



COORDENADORIA DE ENGENHARIA DA COMPUTAÇÃO

**FABRICIO TORQUATO LEITE
LUCAS DE OLIVEIRA DUTRA**

**SISTEMA DE CONTROLE BASEADO EM INTERFACE
CÉREBRO-MÁQUINA**

SOROCABA/SP

2019

**Fabricio Torquato Leite
Lucas de Oliveira Dutra**

**SISTEMA DE CONTROLE BASEADO EM INTERFACE
CÉREBRO-MÁQUINA**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Facens, como exigência parcial para obtenção do diploma de graduação em Engenharia de Computação.

Orientador: Prof. Me. André Breda Carneiro

Coorientador: Prof. Me. Johannes Von Lohchter

**SOROCABA/SP
2019**

**FICHA CATALOGRÁFICA ELABORADA PELA
“BIBLIOTECA FACENS”**

L533s

Leite, Fabricio Torquato.

Sistema de controle baseado em interface cérebro-máquina /
por Fabricio Torquato Leite, Lucas de Oliveira Dutra. – Sorocaba,
SP: [s.n.], 2019.

78f.

Trabalho de Conclusão de Curso (Graduação) – Centro
Universitário Facens – Curso de Engenharia de Computação,
2019.

Orientador: Prof.(a) Me.(a) André Breda Carneiro

Coorientador: Prof.(a) Me.(a) Johannes Von Lochter

1. Aprendizado de maquina. 2. Interface cérebro maquina. 3.
Robot Operation System. I. Dutra, Lucas de Oliveira. II. Centro
Universitário Facens. III. Título.

CDD 621.39

**Fabricio Torquato Leite
Lucas de Oliveira Dutra**

**SISTEMA DE CONTROLE BASEADO EM INTERFACE
CÉREBRO-MÁQUINA**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Facens, como exigência parcial para obtenção do diploma de graduação em Engenharia de Computação.

Orientador: Prof. Me. André Breda Carneiro

Coorientador: Prof. Me. Johannes Von Lochter

Sorocaba, 04 de Novembro de 2019.

BANCA EXAMINADORA

Prof. Me. André Breda Carneiro

Prof. Me. Johannes Von Lochter

Prof. Dr. Glauco Todesco

Este trabalho é dedicado especialmente para as pessoas que possuem alguma complicação nos movimentos motores.

AGRADECIMENTOS

A Deus por ter me dado saúde, família, amigos e força para superar as dificuldades. A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes. Ao meu orientador Me. André Breda Carneiro e coorientador Me. Johannes Von Lochter, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos. Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos. Ao meu pai, meu irmão, minha madrinha e minha namorada que apesar de todas as dificuldades me fortaleceu e que para mim foi muito importante. Em memória agradeço a minha mãe Rosalina Bueno da Silva Leite, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo, cansaço e que me ajudou em toda minha carreira acadêmica. Meus agradecimentos aos amigos, companheiros de trabalhos e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida com certeza. E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Fabricio Torquato Leite

AGRADECIMENTOS

A Deus que me nutriu de forças, perseverança e saúde. À minha mãe, Silmara Moysés de Oliveira e ao meu pai, Joemir Dutra Júnior, por serem meu alicerce nessa vida. Ao meu orientador Me. André Breda Carneiro e o coorientador Me. Johannes Von Lochter pelo suporte e valiosos conselhos dados para a realização deste projeto. Aos meus amigos que sempre estiveram comigo. À minha namorada pela paciência e apoio. A todos os professores que fizeram parte da minha formação, desde o Fundamental até a Universidade, pois foram essenciais no meu desenvolvimento humano e acadêmico.

Lucas de Oliveira Dutra

*“Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)*

LISTA DE FIGURAS

Figura 2.1 – Sistema BCI baseados em realimentação incorporada e não incorporada	19
Figura 2.2 – Diagrama básico de uma BCI	20
Figura 2.3 – O neurônio e a sinapse	22
Figura 2.4 – Representação da montagem de eletrodos 10–20	23
Figura 2.5 – Representação do posicionamento dos eletrodos de acordo com o Sistema Internacional 10-20, com eletrodos de referência biauriculares.	23
Figura 2.6 – Representação do homúnculo cortical, relacionando com as partes do corpo	25
Figura 3.1 – Modelo computacional de um neurônio	27
Figura 3.2 – Modelo de arquitetura da <i>multilayer perceptron</i>	30
Figura 3.3 – Modelo do neurônio	31
Figura 3.4 – MLP com <i>backpropagation</i>	33
Figura 3.5 – Exemplo de aplicação da convolução em uma matriz de valores. À esquerda, uma matriz hipotética representada por um único canal com dimensões 5x5 que recebe a aplicação de um filtro 3x3. À direita, uma matriz ilustrando o somatório que fornece o resultado da convolução.	35
Figura 3.6 – Aplicação de <i>max pooling</i> em uma imagem 4x4 utilizando um filtro 2x2.	36
Figura 3.7 – Arquitetura tradicional das Redes Neurais Convolucionais.	36
Figura 3.8 – Representação geral de uma RNN (BISHOP, 2006)	38
Figura 3.9 – Representação da entrada no tempo de uma RNN (BISPO, 2018)	39
Figura 3.10 – Estrutura de uma célula LSTM (BISPO, 2018)	41
Figura 3.11 – LSTM: <i>Forget Gate</i> (BISPO, 2018)	42
Figura 3.12 – LSTM: <i>Input Gate</i> (BISPO, 2018)	42
Figura 3.13 – LSTM: <i>Output Gate</i> (BISPO, 2018)	43
Figura 4.1 – <i>EMOTIV EPOC+ 14 Channel</i> (EMOTIV, 2019)	46
Figura 4.2 – Configurações dos amplificadores usados para aquisição de EEG, sendo a) bipolar, b) monopolar, c) bipolar unida e d) unipolar com referência média. (CANTARELLI; JÚNIOR; JR, 2016)	47
Figura 4.3 – Demonstração da conversão dos dados brutos na matriz 30x14	50
Figura 4.4 – Demonstração da conversão da matriz 30x14 para matriz 14x4x2	50
Figura 4.5 – Técnica de sequências de 32 elementos com 128 valores	51

Figura 5.1 – Mão em formato de pinça	53
Figura 5.2 – Palma da mão para cima e para baixo	53
Figura 5.3 – Apertando bola estimuladora com polegar para cima rotacionando para dentro	54
Figura 5.4 – Estalando os dedos	54
Figura 5.5 – Apertando a bola estimuladora	55
Figura 5.6 – Abrindo e fechando a mão.	55
Figura 5.7 – Direções mostradas durante os ciclos de aquisição.	56
Figura 5.8 – Espaço físico da realização do experimento.	57
Figura 5.9 – Interface Inicial da Simulação	58
Figura 5.10 – Simulação sendo mostrada na TV	58
Figura 5.11 – Simulação finalizada com sucesso.	59
Figura 5.12 – Locais onde a segunda tartaruga aparece	59
 Figura 6.1 – Trajetória percorrida pelo quarto voluntario na simulação do CNN	65
Figura 6.2 – Trajetória percorrida pelo segundo voluntario na simulação do RNN	65
Figura 6.3 – Percurso percorrido pelos participantes na primeira rodada. . . .	68
Figura 6.4 – Percurso percorrido utilizando o controle independente na primeira rodada	68
Figura 6.5 – Percurso percorrido pelos participantes na primeira terceira. . . .	69
Figura 6.6 – Percurso percorrido utilizando o controle independente na terceira rodada	69

LISTA DE TABELAS

Tabela 1 – Classificação das ondas em bandas de frequências	24
Tabela 2 – Resultados da simulação utilizando a MLP	63
Tabela 3 – Resultados da simulação utilizando a CNN	63
Tabela 4 – Resultados da simulação utilizando a RNN	63
Tabela 5 – Tempo de realização das simulação com o MLP	64
Tabela 6 – Tempo de realização das simulação com o CNN	64
Tabela 7 – Tempo de realização das simulação com o RNN	64
Tabela 8 – Resultados do sistema de controle independente	67

LISTA DE SIGLAS E ABREVIATURAS

A	Auricular
AM	Aprendizado de Máquina
ANN	Artificial Neural Network
BCI	Brain Computer Interface
C	Central
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
EEG	Eletroencefalograma
FP	Pré-frontal
F	Frontal
GPU	graphics Processing Unit
IBGE	Instituto Brasileiro de Geografia e Estatística
ICM	Interface Cérebro-Máquina
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MV	Movimento Voluntário
O	Occipital
P	Parietal
PCD	Pessoas Com Deficiência
PNS	Pesquisa Nacional de Saúde
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROS	Robot Operating System

SOM	Self-Organized Map
SSTb	Stanford Sentiment Treebank
SVM	Support-Vector Machine
T	Temporal
TA	Tecnologia Assistiva

SUMÁRIO

1	INTRODUÇÃO	16
2	TECNOLOGIAS ASSISTIVAS	18
2.1	Interface Cérebro Máquina	19
2.1.1	Trabalhos relacionados	20
2.2	Eletroencefalograma	21
2.2.1	Bases biológicas	21
2.2.2	Sistema 10-20	22
2.3	Bandas de Frequência	24
2.4	Movimento Voluntário e Movimento Imaginário	24
3	APRENDIZADO DE MÁQUINA	26
3.1	Aprendizado Supervisionado e Classificação de Padrões	26
3.2	Rede Neural Artificial	27
3.3	Perceptron de Múltiplas Camadas	30
3.3.1	<i>Backpropagation</i>	31
3.4	Rede Neural Convolucional	33
3.4.1	Mapa de convolução	34
3.5	Rede Neural Recorrente	37
3.5.1	<i>Backpropagation through time</i>	39
3.5.2	<i>Long short-term memory</i>	41
3.6	Arquiteturas e Modelos	43
4	AQUISIÇÃO E PREPARAÇÃO DOS DADOS	46
4.1	Aquisição de Dados	46
4.2	Pré-Processamento dos Dados	48
4.2.1	Transformada de Fourier	49
4.3	Extração de Características	49
4.3.1	Arranjo experimental	50
5	DESENVOLVIMENTO	52
5.1	Treinamento Experimental	52
5.2	Teste Experimental	57
6	RESULTADOS E DISCUSSÕES	63
7	CONCLUSÃO	71
8	REFERÊNCIAS	72

RESUMO

O avanço tecnológico nos disponibilizou novas possibilidades e horizontes para serem estudados entre elas a leitura de impulsos cerebrais. Este trabalho avalia diferentes arquiteturas de redes neurais para realizar o controle de um objeto virtual construído a partir do *Robot Operating System* utilizando o eletroencefalograma como sistema de aquisição. Para as funções de controle da interface foram utilizadas ações motoras voluntárias das mãos, onde cada mão indicava uma direção. Este trabalho contempla os métodos de tratamento aplicados nos dados coletados pelo eletroencefalograma, junto ao protocolo para criação do experimento. Mesmo a CNN e a RNN apresentando bons resultados, os *feedbacks* coletados dos voluntários do experimento permitiu compreender que existe vantagens em arquiteturas RNN, sendo elas uma melhor resposta aos comandos do usuário, isso quando os mesmos não estão sob interferência emocional, e uma melhor adaptação do indivíduo ao melhorar o resultado após diferentes tentativas sem precisar do retreinamento da rede.

Palavras-chaves: Aprendizado de máquina, Interface cérebro maquina, Robot Operating System.

ABSTRACT

Technological advancement has provided us with new possibilities and horizons to study the reading of brain impulses. This work evaluates different neural network architectures to control a virtual object built from Robot Operating System using the electroencephalogram as acquisition system. For the interface control functions, voluntary hand motor actions were used, where each hand indicated a direction. This work contemplates the treatment methods applied to the data collected by the electroencephalogram, together with the protocol for the creation of the experiment. Even though CNN and RNN showed good results, the feedbacks collected from the experiment volunteers allowed us to understand that there are advantages to using RNN architecture , which is a better response to user commands, when they are not under emotional interference, and a better adaptation of the individual by improving the result after different attempts without retraining the net.

Key-words: Machine learning, Brain computer interface, Robot Operating System.

1 INTRODUÇÃO

A tecnologia está em constante evolução, disponibilizando novas possibilidades e horizontes para serem estudados. De modo, que os avanços na ciência permitiram a existência de energia elétrica dentro das residências, a comunicação entre pessoas independente da distância e, também, a identificação das doenças de forma rápida e precisa.

Ao observar os estudos de novas tecnologias, uma das áreas que obteve peculiar progresso foi a inclusão social. Segundo os dados do Instituto Brasileiro de Geografia e Estatística (IBGE, 2013), 6,9% dos brasileiros alegaram ter alguma deficiência; a Pesquisa Nacional de Saúde (PNS) analisou deficiência visual, auditiva, motora e mental/intelectual.

Existem diversas tecnologias para auxiliar pessoas com alguma deficiência, como as próteses mecânicas/biônicas, os aplicativos que leem os textos dos computadores e aparelhos auditivos (LEITE et al., 2019). Conforme as causas sociais ganham força ao redor do mundo, e novos investimentos são direcionados para a inovação e aplicação de tecnologias voltadas para essa área, torna-se fundamental que pesquisas e desenvolvimento sejam feitos para beneficiar, cada vez mais, esse público.

Segundo Carlos Roberto França, a ocorrência de deficiências motoras no Brasil é considerável, pois dos 6,9% dos brasileiros da pesquisa do IBGE, 1,3% dos participantes da pesquisa têm alguma deficiência física, além disso 46,8% deste total tem grau intenso ou muito intenso de limitações, tornando-se o maior grupo da pesquisa (FRANÇA; BORGES; SAMPAIO, 2005).

Um equipamento que pode ser utilizado nessas pesquisas é o eletroencefalograma (EEG), um sensor de monitoramento não invasivo que analisa a atividade elétrica cerebral espontânea, geralmente utilizado para detectar doenças psiquiátricas e neurológicas como epilepsia, lesões cerebrais, tumores cerebrais, está sendo um dos equipamentos que mais possuem resultados em pesquisas com interface cérebro-máquina ou do inglês *brain computer interface* (BCI).

Este trabalho tem como objetivo utilizar o EEG com estudos na área de Interface Cérebro-Máquina (ICM) para criar um sistema de controle. Será analisado as atividades cerebrais do usuário para identificar comandos de ações motoras voluntárias como movimentos motores dos braços esquerdo e direito tais ações serão correspondente à rotacionar para esquerda ou para direita, desse modo para o sistema comandos diferentes serão interpretados como seguir em frente.

O estudo para identificação de qual comando o usuário está realizando é di-

vidido em duas partes sendo a primeira o pré-processamento dos dados onde será aplicado técnicas de processamento de sinais como o filtro de passa banda e a transformada de Fourier. E a segunda parte propõe quatro soluções baseadas em diferentes técnicas de aprendizagem de máquina responsáveis por gerar diferentes tipos de modelos preditivos.

Os resultados deste trabalho compreende, em uma avaliação comparativa entre as soluções propostas sob um mesmo protocolo de treinamento utilizando testes de controle. Com isso este projeto é dedicado a contribuir com a qualidade de vida de pessoas que possuem alguma complicação nos movimentos motores, isso pelo fato da possibilidade da reutilização do princípio de movimentação do sistema desenvolvido para outros meios de interação motora.

2 TECNOLOGIAS ASSISTIVAS

Tecnologia Assistiva (TA) é um termo utilizado para identificar todas as ferramentas e recursos que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência e, consequentemente, promover a vida independente e com inclusão (BERSCH; TONOLLI, 2006).

Para entender o propósito de estudos na área de TA, Dahlhausen e Radabaugh abordam o seguinte conceito: "Para as pessoas sem deficiência a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis" (DAHLHAUSEN; RADABAUGH, 1993).

A TA deve ser interpretada como um meio para promover a ampliação de uma funcionalidade, de modo que as tecnologias possibilitem a recuperação, de forma alternativa, de funções que se encontram limitadas devido a alguma deficiência.

Pode-se dizer que o objetivo maior da TA é proporcionar maior independência, qualidade de vida e inclusão social às pessoas com deficiência , através da ampliação de sua comunicação, mobilidade, controle de seu ambiente, habilidades de seu aprendizado e trabalho (BERSCH, 2008).

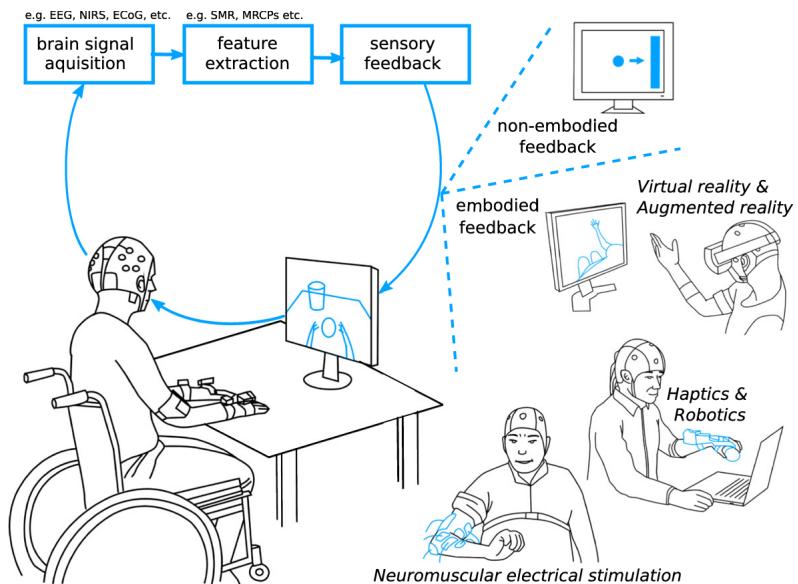
Por existirem diversas áreas de atuação da TA, como menciona Rita Bersch (2008), necessário classificar e dividir essas áreas em categorias de estudo. Desse modo, diversas classificações de TA foram desenvolvidas com finalidades distintas como exemplo temos a ISO 9999/2002, ou a classificação proposta pelo Sistema Nacional de Classificação dos Recursos e Serviços de TA, dos Estados Unidos, ou a classificação HEART entre outros (BERSCH, 2008). Entretanto a classificação que esse trabalho aborda foi escrita, em 1998, por José Tonolli e Rita Bersch.

Segundo José Tonolli e Rita Bersch uma das categorias do ramo da TA é a de sistemas de controle de ambiente que é descrito como um sistema de controle voltado para pessoas com limitações motoras, possibilitando, assim, que os usuários liguem, desliguem e ajustem aparelhos eletroeletrônicos como a luz, televisores, computadores, próteses ou até meios de locomoção (BERSCH, 2008).

Dentro dessa área de classificação, surgiu um novo conceito que une estudos da Neurociência com a Computação, a chamada Interface Cérebro-Máquina: uma área que busca utilizar as tecnologias ascendentes como Inteligência Artificial, Visão Computacional ou Computação Gráfica para construir soluções de inovação contribuindo, assim, com a qualidade de vida de pessoas com deficiência. A Figura 2.1 mostra a utilização de uma Interface Cérebro-Máquina em conjunto com as tecnologias citadas anteriormente, as etapas de uma BCI serão descritas nos próximos

capítulos.

Figura 2.1 – Sistema BCI baseados em realimentação incorporada e não incorporada



Fonte: (CERVERA et al., 2018)

2.1 Interface Cérebro Máquina

Utilizando a evolução tecnológica da Computação como base de sustentação, a Neurociência conseguiu ter um avanço rápido em sua área de estudo. Um exemplo de trabalho nesse campo é a percepção de que cada área do cérebro humano é responsável por executar diferentes funções do corpo, tais como: motricidade, afetividade e cognição (SOUZA et al., 2015).

