UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE MATEMÁTICA

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Rennan de Lucena Gaio

STILL: Sistema Tradutor Inteligente de LIBRAS com Luva

RIO DE JANEIRO

2020

Rennan de Lucena Gaio

STILL: Sistema Tradutor Inteligente de LIBRAS com Luva

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Valéria Bastos

Co-orientadora: Profa. Vanessa Quadros

RIO DE JANEIRO

2020

CIP - Catalogação na Publicação

GGl43s

Gaio, Rennan
STILl: Sistema Tradutoi Inteligente de IIBRAS com [uva / Rennan Gaio. -- Rio de Janezzo. 2020.

Orientadora: Valeiia Bastos.
toozientadora: Vanessa Quadzos.
7rabalho de concluséo de curso (graduaqâo) - Univezsidade Federal do Rio de Janeiro, Instituto de Matemâtica, Bachaiel em "iéncia dd Computagâo, 2020.

1. Arquitetuza de Sistemas. 2. Inteligéncia Artificial. 3. LIBRAS. 4. lingua Poztuguesa. 3. TraduTéo. I. Bastos, Valeiia, orient. II. Quadros, Vanessa. coorlent. Hi. litulo.

Elaborado pelo Sisiema de Geração Automátlca da UFRJ com os dados fomecldos pelo{a) autor{a), sob a responsablIidade de Miguel Romeu Amorim Neto - CRB•7/8283.

Rennan de Lucena Gaio

STILL: Sistema Tradutor Inteligente de LIBRAS com Luva

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 07 de agosto de 2020.

BANCA EXAMINADORA:

Valéria Menezes Bastos
Valéria Menezes Bastos, D. Sc. (UFRJ)

Participação por vídeo-conferência
Vanessa Quadros Gondim Leite, M. Sc. (IME)

Participação por vídeo-conferência
Jorge Bidarra, D. Sc. (Unioeste/PR)

Participação por vídeo-conferência
João Antônio Recio da Paixão, D. Sc. (PUC-RJ)

Dedicatória: Eu dedico este trabalho a todos que me apoiaram, e que mesmo em momentos difíceis não desistiram da ideia.

AGRADECIMENTOS

Inicialmente, gostaria de agradecer a todos que me apoiaram diretamente ou indiretamente na realização do projeto. Um agradecimento especial às minhas ótimas orientadoras Valéria Bastos e Vanessa Quadros, por terem me trazido este grande desafio e me ajudado com todas as questões e problemas que eu não conseguiria concluir sozinho. Gostaria de agradecer a todos os participantes do projeto e do LABIS (UFRJ) por sempre estarem contribuindo com grandes ideias. Um agradecimento à Bianca Gaio, Anselmo Gaio, Raquel Gaio e Felipe Morandi por terem se oferecido a ajudar na fabricação dos dados para a utilização do modelo da luva, assim como a professora Valéria já mencionada anteriormente. E por último, um agradecimento a todos meus amigos e familiares que por mais que tenham me chamado de louco por ter aceitado esse desafio, sempre me apoiaram e me incentivaram a continuar com o projeto para ele se tornar o melhor e mais legal possível.

Epígrafe: O único dia fácil foi ontem.

SEALS

RESUMO

Vivemos atualmente em um mundo de inclusão, em que apesar de todas as dificuldades, a população vem se esforçando para criar um ambiente mais propício para a interação de todos. Com a evolução das ferramentas eletrônicas e dos estudos de técnicas de aprendizado de máquina, os profissionais da área de computação estão sendo capacitados cada vez mais a criar ferramentas que podem auxiliar na busca por uma interação mais igualitária entre todos os cidadãos, independentemente das suas dificuldades ou deficiências.

O foco deste projeto é desenvolver uma pesquisa atrelada a um produto capaz de facilitar a comunicação entre um usuário de LIBRAS e indivíduos que não possuem fluência alguma nesta língua. Para criar um sistema que abrangesse o máximo possível das nuances da língua de sinais, foi projetado um protótipo que leva em consideração tanto as expressões faciais do usuário quanto as movimentações e composições das mãos desse usuário.

Para tal, foi desenvolvida e construída uma luva com diversos sensores que, em conjunto com uma câmera integrada do notebook, coletam os dados para o sistema. Com o auxílio de algoritmos de inteligência artificial foi feita a tradução, em tempo real, de um conjunto limitado de sinais em LIBRAS para palavras do português, que leva em consideração todos os dados capturados.

Em relação às contribuições para a área de atuação desta pesquisa, existe o desenvolvimento de uma arquitetura de sistemas com a luva sensorial e câmera a fim de realizar a tradução de LIBRAS para o português. Além disto, a construção de um conjunto de dados ("*dataset"*) específico de LIBRAS para mapear sinais e movimentação de mão com a luva sensorial desenvolvida. Este conjunto de dados poderá ser utilizado para a continuação e criação de novas iniciativas, anteriormente impossibilitadas pelo fato de não existir um conjunto de dados rotulados sobre tal problema.

Palavras-chave: Arquitetura de sistemas. Inteligência artificial. LIBRAS. Língua Portuguesa. Tradução.

