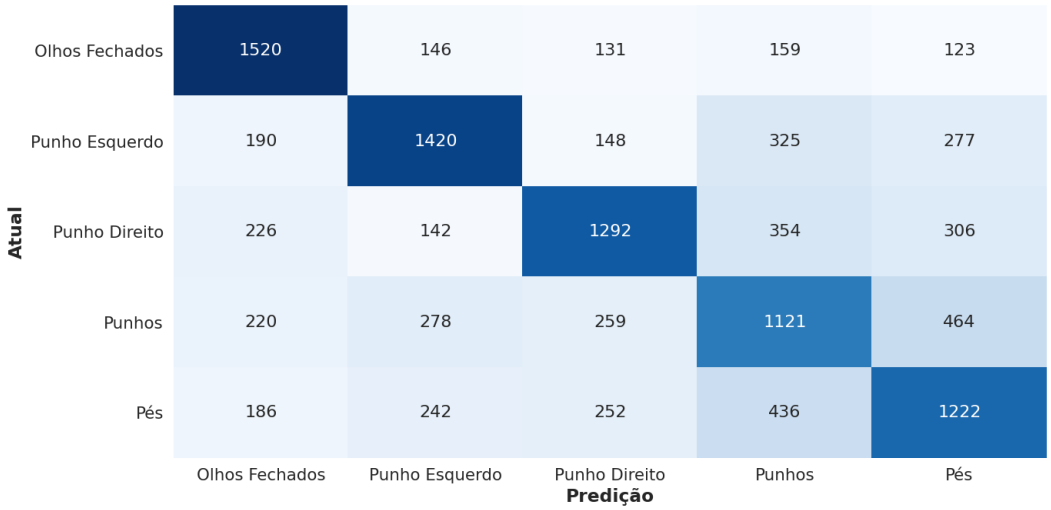
Fonte: O Autor (2020).

A única modificação realizada no modelo foi no número de neurônios da camada de saída, alterado de 2 para 5 (por conta do número de classes). O sistema obteve uma acurácia média

de $57.47\%(\pm 4.2\%)$, representando uma melhoria relativa de 69.63% em relação ao resultado obtido pela CRNN.

A matriz de confusão (Figura 45) permite tirar algumas conclusões a respeito do desempenho do sistema em cada classe. Considerando apenas as classes de imagética motora (i.e., ignorando a classe olhos fechados), pode-se notar uma grande incerteza do sistema nas predições entre as classes punhos e pés. Os maiores acertos dentre as classes de imagética motora se deram nas classes punho esquerdo e punho direito. Além disso, a maioria dos erros dessas classes não ocorreram entre si e sim nas classes punhos e pés. Esses comportamentos podem ser o resultado da extensão e posicionamento de cada membro no mapa somatotópico (também conhecido como Homúnculo) da superfície corporal no córtex somatossensorial (BEAR; CONNORS; PARADISO, 2017). Para a análise no mapa somatotópico das classes punho esquerdo, punho direito e punhos, é plausível considerar a área correspondente ao punho e mão, uma vez que a atividade referente a estas classes é a imaginação da abertura e fechamento dos punhos, movimento este que obviamente envolve as mãos. Os punhos e as mãos são representados no mapa somatotópico ao longo de uma grande extensão e lateralizados. Isto explica o maior número de verdadeiros positivos nas classes punho esquerdo e punho direito, bem como a grande incerteza nas predições entre estas classes e a classe punhos. Os pés são representados no mapa somatotópico em uma única região. Entretanto, sua extensão de representação é menor em relação a extensão dos punhos e mãos. Como a extensão representada no mapa somatotópico está relacionada à importância sensorial daquela área (BEAR; CONNORS; PARADISO, 2017), faz sentido um menor número de predições corretas correspondentes às classes de menor extensão no mapa somatotópico.

Figura 45 – Matriz de confusão dos resultados da classificação de 5 classes na base de dados da *PhysioNet*.



Fonte: O Autor (2020).

Para comparações com trabalhos da literatura foi encontrado apenas o proposto por Fadel

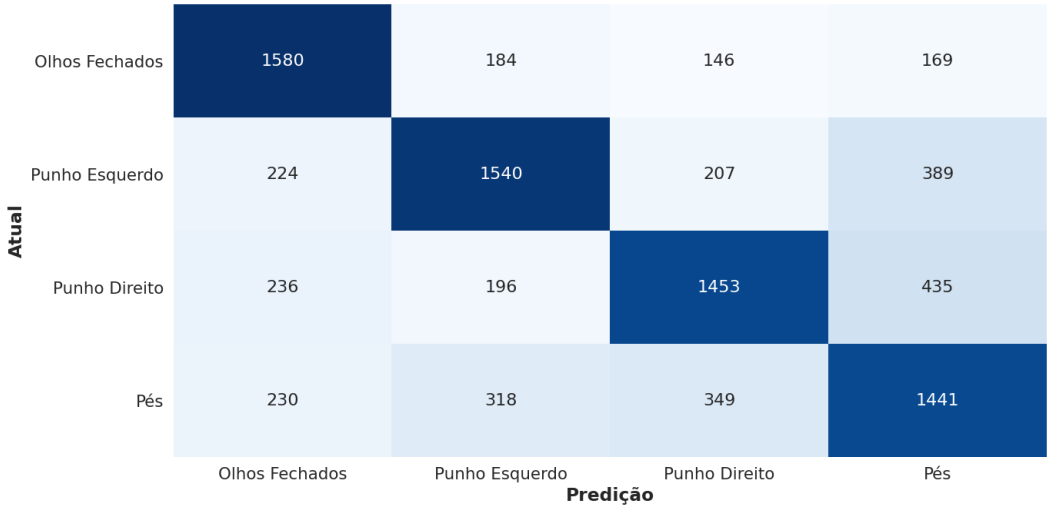
et al. (2020), embora utilizem a classe *rest* ao invés da classe olhos fechados. Os dados desta classe são aqueles adquiridos no período de descanso entre as atividades das classes relacionadas aos movimentos dos punhos e pés. Eles obtiveram uma acurácia média de 70.64%, que representa uma melhoria relativa de 22.92% em relação ao sistema proposto. Entretanto, a abordagem de avaliação de desempenho utilizada foi a *Leave-One-Out*, utilizando dados de $n - 2$ sujeitos para o treinamento, dados de 1 sujeito selecionado aleatoriamente para validação e dados de 1 sujeito para teste. O trabalho propõe a utilização de técnicas de processamento de sinais para a geração de imagens dos sinais de EEG em múltiplas bandas de frequência. Estas imagens formam os dados de entrada para uma rede neural artificial composta por uma CNN de múltiplas camadas seguida por uma LSTM e pela camada de classificação. Dentre as comparações realizadas até o presente momento, é a primeira em que o sistema proposto obteve um desempenho inferior por uma larga margem. Nota-se também que é a primeira comparação baseada em um modelo que não é ponta a ponta, uma vez que utiliza técnicas de processamento de sinais para a geração dos dados de entrada.

3.3.7 RESULTADOS NO CENÁRIO DE CLASSIFICAÇÃO DE 4 CLASSES

Para a avaliação de desempenho no cenário de classificação de 4 classes, foram removidos os ensaios correspondentes à classe punhos. A única modificação realizada no modelo foi no número de neurônios da camada de saída, alterado para 4. O sistema obteve uma acurácia média de $66.11\%(\pm 4.46\%)$. Pela matriz de confusão (Figura 46), pode-se notar um aumento de acertos em todas as classes, mas principalmente nas classes punho direito e pés, que agora apresentam um maior equilíbrio no número de acertos. Este resultado corrobora a análise anterior realizada com base no mapa somatotópico e sugere que protocolos experimentais de interfaces cérebro-máquina baseadas em imagética motora não devem utilizar a classe punhos em conjunto com a classe pés ou com classes relacionadas aos punhos ou às mãos. Agora, a maior incerteza do sistema reside na classe pés com as classes punho esquerdo e punho direito, a qual pode-se explicar pela extensão da área dos pés representada no mapa somatotópico, como já explicado na análise anterior.

Para comparações com trabalhos da literatura foram encontrados apenas trabalhos que utilizaram a classe olhos abertos ao invés da classe olhos fechados. Os resultados (Tabela 7) são dos trabalhos de Dose *et al.* (2018) e Wang *et al.* (2020), já apresentados nas comparações no cenário de classificação binária. Neste cenário, o sistema proposto apresenta uma melhoria relativa de 1.6% em relação ao trabalho de Wang *et al.* (2020). Essas pequenas variações apresentadas em diferentes cenários, demonstram que a capacidade de ambos os sistemas em realizar a classificação de imagética motora independente do sujeito é muito parecida.

Figura 46 – Matriz de confusão dos resultados da classificação de 4 classes na base de dados da *PhysioNet*.



Fonte: O Autor (2020).

Tabela 7 – Resultados da classificação das classes olhos abertos/fechados, punho esquerdo, punho direito e pés.

Trabalho	Janela	Acurácia (%)	Parâmetros
Dose *et al.* (2018)	3 s	59.71	203284
Wang *et al.* (2020)	3 s	65.07	3144
CNN-1D (Sistema proposto)	3 s	$\mathbf{66.11}(\pm\mathbf{4.46})$	144060

Fonte: O Autor (2020).

3.4 RESULTADOS NA BASE DE DADOS II-A

O melhor sistema obtido pelas experimentações na base de dados da *PhysioNet* foi utilizado para avaliar o desempenho na base de dados II-A da IV Competição de Interfaces Cérebro-Máquina. Esta base de dados possui um total de 4 classes (mão esquerda, mão direita, pés e língua). A única modificação realizada no modelo foi no número de neurônios utilizados na camada de saída de acordo com cada experimento realizado. O processo de preparação dos dados foi mantido exatamente como o realizado na base de dados da *PhysioNet*, ou seja, para cada ensaio uma amostra é gerada com os primeiros 3 segundos do sinal de EEG. A dimensão dos dados de entrada foi alterada de acordo com a frequência de amostragem e o número de eletrodos utilizados para a aquisição dos sinais de EEG. Para uma taxa de amostragem de 250 Hz e 22 eletrodos, a dimensão dos dados de entrada tornou-se 750×22. As amostras marcadas como rejeitadas por inspeção visual realizada por especialistas foram desconsideradas. Mesmo assim, a base continua aproximadamente balanceada, como pode ser visualizado na Tabela 8.