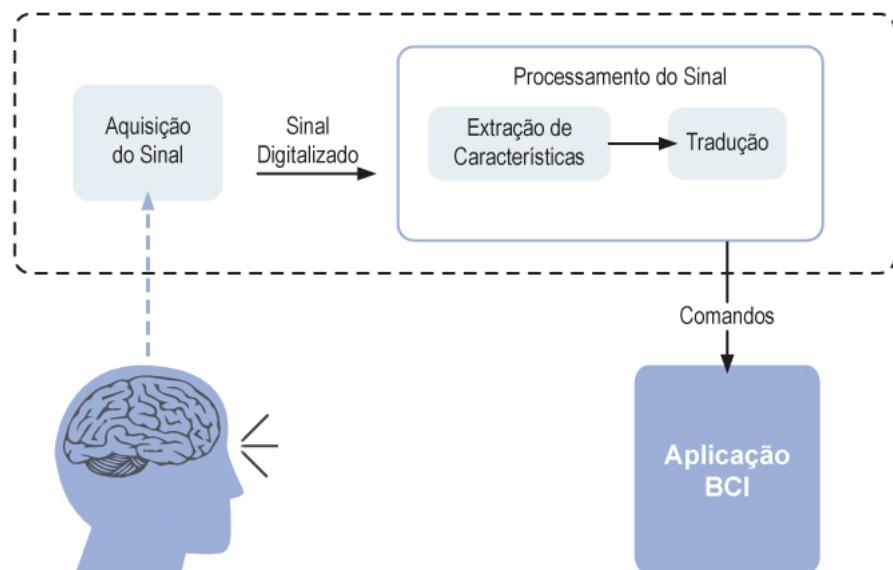
De modo assíncrono a esse avanço surgiu a área de estudo chamada Interface Cérebro-Máquina ou, do inglês, *Brain Computer Interface* (BCI), a qual se baseia em, estudar a capacidade de indivíduos em usar a atividade do cérebro para mover dispositivos. Segundo Gerwin Schalk (2004), esses estudos mensuram características específicas da atividade cerebral e como fazer suas traduções em sinais de controle de dispositivos.

A BCI transforma um padrão de atividade cerebral em um comando externo. São necessários quatro blocos básicos (como mostrado na Figura 2.2): aquisição de dados, extração de características, classificação e comando de dispositivo (WOLPAW et al., 2002).

Como é possível ver na Figura 2.2, a primeira etapa de uma BCI é a aquisição de dados, momento no qual será coletado as atividades cerebrais. Atualmente, através das tecnologias desenvolvidas, duas metodologias de captação de dados cerebrais

ganharam destaque, sendo a primeira chamada forma invasiva, onde através de uma neurocirurgia são implantados microeletrodos no cérebro, e a segunda a chamada forma não invasiva, onde os sinais são coletados sem a utilização de cirurgias.

Figura 2.2 – Diagrama básico de uma BCI



Fonte: (HEIDRICH et al., 2016)

Segundo Ramadan e Vasilakos o modo invasivo, mesmo conseguindo melhores resultados na qualidade dos sinais, tem a desvantagem de não ser possível mudar os implantes após estarem instalados, de modo a medir outras áreas do cérebro. Além disso, os microeletrodos estão propensos a acumular tecido cicatricial e com o decorrer do tempo o sinal pode se perder (RAMADAN; VASILAKOS, 2017).

Considerando essas características, é notório que na academia encontra-se uma proporção maior de trabalhos que utilizam tecnologias não invasivas, apesar de ter sinal com qualidade menor em relação a invasiva, pois não há necessidade de uma cirurgia para serem utilizadas (LEBEDEV; NICOLELIS, 2006).

2.1.1 Trabalhos relacionados

Os estudos na área da Neurociência vêm tendo avanços promissores, como o trabalho (SOUZA et al., 2015) que mostra que o cérebro humano possui diversas regiões e cada uma delas é responsável por realizar diferentes funções do corpo, como: motricidade, afetividade e cognição.

Os estudos relacionados a BCI estão vinculados com acessibilidade (SCHUH et al., 2013). Analisa o uso do EEG no controle de um simulador de cadeira de rodas,

onde os comandos são dados através do piscar dos olhos, um outro projeto similar é o (ROQUE; CEZAR et al., 2017) onde foi avaliado o uso de EEG para controlar o protótipo real de uma cadeira de rodas.

Em Gonçalves(2017) foi proposto um sistema tradutor dos sinais límbicos em respostas digitais, possibilitando o desenvolvimento de uma gama de novas aplicações para o aumento da autonomia de pessoas com limitações motoras.

Além disso, também são encontrados trabalhos voltados para inclusão social, como o trabalho acadêmico de Regina de O. Heidrich (2016) que desenvolveu um sistema que permite que uma pessoa possa transferir comandos a um computador diretamente. Em vez de utilizar um teclado, mouse ou outro dispositivo de entrada, o usuário desta interface simplesmente emite os comandos através de ondas cerebrais e o computador responde a elas, de modo que o foco do sistema foi a utilização do mesmo para auxiliar no processo de inclusão escolar de pessoas com paralisia cerebral.

2.2 Eletroencefalograma

Os sinais de um eletroencefalograma (EEG) foram observados pela primeira vez em 1929, pelo psiquiatra alemão Hans Berger(1929). Seu trabalho demonstrou a existência de atividades elétricas no cérebro, de forma que era possível medir suas grandezas através de eletrodos posicionados no couro cabeludo (LUCK, 2014). Nos anos seguintes, a leitura de sinais de EEG passou a ser aplicada para diversas finalidades, desde a medicina até a engenharia.

Como explicado por Thiago Caparelli (2007), o EEG pode ser descrito como um equipamento que permite coletar amostras bioelétricas das terminações nervosas do cérebro. Para um melhor entendimento do funcionamento do processo de aquisição dos dados do sistema proposto pelo trabalho, será abordado, nesta seção, o funcionamento dessas amostras bioelétricas e da representatividade da informação coletada pelo EEG.

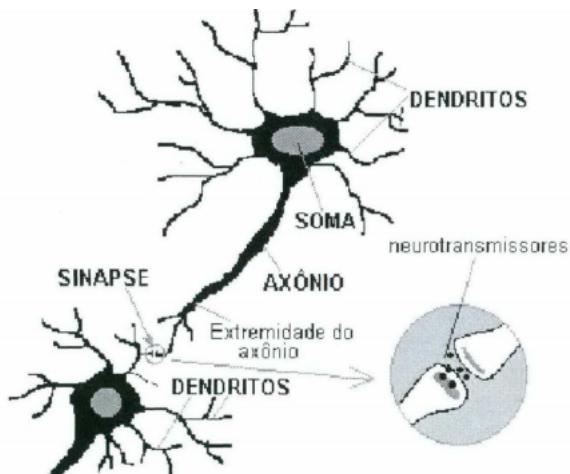
2.2.1 Bases biológicas

De acordo com Guyton e Hall (2010), o sistema nervoso recebe a cada minuto milhões de bits de dados provenientes de diferentes nervos sensoriais. O cérebro é o maior órgão do encéfalo e, também, compõe a maior parte do sistema nervoso, de modo que é dividido em dois hemisférios, sendo eles: esquerdo e direito. No meio desses hemisférios, existe uma camada de 5 mm de espessura de massa cinzenta, composta por células da glia, fibras nervosas e neurônios.

Um neurônio é uma célula composta por três partes: axônio, dendritos e soma

como ilustrado na Figura 2.3. Como o cérebro humano possui milhões de neurônios, as conexões sinápticas ficam responsáveis por garantir um meio de comunicação entre eles, de modo que os neurônios que enviam informações são chamados de pré-sinápticos, enquanto os que recebem são nomeados pós-sinápticos (ARGOUD et al., 2001). Desse modo, o neurônio se torna a célula fundamental para o estudo dos sinais do EEG, já que as informações coletadas pelo EEG são originadas dos impulsos nervosos.

Figura 2.3 – O neurônio e a sinapse



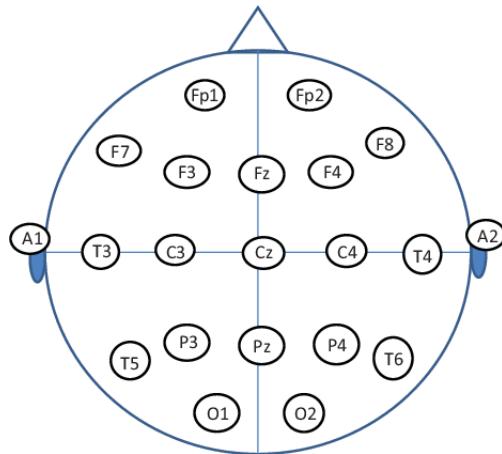
Fonte: (ARGOUD et al., 2001)

2.2.2 Sistema 10-20

Quando inicializado os estudos do modelo de aquisição dos dados do EEG, foi necessário adotar um padrão para o posicionamento dos eletrodos ao longo do couro cabeludo dos voluntários, de modo que pudesse uniformizar a recepção da qualidade dos sinais, que a estrutura de posicionamento deveria ser obtida através de uma medição específica, um padrão que permitisse uma cobertura adequada do espaço ao longo de todo o topo da cabeça e provesse uma nomenclatura simplificada dos locais de medição dos bio sinais. Este padrão é descrito por (KLEM et al., 1999) e é chamado de padrão ou sistema 10–20.

Conforme apresentado na Figura 2.4, neste padrão os eletrodos recebem nomes relacionados com a área do cérebro em que estão posicionados, conforme apresentado na Figura 2.5: Pré-frontal (FP), Frontal (F), Temporal (T), Parietal (P), Occipital (O), Central (C) e Auricular (A). Além disso estes nomes são relacionados à números, onde os números pares representam o lado direito do cérebro, os números ímpares o lado esquerdo e a linha central do crânio recebe a letra Z representando o número zero.

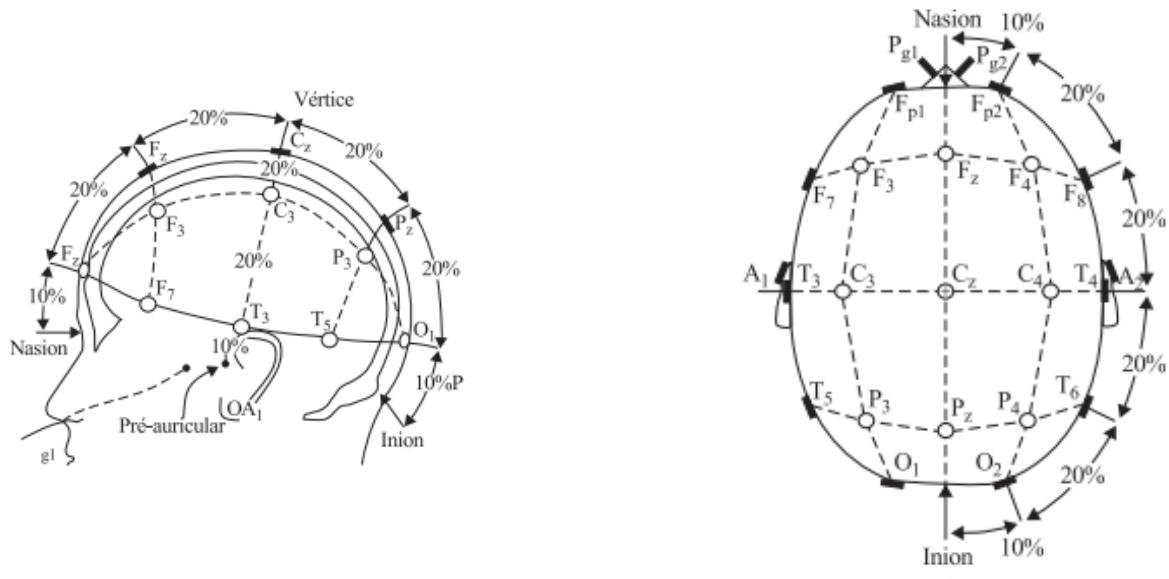
Figura 2.4 – Representação da montagem de eletrodos 10–20



Fonte: (KRUUK et al., 2014)

É conhecido que a atividade elétrica do cérebro é expressa como a diferença de potencial entre dois eletrodos, sendo um localizado na superfície da região do córtex e outro de referência em um local eletricamente neutro, na Figura 2.5 representado pelo eletrodo O0 e O1. O sinal do EEG possui frequências entre 0.5 a 30 Hz e de baixa amplitude variando entre 5 e 300 micro V (CARVALHO, 2008). Entretanto pode ocorrer variações nesses dados devido que possui diferença nas velocidade e tipos das conexões sinápticas de cada indivíduo (PEREIRA et al., 2003).

Figura 2.5 – Representação do posicionamento dos eletrodos de acordo com o Sistema Internacional 10-20, com eletrodos de referência biauriculares.



Fonte: (RIBEIRO, 2014)

2.3 Bandas de Frequência

Ao analisar isoladamente as informações geradas pelos eletrodos o trabalho se torna complicado, devido que esses dados possuem pouco valor agregado. Por razão deste fato, trabalhos na área utilizam os sinais baseados em bandas de frequência.

Bandas de frequência é uma representação na qual utiliza de filtros dos sinais coletados do EEG separados em quatro faixas de frequência de ondas: delta, teta, alfa e beta, sendo representados na tabela 1. Devido que essas ondas representam estados momentâneos do cérebro, é possível agregar mais valor às informações adquiridas pelo EEG.

Tabela 1 – Classificação das ondas em bandas de frequências

Delta (0-4 Hz)	Vigília com estado de sono profundo, meditação, ações involuntárias e estado inconsciente.
Teta (4-8 Hz)	Vigília com estado de atenção forçada (concentração), resolução de problemas lógicos e processamento de memórias.
Alfa (8-12 Hz)	Vigília com relaxamento.
Beta (12-30 Hz)	Vigília com estado de atenção.

Fonte: (BONINI-ROCHA et al., 2008)

Uma vez que o sinal é tratado dessa forma, é possível identificar que diferentes ritmos cerebrais estão sendo executados de forma paralela e independente. Comprovando que o cérebro humano apresenta uma capacidade de emitir sinais elétricos com representatividade de fenômenos neurológicos diferentes no mesmo instante de tempo.

Destes sinais, podem-se destacar algumas respostas fisiológicas que possuem características específicas de latência, amplitude e frequência captados nas medidas de tensão elétrica das bandas de frequência, modo que é possível identificar momentos específicos relacionados à estímulos que criam um processo cognitivo, como os estímulos de um indivíduo realizando ações motoras, na qual a amplitude diminui com o movimento ou intenção de movimento e pode ser controlada através de esforço mental (MUTHUKUMARASWAMY; JOHNSON, 2004).

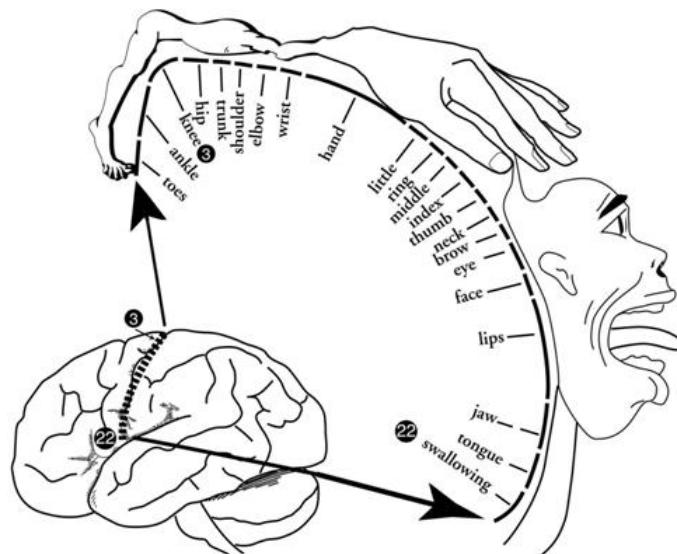
2.4 Movimento Voluntário e Movimento Imaginário

Levando-se em conta que o movimento voluntário envolve as etapas de planejamento, execução e controle, uma ação exigirá a ativação de um grande número de áreas corticais (FILHO et al., 2010). Por outro lado, o movimento involuntário se origina principalmente na medula espinhal. O estudo do movimento voluntário (MV) pode

ser considerado como um bom modelo para trabalhar com gatilhos de ativação de um sistema de controle, visto que o MV é controlado por áreas corticais e subcorticais relativamente bem definidas (KANDEL et al., 2000).

Na Figura 2.6 podemos observar o modelo de um córtex motor, destacando principalmente as áreas do corpo em relação com seu mapeamento do córtex motor. Dessa forma, é importante notar que a área da mão tem uma extensão maior do que a área do braço e do pé.

Figura 2.6 – Representação do homúnculo cortical, relacionando com as partes do corpo



Fonte: (ARKELL, 2019)

De acordo com Detety e Ingvar (1990), o movimento imaginário (MI) pode ser considerado como um abstração imaginária de um MV, sem qualquer entrada sensorial ou qualquer saída na forma de ativação muscular. Jeannerod e Decety (1995) afirmam que a MI é baseado em processos similares àqueles que estão envolvidos na programação e preparação do MV.

Por essas razões é notado que as melhores combinações para terem uma melhor diferenciação por classificadores seria através de ações motoras voluntárias com a captura de dados baseado em frequência de bandas.

3 APRENDIZADO DE MÁQUINA

Segundo Tom M. Mitchell (MITCHELL et al., 1997) , um programa computacional aprende a partir da experiência E, em relação a uma classe de tarefas T, com medida de desempenho D, se seu desempenho nas tarefas T, medida por D, melhora com a experiência E. Essa é a base da definição do aprendizado a partir de tarefas computacionais, resultando nos métodos de aprendizado de máquina.

Uma das principais característica dos métodos de aprendizado de máquina (AM) é adquirir experiência a partir dos dados, onde segundo Batista (2003) a relação entre o desempenho dos algoritmos de AM e a qualidade dos dados são proporcionais. Assim sendo, os dados se tornam um aspecto fundamental para a performance de um sistema de aprendizado.

Entretanto, em base de dados reais é comum se deparar com diferentes tipos de problemas, devido aos diversos estilos de organização dos dados e das qualidades das informações. Por essa razão, pesquisadores separaram as técnicas em paradigmas de AM utilizando como critério de separação os problemas abordados por elas, sendo os principais, aprendizado supervisionado e aprendizado não-supervisionado.

O aprendizado não-supervisionado é descritivo, isto é, ele procura uma maneira de descrever grupos a partir das características dos dados, sendo assim o algoritmo faz suas inferências baseando-se apenas na semelhança estatística dos padrões de entrada.

Já no aprendizado supervisionado, o objetivo é ser preditivo, desse modo os algoritmos tipo de paradigmas são baseados em um modelo que recebe um conjunto de informações, onde cada amostra é formada por atributos de entrada e um rótulo (SOUTO et al., 2003). A finalidade desse grupo é construir um classificador que tem como função determinar o rótulo de uma instância ainda não rotulada. Como exemplos das técnicas vale citar: Perceptron de Múltiplas Camadas (do inglês *Multi-layer Perceptron*, MLP), Regressão Logística (do inglês *Logistic Regression*) e Árvores de Decisão.

3.1 Aprendizado Supervisionado e Classificação de Padrões

Os algoritmos de reconhecimento que este trabalho visa abordar usam como base o conceito de identificação de padrões sob o regime de aprendizado supervisionado, onde aprendem a identificar comportamentos e padrões por similaridade dos atributos de entrada com seus respectivos rótulos. Todavia, o estimador gerado desse grupo de algoritmos são divididos em dois tipos de modelos, sendo eles regressor

quando a saída é baseada em um valor contínuo ou classificador quando a função (modelo) gera uma saída sob forma de classes ou categorias.

Esse tipo de aprendizado é avaliado no trabalho (LEMOS, 2009) onde utilizando o KNN, foi notado que a classificação por similaridade é uma tarefa natural e com bons resultados, isso devido a uma máquina conseguir associar um indivíduo que nasce no Rio de Janeiro (Brasil) como brasileiro, se for engenheiro é classificado como alfabetizado e se tem 1,80 m de altura é considerado alto.

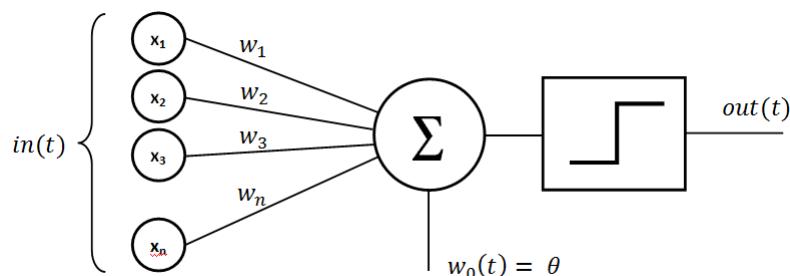
Trabalhos voltados a diversas áreas de reconhecimento de padrões utilizam do conceito dos classificadores de AM para resolver seus problemas, como (FERREIRA, 2008) com a proposta de um sistema de controle de uma cadeiras de rodas utilizando o *support vector machine* (SVM).