ABSTRACT

We currently live in a world of inclusion, in which despite all the difficulties, the population has been striving to create a more conducive environment for everyone to interact. With the evolution of electronic tools and the study of machine learning techniques, computer professionals are being enabled to create tools that can help in the search for a more equal interaction between different people, regardless of their difficulties or deficiencies.

The focus of this project is to develop research linked to the development of a product capable of facilitating communication between LIBRAS users and a person who does not have any fluency in this language. To create a system that covers as much as possible the nuances of the sign language, a prototype was created that takes into account both the user's facial expressions, as well as the movements and compositions of the user's hand.

To this end, a glove was developed with several sensors build by the author, which together with an integrated notebook câmera collect data for the system. With the aid of artificial intelligence algorithms we were able to translate a limited amount of LIBRAS signals into Portuguese in real time, taking into account all captured data.

Regarding the contributions to the area of operation of this research, there is the development of a system architecture with a sensory glove and camera to carry out the translation of LIBRAS into Portuguese. In addition, the creation of a LIBRAS-specific dataset to map signals and movement of the user’s hand using the developed sensory glove. This data set can be used for the continuation and creation of new initiatives, which were previously impossible due to the fact that there wasn’t labeled datasets on this problem.

Keywords: Systems Architecture. Artificial Inteligence. LIBRAS. Portuguese Language. Translation.

LISTA DE ILUSTRAÇÕES

Figura 1: Exemplo de uma distribuição de pontos rotuladas linearmente separáveis 25
Figura 2: Distribuição de pontos rotuladas linearmente separáveis com g(x) 26
Figura 3: Fluxograma de criação de uma função objetivo com base nos dados 26
Figura 4: Exemplo de *overfitting* de dados 27
Figura 5: Relação entre o aumento da complexidade do modelo e o erro esperado 28
Figura 6: Separação de dados utilizando KNN 30
Figura 7: Exemplo de criação de um modelo utilizando aregressão logística 31
Figura 8: Árvore de decisão para a classificação de pacientes 32
Figura 9: Divisão geométrica do espaço com as árvores de decisão 32
Figura 10: Exemplo da predição das k diferentes árvoresno *random forest* 34
Figura 11: Arquitetura de uma rede *LeNet* 35
Figura 12: Exemplo de formação de *haar features* 37
Figura 13: Conta rápida para imagem integral 37
Figura 14: Exemplo de seleçãode atributos 38
Figura 15: Processo de classificação de sub-janelas do *haar cascade* 39
Figura 16: Fluxograma completo do funcionamento do *haar cascade* 39
Figura 17: Exemplo do funcionamento de separação marginal das SVMs 41
Figura 18: Placa do funduino UNO 42
Figura 19: Sensor flexível 43
Figura 20: Acelerômetro 43
Figura 21: Fluxograma do sistema deidentificação facial 47
Figura 22: Funcionamento da segmentação da face utilizando o *haar cascade* 48
Figura 23: Exemplo de imagens presentes no conjunto de dados FER 50
Figura 24: Fluxo do processo de classificação de expressões faciais 51
Figura 25: Resultado do programa executável de classificação de expressões faciais 52
Figura 26: Arquitetura dos sensores da luva 54
Figura 27: Visão frontal do protótipo final de luva sensorial 55
Figura 28: Visão lateral do protótipo final de luva sensorial 55
Figura 29: Representação da palavra “marido” em LIBRAS 56
Figura 30: Fluxograma de seleção do modelo classificador 59

Figura 31: Fotos classificadas pelo algoritmo de reconhecimento de expressões 62

LISTA DE TABELAS

Tabela 1: Modelos utilizados em cada sistema 45
Tabela 2: Tabela de atributos de luva sensorial 57
Tabela 3: Tabela de resultados do modelo de reconhecimento facial 61
Tabela 4: Tabela de resultados do modelo da luva com ruído 63
Tabela 5: Tabela de resultados do modelo da luva sem ruído 64

LISTA DE SIGLAS

LIBRAS – Língua Brasileira de Sinais

SVM – *Support Vectors Machine* (máquina de vetores de suporte)