Como esta base de dados possui somente 9 sujeitos, a avaliação de desempenho foi

Tabela 8 – Quantidade de amostras da base de dados II-A.

Sujeito	Mão Esquerda	Mão Direita	Pés	Língua
1	140	139	137	138
2	138	140	135	140
3	136	138	134	135
4	121	124	126	119
5	133	131	138	136
6	109	112	103	110
7	138	135	138	137
8	132	134	133	136
9	118	128	130	125
Total	1165	1181	1174	1176

Fonte: O Autor (2020).

realizada utilizando a técnica de validação cruzada *Leave-One-Out*[10] sobre os sujeitos. Em cada iteração, dados de $n-1$ sujeitos são utilizados para o treinamento e os dados do sujeito restante para a avaliação do sistema. O treinamento foi realizado por 1500 épocas. Isso foi possível devido ao menor número de amostras da base de dados.

Em um primeiro experimento foram utilizadas somente as amostras das classes mão esquerda e mão direita. A acurácia média obtida foi de $78.31\%(\pm 5.49\%)$. O único trabalho encontrado na literatura realizando a classificação destas classes nesta base de dados foi o de Lotte, Guan & Ang (2009). Eles propuseram uma comparação entre diversas abordagens baseadas em engenharia de características nos cenários de classificação dependente e independente do sujeito. A avaliação de desempenho independente do sujeito foi realizada da mesma forma que no presente trabalho. O melhor resultado obtido foi de 67.59%. O sistema proposto no presente trabalho apresenta uma melhoria relativa de 15.86%. Isso demonstra o potencial dos modelos baseados em aprendizado profundo na obtenção de representações robustas dos sinais de EEG, para a classificação de imagética motora independente do sujeito. Algoritmos baseados em aprendizado profundo possuem a capacidade de capturar características de alto nível e dependências latentes presentes nos dados, sem a necessidade de um grande conhecimento do domínio. Por outro lado, os algoritmos baseados em engenharia de características demandam um grande conhecimento do domínio para definir os melhores algoritmos e técnicas para determinado problema. Estas suposições podem acabar descartando informações discriminativas contidas nos dados dificultando a tarefa de classificação.

Em um segundo experimento foram utilizadas as amostras das 4 classes disponíveis. A acurácia média obtida foi de $55.74\%(\pm 5.71\%)$. Pela matriz de confusão (Figura 47), pode-se observar novamente que, assim como na base de dados da *PhysioNet*, as classes correspondentes aos membros de menor extensão no mapa somatotópico (i.e., pés e língua) são aquelas de maior

[10] A técnica de validação cruzada *Leave-One-Out* é um caso especial da técnica de validação cruzada *K-fold*, onde K é configurado como o número de amostras ou de sujeitos quando aplicado sobre os sujeitos.

incerteza nas predições realizadas pelo sistema.

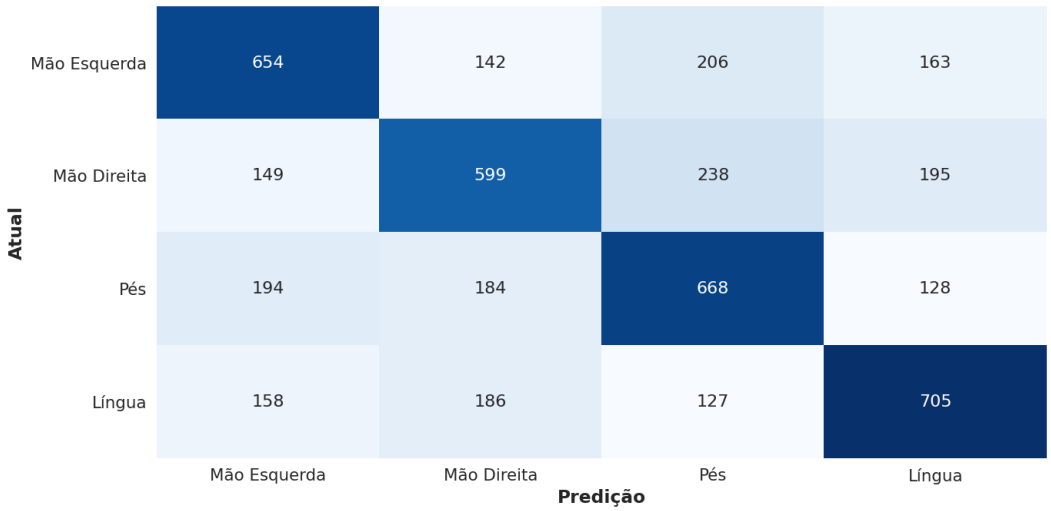
Os resultados obtidos na literatura para comparação e do sistema proposto podem ser visualizados na Tabela 9.

Tabela 9 – Resultados da classificação das classes mão esquerda, mão direita, pés e língua.

Trabalho	Janela	Acurácia (%)	Parâmetros
Lawhern *et al.* (2016)	2s (a partir do tempo 0,5s)	40.0	796
Zhang *et al.* (2020)	1,6s (deslocamento de 0,2s)	$\mathbf{60.11}(\pm\mathbf{9.96})$	420356
CNN-1D (Sistema proposto)	3 s	$55.74\%(\pm 5.71\%)$	224972

Fonte: O Autor (2020).

Figura 47 – Matriz de confusão dos resultados na base de dados II-A da IV Competição de Interfaces Cérebro-Máquina.



Fonte: O Autor (2020).

Lawhern *et al.* (2016) propõe a *EEGNet*, uma rede neural convolucional compacta para o reconhecimento de diferentes tipos de sinais de EEG. A avaliação de desempenho independente do sujeito não foi realizada da mesma forma como no presente trabalho. O subconjunto de dados de teste é formado pelas amostras de teste de 1 sujeito. As amostras de treinamento de 5 outros sujeitos selecionados aleatoriamente são utilizadas para formarem o subconjunto de dados de treinamento. As amostras de treinamento dos 3 sujeitos restantes são utilizadas para formarem o subconjunto de dados de validação. Este processo é repetido 10 vezes para cada sujeito produzindo um total de 90 resultados, a partir dos quais é estimada a acurácia do sistema. O sistema proposto obteve uma melhoria relativa de 39.35%. Um fator que pode explicar esta diferença de desempenho tão grande é a baixa complexidade da *EEGNet* em relação ao sistema proposto. Embora tenha como objetivo ser um sistema genérico para diferentes tipos de sinais de EEG, não é capaz de obter uma representação robusta o suficiente para a classificação de sinais de imagética motora independente do sujeito.

Zhang *et al.* (2020) propuseram um modelo de atenção recorrente convolucional baseado em grafos. Esta categoria de modelos não está contemplada no escopo do presente trabalho. Mas como trata-se de um modelo baseado em aprendizado profundo e a avaliação de desempenho independente do sujeito foi realizada da mesma forma que no presente trabalho, é uma comparação justa. O desempenho obtido representa uma melhoria relativa de 7.84% em relação ao sistema proposto mesmo utilizando uma janela menor e aplicando a técnica de deslocamento, que nos experimentos realizados no presente trabalho não mostrou-se efetiva.

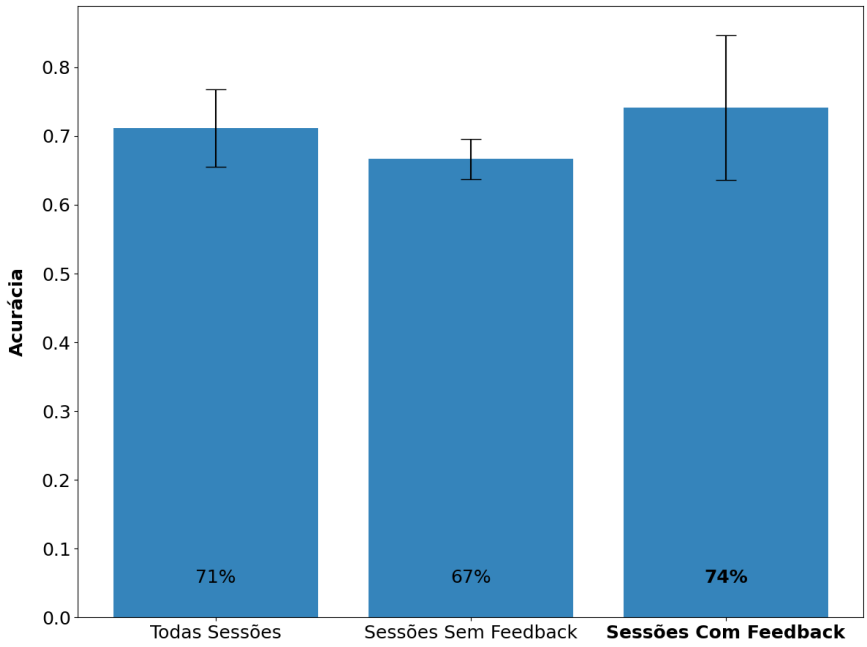
3.5 RESULTADOS NA BASE DE DADOS II-B

A base de dados II-B da IV Competição de Interfaces Cérebro-Máquina possui sinais de EEG de 2 classes de imagética motora (mão esquerda e mão direita) provenientes de sessões de aquisição com e sem *feedback* aos sujeitos. O treinamento e avaliação de desempenho foram realizados da mesma forma que na base de dados II-A.