3.2 Rede Neural Artificial

Especificamente ao trabalhar com a tradução de sinais de mentais, como é o caso deste trabalho, é muito comum encontrar na academia projetos que utilizam de redes neurais. As ANN são baseadas no conceito de perceptron, que é um modelo computacional inventado em 1957 por Frank Rosenblatt no *Cornell Aeronautical Laboratory*, onde segundo a literatura é considerada o tipo mais simples de rede neural *feedforward* (CASTRO; CASTRO, 2001).

De modo a facilitar a explicação sobre a estrutura de um perceptron, deve-se esclarecer primeiramente a estrutura e funcionamento do neurônio. O modelo mais bem aceito foi proposto por Warren McCulloch e Walter Pitts (1943), o qual implementa de maneira simplificada os componentes e o funcionamento de um neurônio biológico sob princípios de modelos matemáticos. Em termos simples, um neurônio matemático de uma rede neural artificial é um componente que calcula a soma ponderada de vários *inputs*, aplica uma função e passa o resultado adiante (ALVES, 2018). Na Figura 3.1 é possível ver um modelo usual para um neurônio.

Figura 3.1 – Modelo computacional de um neurônio



Fonte: (MCCULLOCH; PITTS, 1943)

Neurônio de perceptron é uma técnica de aprendizado de máquina biologicamente inspirada, onde tem-se o perceptron como um classificador binário que correlaciona sua entrada x para um valor de saída $f(x)$, um valor binário simples, através da equação 3.1, onde w é um vetor de peso real e $w \cdot x$ é o produto escalar e b é o viés (*bias*) que pode ser visto como o peso associado com uma entrada x (OLIVEIRA; FAVERO, 2002).

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b \geq 0 \\ 0 & \text{else} \end{cases} \quad (3.1)$$

Dando continuidade da linha de pensamento de funções matemáticas inspiradas em modelos biológicos, uma rede neural artificial ou rede neural ou simplesmente rede, é uma técnica constituída por neurônios de perceptron que aplicam uma determinada função matemática aos dados, chamada na literatura de função de ativação, gerando uma única resposta, na qual são dispostos em camadas e ligados entre si, sendo estas conexões, geralmente associadas a coeficientes denominados de pesos (BINOTI; BINOTI; LEITE, 2014).

O ciclo de processamento de uma rede neural é constituído de duas etapas, a primeira chamada de fase de treinamento ou aprendizado, onde durante esse processo será realizado o ajuste dos pesos, momento onde ocorre a extração das características dos dados e armazenamento do conhecimento na rede. Já a segunda etapa denominada fase de teste, é a aplicação da rede no processo de generalização, ou seja, utilização de uma rede treinada para dar resposta para dados inéditos (BRAGA, 2000a).

Função de ativação é uma expressão matemática ligada a saída de um perceptron, sendo importante enfatizar que a não-linearidade deve ser suave, diferente da função sinal utilizada pelo perceptron original (SILVA et al., 1998).

Esta função matemática caracteriza o efeito que a entrada interna e o estado atual de ativação exercem na definição do próximo estado de ativação do neurônio. Quando existem atributos dinâmicos pertencentes a definição do estado de ativação, em problemas na qual a saída da rede possui saídas contínuas, equações diferenciais são adicionadas ao modelo matemático (KOSKO, 1992).

Ao fato que um programador visa busca a simplicidade do modelo de rede neural, geralmente define-se sua função de ativação como uma função algébrica da entrada interna atual. A seguir alguns tipos de função de ativação comumente encontrados na literatura para ANN (KOSKO, 1992).

- Função Linear

Sendo que a expressão para esta função de ativação é possível ver pela equação 3.2).

$$f(x) = p \cdot x, f'(x) = p \quad (3.2)$$

- Função Linear Rectificada (ReLU)

A principal vantagem desse tipo de função é o fato de que ela não ativa todos os neurônios ao mesmo tempo, isto é, se a entrada da ReLU for negativa, ela será convertida em zero e o neurônio não será ativado. (ver equação 3.3):

$$f(x) = \max(0, x) \quad (3.3)$$

- Função Logística

A utilização desse tipo de função como estado de ativação se vem com a preocupação de pesquisadores em limitar o intervalo de variação da derivada da função, pela inclusão de um efeito de saturação (ver equação 3.4).

$$f(x) = \frac{e^{px}}{1 + e^{px}} = \frac{1}{1 + e^{-px}}, \quad f'(x) = p f(x) \cdot (1 - f(x)) \quad (3.4)$$

- Função Tangente Hiperbólica

Sendo que a função logística é constituída por valores de ativação apenas no intervalo (0, 1), em muitos casos ela é alterada pela função tangente hiperbólica, que preserva a característica sigmoidal da função logística, mas possui seu domínio os valores positivos e negativos ($f(x) \in (-1, 1)$). (ver equação 3.5):

$$f(x) = \frac{e^{px} - e^{-px}}{e^{px} + e^{-px}} = \tanh(px), \quad f'(x) = p (1 - f(x)^2) \quad (3.5)$$

Atualmente o tipo da função de ativação, o número de neurônios da rede, a quantidade de camadas ocultas ou até mesmo o tipo de conexão, é o que determina as diferentes formas de redes e arquitetura encontrados na literatura. Além dessas características apresentadas anteriormente, uma outra individualidade que interfere significativamente no tempo de treinamento e na eficiência de uma rede são os algoritmos utilizados, bem como os parâmetros e coeficientes utilizados em seu treinamento (BINOTI; BINOTI; LEITE, 2014).

A definição do algoritmo de treinamento a ser utilizado, bem como de seus parâmetros influenciam principalmente na fuga de mínimos locais, no desempenho da tarefa desejada e no tempo de treinamento (BINOTI; BINOTI; LEITE, 2014). Devido a esse empecilho durante o treinamento, surgiram na literatura algoritmos como o de *backpropagation* (HAYKIN, 2007), voltado a diminuir problemas como, taxas de

aprendizado muito baixas onde acarretam um aprendizado muito lento, ou até mesmo taxas de aprendizado muito altas geram oscilações no processo de treinamento.

Dentre as diversas configurações de redes encontradas na academia, foi adotado para esse trabalho três diferentes técnicas de ANN, sendo elas Perceptron de Múltiplas Camadas, Rede Neural Convolucional e Redes Neurais Recorrentes.

A Perceptron de Múltiplas Camadas é uma rede do tipo rede *feedforward* (GOMES, 2003), publicada no final da década de 69 com o livro *Perceptrons* (MINSKY; PAPERT, 1969).

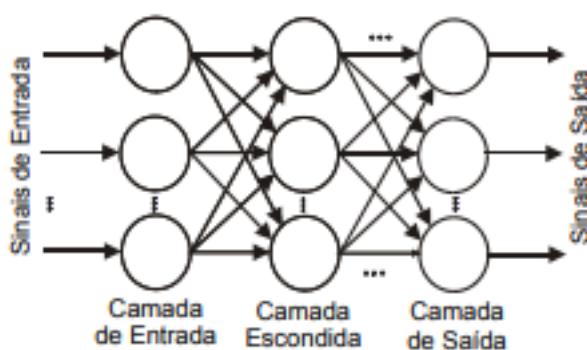
A Rede Neural Convolucional (do inglês *Convolutional Neural Network*, CNN) atualmente considerado por muitos pesquisadores sendo o estado da arte para atividades como reconhecimento de padrões, Touvron (2019) utiliza uma CNN para resolver o problema da Imagenet (Krizhevsky; Sutskever; Hinton, 2012).

E a Rede Neural Recorrente (do inglês *Recurrent Neural Network*, RNN) onde é conhecida na literatura por resolver problemas de natureza intrinsecamente temporal.

3.3 Perceptron de Múltiplas Camadas

Perceptron de Múltiplas Camadas possui três características principais segundo (CASTRO; CASTRO, 2001) sendo elas, os neurônios das camadas ocultas e de saída possuírem uma função de ativação não-linear do tipo sigmoidal, a segunda característica se vem com o fato da rede possuir uma ou mais camadas intermediárias, e a terceira característica a necessidade da arquitetura possuir altos graus de conectividade, ou seja, todos os neurônios de qualquer camada estão conectados a todos os neurônios da camada anterior (HAYKIN, 2001), é possível observar essas individualidades na Figuras 3.2 e 3.3.

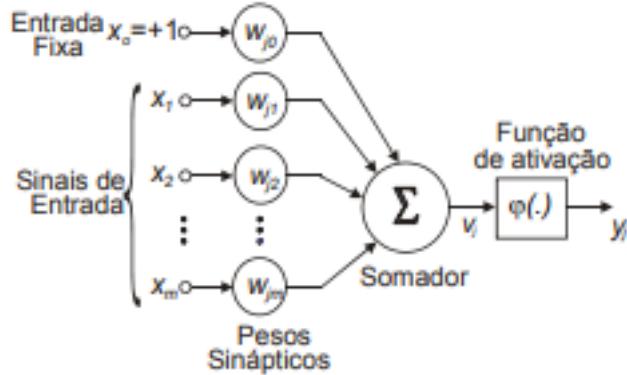
Figura 3.2 – Modelo de arquitetura da *multilayer perceptron*



Fonte: (SOUZA et al., 2003)

Utilizando como base a Figura 3.3, Souza explica, que sendo $x = (x_1, x_2, \dots, x_m)^t$ a representação do vetor de entrada, $W_{j,i} = W_{j,1}, W_{j,2}, \dots, W_{j,m}$ o vetor de pesos sinápticos e $W_{j,0}$ o bias (b_j), na qual $\varphi_j(\cdot)$ é a representação da função de ativação, a saída do neurônio j pode ser representada sendo $y_j = \varphi_j(v_j)$ (SOUZA et al., 2003).

Figura 3.3 – Modelo do neurônio



Fonte: (SOUZA et al., 2003)

De modo a entender o funcionamento da MLP, tem-se que o valor de entrada da rede é transmitida camada a camada, até a saída. O vetor de saída de uma camada k da rede pode ser calculado com a Equação 3.6.

$$y^k = \varphi(W^k \cdot (x^k)^t + b^k) \quad (3.6)$$

Onde W^k é representado como a matriz de pesos da camada k e b^k o vetor de bias da camada k , na qual teremos os vetores x^k e y^k como os valores de entrada e saída da camada k , respectivamente (SOUZA et al., 2003).

Vale ressaltar que as camadas ocultas são as principais responsáveis pela computação do estímulo da camada de entrada, com isso possibilitando a MLP realizar cálculos mais complexos do que uma rede *feedforward* (SOUZA et al., 2003).

Entretanto, a adição de camadas extras faz com que o ajuste do erro, se torne uma tarefa complexa, dificultando determinar qual é a saída desejada de um neurônio interno. Para melhorar a performance devido esse problema a MLP utiliza o algoritmo de *backpropagation*, que tem a finalidade de ajustar os pesos da rede (BISHOP et al., 1995).

3.3.1 Backpropagation

O algoritmo *backpropagation* utiliza pares (entrada e saída desejada) para ajustar os pesos da rede por meio de funções de erro (SOUTO et al., 2003). O treinamento

por esse algoritmo ocorre em duas fases. Em um primeiro momento a computação para frente, onde os valores de entrada da rede se propaga em todas camadas até chegar na de saída, também chamado de *feedforward* por (BRAGA, 2000b). No segundo momento, a rede propaga o ajuste no sentido contrário, referente ao erro calculado caso a saída da rede seja incorreta (SILVA et al., 1998).

Na segunda fase chamada de *backward* por (BRAGA, 2000b), ocorre inicialmente o ajuste dos pesos dos neurônios da camada de saída, sendo que este ajuste acontece em todos os neurônios da rede, sendo que o diferencial é que os valores de entrada da Equação 3.7 na última camada é a diferença entre o resultado desejado pelo obtido, e a partir desse valor resultante ocorrerá a propagação dos valores resultantes para as camadas mais internas.

A MLP possui como característica que a saída de uma camada torna-se a entrada da camada seguinte, de modo que a Equação 3.7 é representada sob esse fundamento a partir da Equação 3.6.

$$u^{m+1} = f^{m+1}(W^{m+1} \cdot y^m + b^{m+1}), \text{ para } m = 0, 1, \dots, M - 1, \quad (3.7)$$

Sendo que M é o número de camadas da rede. As unidades na primeira camada recebem um vetor externo segundo a Equação 3.8 como forma de entrada.

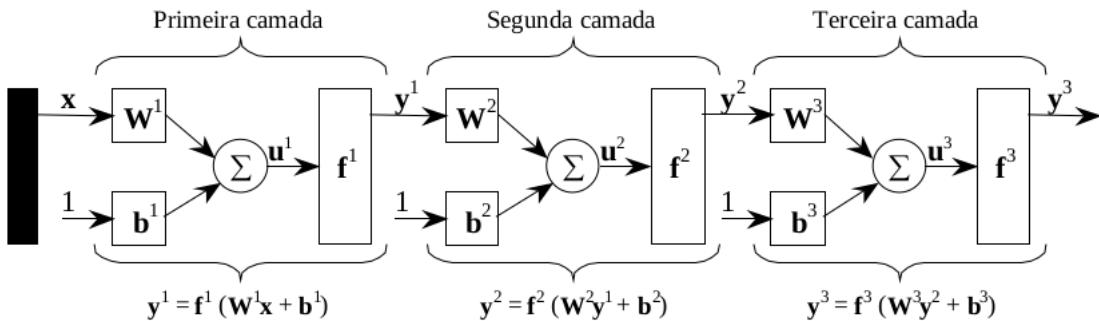
$$y_0 = x, \quad (3.8)$$

Desse modo, a representação condicional inicial para a Equação 3.7, as saídas das unidades localizadas na última camada são consideradas as saídas da rede, veja Equação 3.9.

$$y = y^M. \quad (3.9)$$

Pela Equação 3.10 é possível que a saída da rede pode ser expressa apenas em função do vetor de entradas x , das matrizes de pesos W^m e dos vetores de limiares b^m , e de forma menos abstrata é possível encontrar a Equação 3.10 na Figura 3.4.

$$y^3 = f^3(W^3 f^2(W^2 f^1(W^1 x + b^1) + b^2) + b^3). \quad (3.10)$$

Figura 3.4 – MLP com *backpropagation*

Fonte: (SILVA et al., 1998)

A explicação apresentada sobre *backpropagation* é um resumo com o foco na lógica de implementação computacional, e tal forma que o algoritmo é melhor abordado por (SILVA et al., 1998) em seu trabalho.

Por sua vez, alinhado com a literatura está a proposta de utilizar Perceptron de Múltiplas Camadas para traduzir as ondas cerebrais em sinais de controle de máquinas, (2005) utiliza a MLP juntamente com o pré-processamento da potência espectral e diferença de potência nas 4 bandas: delta e teta, beta, alfa e gama com a intenção de fazer o controle de um cursor na tela, o mesmo utilizado por esse trabalho.

3.4 Rede Neural Convolucional

Um modelo neural segundo LeCun (2015), é baseado em estruturas chamadas de camadas, agrupamento de neurônios, onde a junção de várias camadas na arquitetura é o que define a profundidade da rede neural, por esse motivo que na literatura as ANN com inúmeras camadas são denominadas redes neurais profundas (do inglês *deep learning*). Desse modo em razão das CNN terem a peculiaridade de possuírem muitas camada em suas arquiteturas, o termo de *deep learning* está bastante ligado a essas arquiteturas na literatura (LECUN; BENGIO; HINTON, 2015).

Entre as redes neurais, as convolucionais têm como principal particularidade a habilidade de fazer a extração de características dos dados de entrada através dos mapas de convolução (FERREIRA, 2017). Os estudos baseados em utilizar mapas de convolução em redes neurais começaram a ganhar fama dentro da academia após 2006 com pesquisadores do *Canadian Institute for Advanced Research* (CIFAR), na qual trabalhando com redes neurais artificiais introduziram conceitos de aprendizagem não-supervisionada, onde o resultado desses experimentos foram camadas capazes de detectar atributos sem a necessidade de informação pré-rotulada (LECUN; BEN-

GIO; HINTON, 2015).

3.4.1 Mapa de convolução

Na academia, é notado que diversas vezes é apresentado em seus algoritmos técnicas e funções matemáticas onde seu embasamento teórico é retirado de outras áreas do conhecimento humano.

Sendo que a convolução segue o mesmo princípio onde buscando da matemática, particularmente na área de análise funcional e processamento de sinais, a teoria se baseia em uma operação matemática entre duas funções f e g , na qual resulta em uma terceira função, de modo que pode ser interpretada como uma função modificada de f .

Entretanto quando se trata de reconhecimento de padrões de dados no formato de função bidimensional essa técnica é extremamente eficiente, isso vem do fato que a convolução possui a habilidade de fazer a extração de atributos (PARKER, 2010).

Ferreira (2017) explica em seu trabalho, seja as funções f e g , para uma variável contínua x , a convolução é definida como:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x - \tau) d\tau \quad (3.11)$$

Sendo que o operador $*$ representa a convolução para as funções f e g , de modo que x está definido no conjunto \mathbb{Z} , desse modo a equação da convolução discreta pode ser definida como:

$$f[x] * g[x] = \sum_{n=-\infty}^{\infty} f[n] \cdot g[x - n] \quad (3.12)$$

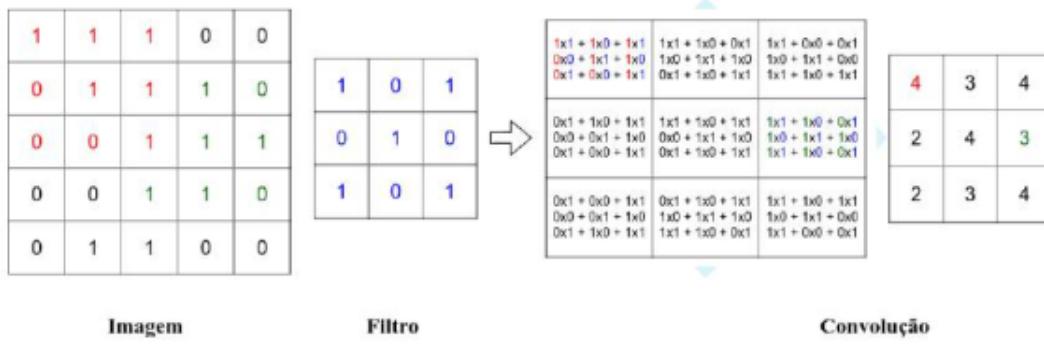
Estendendo esta definição para funções com duas variáveis x e y , obtém-se as seguintes equações:

$$f(x, y) * g(x, y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1, \tau_2) \cdot g(x - \tau_1, y - \tau_2) d\tau_1 d\tau_2 \quad (3.13)$$

$$f[x, y] * g[x, y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2] \quad (3.14)$$

A convolução de uma matriz de valores pode ser interpretada como o somatório da multiplicação de cada elemento da entrada, ao seus vizinhos locais, pelos elementos da matriz que representa o filtro de convolução (FERREIRA, 2017), esse cálculo é ilustrado na Figura 3.5.

Figura 3.5 – Exemplo de aplicação da convolução em uma matriz de valores. À esquerda, uma matriz hipotética representada por um único canal com dimensões 5x5 que recebe a aplicação de um filtro 3x3. À direita, uma matriz ilustrando o somatório que fornece o resultado da convolução.



Fonte: (FERREIRA, 2017)

De modo ao observar a abordagem da Figura 3.5, a aplicação do filtro de convolução à matriz de valores original é reduzida em tamanho proporcional dimensões do filtro utilizado. Entretanto segundo Ferreira (2017) é encontrado na literatura outras abordagens da convolução que fazem a criação de novas células adjacentes, fazendo com que a matriz resultante mantenha as mesmas dimensões de entrada.

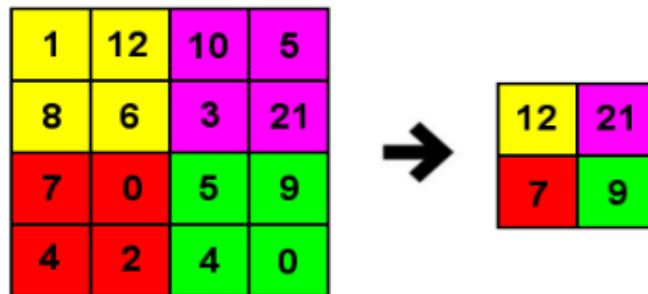
Embora as CNN tenham sido inventadas na década de 1980, com a primeira aplicação de uma ANN utilizando essa abordagem registrada por (WAIBEL, 1987), seu avanço só foi possível após os anos 2000 isso devido a falta de hardware que existia antes desse período. Em 2004, KS Oh e K. Jung (2004) mostraram em seu trabalho que era possível fazer uma aceleração na velocidade de treino de modelos neurais através atuais *graphics processing unit* (GPU). Em seu projeto foi comprovado que a implementação foi vinte vezes mais rápida que uma implementação equivalente na CPU (HINTON; OSINDERO; TEH, 2006).