RF – *Random Forest*

ASL – *American Sign Languange*

LSF – *Langue des Signes Française*

DGS – *Deutsche Gebärdensprache*

MIT - Instituto de Tecnologia de Massachusetts

HMM – *Hidden Markov Model*

CNN – *convolutional neural networks*

KNN – *K nearest neighbor*

IA – Inteligência artificial

STILL – Sistema Tradutor Inteligente de LIBRAS com Luva

UFRJ – Universidade Federal do Rio de Janeiro

USB – U*niversal Serial Bus*

LISTA DE SÍMBOLOS

$\sum_{i}^{n}$ – Somatório de i até n

$\vec{x}$ – Vetor de atributos

y – Classe de uma amostra

X – Matriz de atributos

$\vec{y}$ – Vetor de casses

ε – Erro

$\mathbb{R}$ – Conjunto

Ω – Unidade de medida Ohms

SUMÁRIO

1 INTRODUÇÃO 17
2 TRABALHOS RELACIONADOS 20
3 CONCEITOS BÁSICOS 23
3.1 NOÇÕES GERAIS 24
3.1.1 Ferramentas de implementação 24
3.1.2 Notações e introdução ao aprendizado de máquina 24
3.2 EXPRESSÃO FACIAL 28
3.2.1 KNN 29
3.2.2 Regressão logística 30
3.2.3 Árvores de decisão 31
3.2.4 XGBoost 33
3.2.5 Random forest 33
3.2.6 Redes neurais convolucionais 34
3.2.7 Haar Cascade 36
3.3 LUVA SENSORIAL 39
3.3.1 SVM 40
3.3.2 Arduino 41
3.3.3 Sensor Flexível 42
3.3.4 Acelerômetro 43
4 IMPLEMENTAÇÃO 44
4.1 IDENTIFICAÇÃO DE REQUISITOS DA APLICAÇÃO 44
4.2 ESCOPO DO PROJETO 46
4.3 RECONHECIMENTO FACIAL EM IMAGEM 47
4.4 RECONHECIMENTO DE EXPRESSÃO FACIAL 49
4.4.1 Conjunto de dados 49
4.4.2 Treinamento do modelo de aprendizado de máquina 50
4.4.3 Identificação das expressões faciais 52
4.5 RECONHECIMENTO DE SINAIS E MOVIMENTAÇÃO DA MÃO 53
4.5.1 Arquitetura da luva sensorial 53
4.5.2 Conjunto de dados 55
4.5.3 Treinamento do modelo de aprendizado de máquina 59

4.5.4 Identificação dos sinais gestuais 60
5 RESULTADOS 61
LISTA DE SÍMBOLOS
5.1 RECONHECIMENTO DE EXPRESSÕES FACIAIS 61
5.2 RECONHECIMENTO DE SINAIS DA LUVA 62
5.3 COMPOSIÇÃO DOS PROCEDIMENTOS 65
6 CONCLUSÃO E PROJETOS FUTUROS 66
REFERÊNCIAS 69

1 INTRODUÇÃO

O advento dos computadores com o auxílio da inteligência artificial trouxe a automatização para ajudar nos desafios que antes eram realizados manualmente por uma ou mais pessoas. Essa realidade traz também impactos diretos à sociedade, principalmente aos cidadãos portadores de algum tipo de deficiência. Isso porque os projetos de acessibilidade são relevantes e essenciais para a maior inclusão e facilitação do convívio diário e profissional. Apesar dos esforços, ainda existem muitos problemas que necessitam de tempo e investimento, pular etapas para uma solução rápida e ineficiente pode causar uma falsa impressão de inclusão.

De acordo com o art. 27º da Lei nº 13.146 – Lei Brasileira de Inclusão, de 6 de julhode 2015 [10]: "a educação é um direito da pessoa com deficiência" e "o sistema educacional deve ser inclusivo em todos os níveis". Entretanto, as universidades públicas brasileiras não estão preparadas para absorver esses estudantes em seus ambientes físicos, a comunicação com esses alunos não é efetiva, os materiais e a didática dos professores também não estão adaptados para esta demanda. Ou seja, a acessibilidade requer planejamento, já que cada deficiência (visual, física, cognitiva, auditiva, etc.) exige diferentes ações de acessibilidade.

A fim de criar mais um canal de comunicação entre surdos que se comunicam em LIBRAS (Língua Brasileira de Sinais) [45] com o restante da sociedade, é necessário o entendimento da língua, inclusive de seus requisitos e possibilidades. A língua de sinais é dividida em cinco principais partes: expressão facial e corporal, configuração da mão, pontos de articulação, orientação da mão e movimentação da mão.

Isso significa que quanto mais atributos supracitados são levados em consideração, melhor será uma representação computacional. Neste estudo foi abordado, principalmente, o tratamento das expressões faciais em conjunto com a configuração, movimentação e orientação da mão, através da comparação de diferentes técnicas para a identificação das melhores alternativas.

A despeito de muitos assim entenderem, não há uma língua de sinais universal e cada país pode ter uma ou mais língua de sinais utilizada pelos cidadãos (ASL - *American Sign Language*, LSF - *Langue des Signes Française*, DGS - *Deutsche Gebärdensprache*) além de existirem variações regionais. Neste estudo, foi utilizada LIBRAS com sua versão característica do Estado do Rio de Janeiro, validando os sinais identificados com o auxílio de deficientes auditivos.

No censo de 2010 do IBGE foi constatado que 5,1% da população brasileira possui algum tipo de deficiência auditiva, e que cerca de 2,2 milhões de pessoas (1,1% dessa mesma

população) possuem uma deficiência auditiva grave [23]. Tal realidade ficou muito evidente na comunidade científica com maior inclusão destas pessoas à sociedade acadêmica em virtude da Lei Brasileira de Inclusão [10]. Pois anteriormente, essas eram tratadas separadamente, como um caso isolado.

Em função dessa dificuldade de acessibilidade entre alunos com deficiência auditiva, foi estudada uma maneira de tentar amenizar esse déficit de comunicação entre alunos e professores, através da criação de um tradutor. Este sistema foi elaborado e produzido pelo autor com finalidade de mitigar alguns desses problemas de comunicação, inicialmente em um escopo reduzido. Futuramente, planeja-se a expansão deste projeto para toda a comunidade de forma ampla, para utilização no cotidiano.