Baseado na hipótese de que, o fornecimento de *feedback* aos sujeitos durante a aquisição dos sinais de EEG auxilia na melhoria da sua capacidade de modulação dos sinais de imagética motora, foram realizados experimentos utilizando diferentes estratégias de avaliação de desempenho. Em um primeiro experimento, foi utilizada a mesma estratégia de avaliação aplicada na base de dados II-A, utilizando os sinais de todas as sessões de aquisição. Em um segundo experimento, foi utilizada a estratégia de avaliação *offline*, utilizando somente os dados provenientes das sessões de aquisição sem *feedback*. Em um terceiro experimento, foi utilizada a estratégia de avaliação *online* simulado, utilizando somente os dados provenientes das sessões de aquisição com *feedback*. Os resultados (Figura 48), demonstram uma melhoria relativa de 11.19% na acurácia do sistema quando treinado nos dados em que os sujeitos receberam *feedback*. Este resultado sugere que a hipótese de que, fornecer *feedback* aos sujeitos auxilia na melhoria da sua capacidade de modulação dos sinais de imagética motora, é verdadeira. No entanto, o desvio padrão dos resultados utilizando a estratégia *online* simulado foi muito elevado (10.54%). E além disso, é importante frisar que trata-se de uma base de dados pequena, com apenas 9 sujeitos. Para confirmar esta hipótese, seria necessário uma base de dados com um protocolo experimental nos mesmos moldes desta e um número maior de sujeitos.

Para comparação com resultados obtidos na literatura foi utilizado o resultado obtido utilizando os dados provenientes de todas as sessões de aquisição ($71.14\%(\pm 5.63\%)$). O único trabalho encontrado para comparação foi o de Roy *et al.* (2020). Eles propuseram uma arquitetura composta por múltiplos pares de mega blocos e *pooling*. Os mega blocos, segundo os autores, são componentes compostos por múltiplas camadas convolucionais capazes de repetir sua estrutura sobre o domínio do tempo. O objetivo dos mega blocos é tornar o modelo adaptável para considerar o ruído presente nos sinais e a variabilidade entre diferentes sujeitos e sessões. Os dados de entrada são imagens obtidas por espectogramas gerados a partir dos sinais de EEG. Cada imagem é formada combinando os espectogramas das bandas *theta-alpha* e *beta*, verticalmente.

Figura 48 – Resultados na base de dados II-B da IV Competição de Interfaces Cérebro-Máquina.



Fonte: O Autor (2020).

A dimensão dos dados de entrada é: pixels de altura x pixels de largura x quantidade de eletrodos. Para cada ensaio, a técnica de janelamento é aplicada com um deslocamento de 200 ms considerando os 4 segundos iniciais do ensaio (a partir do recebimento do estímulo de imagética motora). Assim, para cada ensaio, são geradas 11 amostras e a classificação de um ensaio é considerada correta se ao menos 6 de suas amostras são classificadas corretamente. A avaliação de desempenho independente do sujeito é realizada da mesma forma que no presente trabalho. O desempenho reportado foi uma acurácia média de $70.94\%(\pm 9.89\%)$, diferença insignificante em relação ao desempenho obtido no sistema proposto no presente trabalho.

4 CONCLUSÃO

O objetivo principal deste trabalho consistiu na implementação de um sistema para o reconhecimento de múltiplas classes de imagética motora independente do sujeito. Sabe-se que milhares de pessoas ao redor do mundo sofrem de algum tipo de dificuldade motora, seja devido ao acometimento de doenças degenerativas do sistema motor ou ao sofrimento de algum acidente que tenha ocasionado a paralisia de membros. Essas pessoas possuem enormes dificuldades para a realização de suas tarefas diárias e, nos casos mais extremos, ficam completamente inabilitadas. Nesse sentido, torna-se essencial o desenvolvimento de tecnologias assistivas que auxiliem na recuperação e no dia a dia dessas pessoas. Interfaces cérebro-máquina são dispositivos que permitem a comunicação com o mundo externo através da análise dos sinais cerebrais. A eletroencefalografia (EEG) é uma tecnologia que permite a coleta desses sinais de forma prática e não invasiva. Interfaces cérebro-máquina baseadas em imagética motora permitem aos sujeitos a comunicação com dispositivos inteligentes de forma completamente independente, necessárias para os cenários de doenças degenerativas muito graves em que os sujeitos alvos não possuem qualquer atividade nas vias normais de comunicação responsáveis pelo controle dos nervos e músculos. O componente chave de interfaces cérebro-máquina baseadas em imagética motora é um sistema de reconhecimento que possa, de forma confiável e em tempo razoável, realizar a predição das intenções dos sujeitos. Grandes desafios ainda permanecem para o desenvolvimento de sistemas de reconhecimento de imagética motora que possam ser utilizados em interfaces cérebro-máquina comercializáveis, sendo os principais: a confiabilidade de predição e o tempo de resposta. Nesse sentido, é de suma importância a exploração de novas arquiteturas e técnicas das subáreas da inteligência artificial que possam vir a solucionar esses desafios.

Abordagens baseadas em engenharia de características demonstraram uma boa capacidade de predição nos cenários de classificação de imagética motora dependente do sujeito. Entretanto, estas abordagens demandam um grande conhecimento do domínio para a definição dos melhores algoritmos e técnicas. Estas suposições podem acabar descartando informações discriminativas contidas nos dados dificultando a tarefa de classificação. Abordagens baseadas em aprendizado profundo têm demonstrado um maior potencial nos cenários de classificação de múltiplas classes e independente do sujeito. Esta abordagem permite o desenvolvimento de sistemas ponta a ponta que não necessitam de um profundo conhecimento do domínio, e devido a não possuir etapas de pré-processamento computacionalmente complexas, podem realizar as predições em um tempo de resposta mais adequado para sistemas *online*.

A partir da revisão bibliográfica realizada, foi possível o desenvolvimento de um sistema de reconhecimento de imagética motora independente do sujeito. O sistema obteve um desempenho competitivo com os trabalhos encontrados na literatura para o cenário de classificação binária, tanto na base de dados da *PhysioNet* quanto na base de dados II-B da IV Competição de

Interfaces Cérebro-Máquina. Em comparação com resultados de sistemas baseados em engenharia de características na base de dados II-A da IV Competição de Interfaces Cérebro-Máquina, o sistema proposto obteve uma melhoria relativa de 15.86%. Esse resultado demonstra o potencial dos sistemas baseados em aprendizado profundo para a classificação de imagética motora independente do sujeito.

Nas análises realizadas dos resultados nos cenários de classificação de 4 e 5 classes na base de dados da *PhysioNet*, observou-se uma grande incerteza do sistema na predição da classe punhos com as classes pés, punho esquerdo e punho direito. Isso pode ser explicado devido à representação dos punhos e das mãos ser lateralizada no mapa somatotópico, enquanto que os pés são representados em uma região central entre a região do punho e da mão esquerda e a região do punho e da mão direita. Assim, pode-se concluir que protocolos experimentais de interfaces cérebro-máquina baseadas em imagética motora, não devem utilizar a classe punhos em conjunto com a classe pés ou com classes relacionadas aos punhos ou às mãos, pois isso tende a gerar uma grande incerteza no sistema nas predições entre essas classes. No cenário de classificação de 4 classes (i.e., sem a classe punhos), o sistema obteve uma melhoria relativa de 1.6% em relação ao melhor resultado encontrado na literatura.

Com as experimentações realizadas no decorrer do desenvolvimento deste trabalho, observou-se que o fornecimento de um maior contexto da informação temporal dos sinais de EEG agrega informações relevantes ao modelo. O tamanho ideal de janela obtido nas experimentações foi de 3 segundos. A utilização de deslocamento como técnica para o aumento dos dados disponíveis para o treinamento não mostrou-se efetiva para o sistema proposto. Em todos os cenários avaliados, isso causou uma grande degradação no desempenho do sistema. Esse resultado permite concluir que existe informações importantes para o sistema diferenciar diferentes tipos de imagética motora no início dos ensaios, logo após o sujeito ser exposto ao estímulo de imagética motora.

Dentre as redes neurais utilizadas, a CNN mostrou-se mais efetiva na geração de uma representação robusta para a classificação de imagética motora independente do sujeito. Ao fornecer um maior contexto da informação temporal dos sinais de EEG, a CNN conseguiu obter acurácias acima de 80%, enquanto que a LSTM ficou com acurácias entre 59% e 61%. Isso demonstra a maior capacidade da CNN para a engenharia de características temporais dos sinais de EEG.

Embora a base de dados II-B da IV Competição de Interfaces Cérebro-Máquina seja pequena, os resultados obtidos sugerem que o fornecimento de *feedback* aos sujeitos auxilia na melhoria da sua capacidade de modulação dos sinais de imagética motora. Foi obtida uma melhoria relativa de 11.19% na acurácia do sistema quando treinado nos dados em que os sujeitos receberam *feedback*.

Nos resultados obtidos na literatura da classificação de múltiplas classes de imagética motora independente do sujeito na base de dados da *PhysionNet*, verificou-se que a aplicação

de técnicas de processamento de sinais para a preparação dos dados de entrada contribui para a obtenção de bons resultados. Como continuação deste trabalho pode ser realizada a exploração de sistemas que não são ponta a ponta, realizando um maior pré-processamento dos dados de entrada. Outro desafio que fica para a exploração em um trabalho futuro, é a redução do problema de escassez de sinais de EEG de imagética motora disponíveis para o treinamento dos sistemas. Neste trabalho foi realizada a tentativa sem sucesso do aumento dos dados pela técnica de janelamento. Redes neurais generativas podem ser empregadas para a geração de novos dados baseado nos dados disponíveis sem a necessidade do emprego do deslocamento nos sinais de EEG, que no presente trabalho resultou na degradação do sistema.

REFERÊNCIAS

ALI, S. *et al.* A novel features selection approach with common spatial pattern for eeg based brain–computer interface implementation. **IETE Journal of Research**, Taylor & Francis, p. 1–15, 2019.

ALJALAL, M.; DJEMAL, R.; IBRAHIM, S. Robot navigation using a brain computer interface based on motor imagery. **Journal of Medical and Biological Engineering**, v. 39, n. 4, p. 508–522, ago. 2019.

AMIN, S. U. *et al.* Deep learning for eeg motor imagery classification based on multi-layer cnns feature fusion. **Future Generation Computer Systems**, v. 101, p. 542–554, 2019.