Após a evolução nas pesquisas das ANN com a utilização das GPUs em seus treinamentos, as redes neurais convolucionais profundas começaram a ganhar um sucesso prático de forma rápida dentro da academia na qual segundo LeCun (2015) mesmo utilizando uma abordagem de múltiplas camadas, seu treinamento acontecia de maneira robusta e com bons resultados práticos em um tempo aceitável. Segundo Ferreira (2017) o sucesso das redes profundas não é apenas devido a essas características apresentadas, mas também devido a topologia projetada para reduzir o número de parâmetros, desse modo otimizando o tempo do *backpropagation*.

Segundo (LECUN; BENGIO; HINTON, 2015), uma arquitetura de CNN clássica é dividida em estágios, onde o primeiro estágio é composto de dois tipos de camadas, as camadas de convolução e as camadas de *pooling*. A camada de convolução

consiste no processo de mapeamento de atributos, enquanto a camada de *pooling* é uma forma de *down-sampling*, isso é, uma técnica matemática onde a cada camada de *pooling* computa uma métrica de uma determinada região do mapa de atributos, como pode ser visto na Figura 3.6. A principal função é eliminar valores não máximos, reduzindo a dimensão da representação dos dados e consequentemente a computação necessária para as próximas camadas, além de criar uma invariância a pequenas mudanças e distorções locais (FERREIRA, 2017).

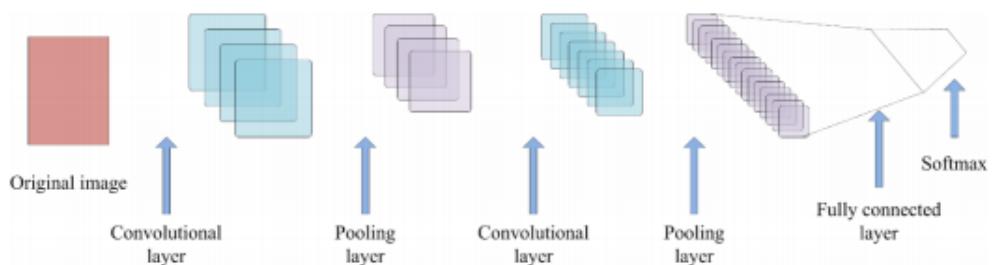
Figura 3.6 – Aplicação de *max pooling* em uma imagem 4x4 utilizando um filtro 2x2.



Fonte: (FERREIRA, 2017)

LeCun (2015) ainda aponta dois ou três estágios de convolução, normalização e *pooling* podem ser empilhados, seguidos por mais camadas completamente conectadas, vide Figura 3.7. As camadas convolucionais e de *pooling* são diretamente inspiradas por noções clássicas de células simples e células complexas na neurociência visual.

Figura 3.7 – Arquitetura tradicional das Redes Neurais Convolucionais.



Fonte: (HE et al., 2019)

Outros dois componentes comuns em CNN são: normalização por batelada e função *softmax* na saída da rede para transformá-la em um classificador

A função de *softmax* é uma função matemática que ao ser adicionada no final da rede produz um efeito probabilístico sob o resultado obtido, isto é, ocorre a distribuição probabilística sobre a quantidade total de classes do problema (GRAVES; MOHAMED; HINTON, 2013).

A normalização por batelada (do inglês *batch normalization*) (IOFFE; SZEGEDY, 2015) é uma transformação que ocorre durante o treinamento de uma ANN, onde sua função é normalizar o conjunto de dados de treinamento de acordo com a distribuição normal padrão, média igual a zero e variância igual a 1, para evitar problemas de comparação devido às diferentes escalas usadas nos dados. Segundo (BEZERRA, 2016) é conhecido na literatura um problema chamado internal *covariate shift*, que é quando durante a passagem dos exemplos normalizados através das camadas da rede.

Esses valores são novamente transformados, devido às pré-ativações e ativações, podendo fazer com que os dados de entrada de algumas camadas ocultas fiquem desnormalizados novamente, podendo agravar dependendo da profundidade da rede, além do fator que esse problema aumenta o tempo de treinamento porque implica na definição de uma taxa de aprendizagem pequena, além de propiciar a dissipação dos gradientes.

Variações de CNN vêm obtendo excelentes resultados em diversos paradigmas de AM. Kalchbrenner e Grefenstette (2014) utilizaram uma rede profunda para solucionar problemas de tarefas de processamento natural de linguagens como análise de sentimentos; Santos e Gatti (2014) onde alcançaram o estado de arte no *Stanford Sentiment Treebank* (SSTb), no reconhecimento de sentimento de frases de filmes e trabalhos de reconhecimento de paradores utilizando EEG temos o trabalho (OH et al., 2018) utilizaram de *deep learning* para auxiliar médicos no diagnóstico de Parkinson a partir de sinais de EEG.

3.5 Rede Neural Recorrente

Uma Rede Neural Recorrente, do inglês *Recurrent Neural Network* (RNN), é uma rede neural artificial em que as conexões entre as unidades ocultas formam ciclos. As Redes Neurais Recorrentes são um tipo específico de rede neural projetada para problemas que envolvem sequências (KAWAKAMI, 2008). Uma de suas principais vantagens nesta tarefa, é o reconhecimento de cadeias de sequências, as quais seriam de difícil reconhecimento em redes não especializadas, como é o caso das redes *feedforward* e redes convolucionais (GOODFELLOW; BENGIO; COURVILLE, 2016).

Ao ser comparada a uma rede perceptron *feedforward*, uma rede neural recor-

rente pode ser pensada como a adição de *loops* à arquitetura (SOARES et al., 2018). Assim sendo, a RNN possui dois dados de entrada, atual e a da etapa anterior. A rede trabalha em cima do valor da entrada atual, e depois usa um *loop de feedback* de modo a considerar as entradas do ciclo anterior. As conexões, ou camadas, recorrentes adicionam estado, ou memórias, na rede de modo que permitem que elas lidem com abstrações mais amplas das sequências de entrada.

As sequências de entrada de uma rede recorrente é representada como um conjunto em função do tempo $x^{(1)}, \dots, x^{(t)}$, onde t é uma variação do tempo de 1 até τ . As RNN compartilham matrizes de pesos ao longo das etapas, também chamado de passos, do processamento da cadeia de entrada.

Desse modo lhe permite aprender padrões que surgem em posições distintas nas sequências. Essa habilidade das camadas recorrentes foi um dos avanços desse tipo de rede em relação às *feedforward* tradicionais (BISPO, 2018).

Na Figura 3.8, é representado uma *recurrent neural network* genérica de modo a ilustrar um exemplo de uma arquitetura geral (BISHOP, 2006). Na imagem, a rede opera sobre $x(t)$, onde vetor de entrada no tempo t é dado pelo conjunto $x(1), x(2), x(3), x(4)$, e resulta no conjunto de saída $z(1), z(2), z(3), z(4)$, da rede após processar sua entrada. Sua saída também é conhecida como o estado da rede no tempo t . O *loop* demonstrado na Figura 3.8 saindo da *Output Layer* (do português camadas de saída) para *Hidden Layer* (do português camadas ocultas) e onde se localiza a capacidade da RNN de transferir informações de um passo para o outro (BISHOP, 2006).

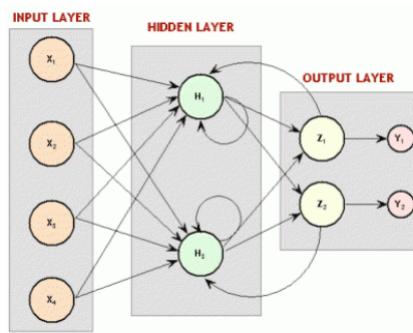


Figura 3.8 – Representação geral de uma RNN (BISHOP, 2006)

Segundo Bispo (BISPO, 2018), para uma sequência de tamanho t , é possível representar a entrada de um RNN no tempo a partir da representação da Figura 3.9. Sendo, cada entrada no tempo t é mapeada para a entrada $t + 1$ através de um função f que opera sobre o estados t .

Visto que os estados s_t são tratados como a memória da RNN, de modo que uma vez onde ela obtém a informações sobre o histórico de ocorrências, ou seja,

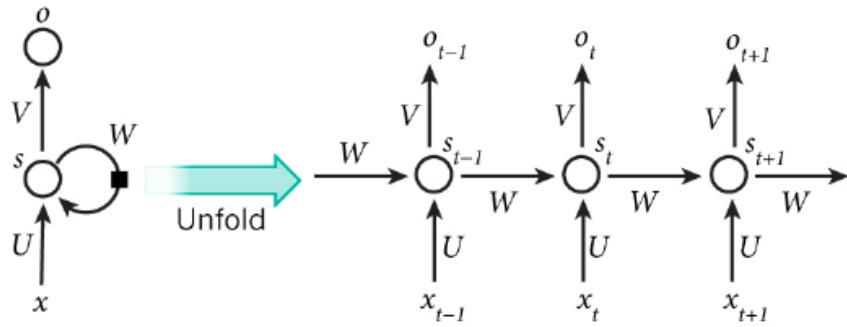


Figura 3.9 – Representação da entrada no tempo de uma RNN (BISPO, 2018)

valores dos estados anteriores, parâmetros de f podem ser usados entre os passos.

Na Figura 3.9, a operação *Unfold* representa o desmembramento da RNN de sua forma recorrente para sua forma em função da entrada em função de t . Segundo Bispo (BISPO, 2018), a entrada do estado s no tempo t é ponderada pela matriz de pesos representado por U , onde sua saída é multiplicada pela matriz V de fim a gerar a saída o , que por sua vez o valor do estado s_t é usado com entrada para o estado s_{t+1} ponderado pela matriz W .

Segundo (BISHOP, 2006) a *vanilla RNN* é um dos modelos mais simples de RNN e sua arquitetura convencional é representada na Figura 3.8. Uma maneira comum de se calcularem os valores de s_t e o é através da Equação 3.15:

$$\begin{aligned} s_t &= f(Ux_t + Ws_{t-1}) \\ o_t &= Vs_t \\ y_t &= \text{softmax}(o_t) \end{aligned} \tag{3.15}$$

De modo que é comum encontrar em trabalhos sobre RNN a função de ativação f sendo uma função não linear, como a ReLU (Equação 3.3) ou a tangente hiperbólica (Equação 3.5).

3.5.1 Backpropagation through time

Pesquisadores quando começaram a estudar a arquitetura da RNN, perceberam que era necessário desenvolver um novo algoritmo de erro para o cálculo do gradiente descendente para as RNN, de modo que essa nova implementação foi necessária devido ao compartilhamento de parâmetros inerente presentes na arquitetura da RNN. De modo que, da mesma forma que a arquitetura da RNN foi desenhada sob a MLP (Secção 3.3), o algoritmo de erro da RNN também chamado *Backpropagation Through Time*, representada pela Fórmula 3.16, foi feito baseada no *backpropagation*

(Subsecção 3.3.1) das redes *feedforward* (BISPO, 2018).

$$\begin{aligned} E_t(y_t, \hat{y}_t) &= -y_t \log(\hat{y}_t) \\ E_t(y, \hat{y}) &= \sum_t E_t(y_t, \hat{y}_t) \\ &= -\sum_t y_t \log(\hat{y}_t) \end{aligned} \quad (3.16)$$

Bispo (2018) explica em seu trabalho que, y_t representa o valor da função no tempo t , e \hat{y}_t é o valor predito pela rede conforme Fórmula 3.15. Onde o erro total é formado pela soma dos erros em cada estado s_t , sendo a matriz V um dos parâmetros do modelo, o erro em função de V é calculado como o somatório das derivadas parciais de E_t em relação a V , conforme Equação 3.17.

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \quad (3.17)$$

Ao expandir a Fórmula 3.17, é possível perceber que para V , a função do erro depende somente dos valores de y_t , \hat{y}_t , s_t , transpondo o resultados de $\frac{\partial E_t}{\partial V}$ calculável a partir de uma única multiplicação matricial, Equação 3.18.

$$\begin{aligned} \frac{\partial E_t}{\partial V} &= \frac{\partial V}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial V} \\ &= \frac{\partial V}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} \frac{\partial z_t}{\partial V}, z_t = Vs_t \end{aligned} \quad (3.18)$$

Entretanto, o erro em função de W e U , visto que ambos dependem de s_t , de modo que isso requer a aplicação da regra da cadeia (STEWART, 2010), considerado que s_t é uma constante, por esta razão, $\frac{\partial E_t}{\partial W}$ é calculado da a partir da Equação 3.19.

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^t \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W} \quad (3.19)$$

Desse modo, o gradiente para o cálculo do erro E no tempo t é propagado recursivamente até o tempo $t = 0$, o que ocasiona o chamado *Vanishing Gradient Problem* (HOCHREITER, 1998), que consiste na degradação dos valores do gradiente para zero em poucos passos, como efeito das sucessivas multiplicações matriciais. Sendo que, o gradiente em cada unidade celular recorrente tendendo a zero, será impulsionado os gradiente das células anteriores para zero também, de modo que quanto maior o valor t da cadeia de entrada, maior as probabilidades da rede ser penalizada com esse problema (BISPO, 2018).

Na literatura diversas soluções foram desenvolvidas para evitar a aproximação do gradiente a zero, sendo que a cada nova solução é uma adaptação da arquitetura

da RNN, a abordagem utilizada por esse trabalho é a arquitetura da *Long Short-Term Memory* (HOCHREITER; SCHMIDHUBER, 1997).

3.5.2 Long short-term memory

De modo a evitar a *vanishing gradient problem* surgiram na literatura as RNN gates, unidades que possuem a característica de conseguirem calculam os pesos de seus conectores através de valores manualmente escolhidos ou parametrizados (GOODFELLOW; BENGIO; COURVILLE, 2016). A partir desse componente neural surgiu a *Long Short-Term Memory*, uma célula de RNN baseada em um tipo especial de gate proposta por Hochreiter (1997).

Como toda *gated RNN*, as *Long Short-Term Memory* (LSTM) têm a capacidade tanto de lembrar quanto de esquecer o estado anterior quando essa informação não for mais necessária (BISPO, 2018). Essa é uma das principais funcionalidades da LSTM, de modo que isso ocorre durante o tempo de treinamento através da execução do *forget gate*, na Figura 3.10 é ilustra os principais elementos da estrutura de uma célula LSTM.

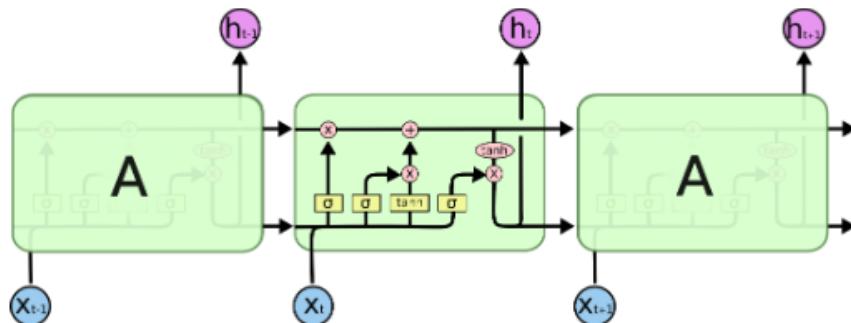
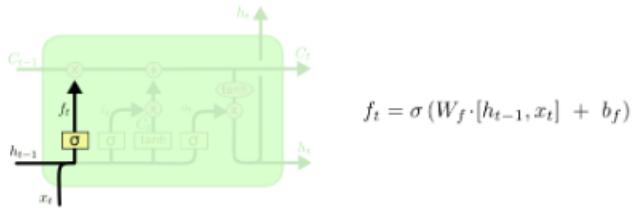


Figura 3.10 – Estrutura de uma célula LSTM (BISPO, 2018)

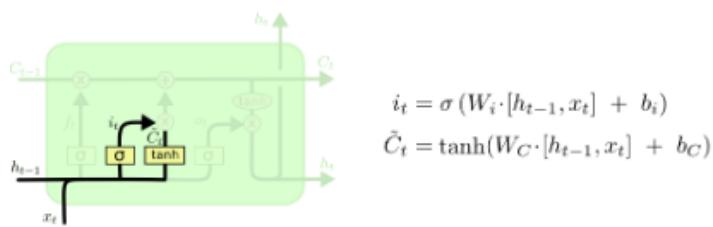
Uma célula LSTM é composta por três gates internos que controlam diferentes comportamentos: *input gate*, *forget gate* e *output gate*. Importante evidenciar que uma das funções desse tipo de célula é que os *gates* internos têm uma sigmoid (σ) como função de linearidade para melhor controlar seu fluxo de dados.

O *forget gate*, vide Figura 3.11, controla a entrada da célula C_t de forma a determinar quais dos valores de índices do vetor de saída da célula anterior C_{t-1} serão mantidos, através de sua função (σ), que retorna valores no intervalo entre 0 e 1. W_f e b_f são os pesos e o valor bias para o *input gate*, respectivamente (BISPO, 2018).

A LSTM tem a aptidão de realizar o cálculo de quais dados de entrada serão armazenado ou atualizado. Como é possível observa pela Figura 3.12, essa funcionalidade é dividida em duas etapas. Sendo que primeiramente a função (σ) irá decidir quais

Figura 3.11 – LSTM: *Forget Gate* (BISPO, 2018)

das informações de C_{t-1} irá atualizar, no *forget gate* essa etapa ocorre para determinar qual informação será esquecida. Logo após, um novo valor de entrada (\tilde{C}_t) é calculado pela função tanh(Equação 3.5), para então ser multiplicado com vetor resultante da primeira etapa. Vale ressaltar que as operações i e \tilde{C} têm seus próprios parâmetros, onde também podem ser aprendidos pela rede durante o treinamento (BISPO, 2018).

Figura 3.12 – LSTM: *Input Gate* (BISPO, 2018)

Após esses processos, a LSTM tem os dados suficientes para atualizar ou esquecer o valor de C_t , entretanto, para essa ação poder acontecer ela precisa esquecer/apagar as informações irrelevantes, sendo que isso ocorre através da Equação 3.20.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.20)$$

Segundo Bispo (BISPO, 2018), duas etapas são necessários para produzir a saída h_t da célula. Sendo a primeira etapa a realização da função (σ) que determina quais elementos do vetor C_t irá fazer parte da saída, o_t na Figura 3.13). Posteriormente, o estado C_t é submetido a função tanh(Equação 3.5), para então ser multiplicado pelo o_t .

A escolha das redes neurais recorrentes é apoiada na literatura. Trabalhos semelhantes ao proposto aqui, no qual há a utilização de um EEG e um componente de aprendizado de máquina, incluem (MICHELLI; ACHARYA; MOLINARI, 2019), no qual foi avaliado o uso de redes neurais recorrentes para classificar o estagio de sono a partir dos dados coletados com EEG; e (ALHAGRY; FAHMY; EL-KHORIBI, 2017), onde

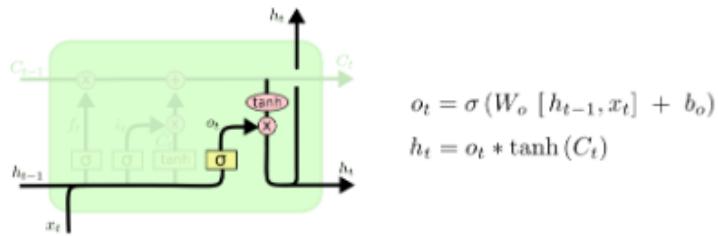


Figura 3.13 – LSTM: Output Gate (BISPO, 2018)

foi avaliada a mesma arquitetura de redes neurais para a tarefa de reconhecimento de emoções.

3.6 Arquiteturas e Modelos

De modo padronizar o treinamento das redes, foi escolhido para as três arquiteturas a mesma função de otimização, no caso a escolhida foi Stochastic Adam por possuir vantagens para o aprendizado de problemas com grandes número de parâmetros e apropriado para problemas com gradientes muito ruidosos ou esparsos (KINGMA; BA, 2014).