Para fazer a identificação e classificação da face do usuário, foi utilizada a câmera do computador que, em conjunto com um algoritmo de reconhecimento, descreve as possíveis configurações de rosto do usuário. Para realizar a extração da face diante de toda a imagem capturada, foi aplicado o algoritmo de classificação *HaarCascade*, também conhecido como Viola-Jones, por ser muito renomado e utilizado nesta área [35][47]. A partir disto, efetuou-se um comparativo entre os algoritmos de aprendizado de máquina testados para a classificação de expressão faciais e foi indicado qual obteve a melhor taxa de acerto.

Em relação aos movimentos e configurações da mão, se propôs a criação de uma luva que utiliza um microcontrolador Funduino Uno e alguns sensores para a captação de dados, como posição espacial e dobradura dos dedos. Com este dispositivo se tornou viável captar os dados para obter sinais em LIBRAS pelo sistema e, com ajuda de algoritmos de aprendizado, realizou-se a classificação de algumas configurações de mão utilizadas para compor as palavras na língua portuguesa. Assim como no caso das expressões faciais, foram testados e comparados vários algoritmos e posteriormente escolhido o que obteve maior desempenho para fazer parte do produto.

O objetivo final do projeto foi criar um estudo de caso composto por captura de imagem em conjunto com a luva sensorial. O protótipo foi desenvolvido o mais eficiente possível perante as limitações (explicitadas ao longo do texto) e de maneira versátil para a pessoa que está utilizando. Em virtude disso, foi produzida uma luva em conjunto com dados que contemplam 8 diferentes configurações de mão. Realizou-se uma análise, com diferentes algoritmos de classificação para buscar o modelo que se adaptasse melhor ao problema apresentado. Finalmente, fomentou-se uma discussão sobre a viabilidade desta abordagem perante ao problema com base nos resultados.

Como consequência, foi desenvolvido um *dataset* específico da LIBRAS, que contém dados recebidos dos sensores da luva e podem ser utilizados também para outras pesquisas. Com os devidos ajustes a nível de design de produto e aumento de classes (vocabulário identificado pelo modelo) do conjunto de dados, o sistema poderá ser implementado e utilizado pela sociedade. Isso resulta em um grande impacto na solução dos problemas de comunicação entre deficientes auditivos e ouvintes que hoje existem na UFRJ.

Resumidamente, os capítulos seguintes são descritos a seguir.

O **capítulo 2** apresenta-se trabalhos relacionados à área de tradução de sinais e expressões faciais que foram utilizados como base de conhecimento.

No **capítulo 3**, realiza-se uma breve introdução de cada um dos algoritmos utilizados. Além disso, se introduz a notação utilizada no documento.

No **capítulo 4** descreve-se como foi a implementação do trabalho, as dificuldades enfrentadas e alternativas escolhidas. Além disto, apresenta-se como foram realizados os experimentos propostos, seus objetivos e mais detalhes sobre cada um dos desafios encontrados.

Posteriormente, no **capítulo 5**, são apresentados os resultados obtidos referentes aos experimentos propostos. Sendo realizada uma discussão sobre os resultados encontrados, analisando-os e comentando seus pontos positivos e negativos.

O **capítulo 6** contém a conclusão do trabalho e os projetos futuros. Propondo, ainda, uma breve discussão sobre os resultados encontrados e as alternativas válidas para a continuação do projeto.

2 TRABALHOS RELACIONADOS

Para a elaboração do projeto, foram avaliados diversos trabalhos [27] [46] [30] [26] [2] [9] [36] como inspiração e referência às opções adotadas. Esses são todos referentes à ASL ("American Signal Language"), a língua de sinais americana, que possui desafios parecidos com os presentes enfrentados na tradução de LIBRAS. Esses foram escolhidos por serem mais abundantes na literatura e por serem mais explorados. Além disto, existem alguns estudos nacionais que também buscam alternativas ao problema [42] [44].

Primeiramente, analisou-se o trabalho de Thomas Pryor e Navid Azodi [27] nomeado de "*signaloud*" ("sinal alto/falado"), criado para concorrer ao prêmio Lemelson do MIT. Este estudo possibilitou a tradução em tempo real de alguns termos da ASL para o inglês apenas com o auxílio de duas luvas sensoriais. Para sua utilização, era necessária uma calibragem dos sensores para o uso de diferentes pessoas. Toda a comunicação dos microcontroladores é realizada via *Bluetooth* até seu sistema em um computador central. Para reconhecimento dos gestos, utiliza-se uma sequência estática de regressões lineares pareando a palavra que mais corresponde ao movimento.

No artigo "*A Fuzzy Rule-Based Approach to Spatio-Temporal Hand Gesture Recognition*" [46] de Mu-Chun é feita uma análise que utiliza a lógica *fuzzy* para o reconhecimento espaço-temporal dos movimentos realizados pela mão. Para o treinamento de seu modelo, aplica-se uma rede neural baseada em estruturas hiper-retangulares que se ajusta aos seus dados compostos por 90 exemplos de gestos espaço-temporais. Em seu estudo, o pesquisador considera os gestos de mão divididos em 4 partes: a configuração da mão, a orientação da palma, a posição/localização e a movimentação.