ANG, K. K. *et al.* Filter bank common spatial pattern (fbcsp) in brain-computer interface. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, 2008, Hong Kong, China. **Proceedings...** [S.l.]: IEEE, 2008.

BASHAR, S. K.; BHUIYAN, M. I. H. Classification of motor imagery movements using multivariate empirical mode decomposition and short time fourier transform based hybrid method. **Engineering Science and Technology, an International Journal**, v. 19, n. 3, p. 1457–1464, 2016.

BEAR, M. F.; CONNORS, B. W.; PARADISO, M. A. **Neurociências: Desvendando o Sistema Nervoso**. 4. ed. Porto Alegre: Artmed, 2017. 1016 p.

CHEN, Y.; ZHANG, C.; WU, X. To assess the influence of artifacts on motor imagery based bci. In: INTERNATIONAL CONFERENCE ON SIGNAL AND IMAGE PROCESSING (ICSIP), 4., 2019, Wuxi, China. **Proceedings...** [S.l.]: IEEE, 2019.

CHO, H. *et al.* EEG datasets for motor imagery brain–computer interface. **GigaScience**, v. 6, n. 7, 05 2017.

COHEN, M. **Analyzing neural time series data : theory and practice**. 1. ed. Cambridge, Massachusetts: The MIT Press, 2014.

COYLE, D. *et al.* Eeg-based continuous control of a game using a 3 channel motor imagery bci: Bci game. In: COMPUTATIONAL INTELLIGENCE, COGNITIVE ALGORITHMS, MIND, AND BRAIN (CCMB), 2011, Paris, France. **Proceedings...** [S.l.]: IEEE, 2011.

Dietterich, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. **Neural Computation**, v. 10, n. 7, p. 1895–1923, 1998.

DING, B.; QIAN, H.; ZHOU, J. Activation functions and their characteristics in deep neural networks. In: CHINESE CONTROL AND DECISION CONFERENCE (CCDC), 2018, Shenyang, China. **Proceedings...** [S.l.]: IEEE, 2018.

Dornhege, G. *et al.* Evaluation criteria for bci research. In: **Toward Brain-Computer Interfacing**. [S.l.: s.n.], 2007. p. 327–342.

DOSE, H. *et al.* A deep learning mi - eeg classification model for bcis. In: EUROPEAN SIGNAL PROCESSING CONFERENCE (EUSIPCO), 26., 2018, Rome, Italy. **Proceedings...** [S.l.]: IEEE, 2018.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification**. 2. ed. New York: Wiley-Interscience, 2000.

FADEL, W. *et al.* Multi-class classification of motor imagery eeg signals using image-based deep recurrent convolutional neural network. In: INTERNATIONAL WINTER CONFERENCE ON BRAIN-COMPUTER INTERFACE (BCI), 8., 2020, Gangwon, Korea (South), Korea (South). **Proceedings...** [S.l.]: IEEE, 2020.

GAUR, P. *et al.* An empirical mode decomposition based filtering method for classification of motor-imagery eeg signals for enhancing brain-computer interface. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2015, Killarney, Ireland. **Proceedings...** [S.l.]: IEEE, 2015.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. 1. ed. Cambridge: MIT Press, 2016.

GRAIMANN, B.; ALLISON, B.; PFURTSCHELLER, G. Brain–computer interfaces: A gentle introduction. In: GRAIMANN, B.; PFURTSCHELLER, G.; ALLISON, B. (Ed.). **Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27.

GéRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems**. 2. ed. Sebastopol, CA: O'Reilly Media, Inc, 2019.

HAMEDI, M. *et al.* Neural network-based three-class motor imagery classification using time-domain features for bci applications. In: REGION 10 SYMPOSIUM, 2014, Kuala Lumpur, Malaysia. **Proceedings...** [S.l.]: IEEE, 2014.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, p. 1735–80, 12 1997.

HWANG, H.-J. *et al.* Eeg-based brain-computer interfaces: A thorough literature survey. **International Journal of Human–Computer Interaction**, Taylor & Francis, v. 29, n. 12, p. 814–826, 2013.

JAMES, G. *et al.* **An Introduction to Statistical Learning**. 1. ed. New York: Springer New York, 2013.

JOADDER, M. *et al.* A new design of mental state classification for subject independent bci systems. **IRBM**, v. 40, n. 5, p. 297–305, 2019.

JOADDER, M. A. M. *et al.* A performance based feature selection technique for subject independent mi based bci. **Health Information Science and Systems**, v. 7, n. 1, p. 15, ago. 2019.

KEIRN, Z. A.; AUNON, J. I. A new mode of communication between man and his surroundings. **IEEE Transactions on Biomedical Engineering**, v. 37, n. 12, p. 1209–1214, 1990.

KHORSHIDTALAB, A.; SALAMI, M. J. E.; HAMEDI, M. Evaluation of time-domain features for motor imagery movements using fcm and svm. In: CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING (JCSSE), 9., 2012, Bangkok, Thailand. **Proceedings...** [S.l.]: IEEE, 2012.

____. Robust classification of motor imagery eeg signals using statistical time-domain features. **Physiological measurement**, England, v. 34, n. 11, p. 1563–1579, nov. 2013.

KIM, C. *et al.* An effective feature extraction method by power spectral density of eeg signal for 2-class motor imagery-based bci. **Medical & Biological Engineering & Computing**, v. 56, n. 9, p. 1645–1658, set. 2018.

KÜBLER, A. The history of bci: From a vision for the future to real support for personhood in people with locked-in syndrome. **Neuroethics**, v. 13, p. 163–180, maio 2019.

KUMAR, S.; SHARMA, A. A new parameter tuning approach for enhanced motor imagery eeg signal classification. **Medical & Biological Engineering & Computing**, v. 56, n. 10, p. 1861–1874, out. 2018.

KUMAR, S.; SHARMA, A.; TSUNODA, T. An improved discriminative filter bank selection approach for motor imagery eeg signal classification using mutual information. **BMC bioinformatics**, BioMed Central, v. 18, n. Suppl 16, p. 545–545, dez. 2017.

____. Subject-specific-frequency-band for motor imagery eeg signal recognition based on common spatial spectral pattern. In: PACIFIC RIM INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (PRICAI), 16., 2019, Pacific Rim, Cuvu, Yanuca. **Proceedings...** [S.l.]: Springer International Publishing, 2019.

KWON, O. *et al.* Subject-independent brain-computer interfaces based on deep convolutional neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–14, 2019.

LAWHERN, V. J. *et al.* Eegnet: A compact convolutional network for eeg-based brain-computer interfaces. **CoRR**, abs/1611.08024, 2016. Disponível em: <http://arxiv.org/abs/1611.08024>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, maio 2015.

LECUN, Y. A. *et al.* Efficient backprop. In: MONTAVON, G.; ORR, G. B.; MÜLLER, K.-R. (Ed.). **Neural Networks: Tricks of the Trade: Second Edition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 9–48.

LEE, H. K.; CHOI, Y. A convolution neural networks scheme for classification of motor imagery eeg based on wavelet time-frequecy image. In: INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING (ICOIN), 2018, Chiang Mai, Thailand. **Proceedings...** [S.l.]: IEEE, 2018.

LEMM, S. *et al.* Spatio-spectral filters for improving the classification of single trial eeg. **IEEE Transactions on Biomedical Engineering**, v. 52, n. 9, p. 1541–1548, 2005.

LI, F. *et al.* A novel simplified convolutional neural network classification algorithm of motor imagery eeg signals based on deep learning. **Applied Sciences**, MDPI AG, v. 10, n. 5, p. 1605, fev. 2020.

LOTTE, F.; GUAN, C.; ANG, K. K. Comparison of designs towards a subject-independent brain-computer interface based on motor imagery. **Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference**, United States, v. 2009, p. 4543–4546, 2009.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115–133, dez. 1943.

MILLS, A.; DUREPOS, G.; WIEBE, E. **Encyclopedia of Case Study Research**. 1. ed. Thousand Oaks: SAGE Publications, Inc., 2010.

NAM, C.; NIJHOLT, A.; LOTTE, F. **Brain-Computer Interfaces Handbook: Technological and Theoretical Advances**. 1. ed. Boca Raton: CRC Press, 2018.

NEUPER, C. *et al.* Imagery of motor actions: Differential effects of kinesthetic and visual-motor mode of imagery in single-trial eeg. **Brain Research / Cognitive Brain Research**, Elsevier B.V., v. 25, n. 3, p. 668–677, 2005.

NICOLAS-ALONSO, L. F.; GOMEZ-GIL, J. Brain computer interfaces, a review. **Sensors**, MDPI AG, v. 12, n. 2, p. 1211–1279, jan. 2012.

NOVI, Q. *et al.* Sub-band common spatial pattern (sbcsp) for brain-computer interface. In: INTERNATIONAL IEEE/EMBS CONFERENCE ON NEURAL ENGINEERING, 3., 2007, Kohala Coast, HI, USA. **Proceedings...** [S.l.]: IEEE, 2007.

OLAH, C. **Understanding LSTM Networks**. 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: jul. 2020.

OLIVEIRA, L. M. B. *et al.* **Cartilha do Censo 2010: pessoas com deficiência**. 1. ed. Brasília: SDH-PR/SNPD, 2012. Disponível em: <https://bibliotecadigital.mdh.gov.br/jspui/handle/192/754>.

PADFIELD, N. *et al.* Eeg-based brain-computer interfaces using motor-imagery: Techniques and challenges. **Sensors**, MDPI AG, v. 19, n. 6, p. 1423, mar. 2019.

PATTERSON, J. **Deep learning : a practitioner's approach**. 1. ed. Sebastopol, CA: O'Reilly, 2017.

PATTNAIK, S.; DASH, M.; SABUT, S. K. Dwt-based feature extraction and classification for motor imaginary eeg signals. In: INTERNATIONAL CONFERENCE ON SYSTEMS IN MEDICINE AND BIOLOGY (ICSMB), 2016, Kharagpur, India. **Proceedings...** [S.l.]: IEEE, 2016.