Para a arquitetura da MLP desse projeto foi utilizado o trabalho (BALAKRISHNAN; PUTHUSSEYAPADY, 2005) como base, com variações voltadas ao número de parâmetros, na utilização da tangente hiperbólica como função de ativação e na adição da função de *softmax* na saída da rede. A implementação foi feita com a biblioteca PyTorch (PASZKE et al., 2017), ver implementação do programa 3.1.

Listing 3.1 – Perceptron de Múltiplas Camadas

```

class Mlp(nn.Module):
    def __init__(self, device=None):
        super(Mlp, self).__init__()
        self.dense = nn.Sequential(
            nn.Linear(112, 32),
            nn.Tanh(),
            nn.Linear(32, 32),
            nn.Tanh(),
            nn.Linear(32, 3),
        )

    def forward(self, x):
        x = self.dense(x)
        return F.softmax(x, dim=1)

```

De modo a seguir a arquitetura tradicional da CNN proposta por (LECUN; BEN-GIO; HINTON, 2015), para este trabalho foram utilizadas camadas de convolução com *max pooling*, e como variação da arquitetura de LeCun foi adicionado o *batch nor-*

malization para diminuir o efeito do *internal covariate*. A implementação da rede em PyTorch (PASZKE et al., 2017) pode ser visto pelo programa 3.2.

Listing 3.2 – Rede Neural Convolucional

```

class Cnn(nn.Module):
    def __init__(self, device=None):
        super(Cnn, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(3, 8, 4, 1, 1),
            nn.BatchNorm2d(8, False)
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(8, 16, 4, 1, 1),
            nn.BatchNorm2d(16, False),
            nn.MaxPool2d(2)
        )
        self.dense = nn.Sequential(
            nn.Linear(8 * 8 * 4, 32),
            nn.Tanh(),
            nn.Linear(32, 32),
            nn.Tanh(),
            nn.Linear(32, 3),
        )

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(x.size(0), -1)
        x = self.dense(x)
        return F.softmax(x, dim=1)

```

Para esse trabalho foi utilizando a arquitetura de RNN proposta por (ALHAGRY; FAHMY; EL-KHORIBI, 2017) como base, sendo que possui como diferenças a retirada do *dropout* (SEMEINIUTA; SEVERYN; BARTH, 2016) e da função de ativação ReLu (Equação 3.3). Assim, a rede é composta de uma LSTM com duas camadas empilhadas, seguida de uma camada linear simples acoplada a uma saída *softmax*. A implementação da rede em PyTorch (PASZKE et al., 2017) pode ser visto pelo programa 3.3.

Listing 3.3 – Rede Neural Recorrente

```

class Rnn(nn.Module):
    def __init__(self):
        super(Rnn, self).__init__()
        self.lstm = nn.LSTM(11, 16, 2, batch_first=True)
        self.fc = nn.Linear(16, 3)

```

```
def forward(self, x):
    x, _ = self.lstm(x)
    x = x[:, -1, :]
    x = self.fc(x)
    return F.softmax(x, dim=1)
```

Neste capítulo foram abordados os conceitos sobre AM dando ênfase ao aprendizado supervisionado por classificação, relacionando com os métodos adotados pelo trabalho que são: Perceptron de Múltiplas Camada, Rede Neural Convolucional e Rede Neural Recorrente.

4 AQUISIÇÃO E PREPARAÇÃO DOS DADOS

Com o intuito de validar a proposta deste trabalho e tornar os experimentos reprodutíveis, nas seções a seguir são descritas as etapas de coleta de dados com os sensores de EEG, processamento de sinais e a metodologia de desenvolvimento do *dataset*. O protocolo experimental dos comandos, por sua vez, na qual foram utilizados para manipular um agente em ambiente virtual na plataforma *Robot Operating System* (ROS) serão apresentados na Seção 5.2.

4.1 Aquisição de Dados

Para a realização da captura dos impulsos cerebrais foi utilizado um EEG *EMOTIV EPOC+ 14 Channel*, da empresa EMOTIV (Figura 4.1). Esse equipamento possui frequência de amostragem de 128 Hz com conversor analógico-digital de 16 bits com 14 canais de eletrodos: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 e AF4 (EMOTIV, 2019). O mesmo equipamento já foi utilizado em outros trabalhos na literatura, tanto pela alta sensibilidade quanto o custo atrativo (ROQUE; CEZAR et al., 2017).



Figura 4.1 – *EMOTIV EPOC+ 14 Channel* (EMOTIV, 2019)

Para transmissão e captura dos dados obtidos pelo equipamento para o computador utilizou-se a biblioteca de código fonte aberto EMOKIT (BROCIOUS; MACHULIS, 2019).

Para a aquisição dos sinais de EEG, geralmente é obedecida a regra do sistema internacional 10-20(Subcapítulo 2.2.2). Além disso, a aquisição dos sinais de EEG é realizada por meio de amplificadores diferenciais característicos de cada modelo de EEG, podendo ser de quatro tipos diferentes: bipolar, bipolar unida, unipolar (ou monopolar) e unipolar com referência média (ADUR et al., 2008); (CARVALHO, 2008), respectivamente conforme é apresentado na Figura 4.2.

Em virtude de entender melhor o sensor utilizado, será apresentado brevemente as características e vantagens de cada tipo de configuração. A configuração bipolar realiza a medição da diferença de potencial que existe entre dois pontos, po-

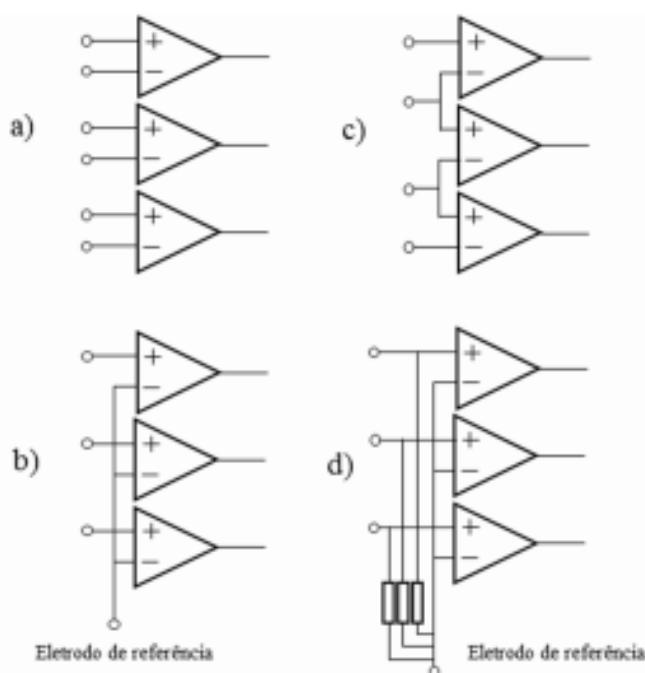


Figura 4.2 – Configurações dos amplificadores usados para aquisição de EEG, sendo a) bipolar, b) monopolar, c) bipolar unida e d) unipolar com referência média. (CANTARELLI; JÚNIOR; JR, 2016)

dendo captar o sinal de um lugar específico. Na configuração monopolar, é inserido um eletrodo de referência, geralmente no lóbulo da orelha ou no queixo, a qual capta sinais distribuídos em uma área maior do escopo, porém possuem maior aquisição de ruídos. Todavia a configuração bipolar unida, pode-se acompanhar a evolução e a transmissão do sinal de EEG por entre as regiões do escopo; e por fim, existem amplificadores que acoplam um resistor na região do eletrodo de referência para oferecer proteção ao indivíduo (CANTARELLI; JÚNIOR; JR, 2016), para esse trabalho ao se usar o EEG *EMOTIV EPOC+ 14 Channel* ele possui sua configuração baseada no monopolar.

Porém, como todo sinal de origem biológica é sujeito a ruídos e interferências, além dos apresentados pelo equipamento, o mesmo ocorre com o sinal de EEG (CANTARELLI; JÚNIOR; JR, 2016). Segundo Carvalho (2008) as interferências são classificadas de três formas: As geradas pelo paciente, na qual pode ser ocasionadas devido a movimentações da cabeça ou do corpo, por bio sinais (pulsação arterial) ou até mesmo por movimentação dos olhos; As geradas na interface eletrodo-pele, devido a movimentação dos eletrodos relativos ao couro cabeludo ou por má aplicação dos eletrodos; E as geradas pelo ambiente, como interferência da redes ou equipamentos eletrônicos. De modo a atenuar a influência desses ruídos, foi desenvolvidos pelos autores deste trabalho protocolos experimentais voltados a conscientização do usuário de tais interferências ocasionadas por ele e de como amenizar tais ruídos, assunto que será abordado na Subcapítulo 5.1, além disso foi utilizado durante os teste ambientes

controlados.

Com o propósito de evitar a ocorrência de falsos positivos em razão de possíveis ruídos elétricos provocados pela movimentação dos olhos, foi planejado pelos autores um protocolo de coleta de dados, denominado fase de treinamento experimental, com duração de aproximadamente vinte e cinco minutos com diferentes cenários, na qual contemplam situações diversas, sendo que esse modelo foi proposto com o objetivo de obter uma rede neural mais genérica em relação ao comportamento das ondas cerebrais em relação aos comandos de controle.

Alinhado com a literatura está a proposta de utilização de um protocolo com cenários variados durante a fase de treinamento experimental, sendo que como exemplo é possível citar os trabalhos de Ferreira (2008), que utiliza um protocolo baseado em olhos abertos e fechados para o treinamento do seu BCI de cadeira de rodas, ou até mesmo o trabalho de Santos Filho (2010), na qual utiliza um protocolo de ações motoras diversas para compor seu *dataset* de treinamento de ondas cerebrais com o EEG.

Também é conhecido na literatura que a utilização de ondas como forma de entrada da rede neural obtém melhores resultados de performance quando comparado a utilização de dados brutos recebidos do EEG, como exemplo de comparação desses dois tipos de técnicas temos o trabalho de Brodu (2011), que realiza a comparação de diferentes técnicas para extrair as informações de dados coletados de um EEG, sendo que Brodu demonstra em seu trabalho, em média, um algoritmo de aprendizado de máquina se comporta de modo superior com fim de extrair as informações de tarefas de ações motoras ao ser comparado a dados de sinais brutos de EEG.

4.2 Pré-Processamento dos Dados

A estratégia de pré-processamento adotada por esse trabalho é dividida em dois momentos, na qual na primeira etapa é realizada a transformada de Fourier, com o propósito de realizar a separação dos dados obtidos a partir do EEG em faixas de ondas, e no segundo momento será realizado o tratamento estatístico das ondas, assim obtendo as médias aritméticas e ponderadas.

O uso do tratamento matemático baseado em definições no domínio da frequência e na determinação dos espectros é um método de análise utilizada para sinais bioelétricos, sendo que isso ocorre devido ao fato de alguns eventos fisiológicos apresentam frequências características, como é o caso das ações motoras (CANTARELLI; JÚNIOR; JR, 2016). Dentre os estudos envolvendo a análise espectral de sinais bioelétricos coletados por EEG, podemos citar: Pinna (1994), na qual em seu trabalho fez uma comparação entre a Transformada Rápida de Fourier e o método Auto-

Regressivo; Chen (2000), na qual utilizou a Transformada Rápida de Fourier para sinais de batimento cardíacos; Brennan (2010), onde propôs uma análise da transformada de Fourier discreta e Minami (1999), que usou diretamente a Transformada Rápida de Fourier sobre o sinal de interesse (PEREIRA et al., 2003).

4.2.1 Transformada de Fourier

A transformada de Fourier em sua forma discreta, mostra algumas limitações ao trabalhar com sinais bioelétricos, que são, em essência, estocásticos não estacionários (PEREIRA et al., 2003). Entretanto, a transformada de Fourier é uma técnica renomada dentro da literatura para definir as componentes de frequência dos sinais. Como exemplos de trabalhos que demonstram as explicações teóricas em torno de Fourier temos : Akay (1996), Cohen (2019), Haykin (2001), Kraniauskas (2005).

Segundo a literatura, o algoritmo de Cooley-Tukey (1965) foi o que estabeleceu a linha de pensamento inicial para execução da transformada com a complexidade computacional da ordem de $O(n \log n)$. Todavia, a contribuição para a computação veio com a ideia que resultaria em um método eficaz em tempo mesmo para grandes conjuntos de amostras de sinais, algoritmo que atualmente é conhecido como Transformada Rápida de Fourier (do inglês *Fast Fourier Transform*, FFT).

Neste trabalho foi utilizada a implementação disponível na biblioteca *Scikit-learn* (PEDREGOSA et al., 2011), na qual suas implementações são baseadas em suas formas de funções otimizadas com foco em uma melhor eficiência.

4.3 Extração de Características

A extração de característica acontece a partir da aquisição de dados e um fluxo é definido para realizar a criação de um *dataset* específico para cada usuário.

Primeiramente vale lembrar que o EEG utilizado no projeto, modelo ilustrado pela Figura 4.1, possui uma frequência de 128 Hz e 14 canais assim fornecendo uma matriz 128x14 com 16 bits a cada leitura. Como as ondas cerebrais de interesse estão na faixa de 0 a 30 Hz (Tabela 1), essas informações coletadas são passadas pelo algoritmo da FFT, na faixa de frequência de 0 a 30 Hz, resultando em uma matriz de 30x14 no domínio da frequência, ver Figura 4.3.

Após a execução da transformada, é realizada a média ponderada e aritmética de cada onda, de modo que a matriz resultante tem as dimensões 14x4x2. Desse modo, cada instante de dados coletados são representados pela média ponderada e aritmética para cada um dos 14 canais do dispositivo e as 4 classificações de onda, ver Figura 4.4.

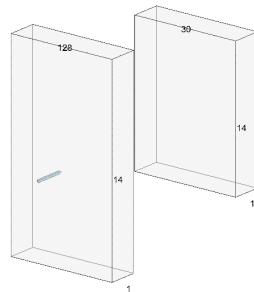


Figura 4.3 – Demonstração da conversão dos dados brutos na matriz 30x14

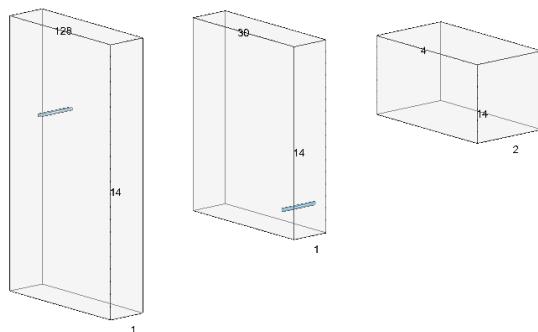


Figura 4.4 – Demonstração da conversão da matriz 30x14 para matriz 14x4x2

Esse processo de extração de características é realizado em dois momentos distintos no processo experimental: a criação do *dataset* e na etapa de teste, explicada posteriormente.

4.3.1 Arranjo experimental

O principal objetivo deste trabalho é avaliar as diferentes arquiteturas de redes neurais e analisar o desempenho delas na tarefa de traduzir comandos mentais em sinais de controle de dispositivos. Os dados obtidos a partir do EEG e pré-processados são transmitidos a um dos três modelos de redes neurais apresentados no Subcapítulo 3.6, de modo a aprender o padrão entre os comandos mentais e como a tradução deverá ser feita para sinais de controle.

O treino das redes neurais foi realizado usando os dados obtidos durante a fase de treino do experimento, protocolo que será abordado na Subcapítulo 5.1. Esta base de dados coletadas é composta por 6 ciclos de aquisição com 24956 amostras cada, totalizando 149736, lembrando que cada base de dados é única para cada voluntário. Neste trabalho para criação do *dataset* através do método explicado nas seções anteriores, foi utilizado uma técnica de sequências de 32 elementos com 128 valores que correspondem a um tempo de coleta de 0,15 segundos de dados pré-processados provenientes do EEG (LEITE et al., 2019), vide Figura 4.5. Com essa técnica, além de garantir que existem 128 dados que são os pré-requisitos para executar o processo

de coleta do espectro, Subcapítulo 2.3, também é garantindo a captura de picos de eventos do EEG. Com a adição dessa técnica o *dataset* resultante possui 2400 registros, 33% de cada classe, isso é 800 dados sem atividade motora, 800 dados atividade motora esquerda e 800 dados motora direita (LEITE et al., 2019).

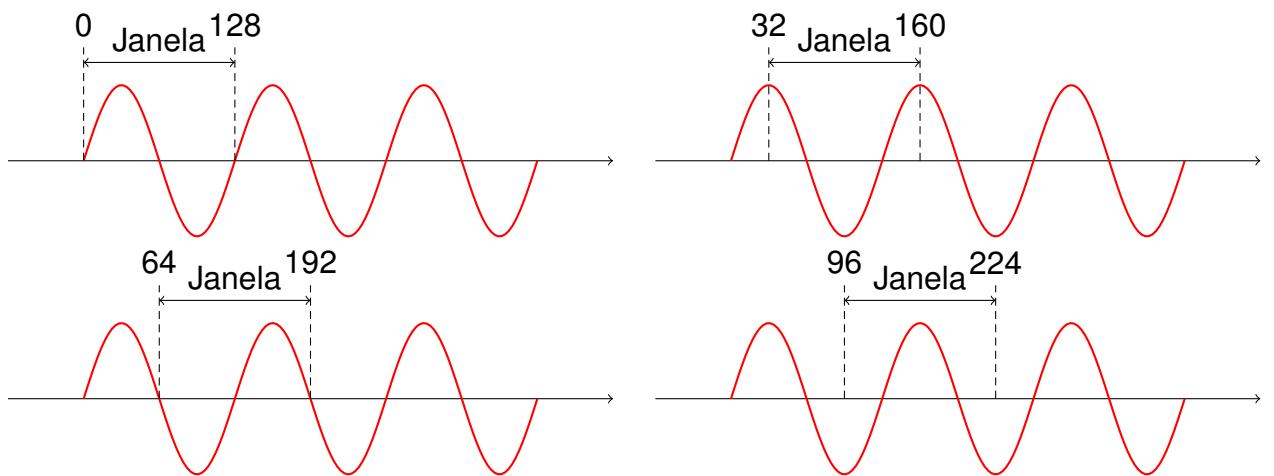


Figura 4.5 – Técnica de sequências de 32 elementos com 128 valores

Desse modo após a preparação dos atributos do *dataset*, será adicionado a cada registro o movimento que o usuário está executando durante o tempo de coleta. Sendo assim, a cada amostra do *dataset* a técnica de aprendizado de máquina terá como objetivo gerar uma resposta que corresponde a um dos movimentos: ação motora com a mão esquerda ou ação motora com a mão direita ou nenhuma ação motora.

As redes foram avaliadas utilizando *hold-out* com uma parcela de dados para validação. As três divisões de dados (treino, teste e validação) foram feitas de forma estratificada, de modo a manter a proporção da ocorrência de cada classe de maneira igual entre as divisões. Para treinamento foram separadas 1536 amostras (64%) e a cada época o desempenho da rede era medido utilizando 480 amostras (20%) diferentes das utilizadas no treino. Ao terminar as épocas de treinamento (25 épocas para MLP, 25 épocas para CNN e 25 épocas para RNN), uma parcela diferente de dados (validação) era utilizada para avaliar o resultado final do treinamento, contendo 384 amostras (16%) (LEITE et al., 2019).

Vale ressaltar que esse processo do sistema de aquisição dos dados juntamente com pré-processamento e treinamento das redes ocorre entre a fase de treino e teste experimental, fases que serão abordadas no capítulo seguinte.

5 DESENVOLVIMENTO

Desta forma, este projeto utiliza o EEG para coletar os impulsos cerebrais de um usuário e, após processar esses dados, utilizá-los como *dataset* para treinar três algoritmos de AM, sendo eles: MLP, CNN e RNN, com o objetivo de realizar uma análise do desempenho dos algorítimos citados no controle de um objeto virtual.

Para elaborar esta análise, foi desenvolvido um ambiente virtual em 2D, a fim de realizar simulações, onde o usuário através dos impulsos cerebrais coordena uma tartaruga, fazendo-a rotacionar para a esquerda, caso determinadas ações motoras sejam realizadas com a mão esquerda, rotacionar para a direita, caso determinadas ações motoras sejam realizadas com a mão direita, ou siga para frente, caso nenhuma ação motora seja realizada.

No primeiro protótipo do experimento, o usuário conseguiria fazer com que a tartaruga fosse para frente, caso o mesmo estivesse concentrado, e para trás, caso o mesmo estivesse relaxado, porém o comando para trás foi removido e a ação para conduzir a tartaruga para frente modificado, por motivos que serão mostrados adiante.