Em "*Sign Language Recognition using Sensor Gloves*" do Yasir Niaz Khan [30] é discutida a possibilidade de construção de uma luva sensorial para a tradução dos sinais em ASL para o inglês. Nele é implementado o projeto "*talking hands*" em que é utilizada uma rede neural para classificar seus dados em 26 diferentes classes. O dispositivo desenvolvido é composto por: cinco sensores para medição da flexão de cada dedo; um sensor para medir a inclinação a mão; e um sensor para medir a rotação. Neste estudo é encontrada uma acurácia de 88% de seu modelo em relação aos seus dados. Porém, seu projeto não conseguia lidar bem com gestos dinâmicos presentes na ASL.

Outro modelo pode ser observado no artigo do Ji-Hwan, "*3-D Hand Motion Tracking and Gesture Recognition Using a Data Glove*" [26]. Nele é desenvolvida uma luva capaz de reconhecer a estrutura em 3 dimensões da mão de um usuário e representá-la de forma digital.

Para a confecção deste material, foram utilizados 3 acelerômetros, 1 controlador e 1 dispositivo de comunicação *bluetooth*. Apesar de não ser um trabalho específico para a ASL ou Libras, ele possui todo o ferramental para este tipo de aplicação. Em seus exemplos, foi produzido um detector de pedra, papel e tesoura.

Outra arquitetura de captura de dados para a obtenção de um modelo de tradução de ASL para o inglês pôde ser observado no artigo do Abhishek Kalpattu [2]. Para sua estrutura, são utilizados apenas sensores de toque para a captação de dados, sendo 8 no total. Apesar da simplicidade, obteve-se uma acurácia de 92% para a classificação de números de 0 a 9 e das letras de A à Z com um custo de produção muito mais acessível que os demais apresentados. Seus sensores só são capazes de detectar 2 estados, ligado e desligado, conforme o sensor reconhece o toque ou não. Essa abordagem tende a ser muito limitado conforme seu banco de palavras aumenta pela baixa quantidade de combinação de atributos.

Na área de reconhecimento facial, mesmo que não específico para LIBRAS, no estudo da Priyanka Dwivedi [39], foi utilizada uma rede neural convolucional [6] para a realização da classificação de expressões faciais presentes em uma imagem. Através de um pré-processamento, e com auxílio do algoritmo *Haar Cascade*, tornou-se possível identificar um rosto e rotulá-lo, mesmo que este não esteja no centro da imagem. Para a construção de seu modelo é utilizado o conjunto de dados FER [25]. Sua rede consiste em inserção de camadas de "*max pooling"*, "*batch normalization"* e camadas de convolução total.

Na abordagem de utilização de câmera para a detecção dos sinais de mão, são utilizados dois estudos como parâmetro, sendo o primeiro deles, referente à ASL, o projeto da Helene Brashear [9] exposto em "*Using Multiple Sensors for Mobile Sign Language Recognition*". Nele a captura das posições e configurações de mão é realizada através da câmera do celular, o que torna o projeto mais acessível financeiramente.

Outros fatores favoráveis ao trabalho acima referem-se aos experimentos operados com uma câmera de pouca resolução e a realização de um tratamento para dados com muito ruído. Em conjunto com a imagem, aplicou-se com um conjunto de acelerômetros os dados espaciais do usuário. Para a classificação, é utilizado um modelo de cadeia de Markov oculta (HMM), alcançando-se uma acurácia de 90.48% para a combinação dessas 2 abordagens (câmera e acelerômetros).

Ainda nesta perspectiva, existe o trabalho nacional do Jonilson Santos [42], "Reconhecimento das configurações de mão de LIBRAS baseado na análise de discriminantes de Fisher bidimensional, utilizando imagens de profundidade". A captura das imagens neste caso foi realizada com uma câmera com sensor Kinect (desenvolvida pela *Microsoft*, utilizada

em seus consoles de jogos, o *Xbox*), capaz de gerar imagens de altíssima qualidade e, ao mesmo tempo, ser acessível.

Para um pré-processamento dessas imagens aplicou-se a técnica de 2D²LDA [43] de redução de dimensionalidade que foi então utilizada para a classificação dos sinais. O sistema é capaz de fazer a segmentação da mão e é provido de 61 diferentes configurações para a classificação. Além disso, sua solução é insensível à luminosidade, o que a torna muito adequada para diferentes tipos de ambientes, independentemente da luz do local. Em seus experimentos, foi alcançada uma acurácia média de 95,7%. Seu lado negativo, todavia, é a pouca capacidade de mobilidade e sua consequente dificuldade de utilização em diversos lugares em consequência disto. Para utilização em uma sala de aula fixa, por exemplo, pode ser uma solução muito interessante.

Por fim, outro trabalho nacional que descreve o uso de uma luva sensorial é a ALFALUVA [44] que, em conjunto com um aplicativo de celular, auxilia no aprendizado de LIBRAS fazendo o reconhecimento de alguns sinais. Apesar de ainda possuir alguns problemas na detecção e apresentar necessidade de calibração para cada usuário, sua ideia pode facilitar o aprendizado desta língua de forma interativa.

Para essa classificação foi utilizada uma árvore de decisões que identifica os dados recebidos por sensores flexíveis em cada dedo e um acelerômetro. Dentre os sinais reconhecidos, estão presentes todas as letras de A à Z e, para a formação de palavras, o usuário precisa fazer a soletração.