PFURTSCHELLER, G.; SILVA, F. L. da. Event-related eeg/meg synchronization and desynchronization: basic principles. **Clinical Neurophysiology**, v. 110, n. 11, p. 1842–1857, 1999.

QUILES, E. *et al.* Low-cost robotic guide based on a motor imagery brain–computer interface for arm assisted rehabilitation. **International Journal of Environmental Research and Public Health**, MDPI AG, v. 17, n. 3, p. 699, jan. 2020.

RAMADAN, R. A.; VASILAKOS, A. V. Brain computer interface: control signals review. **Neurocomputing**, v. 223, p. 26–44, 2017.

RAO, R. P. N. **Brain-Computer Interfacing: An Introduction**. 1. ed. New York: 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>.

RESHMI, G.; AMAL, A. Design of a bci system for piloting a wheelchair using five class mi based eeg. In: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTING AND COMMUNICATIONS (ICACC), 3., 2013, Cochin, India. **Proceedings...** [S.l.]: IEEE, 2013.

REZEIKA, A. *et al.* Brain–computer interface spellers: A review. **Brain Sciences**, v. 8, p. 57, 03 2018.

ROY, S. *et al.* Deep learning based inter-subject continuous decoding of motor imagery for practical brain-computer interfaces. **Frontiers in Neuroscience**, v. 14, p. 918, 2020.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, out. 1986.

SHI, Y. *et al.* Dynamic time-frequency feature extraction for brain activity recognition. **Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference**, United States, v. 2018, p. 3104–3107, jul. 2018.

SIULY, S.; LI, Y.; ZHANG, Y. Electroencephalogram (eeg) and its background. In: **EEG Signal Analysis and Classification: Techniques and Applications**. Cham: Springer International Publishing, 2016. p. 3–21.

SMITH, S. W. **The Scientist and Engineer's Guide to Digital Signal Processing**. 2. ed. San Diego: California Technical Publishing, 1999.

SREEJA, S. R. *et al.* Motor imagery eeg signal processing and classification using machine learning approach. In: INTERNATIONAL CONFERENCE ON NEW TRENDS IN COMPUTING SCIENCES (ICTCS), 2017, Amman, Jordan. **Proceedings...** [S.l.]: IEEE, 2017.

STREHL, U. *et al.* Self-regulation of slow cortical potentials: A new treatment for children with attention-deficit/hyperactivity disorder. **Pediatrics**, American Academy of Pediatrics, v. 118, n. 5, p. e1530–e1540, 2006.

SUBASI, A. **Practical guide for biomedical signals analysis using machine learning techniques : a MATLAB based approach**. 1. ed. London, United Kingdom: Academic Press, an imprint of Elsevier, 2019.

TABAR, Y. R.; HALICI, U. A novel deep learning approach for classification of EEG motor imagery signals. **Journal of Neural Engineering**, IOP Publishing, v. 14, n. 1, p. 016003, nov. 2016.

TAN, C. *et al.* Spatial and spectral features fusion for eeg classification during motor imagery in bci. In: EMBS INTERNATIONAL CONFERENCE ON BIOMEDICAL & HEALTH INFORMATICS (BHI), 2017, Orlando, FL, USA. **Proceedings...** [S.l.]: IEEE, 2017.

TANG, X. *et al.* A novel classification algorithm for mi-eeg based on deep learning. In: JOINT INTERNATIONAL INFORMATION TECHNOLOGY AND ARTIFICIAL INTELLIGENCE CONFERENCE (ITAIC), 8., 2019, Chongqing, China. **Proceedings...** [S.l.]: IEEE, 2019.

TANG, Z.-c. *et al.* Classification of eeg-based single-trial motor imagery tasks using a b-csp method for bci. **Frontiers of Information Technology & Electronic Engineering**, v. 20, n. 8, p. 1087–1098, ago. 2019.

TANGERMANN, M. *et al.* Review of the bci competition iv. **Frontiers in Neuroscience**, v. 6, p. 55, 2012.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern recognition**. 4. ed. Burlington, MA London: Academic Press, 2009.

THOMAS, E.; DYSON, M.; CLERC, M. An analysis of performance evaluation for motor-imagery based BCI. **Journal of Neural Engineering**, IOP Publishing, v. 10, n. 3, p. 031001, maio 2013.

THOMAS*, K. P. *et al.* A new discriminative common spatial pattern method for motor imagery brain–computer interfaces. **IEEE Transactions on Biomedical Engineering**, v. 56, n. 11, p. 2730–2733, 2009.

TU, Y. Machine learning. In: HU, L.; ZHANG, Z. (Ed.). **EEG Signal Processing and Feature Extraction**. Singapore: Springer Singapore, 2019. p. 301–323.

VASILEV, I. *et al.* **Python deep learning: exploring deep learning techniques and neural network architectures with PyTorch, Keras and TensorFlow**. 2. ed. Birmingham: [s.n.], 2019.

VIDAL, J. J. Toward direct brain-computer communication. **Annual Review of Biophysics and Bioengineering**, v. 2, n. 1, p. 157–180, 1973.

WANG, J. *et al.* Toward optimal feature and time segment selection by divergence method for eeg signals classification. **Computers in Biology and Medicine**, v. 97, p. 161–170, 2018.

WANG, X. *et al.* An accurate eegnet-based motor-imagery brain-computer interface for low-power edge computing. In: INTERNATIONAL SYMPOSIUM ON MEDICAL MEASUREMENTS AND APPLICATIONS (MEMEA), 15., 2020, Virtual. **Proceedings...** [S.l.]: arXiv, 2020.

WANG, Z. *et al.* Towards a hybrid bci gaming paradigm based on motor imagery and ssvep. **International Journal of Human–Computer Interaction**, Taylor & Francis, v. 35, n. 3, p. 197–205, 2019.

WOLPAW, J. R. *et al.* Brain–computer interfaces for communication and control. **Clinical Neurophysiology**, v. 113, n. 6, p. 767–791, 2002.

XIA, X.; HU, L. Eeg: Neural basis and measurement. In: HU, L.; ZHANG, Z. (Ed.). **EEG Signal Processing and Feature Extraction**. Singapore: Springer, 2019. p. 7–21.

Zhang, D. *et al.* Motor imagery classification via temporal attention cues of graph embedded eeg signals. **IEEE Journal of Biomedical and Health Informatics**, v. 24, n. 9, p. 2570–2579, 2020.

ZHANG, D. *et al.* Cascade and parallel convolutional recurrent neural networks on eeg-based intention recognition for brain computer interface. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-18), 32., 2018, New Orleans, USA. **Proceedings...** [S.l.]: AAAI Press, 2018.

ZHANG, X. *et al.* A survey on deep learning based brain computer interface: Recent advances and new frontiers. **CoRR**, abs/1905.04149, 2019. Disponível em: <http://arxiv.org/abs/1905.04149>.

____. Multi-person brain activity recognition via comprehensive eeg signal analysis. In: EAI INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS SYSTEMS: COMPUTING, NETWORKING AND SERVICES, 14., 2017, Melbourne, VIC, Australia. **Proceedings...** [S.l.]: arXiv, 2017.

ZHANG, Z. Spectral and time-frequency analysis. In: HU, L.; ZHANG, Z. (Ed.). **EEG Signal Processing and Feature Extraction**. Singapore: Springer Singapore, 2019. p. 89–116.