Além disso, as ações motoras para rotacionar o objeto virtual eram, apenas, o estalar dos dedos, entretanto, com o decorrer do projeto, mais ações motoras foram adicionadas afim de evitar a especialização dos modelos neurais, ações que serão descritas no próximo subcapítulo.

Desse modo, esse capítulo ficou dividido em duas seções, sendo que o primeiro se concentra na descrição da metodologia aplicada sob a fase de treinamento experimental, onde o usuário treina a rede a interpretar suas ações motoras através do EEG. E no segundo subcapítulo se encontra a sistematização da fase de teste experimental, na qual o usuário é desafiado a controlar um agente virtual sob um ambiente de simulação projetado sob o *Robot Operating System* (ROS).

5.1 Treinamento Experimental

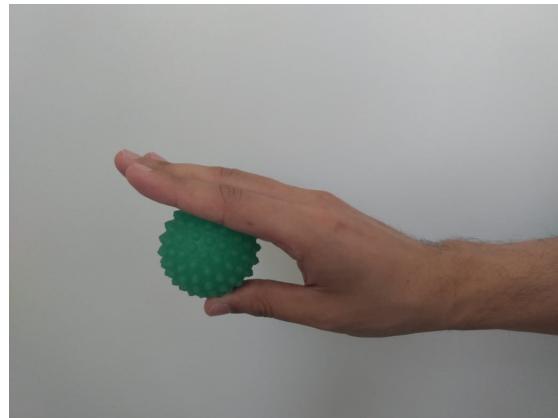
Para um usuário realizar a simulação através dos algoritmos de redes neurais utilizadas nesse trabalho, é necessário que seus impulsos cerebrais sejam coletados e pré-processados na geração do *dataset*, por essa razão cada usuário realizava uma vez o protocolo de coleta de dados com EEG, de modo que esse procedimento é realizado a partir de seis ciclos de aquisição em situações diferentes, afim de evitar a especialização dos modelos e garantir uma generalização da rede em cima dos dados coletados.

Esse protocolo se baseia na combinação das atividades em questão ocorrerem com olhos abertos ou fechados, sentado ou em pé além das ações motoras diversas:

1. Primeiro ciclo de aquisição:

Participante sentado com os olhos abertos, apertando uma bola estimuladora sensorial com a mão dominante em formato de pinça (Figura 5.1) e movimentando a palma da outra mão para cima e para baixo (Figura 5.2).

Figura 5.1 – Mão em formato de pinça



Fonte: Elaborado pelo autor.

Figura 5.2 – Palma da mão para cima e para baixo



Fonte: Elaborado pelo autor.

2. Segundo ciclo de aquisição:

Participante Sentado com os olhos fechados, apertando uma bola estimuladora sensorial com a mão dominante em formato de pinça (Figura 5.1) e movimentando a palma da outra mão para cima e para baixo (Figura 5.2).

3. Terceiro ciclo de aquisição:

Participante em pé com os olhos abertos, apertando uma bola estimuladora sensorial com a mão dominante, com o polegar para fora apontando para cima e rotacionando a mão para dentro(Figura 5.3) e estalando os dedos com a outra mão (Figura 5.4).

Figura 5.3 – Apertando bola estimuladora com polegar para cima rotacionando para dentro



Fonte: Elaborado pelo autor.

Figura 5.4 – Estalando os dedos



Fonte: Elaborado pelo autor.

4. Quarto ciclo de aquisição:

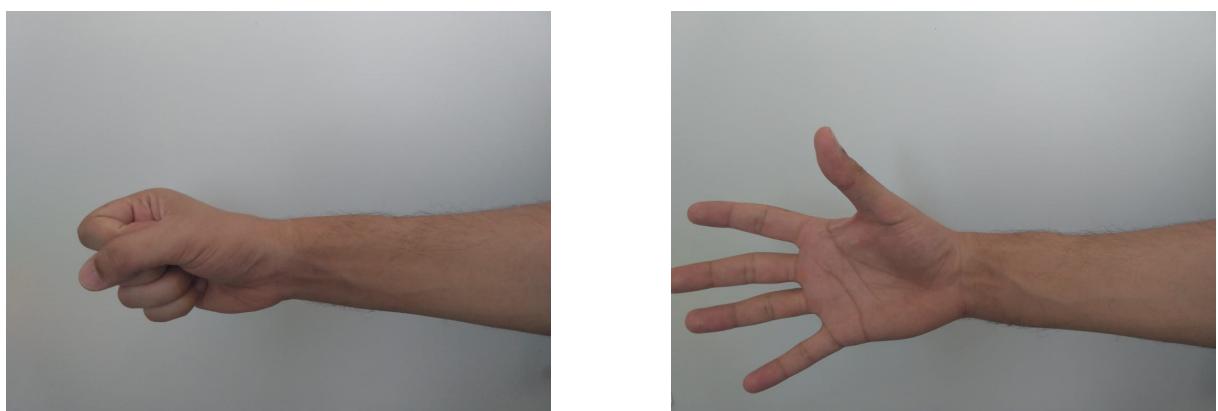
Participante sentado com os olhos abertos, apertando uma bola estimuladora sensorial com a mão dominante (Figura 5.5) e abrindo e fechando a outra mão (Figura 5.6).

Figura 5.5 – Apertando a bola estimuladora



Fonte: Elaborado pelo autor.

Figura 5.6 – Abrindo e fechando a mão.



Fonte: Elaborado pelo autor.

5. Quinto ciclo de aquisição:

Participante em pé com os olhos fechados, apertando uma bola estimuladora sensorial com a mão dominante, com o polegar para fora apontando para cima e rotacionando a mão para dentro (Figura 5.3) e estalando os dedos com a outra mão (Figura 5.4).

6. Sexto ciclo de aquisição:

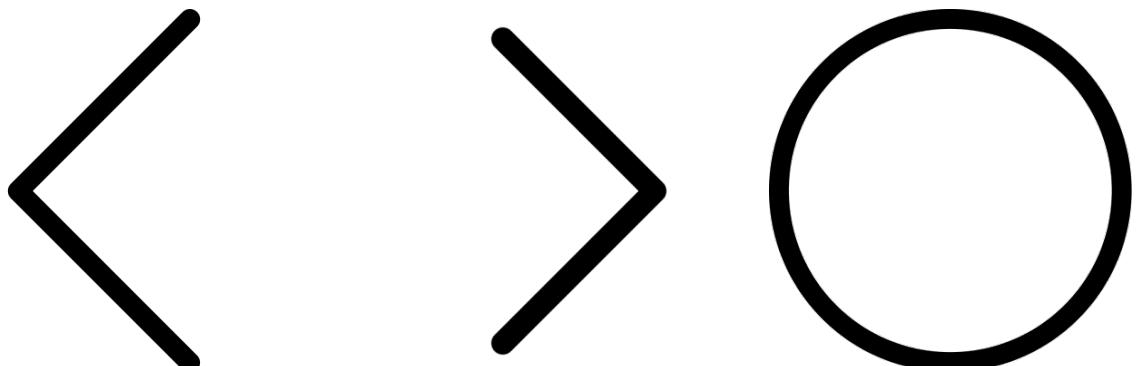
Participante sentado com os olhos fechados, apertando uma bola estimuladora sensorial com a mão dominante em formato de pinça (Figura 5.1) e movimentando a palma da outra mão para cima e para baixo (Figura 5.2).

Os movimentos utilizados no protocolo, com exceção do estalar de dedos, foram escolhidos com o intuito de estimular o cérebro no momento da coleta e assim registrando os picos ocasionados nos eventos, pois são exercícios de terapia para as

mãos, o estalar de dedos foi escolhido arbitrariamente por apresentar bons resultados no começo do projeto.

Durante cada um dos ciclos de aquisição, o participante visualiza imagens que retratam uma ação motora voluntária: uma seta para a direita que representa ação motora da mão direita, uma seta para a esquerda que representa ação motora da mão esquerda e um círculo que representa nenhuma ação motora, Figura 5.7, vale ressaltar que durante a aparição de cada imagem ocorre três fases distintas, sendo a primeira uma amostragem de uma imagem preta, na qual representa o momento em que o participante pode relaxar, em um segundo momento a exposição de uma das três imagens citadas anteriormente entretanto com o fundo branco, esse momento serve como uma preparação para o participante saber qual a ação que ele irá tomar, sendo que durante essa fase os dados ainda não estão sendo coletados. E como última fase é apresentado a imagem com um fundo verde, na qual o participante deve executar a ação proposta, sendo que somente nesse momento os dados coletados do EEG estão sendo salvos para futuramente serem pré-processados. De modo a utilizar o mesmo padrão de amostragem das imagens nos ciclos de aquisição que ocorrem com os olhos fechados, foi colocado um sinal sonoro no inicial da fase da tela com fundo branco, desse modo o usuário sabe qual momento ele pode abrir os olhos para verificar qual ação deve tomar para só então fechar os olhos novamente.

Figura 5.7 – Direções mostradas durante os ciclos de aquisição.

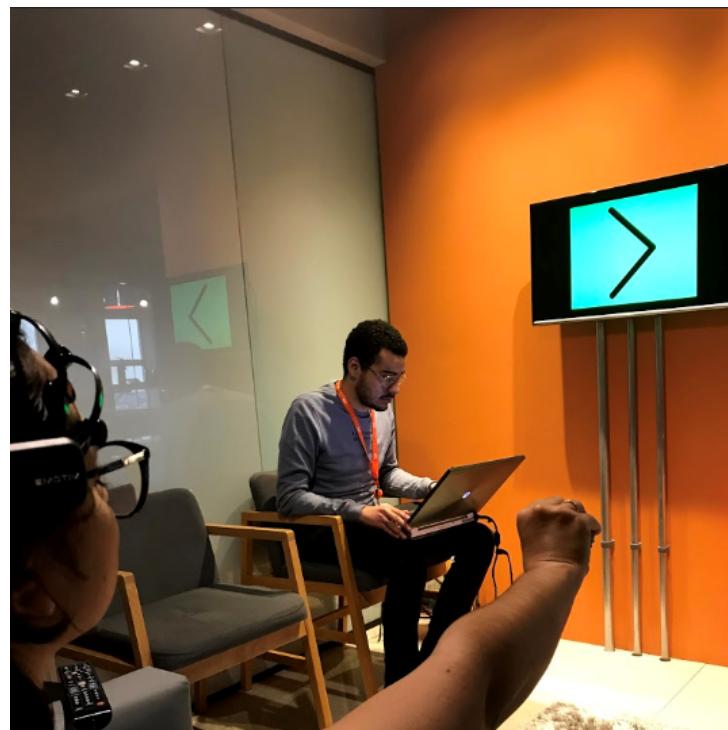


Fonte: Elaborado pelo autor.

Cada participante era conduzido para uma sala onde havia um sofá e uma televisão, em seguida a solução de base salina é colocada nos eletrodos do EEG, de modo a melhorar a aquisição dos impulsos, e posicionado no participante, que era instruído a ficar sentado, no sofá, em uma posição confortável, mantendo-a durante todo o processo, também era solicitado que ficasse com os braços bem relaxados, movimentando-os de acordo com a imagem mostrada na televisão (Figura 5.8), e evitasse falar ou realizar qualquer movimento, além dos informados, durante a coleta.

O protocolo leva, em média, 38 minutos para ser realizado, sendo que apenas

Figura 5.8 – Espaço físico da realização do experimento.



Fonte: Elaborado pelo autor.

durante a fase de aquisição que a imagem com fundo verde aparece na tela que é salvo no computador os dados do EEG, desse modo aumenta a qualidade e pureza dos dados que serão futuramente pré-processamento, como descrito no Capítulo 4

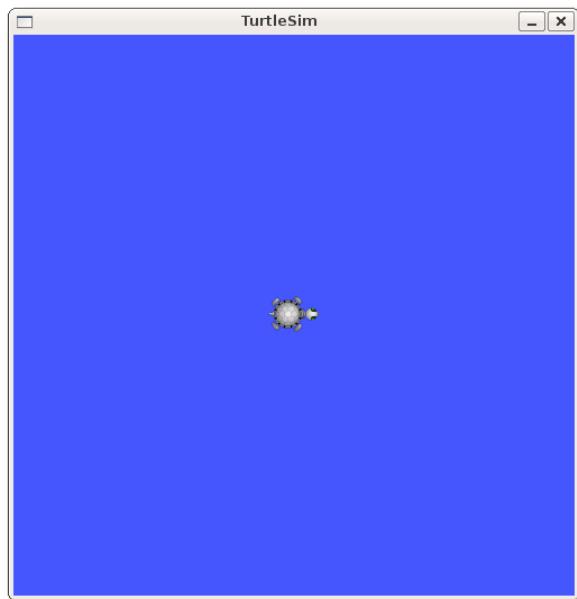
Após o final da fase de treinamento experimental é realizado a criação do *dataset*, o mesmo é enviado para realizar o treinamento dos algoritmos de AM, procedimento abordado no Capítulo 4, que seguem a arquitetura descrita no Subcapítulo 3.6. O procedimento completo, criação da base de dados e o treinamento das três redes neurais, leva cerca de 7 minutos, após isso as redes neurais já estão prontas para serem utilizadas na fase de teste experimental.

5.2 Teste Experimental

Com o treinamento dos algoritmos de AM finalizados, o participante pode realizar o teste, que consiste em uma simulação de fundo azul com uma tartaruga no centro, ver Figura 5.9, os impulsos cerebrais do usuário estarão sendo coletados, através do EEG, enviados para o algoritmo de aprendizado de máquina que irá predizer qual direção o mesmo está executando e encaminhará como comando para movimentar a tartaruga, essa simulação é visualizada na televisão, como pode ser visto na Figura 5.10.

Ao dar início ao teste, uma segunda tartaruga é instanciada, sendo que o ob-

Figura 5.9 – Interface Inicial da Simulação



Fonte: (ROS, 2019).

Figura 5.10 – Simulação sendo mostrada na TV

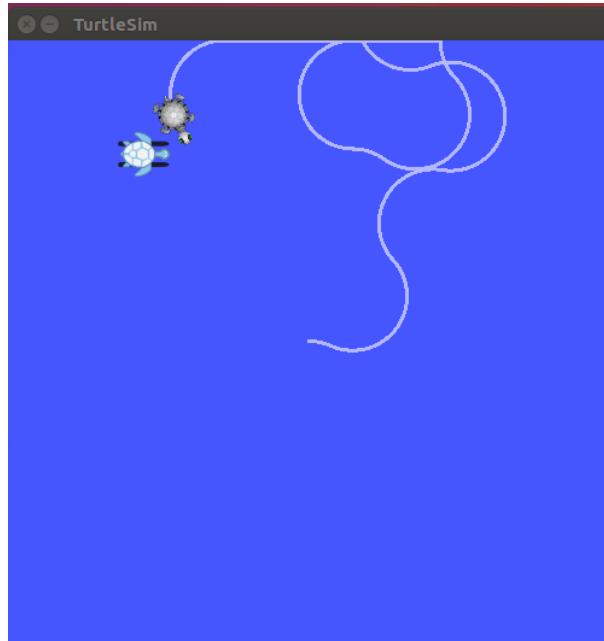


Fonte: Elaborado pelo autor.

jetivo do usuário é conduzir sua tartaruga, a central, até a outra no menor tempo possível, além disso a trajetória que a tartaruga percorre é marcada, com o intuito de analisar a trajetória percorrida pelo participante, vale ressaltar que para fins de análise é retirada uma foto da tela apos a finalização do teste, a Figura 5.11 mostra uma simulação que foi concluída com sucesso.

O participante tentará chegar na outra tartaruga três vezes, tendo 90 segundos para alcançar o objetivo, sendo que esse tempo foi escolhido arbitrariamente pelos autores de modo a não ocasionar a frustração do participante caso ele fique muito tempo na tentativa de sucesso. Mesmo se a simulação atual não seja concluída com sucesso, o participante não conseguiu chegar ao objetivo, é recolhido *feedback* do

Figura 5.11 – Simulação finalizada com sucesso.

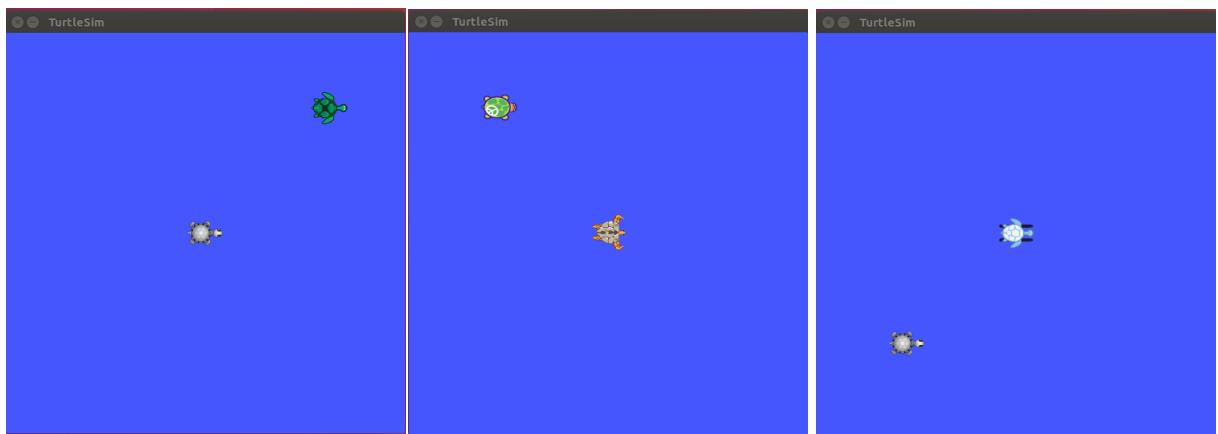


Fonte: Elaborado pelo autor.

participante em ambas situações, sucesso e não sucesso, sobre o que ele sentiu em relação ao sistema de controle baseado naquele algoritmo.

Este processo será realizado com cada um dos algoritmos de AM, totalizando nove simulações por participante. Na primeira simulação de cada algoritmo, a outra tartaruga aparece próxima do canto superior direito, enquanto na segunda aparece próximo do canto superior esquerdo e a última aparece no canto inferior esquerdo como mostrado nas Figuras 5.12.

Figura 5.12 – Locais onde a segunda tartaruga aparece



Fonte: Elaborado pelo autor.

Para o desenvolvimento da simulação realizada neste projeto, utilizou-se o *Robot Operating System* (ROS), um *framework* de código fonte aberto, permitindo que as

pessoas que utilizam essa plataforma compartilhem suas ideias e códigos de forma simples, desse modo, os desenvolvedores que utilizam esse *framework* não precisam de muito tempo para elaborar a infraestrutura do software (QUIGLEY; GERKEY; SMART, 2015).

Existem diversos recursos dentro do ROS para auxiliar os desenvolvedores, entre eles há uma interface de transmissão de mensagens entre processos, que facilita a comunicação entre os programas e processos, além disso o ROS possui recursos similares ao sistema operacional, permitindo o controle de dispositivo de baixo nível e abstração de hardware (JOSEPH; JOSEPH; PAO, 2018).

A *framework* também permite a programação de linguagem de alto nível, como C++, Python e Lisp (JOSEPH; JOSEPH; PAO, 2018), a mesma também tem integração com outras bibliotecas de código aberto como o Gazebo, OpenCV, MoveIt (ROS, 2019). O ROS possui, também, ferramentas voltadas para a simulação como o Gazebo, citado anteriormente, e o *Turtlesim*, um simulador 2D de uma tartaruga (ROS, 2019).

Para conseguir utilizar todos esses recursos é importante entender como o ROS funciona, primeiramente um processo que utiliza os recursos do ROS é chamado de ROS *node*, além disso tem-se também o ROS *master*, um programa responsável por auxiliar na comunicação entre os ROS *nodes* (JOSEPH; JOSEPH; PAO, 2018).

Uma forma dos nodes se comunicarem é através de tópicos, barramentos que possuem nomes para identificá-los. Um node pode enviar informação para o tópico, chamado "publicar", ou receber as informações que foram enviadas, chamado "se inscrever" é possível transportar essas mensagens utilizando os protocolos TCP/IP e UDP (ROS, 2019).

Ao se inscrever em um tópico, o node pode definir uma função que será executada sempre que algo for publicado no tópico, um node pode se inscrever ou publicar em quantos tópicos forem necessários, analogamente um tópico pode receber informação de vários nodes e encaminhá-las para vários nodes (ROS, 2019).