Apesar de tudo, os projetos observados não possuíam um tratamento para as expressões faciais em conjunto com as configurações de mão que também compõem um termo em LIBRAS. Logo, tendo em vista o comportamento dos casos anteriores, este novo projeto tem o objetivo de solucionar alguns dos problemas encontrados supracitados e abranger o máximo possível dos pontos positivos de cada um deles. Dentre eles, foram aplicadas: a aplicação e avaliação de mais algoritmos de aprendizado de máquina para a criação de modelos mais consistentes, a utilização de um mapeamento de dados dos sensores com diferentes tempos, a criação de um *dataset* específico de configurações de mão para o problema e a utilização do reconhecimento de expressões faciais.

3 CONCEITOS BÁSICOS

Nos últimos anos, muitos estudos relacionados à área de inteligência artificial vêm sendo realizados e novos algoritmos vêm sendo viabilizados de forma prática. Em conjunto com este cenário, a evolução dos microcontroladores e sensores possibilitou a união destas duas áreas, que permitiu a criação de novas tecnologias a fim de buscar soluções para problemas mais complexos.

Para viabilizar a tarefa de tradução de forma escalável, sem restrições que necessitem de ajustes manuais, foram utilizados algoritmos de aprendizado de máquina para a classificação de padrões. Dentre o que se busca classificar, podemos citar: a expressão facial do usuário e o sinal de mão realizado. Fazendo a composição dessas características de forma eficiente, é possível identificar um termo em LIBRAS e, a partir deste mapeamento, é relacionado a uma palavra em Português.

No sistema de classificação de expressões faciais utilizaram-se os algoritmos: "*HaarCascade*", "KNN", "Regressão Logística", "XGBoost", "*Randon Forest*" e "Redes Neurais Convolucionais". Para o sistema de classificação de sinais de mão foram utilizados os algoritmos: "KNN", "XGBoost" e "SVM". Todas essas técnicas foram explicadas de forma simplificada neste capítulo, separadas por sistema.

Este trabalho utiliza tais algoritmos como ferramental, porém, para um entendimento mais claro de suas necessidades, são enaltecidos fatores relevantes utilizados no desenvolvimento neste capítulo. Todos os funcionamentos dos métodos descritos são intrínsecos deles e não houve modificação para este projeto, logo ele funciona comoo descrito para qualquer conjunto de dados. Nota-se que este estudo não buscou aperfeiçoar tais algoritmos, mas sim prepara-lo com finalidade de obter a melhor saída/resultado possível para um determinado dado de entrada para o problema.

Além de tais algoritmos, para alimentá-los e reproduzir os resultados foramnecessárias bibliotecas específicas em "*Python*" importantes para o desenvolvimento do projeto e da utilização de um microcontrolador com seus respectivos sensores. Mais detalhes técnicos sobre o funcionamento dos mesmos são explicitados nesta seção.

Para uma melhor organização dos tópicos citados, os assuntos foram divididos em 3 diferentes seções. A primeira apresenta noções gerais necessárias para o entendimento da implementação do sistema e do funcionamento de algoritmos de inteligência artificial. O subcapítulo seguinte separa os conceitos referentes a algoritmos utilizados para a realização da

classificação de expressões faciais. E por último, é apresentado o ferramental necessário para a criação da luva sensorial e seu sistema de classificação.

Como muitos desses assuntos possuem uma complexidade elevada, é recomendada a leitura das referências para mais detalhes do funcionamento de cada tópico. Neste trabalho, o foco está presente na arquitetura elaborada para a solução do problema, bem como nos dados de entrada e saída para cada um deles. Para cada conceito a seguir foi explicitado o conhecimento básico necessário para o entendimento da motivação da utilização de cada um deles.

3.1 NOÇÕES GERAIS

Para o melhor entendimento dos tópicos referentes a sistemas específicos de aprendizado de máquina utilizados no projeto, os itens a seguir tem por objetivo criar uma base de conhecimento unificada, que será utilizada nas explicações subsequentes. Dentre os temas abordados nesta seção estão presentes: As ferramentas de programação utilizados no projeto e o conceito básico de aprendizado de máquina que será utilizado para todos os algoritmos.

3.1.1 Ferramentas de implementação

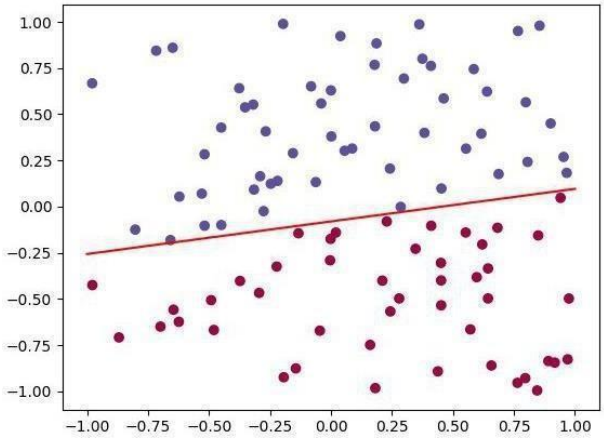
Na confecção do projeto, foi utilizada a versão 3 do "*Python*", pela facilidade e confiabilidade que o conjunto de bibliotecas do "*Sklearn*" [37] provêm ao desenvolvedor, sobretudo quando se trata de implementação de algoritmos de aprendizado com máquina. Para uma maior compatibilidade nos métodos utilizados, optou-se pela utilização somente desta biblioteca para os algoritmos de IA. Logo, comparações relacionadas à eficiência e tempo de execução podem ser feitas sem viés que pode ser causado pela utilização de implementações de bibliotecas distintas.