ANEXO A – CÓDIGO FONTE

SEGMENTAÇÃO DOS SINAIS DE EEG DA BASE DE DADOS DA PHYSIONET

```
import os
import shutil
import pickle
import re
from itertools import groupby
from operator import itemgetter

import numpy as np
import tensorflow as tf
from tensorflow.core.example.example_pb2 import Example
from tensorflow.core.example.feature_pb2 import Features, Feature, BytesList, Int64List

from .EdfFile import EdfFile


def generate_dataset_physionet(dataset_root_dir, shape, window_size=10, start=0, offset=5,
                               noise_samples=0, classes=None, damaged_subjects=None):
    if classes is None:
        classes = ["left-fist", "right-fist"]
    if damaged_subjects is None:
        damaged_subjects = ["S088", "S089", "S092", "S100", "S104"]
    if type(shape) not in (tuple, list):
        shape = [shape]

    window_folder_name = _get_window_folder_name(window_size, start, offset, noise_samples)
    dataset_dir = os.path.join(dataset_root_dir, "normalized-by-sample",
                               window_folder_name,
                               f"{window_size}x{'x'.join(str(dim) for dim in shape)}",
                               "-".join(classes))
    if os.path.exists(path_dir):
        shutil.rmtree(path_dir)
    os.makedirs(path_dir)

    executions = []
    if any([c in classes for c in ["eyes-closed"]]):
        executions.append("R02")
    if any([c in classes for c in ["left-fist", "right-fist"]]):
        for execution in ["R04", "R08", "R12"]:
            executions.append(execution)
    if any([c in classes for c in ["both-fists", "both-feet"]]):
        for execution in ["R06", "R10", "R14"]:
            executions.append(execution)

    label_value = 0
    labels = {}
    if "eyes-closed" in classes:
        labels["eyes-closed"] = label_value
        label_value += 1
    if "left-fist" in classes:
        labels["left-fist"] = label_value
```

```
    label_value += 1
if "right-fist" in classes:
    labels["right-fist"] = label_value
    label_value += 1
if "both-fists" in classes:
    labels["both-fists"] = label_value
    label_value += 1
if "both-feet" in classes:
    labels["both-feet"] = label_value
    label_value += 1

regex_executions = f"({'|'.join(executions)})"
info = {
    "n_samples_by_subject": 0
}
dataset_edf_files_dir = os.path.join(dataset_root_dir, "raw-edf-files")
subjects = filter(lambda s: s not in damaged_subjects,
    sorted(os.listdir(dataset_edf_files_dir)))
for subject in filter(lambda f: re.match("S(\\d+)", f), subjects):

    X_segments = np.empty((0, 64))
    y = np.empty(0, dtype=np.int64)

    edf_subject_path_dir = os.path.join(dataset_edf_files_dir, subject)
    edf_file_names = sorted(os.listdir(edf_subject_path_dir))
    regex_subject_edf_files = f"^{subject}{regex_executions}\\.edf$"
    for edf_file_name in filter(lambda f:
        re.match(regex_subject_edf_files, f), edf_file_names):

        edf_file = EdfFile(edf_subject_path_dir, edf_file_name)
        events_windows = [next(group) for key, group in groupby(
            enumerate(edf_file.labels), key=itemgetter(1))]
        n_events = len(events_windows)
        for index, (event_start_index, event) in enumerate(events_windows):
            if event not in classes:
                continue

            event_start_index += start
            X = edf_file.data[event_start_index:] if index + 1 == n_events \
                else edf_file.data[event_start_index:events_windows[index + 1][0]]
            n_segments = 0
            for (start_segment, end_segment) in _windows(X, window_size, offset):
                x_segment = X[start_segment:end_segment]
                X_segments = np.vstack((X_segments, x_segment))
                y = np.append(y, labels[event])
                for _ in range(noise_samples):
                    noise = np.random.normal(0, 1, x_segment.shape)
                    X_segments = np.vstack((X_segments, x_segment + noise))
                    y = np.append(y, labels[event])
                n_segments += 1

        edf_file.close()

    if len(y) > info["n_samples_by_subject"]:
        info["n_samples_by_subject"] = len(y)

    X_segments = X_segments.reshape((-1, window_size, 64))

    tfrecord_subject_filepath = os.path.join(dataset_dir, f"{subject}.tfrecord")
    options = tf.io.TFRecordOptions(compression_type="GZIP")
```

```
        with tf.io.TFRecordWriter(tfrecord_subject_filepath, options) as writer:
            for n_segment in range(len(y)):
                X_segment = np.array(list(map(
                    lambda x: _process_record(x, shape),
                    X_segments[n_segment])))
                X_segment = tf.io.serialize_tensor(X_segment).numpy()
                X_feature = Feature(bytes_list=BytesList(value=[X_segment]))
                y_feature = Feature(int64_list=Int64List(value=[y[n_segment]]))

                eeg_example = Example(
                    features=Features(
                        feature={
                            "X": X_feature,
                            "y": y_feature
                        }
                    )
                )
                writer.write(eeg_example.SerializeToString())

    info_filepath = os.path.join(dataset_dir, "info.pkl")
    with open(info_filepath, "wb") as fp:
        pickle.dump(info, fp, protocol=4)


def _get_window_folder_name(window_size, start, offset, noise_samples):
    folder_name = f"window-{window_size}"
    if start > 0:
        folder_name += f"-start-{start}"
    elif start < 0:
        folder_name += f"-start-minus-{np.abs(start)}"
    if offset > 0:
        folder_name += f"-offset-{offset}"
    if noise_samples > 0:
        folder_name += f"-times-{noise_samples}-noise-samples"
    return folder_name


def _windows(data, size, offset):
    start = 0
    while (start + size) <= len(data):
        yield int(start), int(start + size)
        if offset > 0:
            start += offset
        else:
            start += len(data)


def _process_record(x, shape):
    x = (x - np.mean(x)) / np.std(x)
    if shape == (10, 11):
        return _map_to_10x11(x)
    elif shape == (7, 9):
        return _map_to_7x9(x)
    elif shape == [64]:
        return x
    raise AttributeError("Unexpected value for parameter 'shape'.")


def _map_to_10x11(x):
    X = np.zeros([10, 11])
```

```
    X[0] = (0, 0, 0, 0, x[21], x[22], x[23], 0, 0, 0, 0)
    X[1] = (0, 0, 0, x[24], x[25], x[26], x[27], x[28], 0, 0, 0)
    X[2] = (0, x[29], x[30], x[31], x[32], x[33], x[34], x[35], x[36], x[37], 0)
    X[3] = (0, x[38], x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[39], 0)
    X[4] = (x[42], x[40], x[7], x[8], x[9], x[10], x[11], x[12], x[13], x[41], x[43])
    X[5] = (0, x[44], x[14], x[15], x[16], x[17], x[18], x[19], x[20], x[45], 0)
    X[6] = (0, x[46], x[47], x[48], x[49], x[50], x[51], x[52], x[53], x[54], 0)
    X[7] = (0, 0, 0, x[55], x[56], x[57], x[58], x[59], 0, 0, 0)
    X[8] = (0, 0, 0, 0, x[60], x[61], x[62], 0, 0, 0, 0)
    X[9] = (0, 0, 0, 0, 0, x[63], 0, 0, 0, 0, 0)

    return X


def _map_to_7x9(x):
    X = np.zeros([7, 9])

    X[0] = (x[24], x[21], x[25], x[22], x[26], x[23], x[27], x[28], 0)
    X[1] = (x[29], x[30], x[31], x[32], x[33], x[34], x[35], x[36], x[37])
    X[2] = (x[38], x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[39])
    X[3] = (x[40], x[7], x[8], x[9], x[10], x[11], x[12], x[13], x[41])
    X[4] = (x[44], x[14], x[15], x[16], x[17], x[18], x[19], x[20], x[45])
    X[5] = (x[46], x[47], x[48], x[49], x[50], x[51], x[52], x[53], x[54])
    X[6] = (x[55], x[60], x[56], x[63], x[61], x[57], x[62], x[58], x[59])

    return X
```

SEGMENTAÇÃO DOS SINAIS DE EEG DAS BASES DE DADOS DA IV COMPETIÇÃO DE INTERFACES CÉREBRO-MÁQUINA

```
import os
import shutil
import pickle
import re

import mne
import numpy as np
import tensorflow as tf
from scipy import io
from tensorflow.core.example.example_pb2 import Example
from tensorflow.core.example.feature_pb2 import Features, Feature, BytesList, Int64List
from .LabelsCounts import LabelsCounts

GDF_FILES_REGEX = "(.)(\\d{2})(\\d{0,2})([ET])\\.gdf"


def generate_dataset_bciiv(dataset_root_dir, events_enum, labels_enum,
                           labels, classes, window_size):
    dataset_dir = os.path.join(dataset_root_dir, "normalized-by-sample",
                               f"window-{window_size}",
                               "-".join(classes).lower())
    if os.path.exists(path_dir):
        shutil.rmtree(path_dir)
    os.makedirs(path_dir)

    info = {
        "n_samples_by_file": 0
```

```
}

subjects_labels_counts = LabelsCounts()
subjects_session_labels_counts = LabelsCounts()
gdf_files_dir = os.path.join(dataset_root_dir, "gdf-files")
gdf_file_names = filter(lambda f: re.match(".*.gdf", f),
    sorted(os.listdir(gdf_files_dir)))
for gdf_file_name in gdf_file_names:
    gdf_filepath = os.path.join(gdf_files_dir, gdf_file_name)
    gdf_file = mne.io.read_raw_gdf(gdf_filepath, preload=True)
    groups_gdf_file = re.match(GDF_FILES_REGEX, gdf_file_name).groups()
    database_prefix = groups_gdf_file[0]
    subject = groups_gdf_file[1]
    session = groups_gdf_file[2]
    session_type = groups_gdf_file[3]

    file_name_pattern = f"{database_prefix}{subject}{session}{session_type}"
    labels_filepath = os.path.join(gdf_files_dir, "labels", f"{file_name_pattern}.mat")
    labels_file = io.loadmat(labels_filepath)["classlabel"]

    annotations = gdf_file.annotations.description
    onset_annotations = gdf_file.annotations.onset
    start_trials_indexes = [event_index for event_index in range(len(annotations))
                            if events_enum.get(annotations[event_index]) == "NEW_TRIAL"]

    indexes_channels_eeg = [index for index, _ in enumerate(
        filter(lambda ch: "EEG" in ch, gdf_file.ch_names))]
    n_channels = len(indexes_channels_eeg)
    frequency = int(gdf_file.info["sfreq"])
    # should consider the complete motor imagery period (4s),
    # not only the cue exhibition period (1.25s)
    duration_event = window_size // frequency
    n_samples = frequency * duration_event

    rejected_trials = 0
    ignored_trials = 0
    X = np.empty((0, n_channels))
    y = np.empty(0, dtype=np.int64)
    for n_trial, event_start_trial_index in enumerate(start_trials_indexes):
        cue_event_index = event_start_trial_index + 1
        onset_event = onset_annotations[cue_event_index]
        onset_index = int(np.ceil(onset_event * frequency))
        end_index = int(np.ceil((onset_event + duration_event) * frequency))
        event_samples = end_index - onset_index
        if event_samples != n_samples:
            end_index += n_samples - event_samples

        # The event correspondent to the trial is the following the start trial event
        if events_enum[annotations[cue_event_index]] == "REJECTED_TRIAL":
            rejected_trials += 1
            continue

        label = labels_enum[labels_file[n_trial][0]]
        if label not in classes:
            ignored_trials += 1
            continue

        # The index 0 returns the data array of the gdf_file
        # The index 1 returns the times array of the gdf_file
        x = gdf_file[indexes_channels_eeg, onset_index:end_index][0].T
```

```
            x = (x - np.mean(x)) / np.std(x)
            X = np.vstack((X, x))

            y = np.append(y, labels[label])

        gdf_file.close()
        X = X.reshape((-1, n_samples, n_channels))

        tfrecord_filepath = os.path.join(dataset_dir, f"{file_name_pattern}.tfrecord")
        options = tf.io.TFRecordOptions(compression_type="GZIP")
        with tf.io.TFRecordWriter(tfrecord_filepath, options) as writer:
            for n_segment in range(len(y)):
                X_segment = tf.io.serialize_tensor(X[n_segment]).numpy()
                X_feature = Feature(bytes_list=BytesList(value=[X_segment]))
                y_feature = Feature(int64_list=Int64List(value=[y[n_segment]]))
                eeg_example = Example(
                    features=Features(
                        feature={
                            "X": X_feature,
                            "y": y_feature
                        }
                    )
                )
                writer.write(eeg_example.SerializeToString())

        valid_trials = len(y)
        if valid_trials > info["n_samples_by_file"]:
            info["n_samples_by_file"] = valid_trials

        labels_counts = np.unique(y, return_counts=True)[1]
        subjects_labels_counts.put(subject, labels_counts)
        subjects_session_labels_counts.put(subject + session_type, labels_counts)

    info_filepath = os.path.join(dataset_dir, "info.pkl")
    with open(info_filepath, "wb") as fp:
        pickle.dump(info, fp, protocol=4)
```