Outra maneira dos nodes se comunicarem é via serviços, uma função que se encontra dentro de um determinado node, denominado *server*. Essa função pode ser chamada por outros nodes, estes chamados de node cliente (JOSEPH; JOSEPH; PAO, 2018), os pacotes do ROS fornecem diversos serviços para serem utilizados (QUIGLEY; GERKEY; SMART, 2015).

Dessa forma, o ROS *master* é responsável por localizar os *nodes publisher* e *subscriber* nos tópicos e serviços, em seguida o *master* informa para os nodes individuais com quem estão se comunicando, a partir de então eles se comunicam entre si (ROS, 2019).

Entre as diversas ferramentas do ROS, tem-se o *Turtlesim*, um simulador de tartaruga em 2D como dito anteriormente, o intuito desta ferramenta é ser simples para que iniciantes na plataforma possam se familiarizar com o ROS (ROS, 2019).

O simulador fornece uma interface gráfica, mostrada na Figura 5.11, e através de tópicos e serviços é possível movimentar a tartaruga, ler a posição atual dela, removê-la, ou até mesmo instanciar uma outra tartaruga (ROS, 2019).

Dessa forma, para a construção do ambiente foi criado um node nomeado de *talker* no ROS, responsável por receber via *socket* as previsões dos algoritmos de aprendizagem de máquina. Ao recebê-las o mesmo as encaminha para o node *listener* que irá converter o comando dado em velocidade linear e angular, após isso o mesmo envia as velocidades para o tópico *turtle1/cmd_vel*, onde *turtle1* representa o nome de uma tartaruga dentro do *Turtlesim*. A primeira tartaruga é inscrita neste tópico e ao receber as informações atualiza suas velocidades.

Além disso, foi criado um node com o nome de *manager* que ao iniciar a simulação instancia a segunda tartaruga, para fazer isso o mesmo faz uma chamada do serviço *spawn* do *Turtlesim* passando como parâmetro a posição e o nome da nova tartaruga.

O Node manager também é responsável por verificar quando o usuário atingiu o objetivo, por isso ele é inscrito no tópico *turtle1/pose*. Constantemente o *turtle1* publica a posição da tartaruga neste tópico, sempre que a posição é atualizada o manager verifica se o objetivo foi atingido.

Para facilitar a coleta dos resultados da simulação, o manager fica responsável por cronometrar / medir o tempo da simulação e, ao finalizar, salvar em um arquivo de texto simples o tempo que o usuário levou na simulação e consequentemente se o objetivo foi atingido, além disso o mesmo salva uma captura da tela, na qual representa o percurso que o participante fez na simulação.

Caso o usuário chegue no objetivo, o manager identifica e coleta os resultados e, em seguida, informa para o node *listner* que a simulação foi finalizada, para que o mesmo pare de enviar as velocidades para o *turtle1*.

Visando organizar o armazenamento desses resultados foi desenvolvido o node *keyCatch*. Ao iniciar o ROS esse node pede para informar, via entrada padrão, o nome do participante que irá realizar a simulação e qual algoritmo de AM está sendo utilizado, com esses dados o mesmo cria uma pasta de resultados dentro do diretório do ROS e encaminha para o Node manager o caminho da pasta, para que o mesmo saiba onde salvar os resultados.

Em seguida, o node *KeyCatch* fica aguardando a entrada padrão do carácter "i", quando a tecla é recebida, ele envia para o node manager para que o mesmo instancie

a segunda tartaruga e comece a cronometrar o tempo e o listner informando-o que o mesmo pode publicar as velocidades recebidas pelo Node talker na turtle1.

Para a execução da simulação utilizou-se o *roslaunch*, um arquivo de extensão *XML*, onde contém os nodes que serão executados pelo ROS e seus respectivos nomes, assim foi criado um *roslaunch* contendo todos os nodes citados anteriormente e o node do *Turtlesim*.

Assim sendo, é possível testar cada um dos algoritmos de aprendizagem de máquina com diversos usuários, salvando os resultados relevantes de cada simulação para poder realizar uma análise de desempenho de cada um deles.

6 RESULTADOS E DISCUSSÕES

Neste trabalho, quatro voluntários, classificados como sem disfunções neurológicas, realizaram o teste com cada um dos algoritmos três vezes, assim totalizando nove simulações. Nas Tabelas 2,3 e 4 é possível notar três colunas que significam respectivamente: quantidade de vezes que o participante em questão conseguiu finalizar o desafio proposto, atingir o alvo na tela em menos de 90 segundos; quantidade de vezes que o participante não concluiu o desafio; porcentagem de acurácia do modelo neural que foi utilizado no teste experimental em questão. Vale ressaltar, que cada tabela representa um dos algorítimos de aprendizagem de máquina utilizado no trabalho.

Tabela 2 – Resultados da simulação utilizando a MLP

ID's	Concluídas	Não Concluídas	Acurácia do Modelo
Voluntario 1	0	3	78%
Voluntario 2	1	2	60%
Voluntario 3	0	3	69%
Voluntario 4	1	2	63%

Tabela 3 – Resultados da simulação utilizando a CNN

ID's	Concluídas	Não Concluídas	Acurácia do Modelo
Voluntario 1	0	3	60%
Voluntario 2	2	1	59%
Voluntario 3	1	2	53%
Voluntario 4	2	1	43%

Tabela 4 – Resultados da simulação utilizando a RNN

ID's	Concluídas	Não Concluídas	Acurácia do Modelo
Voluntario 1	2	1	66%
Voluntario 2	2	1	60%
Voluntario 3	0	3	61%
Voluntario 4	1	2	62%

Ao analisar os resultados em questão é possível notar que o MLP apresenta *overfit* em seus modelos, mesmo possuindo uma acurácia elevada ao comparar com os demais algoritmos, seu desempenho durante o teste não foram positivos, uma vez que todos os voluntários sentiram grande dificuldade em controlar seu agente durante o uso da MLP.

Durante a etapa de teste experimental, a primeira simulação realizada com todos os voluntários foi com a MLP, em razão disso, vale levantar o questionamento em cima da baixa taxa de conclusão deste método, o fato dos participantes estarem se acostumando com o ambiente da simulação poder ter interferido nos resultados. De modo a resolver essa questão, em trabalhos futuros, será realizado o teste experimental com a ordem dos algoritmos trocada e verificar se a taxa de conclusão da MLP continuará baixa.

Ao finalizar todas as simulações, os participantes deram sua opinião sobre qual modelo se sentiram mais confortáveis e com maior controle do agente virtual, dos quatro participantes, dois disseram que preferiram o RNN, enquanto os outros dois preferiram o CNN. As Tabelas 5, 6, e 7 mostram o tempo que cada participante levou na simulação.

Tabela 5 – Tempo de realização das simulação com o MLP

ID's	Rodada 1	Rodada 2	Rodada 3
Voluntario 1	90.00s	90.00s	90.00s
Voluntario 2	90.00s	90.00s	06.18s
Voluntario 3	90.00s	90.00s	90.00s
Voluntario 4	05.47s	90.00s	90.00s

Tabela 6 – Tempo de realização das simulação com o CNN

ID's	Rodada 1	Rodada 2	Rodada 3
Voluntario 1	90.00s	90.00s	90.00s
Voluntario 2	54.53s	17.02s	90.00s
Voluntario 3	90.00s	90.00s	26.10s
Voluntario 4	12.85s	90.00s	37.54s

Tabela 7 – Tempo de realização das simulação com o RNN

ID's	Rodada 1	Rodada 2	Rodada 3
Voluntario 1	90.00s	40.94s	52.95s
Voluntario 2	13.27s	90.00s	59.28s
Voluntario 3	90.00s	90.00s	90.00s
Voluntario 4	72.50s	90.00s	90.00s

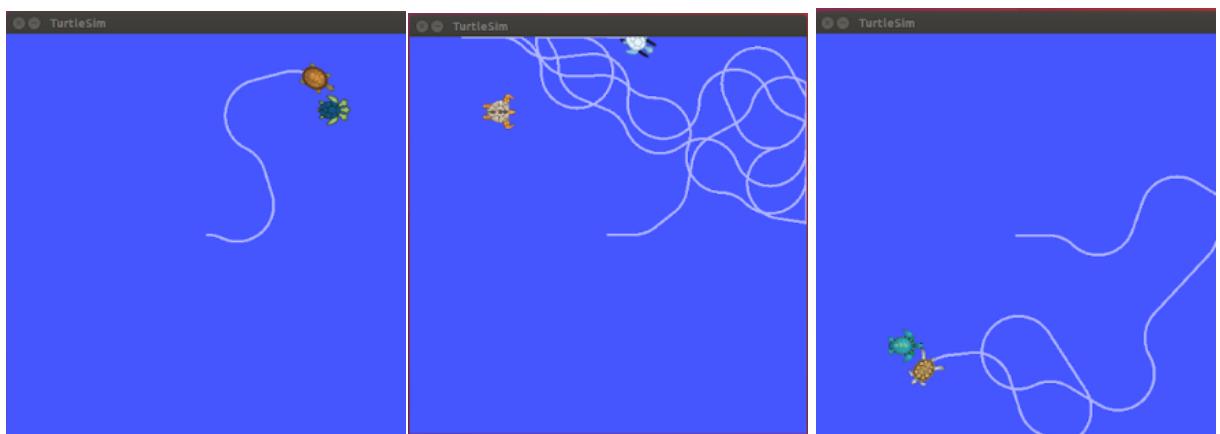
Ao observar os participantes realizando as simulações, foi notado que o estado emocional estava interferindo diretamente no controle do agente virtual, pois quando o participante atingia o objetivo, o mesmo apresentava em um estado de realização, feliz, por ter alcançado o objetivo e ao dar continuidade nas demais simulações o mesmo tinha dificuldades em realizar o controle do agente. Esse fenômeno é possível

ser notado na Figura 6.1, onde mostra a trajetória percorrida pelo quarto voluntário durante a simulação da CNN.

Como visto na Figura 6.1 o participante conseguiu atingir objetivo de forma rápida. Após esse êxito o participante demonstrou, fisicamente, estar feliz e ao tentar realizar a segunda simulação o mesmo teve dificuldade no controle, deixando-o frustrado, tornando mais difícil o controle do agente.

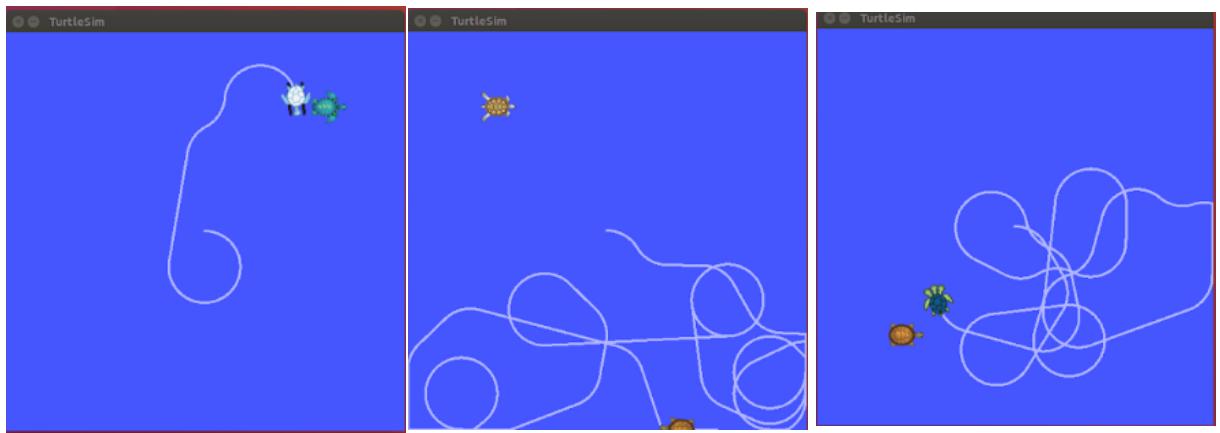
Ao finalizar a segunda simulação, foi orientado para que o participante fechasse os olhos e relaxasse e ao executar a terceira simulação o mesmo conseguiu conduzir a tartaruga até o objetivo. Este comportamento, também, foi notado com o voluntario dois, a imagem Figura 6.2 mostra os resultados obtidos por ele na simulação do RNN.

Figura 6.1 – Trajetória percorrida pelo quarto voluntario na simulação do CNN



Fonte: Elaborado pelo autor.

Figura 6.2 – Trajetória percorrida pelo segundo voluntario na simulação do RNN



Fonte: Elaborado pelo autor.

Ao coletar a opinião dos participantes no final de todas as simulações, os voluntários informaram que ao fechar os olhos e relaxar, obtiveram uma melhora no controle do agente ao longo da simulação. Vale ressaltar que esse fenômeno foi notado

na RNN e CNN, sendo que essa interferência do estado emocional foi retratado pelos voluntários de forma mais prejudicial no controle do agente durante o uso da RNN. Esse relato está alinhado com a literatura, onde devido ao fato de que a RNN utiliza os dados anteriores para predizer seu estado atual, uma vez que o estado emocional tenha interferido na predição esse erro será propagado em suas próximas predições. Sendo que esse fenômeno de retro-propagação não ocorre na CNN, na qual utiliza apenas o estado atual para predizer sua ação.

No inicio da construção do protótipo tinha sido incluso o comando para frente, sendo o usuário focado e o comando para trás sendo o usuário relaxado, entretanto esse último comando foi retirado do teste devido às observações citadas no parágrafo anterior. Pois para o usuário usar este comando ele precisava estar relaxado e a satisfação do usuário de ver o agente respondendo aos comandos dados fazia com que o mesmo entrasse num estado de satisfação/relaxamento, fazendo com que o agente começasse a andar para trás, causando uma frustração ao usuário e consequentemente dificultando o controle do sistema virtual.

Outra confirmação coletada com os *feedbacks* dos voluntários foi que conforme iam realizando as simulações, ficava mais fácil conduzir o agente, entretanto sempre que havia a troca para outro algoritmo de aprendizagem de máquina essa facilidade era perdida, mostrando que o usuário vai se acostumando com os controles conforme realiza as simulações, isso foi retratado pelo terceiro e quarto voluntário.

Quando o algoritmo de aprendizagem é trocado, os comandos que apresentavam boas respostas no anterior podem não responder corretamente, assim como os que não apresentavam boas respostas anteriormente, podem passar a responder de maneira mais eficiente, fazendo com que o participante precise se adaptar novamente com o controle.

O quarto voluntário sentiu que o algoritmo RNN tinha um *delay*, uma demora, no tempo de resposta do comando enviado, causando uma certa estranheza e frustração, tornando difícil concluir a simulação. Essa percepção ocorre devido ao fato do modelo neural da RNN ser mais complexo que o da CNN e MLP, como o voluntário realizou a simulação do RNN por último, sua mente estava acostumada com o tempo de resposta dos anteriores, tornando mais evidente esse *delay* ao controlar a RNN.

Além desses resultados, também foram realizadas três baterias de três simulações, utilizando um programa que escolhia aleatoriamente uma direção e enviava para a tartaruga, sempre que um comando era enviado existia 30% de chance da direção ser trocada, caso a direção não fosse trocada, o comando anterior era enviado novamente, o programa enviava um comando a cada 250 ms, este programa conseguiu concluir a simulação cinco vezes, 55.55% de acerto, mostrando que com 90 segundos para alcançar o objetivo havia o risco do usuário estar atingindo o objetivo de forma

accidental.

Um dos objetivos dos *feedbacks* dos participantes era identificar se o objetivo não foi alcançado por aleatoriedade, como no caso dos voluntários dois e quatro que no algoritmo MLP mesmo chegando uma vez no objetivo, ambos disseram que não se sentiram no controle da tartaruga.

Visando realizar uma comparação dos resultados obtidos pelos participantes com o de um sistema de controle independente, foram realizadas três baterias de três simulações, utilizando o teclado como forma de controle do agente virtual, a Tabela 8 mostra o tempo levado em cada simulação.

Tabela 8 – Resultados do sistema de controle independente

ID's	Rodada 1	Rodada 2	Rodada 3
Bateria 1	06.22s	07.65s	09.89s
Bateria 2	06.13s	07.59s	10.16s
Bateria 3	05.28s	06.60s	09.35s

De modo geral, após a coleta de todos os *feedbacks*, os voluntários um e dois preferiram o algoritmo RNN, enquanto os voluntários três e quatro preferiram o CNN, uma vez que obtiveram resultados positivos com esses respectivos algoritmos, os participantes disseram que se sentiram no controle da tartaruga.

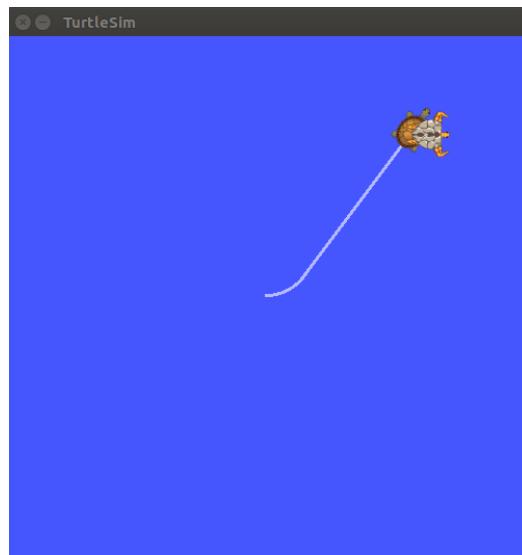
Analizando apenas os resultados obtidos na CNN e RNN é possível perceber que a primeira rodada é a que chega mais próxima dos resultados do controle independente, o voluntário quatro fez em 12.85s utilizando o CNN e o voluntário dois fez em 13.27s utilizando o RNN, a Figura 6.3 mostra a trajetória percorrida nessas simulações, enquanto na Figura 6.4 mostra a trajetória percorrida na primeira rodada da segunda bateria de simulações utilizando o sistema de controle independente.

Figura 6.3 – Percurso percorrido pelos participantes na primeira rodada.



Fonte: Elaborado pelo autor.

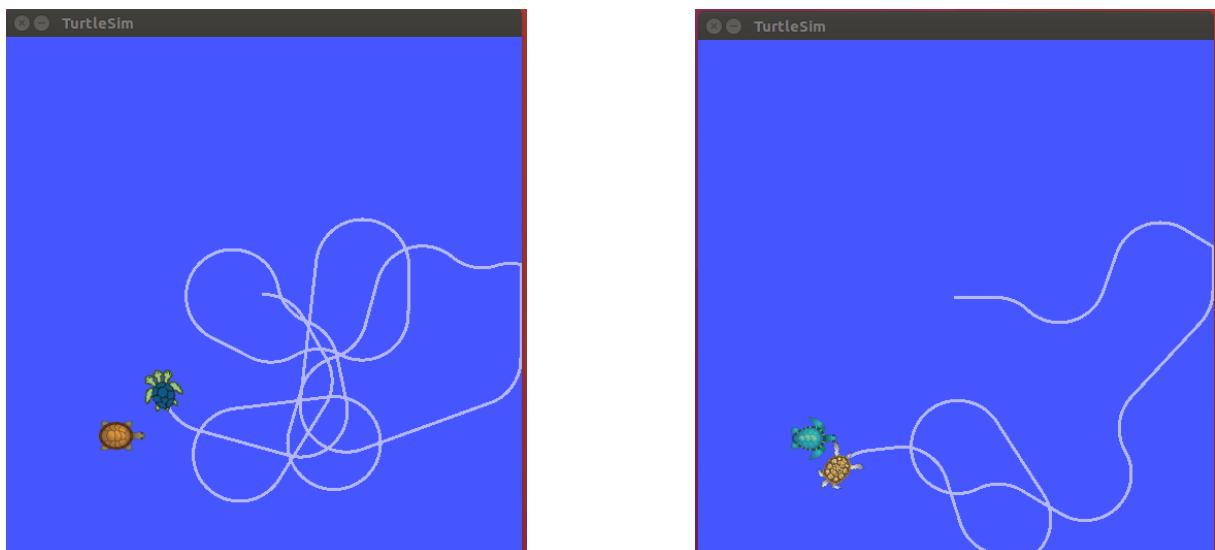
Figura 6.4 – Percurso percorrido utilizando o controle independente na primeira rodada



Fonte: Elaborado pelo autor.