3.1.2 Notações e introdução ao aprendizado de máquina

Para estabelecer um padrão de notação para os próximos capítulos, este item esclarece alguns conceitos básicos que são utilizados na explicação dos demais algoritmos. O principal objetivo ao utilizar um algoritmo de aprendizado de máquina é estipular ou aproximar uma função que consiga fazer a separação dos pontos com seus respectivos rótulos . Tem-se

como exemplo a **Figura 1** a seguir, onde é demonstrada uma representação em duas dimensões de dados gerados artificialmente, em que os valores do vetor correspondentes as coordenadas cartesianas. Para os rótulos, é realçado os pontos de diferentes cores (vermelho e azul) de acordo com sua respectiva classe.

Figura 1 - Exemplo de uma distribuição de pontos rotuladas linearmente separáveis



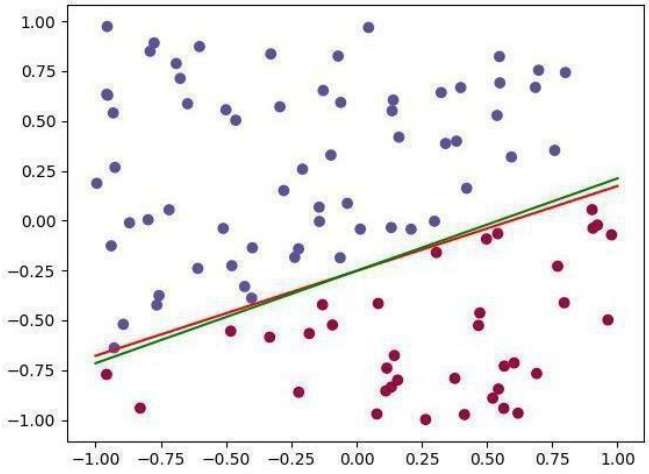
Fonte: Reprodução do autor

Nesta mesma **Figura 1** está representada por uma linha vermelha a função de separação "perfeita", também conhecida como "função objetivo" e esta é caracterizada pela simbologia (). Vale ressaltar que ela não é possuída nos problemas e está apenas ilustrada no exemplo, uma vez que foi criada artificialmente. A curva (ou reta) aproximada que é modelada para ser a mais próxima possível de () será representada pela simbologia (), e ela é nomeada de "função de hipótese".

Com a finalidade de alcançar a melhor representação possível da função objetivo, considera-se um conjunto de hipóteses , composto por diversos algoritmos diferentes com suas respectivas configurações de parâmetros (seja um algoritmo de vizinhos mais próximos ou um algoritmo de divisão do espaço em hiperplanos como o SVM – *Support Vector Machine*).

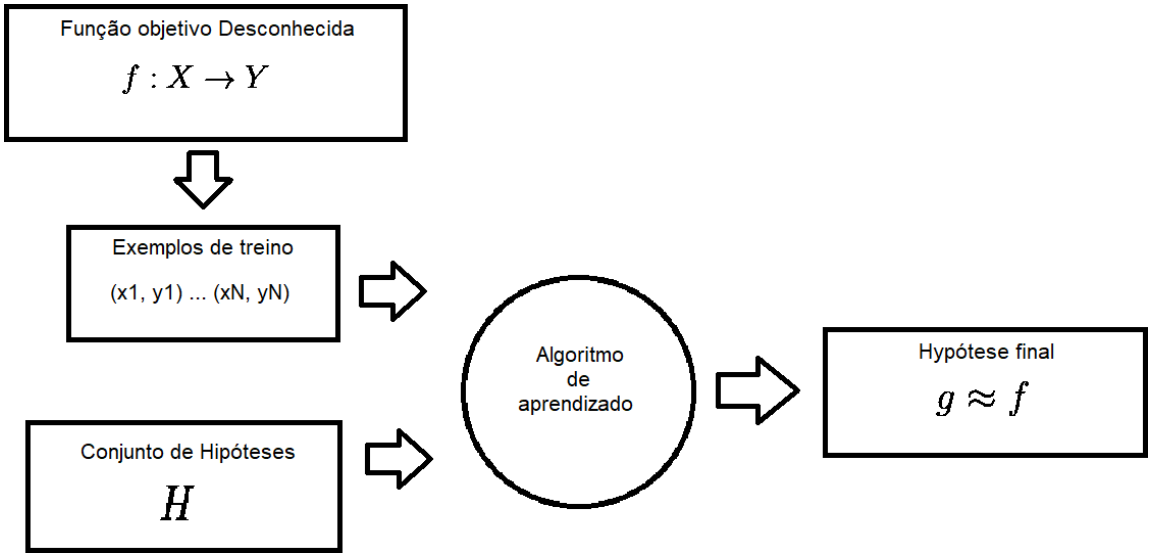
Esse conjunto deve ser explorado (em partes) para se obter a melhor separação dos dados que gera comparações entre diferentes () e seleciona a melhor delas. Um exemplo de representação de como esta se comportaria, neste caso, pode ser observado na **Figura 2**, na reta em **verde**. Além disso, na **Figura 3** está um esquema simplificado que ilustra o funcionamento geral descrito em um fluxograma.