LEITURA DA BASE DE DADOS NO FORMATO ESPERADO PELO ALGORITMO DE PREDIÇÃO

```
import os
import numpy as np
import tensorflow as tf


def load_dataset(dataset_dir, subjects, n_parallel_calls=tf.data.experimental.AUTOTUNE,
                 n_buffer_shuffle=None, batch_size=100, **kwargs):
    """
        The reason to expand_dim parameter is because TensorFlow expects a certain input shape
        for it's Deep Learning Model. For example a Convolution Neural Network expect:

        (<number of samples>, <x_dim sample>, <y_dim sample>, <number of channels>)
    """
    path_files = [os.path.join(dataset_dir, subject)
                  for subject in subjects]
    dataset = tf.data.Dataset.list_files(path_files)
    n_cycle_length = 1
    dataset = dataset.interleave(
```

```
        lambda filepath: tf.data.TFRecordDataset(filepath, compression_type="GZIP"),
        cycle_length=n_cycle_length)
    if n_buffer_shuffle is not None:
      dataset = dataset.shuffle(n_buffer_shuffle)
    dataset = dataset.map(lambda r: _preprocess(r, **kwargs),
                          num_parallel_calls=n_parallel_calls)
    dataset = dataset.batch(batch_size)
    return dataset.prefetch(tf.data.experimental.AUTOTUNE)


@tf.function
def _preprocess(serialized_eeg_records, shape, expand_dim=True):
    """
        The reason to expand_dim parameter is because TensorFlow expects a certain input shape
        for it's Deep Learning Model. For example a Convolution Neural Network expect:

        (<number of samples>, <x_dim sample>, <y_dim sample>, <number of channels>)
    """
    feature_description = {
        "X": tf.io.FixedLenFeature([], tf.string),
        "y": tf.io.FixedLenFeature([], tf.int64)
    }
    parsed_eeg_records = tf.io.parse_single_example(
        serialized_eeg_records, feature_description)
    X = parsed_eeg_records["X"]
    X = tf.io.parse_tensor(X, out_type=tf.float64)
    X.set_shape(shape)
    y = parsed_eeg_records["y"]

    if expand_dim:
        X = X[..., np.newaxis]

    return X, y
```

ARQUITETURA CRNN

```
from tensorflow import keras


def create_model(shape=(10, 10, 11, 1), output_classes=5):
    model = keras.models.Sequential([
        keras.layers.Input(shape=shape),
        keras.layers.TimeDistributed(keras.layers.Conv2D(32, (3, 3),
        activation="elu", padding="SAME")),
        keras.layers.TimeDistributed(keras.layers.Conv2D(64, (3, 3),
        activation="elu", padding="SAME")),
        keras.layers.TimeDistributed(keras.layers.Conv2D(128, (3, 3),
        activation="elu", padding="SAME")),
        keras.layers.TimeDistributed(keras.layers.Flatten()),
        keras.layers.TimeDistributed(keras.layers.Dense(1024, activation="elu")),
        keras.layers.TimeDistributed(keras.layers.Dropout(0.5)),
        keras.layers.LSTM(64, return_sequences=True, dropout=0.5),
        keras.layers.LSTM(64, dropout=0.5),
        keras.layers.Dense(1024, activation="elu"),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(output_classes, activation="softmax")
    ])

    optimizer = keras.optimizers.Adam(lr=1e-4)
```

```
    model.compile(loss="sparse_categorical_crossentropy",
        optimizer=optimizer, metrics=["accuracy"])

    return model
```

ARQUITETURA CNN-2D

```
from tensorflow import keras


def create_model(shape=(480, 64, 1), output_classes=5, model):
    model = keras.models.Sequential([
        keras.layers.Input(shape=shape),
        keras.layers.Conv2D(8, (3, 3), activation="elu", padding="SAME"),
        keras.layers.AveragePooling2D(pool_size=(3, 1)),
        keras.layers.Conv2D(16, (3, 3), activation="elu", padding="SAME"),
        keras.layers.AveragePooling2D(pool_size=(3, 1)),
        keras.layers.Conv2D(32, (3, 3), activation="elu", padding="SAME"),
        keras.layers.AveragePooling2D(pool_size=(3, 1)),
        keras.layers.Flatten(),
        keras.layers.Dense(256, activation="elu"),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(output_classes, activation="softmax")
    ])

    optimizer = keras.optimizers.Adam(lr=1e-4)
    model.compile(loss="sparse_categorical_crossentropy",
        optimizer=optimizer, metrics=["accuracy"])

    return model
```

ARQUITETURA CNN-1D

```
from tensorflow import keras


def create_model(shape=(480, 64), output_classes=5, weights_filepath=None):
    model = keras.models.Sequential([
        keras.layers.Input(shape=shape),
        keras.layers.Conv1D(8, 7, activation="elu", padding="SAME"),
        keras.layers.AveragePooling1D(pool_size=3),
        keras.layers.Conv1D(16, 7, activation="elu", padding="SAME"),
        keras.layers.AveragePooling1D(pool_size=3),
        keras.layers.Conv1D(32, 7, activation="elu", padding="SAME"),
        keras.layers.AveragePooling1D(pool_size=3),
        keras.layers.Flatten(),
        keras.layers.Dense(256, activation="elu"),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(output_classes, activation="softmax")
    ])

    if weights_filepath is not None:
        model.load_weights(weights_filepath)

    optimizer = keras.optimizers.Adam(lr=1e-4)
    model.compile(loss="sparse_categorical_crossentropy",
        optimizer=optimizer, metrics=["accuracy"])
```

```
    return model
```

TREINAMENTO COM VALIDAÇÃO CRUZADA 10-FOLD PARA A BASE DE DADOS DA PHYSIONET

```
import os
import tensorflow as tf

from sklearn.model_selection import KFold
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard

from .load_tf_dataset import load_dataset


def train_for_physionet_dataset(model_results_dir, create_model_fn, n_samples_by_subject,
                                n_splits=10, n_epochs=50, expand_dim=False):
    k_folds = KFold(n_splits=n_splits).split(subjects)
    for fold, (train_index, test_index) in enumerate(k_folds):
        train_subjects = subjects[train_index]

        n_buffer_shuffle = n_samples_by_subject * len(train_index)
        train_dataset = load_dataset(train_subjects,
            n_buffer_shuffle=n_buffer_shuffle, expand_dim=expand_dim)

        model_name = f"independent-{n_splits}-cross-validation-fold-{fold}"
        log_dir = os.path.join(model_results_dir, "tensorboard-logs", model_name)
        model_filepath = os.path.join(model_results_dir, f"{model_name}.h5")
        checkpoint_cb = ModelCheckpoint(model_filepath, save_weights_only=True)
        tensorboard_cb = TensorBoard(log_dir)

        model = create_model_fn()
        model.fit(train_dataset, n_epochs=n_epochs,
            callbacks=[checkpoint_cb, tensorboard_cb], verbose=2)

        tf.keras.backend.clear_session()
```

TREINAMENTO COM VALIDAÇÃO LEAVE-ONE-OUT PARA AS BASES DE DADOS DA IV COMPETIÇÃO DE INTERFACES CÉREBRO-MÁQUINA

```
import os
import re
import numpy as np

from .load_tf_dataset import load_dataset


def train_for_bci_iv_iia_dataset(model_results_dir, create_model_fn, n_samples_by_subject,
                                 n_epochs=1500, expand_dim=False):
    n_subjects = 9
    subjects = np.array(range(n_subjects)) + 1
    for test_subject in subjects:
      train_subjects = subjects[subjects != test_subject]

      regex_train_subjects = f"({'|'.join(['0' + str(n_subject)
          for n_subject in train_subjects])})"
```

```
        train_filenames = list(filter(lambda f:
            re.match(f"A({regex_train_subjects})(T|E).tfrecord", f), filenames))

        n_buffer_shuffle = n_samples_by_subject * len(train_filenames)
        train_dataset = load_dataset(train_filenames,
          n_buffer_shuffle=n_buffer_shuffle, expand_dim=expand_dim)

        model_name = f"test-subject-{test_subject}"
        log_dir = os.path.join(model_results_dir, "tensorboard-logs", model_name)
        model_filepath = os.path.join(model_results_dir, f"{model_name}.h5")
        checkpoint_cb = keras.callbacks.ModelCheckpoint(model_filepath, save_weights_only=True)
        tensorboard_cb = keras.callbacks.TensorBoard(log_dir)
        model = create_model_fn()
        model.fit(train_dataset, epochs=n_epochs,
                  callbacks=[checkpoint_cb, tensorboard_cb], verbose=2)

        tf.keras.backend.clear_session()


def train_for_bci_iv_iib_dataset(model_results_dir, create_model_fn, n_samples_by_subject,
                                 n_epochs=1500, expand_dim=False):
    n_subjects = 9
    subjects = np.array(range(n_subjects)) + 1
    for test_subject in subjects:
      train_subjects = subjects[subjects != test_subject]

      regex_train_subjects = f"({'|'.join(['0' + str(n_subject)
        for n_subject in train_subjects])})"
      train_filenames = list(filter(lambda f:
          re.match(f"B({regex_train_subjects})(\d+)(T).tfrecord", f), filenames))

      n_buffer_shuffle = n_samples_by_subject * len(train_filenames)
      train_dataset = load_dataset(train_filenames,
        n_buffer_shuffle=n_buffer_shuffle, expand_dim=expand_dim)

      model_name = f"test-subject-{test_subject}"
      log_dir = os.path.join(model_results_dir, "tensorboard-logs", model_name)
      model_filepath = os.path.join(model_results_dir, f"{model_name}.h5")
      checkpoint_cb = keras.callbacks.ModelCheckpoint(model_filepath, save_weights_only=True)
      tensorboard_cb = keras.callbacks.TensorBoard(log_dir)
      model = create_model_fn()
      model.fit(train_dataset, epochs=n_epochs,
                callbacks=[checkpoint_cb, tensorboard_cb], verbose=2)

      tf.keras.backend.clear_session()
```