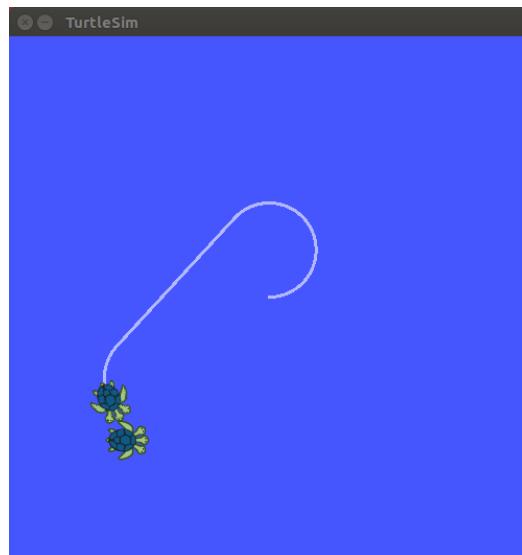
É possível notar que os participantes conseguiram atingir o objetivo utilizando uma rota simples e direta, próxima da trajetória percorrida utilizando o teclado, entretanto conforme a trajetória percorrida pela tartaruga para atingir o objetivo aumentava, os participantes apresentavam dificuldades para concluir a simulação com sucesso, a Figura 6.5 mostra a trajetória percorrida pelos mesmos participantes utilizando os algoritmos citados anteriormente, enquanto a 6.6 mostra a trajetória percorrida na terceira rodada da segunda bateria de simulações utilizando o sistema de controle independente.

Figura 6.5 – Percurso percorrido pelos participantes na primeira terceira.



Fonte: Elaborado pelo autor.

Figura 6.6 – Percurso percorrido utilizando o controle independente na terceira rodada



Fonte: Elaborado pelo autor.

Ao perguntar para os participantes como foi conduzir a tartaruga nessa rodada, eles disseram que a mesma estava seguindo os comandos enviados, entretanto em alguns momentos, ela não respondia corretamente, tornando um pouco difícil atingir o objetivo, já era esperado que os algoritmos errassem as previsões às vezes, isso devido ao fator da acurácia de seus modelos serem baixas, porém o fato desses erros ocorrerem, fazem com que os usuários fiquem frustrados, tornando difícil a recuperação do controle, os voluntários disseram que precisaram fechar os olhos e relaxarem para conseguirem recuperar o controle da tartaruga, aumentando consideravelmente o tempo para atingir o objetivo.

Todos os voluntários sentiram que o agente virtual estava realizando os comandos enviados corretamente em pelo menos um dos algoritmos testados, entretanto com cada participante houve um momento em que teve dificuldades para controlar a tartaruga.

7 CONCLUSÃO

Este trabalho teve como objetivo contribuir com as pesquisas na área de BCI, que vem crescendo muito atualmente, uma boa parte destas pesquisas visa contribuir com a qualidade de vida das pessoas que possuem alguma complicaçāo motora.

Desta forma, neste trabalho foi conduzida a análise do desempenho de três algoritmos de aprendizagem de máquina, sendo eles: MLP, CNN e RNN no controle de um objeto virtual, de modo a demonstrar o princípio de um sistema de controle baseado em informações extraídas de um EEG.

Ao analisar os resultados obtidos pelos voluntários neste projeto, notou-se que o estado emocional dos mesmos interfere diretamente nos resultados dos algoritmos CNN e RNN, sendo mais prejudicial no RNN, pois o mesmo possui erro propagado. Ao coletar os *feedbacks* todos os participantes sentiram que o algoritmo MLP estava completamente aleatório, entretanto como o primeiro contato com a simulação foi no MLP é preciso executar a simulação com mais usuários trocando a ordem dos algoritmos de aprendizagem de máquina para confirmar se esse comportamento se manterá.

Como trabalhos futuros, cogita-se conduzir pesquisas na área de neurociência para entender melhor o *dataset* coletado e encontrar uma maneira de amenizar o impacto do estado emocional do usuário e estudar novas técnicas dos algoritmos RNN e CNN, visando abordar o melhor dos dois, com isso serão testados novamente os algoritmos, entretanto no controle de um objeto real.

8 REFERÊNCIAS

- ADUR, R. et al. Sistema de processamento de sinais biomédicos: Módulo didático de eletroencefalograma. Florianópolis, SC, 2008.
- AKAY, M. **Detection and estimation methods for biomedical signals**. [S.I.]: Academic Press, Inc., 1996.
- ALHAGRY, S.; FAHMY, A. A.; EL-KHORIBI, R. A. Emotion recognition based on eeg using lstm recurrent neural network. **Emotion**, v. 8, n. 10, p. 355–358, 2017.
- ALVES, M. F. Da econometria ao aprendizado de máquina. 2018.
- ARGOUD, F. I. M. et al. Contribuição à automatização da detecção e análise de eventos epileptiformes em eletroencefalograma. Florianópolis, SC, 2001.
- ARKELL, E. I. **How your brain sees your body: Meet the cortical homunculus**. 2019. <https://io9.gizmodo.com/how-your-brain-sees-your-body-meet-the-cortical-homunc-5670064>. (Accessed on 05/25/2019).
- BALAKRISHNAN, D.; PUTHUSERYYPADY, S. Multilayer perceptrons for the classification of brain computer interface data. In: IEEE. **Proceedings of the IEEE 31st Annual Northeast Bioengineering Conference, 2005**. [S.I.], 2005. p. 118–119.
- BATISTA, G. E. d. A. P. et al. **Pré-processamento de dados em aprendizado de máquina supervisionado**. Tese (Doutorado) — Universidade de São Paulo, 2003.
- BERGER, H. Über das elektrenkephalogramm des menschen. **European archives of psychiatry and clinical neuroscience**, Springer, v. 87, n. 1, p. 527–570, 1929.
- BERSCH, R. Introdução à tecnologia assistiva. **Porto Alegre: CEDI**, p. 21, 2008.
- BERSCH, R.; TONOLLI, J. Introdução ao conceito de tecnologia assistiva e modelos de abordagem da deficiência. **Bengala Legal**, 2006.
- BEZERRA, E. Introdução à aprendizagem profunda. **Artigo—31º Simpósio Brasileiro de Banco de Dados—SBBD2016—Salvador**, 2016.
- BINOTI, D. H. B.; BINOTI, M. L. M. da S.; LEITE, H. G. Configuração de redes neurais artificiais para estimação do volume de árvores. **Revista Ciência da Madeira (Brazilian Journal of Wood Science)**, v. 5, n. 1, p. 10–12953, 2014.
- BISHOP, C. M. **Pattern recognition and machine learning**. [S.I.]: springer, 2006.
- BISHOP, C. M. et al. **Neural networks for pattern recognition**. [S.I.]: Oxford university press, 1995.
- BISPO, T. D. Arquitetura lstm para classificação de discursos de ódio cross-lingual inglês-ptbr. Pós-Graduação em Ciência da Computação, 2018.

BONINI-ROCHA, A. C. et al. Metodologia para observação e quantificação de sinais de eeg relativos a evidências cognitivas de aprendizagem motora. **Ciências & Cognição**, v. 13, n. 2, p. 27–50, 2008.

BRAGA, A. d. P. De p.; carvalho, ap de lf; ludemir, tb redes neurais artificiais-teoria e aplicações. **Rio de Janeiro: Livros Técnicos e Científicos**, 2000.

BRAGA, A. d. P. **Redes neurais artificiais: teoria e aplicações**. [S.I.]: Livros Técnicos e Científicos, 2000.

BROCIOUS, C.; MACHULIS, K. Emokit. **OpenYou,[Online]. Available: <http://www.openyou.org>**, 2019.

BRODU, N.; LOTTE, F.; LÉCUYER, A. Comparative study of band-power extraction techniques for motor imagery classification. In: **IEEE. 2011 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)**. [S.I.], 2011. p. 1–6.

CANTARELLI, T. L.; JÚNIOR, J. J. A. M.; JR, S. L. S. Fundamentos da medição do eeg: Uma introdução. **SEMINÁRIO DE ELETRONICA E AUTOMAÇÃO. Ponta Grossa**, 2016.

CAPARELLI, T. B. et al. Projeto e desenvolvimento de um sistema multicanal de biotelemetria para detecção de sinais ecg, eeg e emg. Universidade Federal de Uberlândia, 2007.

CARVALHO, L. C. **Instrumentação médico-hospitalar**. [S.I.]: Manole, 2008.

CASTRO, F. D.; CASTRO, M. D. Redes neurais artificiais. **DCA/FEEC/Unicamp**, 2001.

CERVERA, M. A. et al. Brain-computer interfaces for post-stroke motor rehabilitation: a meta-analysis. In: **Annals of clinical and translational neurology**. [S.I.: s.n.], 2018.

CHEN, K.-M. et al. Microwave life-detection systems for searching human subjects under earthquake rubble or behind barrier. **IEEE transactions on biomedical engineering**, IEEE, v. 47, n. 1, p. 105–114, 2000.

CHEN, Z.; BROWN, E. N.; BARBIERI, R. Characterizing nonlinear heartbeat dynamics within a point process framework. **IEEE Transactions on Biomedical Engineering**, IEEE, v. 57, n. 6, p. 1335–1347, 2010.

COHEN, A. **Biomedical Signal Processing: Volume 2: Compression and Automatic Recognition**. [S.I.]: CRC Press, 2019.

COOLEY, J.; TUKEY, J. An algorithm for machine calculation of complex fourier series. mathematics of computing, reprinted 1972. **Digital signal processing**. IEEE Press, New York, NY, p. 223–227, 1965.

DAHLHAUSEN, B.; RADABAUGH, S. Update on psittacine beak and feather disease and avian polyomavirus testing. In: **Proc Annu Conf Assoc Avian Vet**. [S.I.: s.n.], 1993. p. 5–7.

- DECETY, J. et al. The cerebellum participates in mental activity: tomographic measurements of regional cerebral blood flow. **Brain research**, Elsevier, v. 535, n. 2, p. 313–317, 1990.
- EMOTIV. **EPOC+ 14 Channel - Emotiv**. 2019. <https://www.emotiv.com/epoc-14-channel-eeg-2/>. (Accessed on 05/12/2019).
- FERREIRA, A. Uma proposta de interface cérebro-computador para comando de cadeiras de rodas. Universidade Federal do Espírito Santo, 2008.
- FERREIRA, A. d. S. **Redes neurais convolucionais profundas na detecção de plantas daninhas em lavoura de soja**. Dissertação (Mestrado) — UFMS - Campo Grande, 2017.
- FILHO, S. A. dos S. et al. Magnitude quadrática da coerência na detecção da imaginação do movimento para aplicação em interface cérebro-máquina. UFMG, 2010.
- FRANÇA, C. R.; BORGES, J. A.; SAMPAIO, F. F. Tupi—recursos de acessibilidade para educação especial e inclusiva dos deficientes motores. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.I.: s.n.], 2005. v. 1, n. 1, p. 591–600.
- GOMES, M. L. L. Aula 5. 2003.
- GONÇALVES, W. d. O. et al. Utilização dos sinais de eletroencefalograma e eletrodermal no aprendizado por reforço de uma interface cérebro-máquina. Universidade Federal Rural do Rio de Janeiro, 2017.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning <http://www.deeplearningbook.org>. **MIT Press, Cambridge, MA**, 2016.
- GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. In: **IEEE. 2013 IEEE international conference on acoustics, speech and signal processing**. [S.I.], 2013. p. 6645–6649.
- HALL, J. E. **Guyton and Hall Textbook of Medical Physiology E-Book: with STUDENT CONSULT Online Access**. [S.I.]: Elsevier Health Sciences, 2010.
- HAYKIN, S. **Redes neurais princípios prática. Vol. 2**. [S.I.]: Editora Bookman, Porto Alegre, 2001.
- HAYKIN, S. **Redes neurais: princípios e prática**. [S.I.]: Bookman Editora, 2007.
- HAYKIN, S.; SINAIS, V. **Sistemas**. [S.I.]: Porto Alegre: Bookman, 2001.
- HE, W. et al. Simultaneous human health monitoring and time-frequency sparse representation using eeg and ecg signals. **IEEE Access**, IEEE, v. 7, p. 85985–85994, 2019.
- HEIDRICH, R. d. O. et al. Jogos digitais para interação com brain computer interface para auxiliar no processo de inclusão escolar de pessoas com paralisia cerebral. **Blucher Design Proceedings**, v. 2, n. 9, p. 3397–3407, 2016.

- HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. **Neural computation**, MIT Press, v. 18, n. 7, p. 1527–1554, 2006.
- HOCHREITER, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, World Scientific, v. 6, n. 02, p. 107–116, 1998.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.
- HU, G. **Introduction to digital signal processing**. [S.I.]: Tsinghua University Press, Beijing, 2005.
- IBGE, I. C. **Instituto Brasileiro de Geografia e Estatística - 2013**. [S.I.: s.n.], 2013.
- IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015.
- JEANNEROD, M.; DECETY, J. Mental motor imagery: a window into the representational stages of action. **Current opinion in neurobiology**, Elsevier, v. 5, n. 6, p. 727–732, 1995.
- JOSEPH, L.; JOSEPH, L.; PAO. **Robot Operating System for Absolute Beginners**. [S.I.]: Springer, 2018.
- KALCHBRENNER, N.; GREFENSTETTE, E.; BLUNSOM, P. A convolutional neural network for modelling sentences. **arXiv preprint arXiv:1404.2188**, 2014.
- KANDEL, E. R. et al. **Principles of neural science**. [S.I.]: McGraw-hill New York, 2000.
- KAWAKAMI, K. Supervised sequence labelling with recurrent neural networks. **Ph. D. thesis**, Technical University of Munich, 2008.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.
- KLEM, G. H. et al. The ten-twenty electrode system of the international federation. **Electroencephalogr Clin Neurophysiol**, v. 52, n. 3, p. 3–6, 1999.
- KOSKO, B. **Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence**. [S.I.: s.n.], 1992.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.I.: s.n.], 2012. p. 1097–1105.
- KRUK, K. A. et al. Comparison of brain activity during drawing and clay sculpting: A preliminary qeeg study. **Art Therapy**, Taylor & Francis, v. 31, n. 2, p. 52–60, 2014.
- LEBEDEV, M. A.; NICOLELIS, M. A. Brain-machine interfaces: past, present and future. **TRENDS in Neurosciences**, Elsevier, v. 29, n. 9, p. 536–546, 2006.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

- LEITE, F. T. et al. Evaluation of recurrent neural network architectures to help motor disabled people through brain computer interface. In: **ENIAC 2019** (). [s.n.], 2019. Disponível em: <<http://XXXXXX/198987.pdf>>.
- LEMONS. Metodologia knn-fuzzy: Uma abordagem da classificação de dados por similaridade. 2009.
- LUCK, S. J. **An introduction to the event-related potential technique**. [S.I.]: MIT press, 2014.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.
- MICHIELLI, N.; ACHARYA, U. R.; MOLINARI, F. Cascaded lstm recurrent neural network for automated sleep stage classification using single-channel eeg signals. **Computers in biology and medicine**, Elsevier, v. 106, p. 71–81, 2019.
- MINAMI, K.-i.; NAKAJIMA, H.; TOYOSHIMA, T. Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network. **IEEE transactions on Biomedical Engineering**, IEEE, v. 46, n. 2, p. 179–185, 1999.
- MINSKY, M.; PAPERT, S. **Perceptrons: An Introduction to Computational Geometry**. Cambridge, MA, USA: MIT Press, 1969.
- MITCHELL, T. M. et al. Machine learning. 1997. **Burr Ridge, IL: McGraw Hill**, v. 45, n. 37, p. 870–877, 1997.
- MUTHUKUMARASWAMY, S. D.; JOHNSON, B. W. Primary motor cortex activation during action observation revealed by wavelet analysis of the eeg. **Clinical Neurophysiology**, Elsevier, v. 115, n. 8, p. 1760–1766, 2004.
- OH, K.-S.; JUNG, K. Gpu implementation of neural networks. **Pattern Recognition**, Elsevier, v. 37, n. 6, p. 1311–1314, 2004.
- OH, S. L. et al. A deep learning approach for parkinson's disease diagnosis from eeg signals. **Neural Computing and Applications**, Springer, p. 1–7, 2018.
- OLIVEIRA, M. A.; FAVERO, L. P. L. Uma breve descrição de algumas técnicas para análise de séries temporais: Séries de fourier, wavelets, arima, modelos estruturais para séries de tempos e redes neurais. **VI SEMEAD Ensaio mqi. USP. Anais, São Paulo**, 2002.
- PALANIAPPAN, R. Brain computer interface design using band powers extracted during mental tasks. In: **IEEE. Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005**. [S.I.], 2005. p. 321–324.
- PARKER, J. R. **Algorithms for image processing and computer vision**. [S.I.]: John Wiley & Sons, 2010.
- PASZKE, A. et al. Pytorch. **Computer software. Vers. 0.3**, v. 1, 2017.
- PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

- PEREIRA, M. d. C. V. et al. Avaliação de técnicas de pré-processamento de sinais do eeg para detecção de eventos epileptogênicos utilizando redes neurais artificiais. Florianópolis, SC, 2003.
- PINNA, G. et al. The accuracy of power-spectrum analysis of heart-rate variability from annotated rr lists generated by holter systems. **Physiological measurement**, IOP Publishing, v. 15, n. 2, p. 163, 1994.
- QUIGLEY, M.; GERKEY, B.; SMART, W. D. **Programming Robots with ROS: a practical introduction to the Robot Operating System**. [S.I.]: "O'Reilly Media, Inc.", 2015.
- RAMADAN, R. A.; VASILAKOS, A. V. Brain computer interface: control signals review. **Neurocomputing**, Elsevier, v. 223, p. 26–44, 2017.
- RIBEIRO, C. **O Potencial Pré-Motor Cortical Durante Tarefa Voluntária Bilateral**. Tese (Doutorado) — Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2014.
- ROQUE, K.; CEZAR, P. et al. Controle de um protótipo de cadeira de rodas através de uma interface cérebro-computador. [sn], 2017.
- ROS, R. O. S. **ROS Robot Operating System**. 2019. <https://www.ros.org>. Accessed: 2019-09-30.
- ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. [S.I.]: Cornell Aeronautical Laboratory, 1957.
- SANTOS, C. D.; GATTI, M. Deep convolutional neural networks for sentiment analysis of short texts. In: **Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers**. [S.I.]: s.n., 2014. p. 69–78.
- SCHALK, G. et al. Bci2000: a general-purpose brain-computer interface (bci) system. **IEEE Transactions on biomedical engineering**, IEEE, v. 51, n. 6, p. 1034–1043, 2004.
- SCHUH, Â. R. et al. Desenvolvimento de um simulador controlado por interface cérebro-computador não invasiva para treinamento na utilização de cadeira de rodas. **RENOTE**, v. 11, n. 3, 2013.
- SEMENIUTA, S.; SEVERYN, A.; BARTH, E. Recurrent dropout without memory loss. **arXiv preprint arXiv:1603.05118**, 2016.
- SILVA, L. N. d. C. et al. Análise e síntese de estratégias de aprendizado para redes neurais artificiais. [sn], 1998.
- SOARES, A. S. et al. Reconhecimento de padrões biomédicos utilizando máquinas de aprendizado profundo. **Revista de Sistemas e Computação-RSC**, v. 8, n. 2, 2018.
- SOUTO, M. D. et al. Técnicas de aprendizado de máquina para problemas de biologia molecular. **Sociedade Brasileira de Computação**, 2003.
- SOUZA, B. de et al. Classificação de faltas via redes neurais artificiais. **V SBQEE, Aracaju, Brasil**, p. 163–8, 2003.

SOUZA, J. P. G. de et al. Interface cérebro–máquina (icm): Da transdução do estímulo externo em impulso nervoso a tradução em comandos digitais. **Caderno de Graduação-Ciências Biológicas e da Saúde-UNIT-ALAGOAS**, v. 3, n. 1, p. 139–152, 2015.

STEWART, J. **CÁLCULO VOLUME 1-TRADUÇÃO DA 6® EDIÇÃO NORTE-AMERICANA**. [S.I.]: Cengage Learning Edições Ltda., 2010.

TOUVRON, H. et al. Fixing the train-test resolution discrepancy. **arXiv preprint arXiv:1906.06423**, 2019.

WAIBEL, A. Prosodic knowledge sources for word hypothesization in a continuous speech recognition system. In: IEEE. **ICASSP'87. IEEE International Conference on Acoustics, Speech, and Signal Processing**. [S.I.], 1987. v. 12, p. 856–859.

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