AVALIAÇÃO DE DESEMPENHO COM VALIDAÇÃO CRUZADA 10-FOLD PARA A BASE DE DADOS DA PHYSIONET

```
import os
import tensorflow as tf

from sklearn.model_selection import KFold

from .load_tf_dataset import load_dataset


def train_for_physionet_dataset(model_results_dir, create_model_fn,
```

```
                                n_splits=10, expand_dim=False):
    results = []
    kf = KFold(n_splits=n_splits)
    for fold, (train_index, test_index) in enumerate(kf.split(subjects)):
        test_subjects = subjects[test_index]
        test_dataset = load_dataset(test_subjects, batch_size=1, expand_dim=expand_dim)

        model_name = f"independent-{n_splits}-cross-validation-fold-{fold}"
        model_weights_filepath = os.path.join(model_results_dir, f"{model_name}.h5")

        model = create_model_fn(model_weights_filepath)
        loss, accuracy = model.evaluate(test_dataset)

        results.append({
            "fold": fold,
            "test_subjects": test_subjects,
            "loss": loss,
            "accuracy": accuracy
        })

        tf.keras.backend.clear_session()

    return results
```

AVALIAÇÃO DE DESEMPENHO COM VALIDAÇÃO LEAVE-ONE-OUT PARA AS BASES DE DADOS DA IV COMPETIÇÃO DE INTERFACES CÉREBRO-MÁQUINA

```
import os
import re
import numpy as np

from .load_tf_dataset import load_dataset


def evaluate_for_bci_iv_iia_dataset(model_results_dir, create_model_fn, expand_dim=False):
    n_subjects = 9
    subjects = np.array(range(n_subjects)) + 1
    results = []
    for test_subject in subjects:
        test_filenames = list(filter(lambda f:
            re.match(f"A0{test_subject}(T|E).tfrecord", f), filenames))
        test_dataset = load_dataset(test_filenames, batch_size=1, expand_dim=expand_dim)

        model_name = f"test-subject-{test_subject}"
        model_weights_filepath = os.path.join(model_results_dir, f"{model_name}.h5")

        model = create_model_fn(model_weights_filepath)
        loss, accuracy = model.evaluate(test_dataset)

        results.append({
            "test_subject": test_subject,
            "loss": loss,
            "accuracy": accuracy
        })

        tf.keras.backend.clear_session()
```

```
    return results


def evaluate_for_bci_iv_iib_dataset(model_results_dir, create_model_fn, expand_dim=False):
    n_subjects = 9
    subjects = np.array(range(n_subjects)) + 1
    results = []
    for test_subject in subjects:
        test_filenames = list(filter(lambda f:
            re.match(f"B0{test_subject}(\d+)(E).tfrecord", f), filenames))
        test_dataset = load_dataset(test_filenames, batch_size=1, expand_dim=expand_dim)

        model_name = f"test-subject-{test_subject}"
        model_weights_filepath = os.path.join(model_results_dir, f"{model_name}.h5")

        model = create_model_fn(model_weights_filepath)
        loss, accuracy = model.evaluate(test_dataset)

        results.append({
            "test_subject": test_subject,
            "loss": loss,
            "accuracy": accuracy
        })

    tf.keras.backend.clear_session()

    return results
```

TESTE DE SIGNIFICÂNCIA ESTATÍSTICA T DE STUDENT PAREADO COM VALIDAÇÃO CRUZADA

```
import os
import re
import numpy as np

from sklearn.model_selection import KFold

from .load_tf_dataset import load_dataset


def t_test(datasets_dir, models_dir, create_models_fn, models_shape,
           n_splits=10, expand_dim=False):

    dataset_a_dir, dataset_b_dir = datasets_dir
    model_a_dir, model_b_dir = models_dir
    create_model_a_fn, create_model_b_fn = create_models_fn
    shape_model_a, shape_model_b = models_shape

    subjects_dataset_a = _get_subjects(dataset_a_dir)
    subjects_dataset_b = _get_subjects(dataset_b_dir)
    if len(subjects_dataset_a) != len(subjects_dataset_b):
        raise AttributeError("Subjects dataset's does not match!")

    differences = []
    kf = KFold(n_splits=n_splits)
    for fold, (train_index, test_index) in enumerate(kf.split(subjects_dataset_a)):
        model_name = f"independent-{n_splits}-cross-validation-fold-{fold}"
```

```
        model_a_weights_filepath = os.path.join(model_a_dir, f"{model_name}.h5")
        model_b_weights_filepath = os.path.join(model_b_dir, f"{model_name}.h5")

        model_a = create_model_a_fn(model_a_weights_filepath, shape=shape_model_a)
        model_b = create_model_b_fn(model_b_weights_filepath, shape=shape_model_b)

        _, accuracy_a = _evaluate(dataset_a_dir, subjects_dataset_a[test_index],
            model_a, shape_model_a, expand_dim=expand_dim)
        _, accuracy_b = _evaluate(dataset_b_dir, subjects_dataset_b[test_index],
            model_b, shape_model_a, expand_dim=expand_dim)

        differences.append(accuracy_a - accuracy_b)

        tf.keras.backend.clear_session()

    centered_diff = np.array(differences) - np.mean(differences)
    return np.mean(differences) * (n_splits ** .5) / \
        (np.sqrt(np.sum(centered_diff ** 2) / (n_splits - 1)))


def _get_subjects(dataset_dir):
    subjects = sorted(os.listdir(dataset_dir))
    subjects = filter(lambda s: re.match("S(\d+).tfrecord", s), subjects)
    return np.array(list(subjects))


def _evaluate(dataset_dir, test_subjects, model, shape, expand_dim):
    test_dataset = load_dataset(test_subjects, dataset_dir, shape,
        batch_size=1, expand_dim=expand_dim)
    return model.evaluate(test_dataset, verbose=2)
```

ESTRUTURAS DE DADOS UTILITÁRIAS

```
import json
from json import JSONEncoder

import numpy as np


class NumpyArrayEncoder(JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return JSONEncoder.default(self, obj)


class LabelsCounts:
    def __init__(self):
        self.__counts = {}

    def put(self, key, counts):
        if key not in self.__counts:
            self.__counts[key] = counts
        else:
            self.__counts[key] = self.__counts[key] + counts

    def __repr__(self):
        return json.dumps(self.__counts, sort_keys=True, indent=4, cls=NumpyArrayEncoder)
```

```
import os
import re

import pyedflib

import numpy as np

MOVEMENT_SINGLE_MEMBER_RUNS = [3, 4, 7, 8, 11, 12]
MOVEMENT_BOTH_MEMBERS_RUNS = [5, 6, 9, 10, 13, 14]


class EdfFile:
    def __init__(self, edf_path_dir, edf_file_name, channels=None):
        self.edf_path_dir = edf_path_dir
        self.edf_file_name = edf_file_name

        groups_edf_file = re.match("S(\\d+)R(\\d+).edf", edf_file_name).groups()
        self.subject = groups_edf_file[0]
        self.run_execution = groups_edf_file[1]
        self.__file = pyedflib.EdfReader(os.path.join(edf_path_dir, edf_file_name))

        annotations = self.__file.readAnnotations()
        self.__onset_events = annotations[0]
        self.__duration_events = annotations[1]
        self.__events = annotations[2]

        self.frequency = self.__file.getSampleFrequencies()[0]
        total_duration_events = np.round(np.sum(self.__duration_events), decimals=2)
        self.n_samples = int(total_duration_events * self.frequency)
        self.n_channels = self.__file.signals_in_file if channels is None else len(channels)
        self.channels = np.arange(self.n_channels) if channels is None else channels
        self.channels_labels = np.array(self.__file.getSignalLabels())[self.channels]
        self.data, self.labels = self.__read()

    def get_path_file(self, path_dir, extension):
        return os.path.join(path_dir, f"S{self.subject}R{self.run_execution}.{extension}")

    def close(self):
        self.__file.close()

    def __read(self):
        data = np.zeros((self.n_samples, self.n_channels))
        # Set invalid label to verify skipped samples
        labels = np.full(self.n_samples, "invalid", dtype="U256")
        end_index = None
        for index_event in range(len(self.__onset_events)):
            onset_event = self.__onset_events[index_event]
            duration_event = self.__duration_events[index_event]
            event = self.__events[index_event]

            onset_index = int(onset_event * self.frequency)
            if end_index is not None and onset_index != end_index:
                onset_index = end_index
            total_duration_event = np.round(onset_event + duration_event, decimals=2)
            max_event_samples = int(total_duration_event * self.frequency)
            end_index = np.minimum(max_event_samples, self.n_samples)
            event_samples = end_index - onset_index

            for ch_index, ch in enumerate(self.channels):
                data[onset_index:end_index, ch_index] = self.__file.readSignal(
```

```
                ch, onset_index, event_samples)
            labels[onset_index:end_index] = np.repeat(self.__get_label_for_event(
                event, int(self.run_execution)), event_samples)

        if np.sum(labels == "invalid") > 0:
            print("WARNING: Samples skipped when reading the file " + self.edf_file_name)

        return data, labels

    """
        The events of real movements are not handled because these files are not read
    """
    @staticmethod
    def __get_label_for_event(event, run_execution):
        if event == "T0":
            if run_execution == 1:
                return "eyes-open"
            elif run_execution == 2:
                return "eyes-closed"
            else:
                return "rest"

        if event == "T1":
            if run_execution in MOVEMENT_SINGLE_MEMBER_RUNS:
                return "left-fist"
            elif run_execution in MOVEMENT_BOTH_MEMBERS_RUNS:
                return "both-fists"

        if event == "T2":
            if run_execution in MOVEMENT_SINGLE_MEMBER_RUNS:
                return "right-fist"
            elif run_execution in MOVEMENT_BOTH_MEMBERS_RUNS:
                return "both-feet"

        return None
```