Figura 2 - Exemplo de uma distribuição de pontos rotuladas linearmente separáveis com g(x)



Fonte: Reprodução do autor

Figura 3 - Fluxograma de criação de uma função objetivo com base nos dados



Fonte: Reprodução do autor

Em todos os experimentos realizados neste projeto foram utilizadas técnicas de aprendizado supervisionado [50], tendo em vista que o conjunto de dados é composto por uma matriz de atributos □ e uma única classe □ para cada exemplo. Neste tipo de aprendizado, os algoritmos do conjunto de hipóteses possuem os rótulos (classes) dos exemplos que são recebidos pelo classificador. Logo, o número definido de classes é equivalente ao número real que o problema possui.

O retorno esperado deste tipo de algoritmo é que, dado um conjunto de dados □ que não possuem um rótulo, seja possível mapeá-los para os rótulos observados □, sendo que este seja

o certo. Sua corretude provém do já conhecimento do resultado, anteriormente omitido, ou que por verificação e discernimento da situação era o esperado.

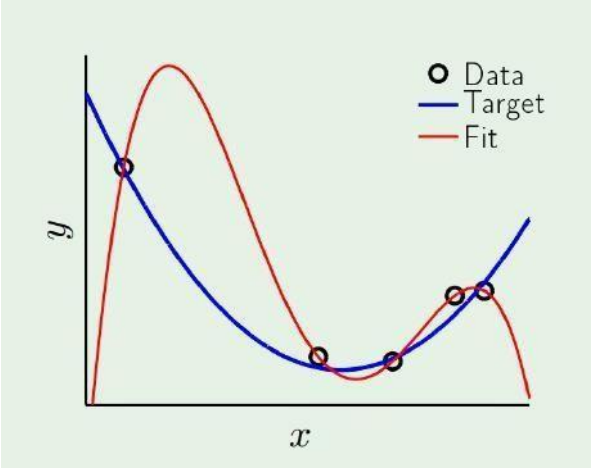
Em busca de bons resultados, é necessário fazer uma divisão dos dados (□). A primeira parte é composta pelos responsáveis por construir a função □(□), também conhecidos como "conjunto de treino". A segunda parte é responsável por verificar se □(□) realmente conseguiu fazer a divisão de forma correta, esses são chamados de "conjunto de teste". Nota-se que o primeiro não pode ser utilizado para a construção da função separadora, pois isso tiraria todo o propósito de conseguir construir uma função capaz de dividir dados nunca antes vistos.

A partir da classificação dos dados presentes no conjunto de teste, é possível extrair métricas como a acurácia do modelo, pois ele possui o vetor de rótulos verdadeiro □. Considera-se que o modelo realiza um acerto caso □(□□) = □(□□) = □□. Vale ressaltar que, no caso de dados rotulados, são dispostos os resultados de □(□□), porém não é de acesso à função verdadeira, que é exatamente o que se busca estimar com □(□).

Um detalhe que gera muitos problemas em várias implementações é que o conjunto de teste não pode ser utilizado na criação do modelo e esse não deve ter qualquer influência na construção do mesmo. Caso isso aconteça, existe uma grande possibilidade de ocorrer um *overfitting*, que consiste no ajuste extremo de uma função aos seus dados descritos, conforme na **Figura 4**.

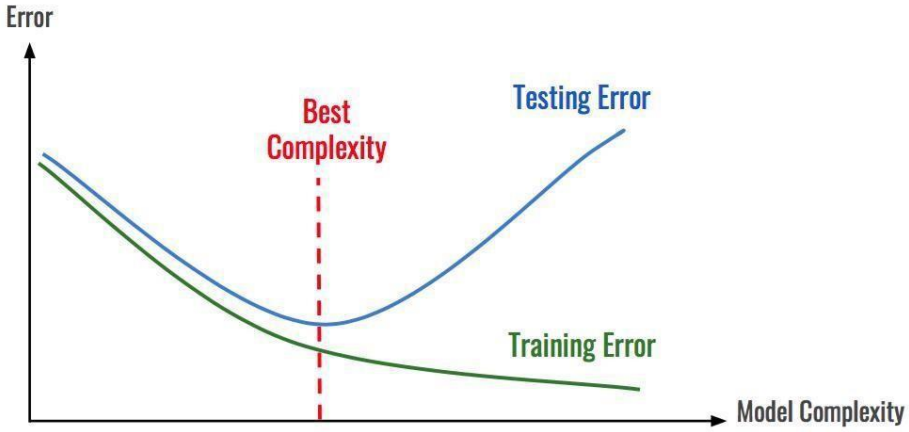
Inicialmente obtém-se uma taxa de acerto maior para os dados, porém, ao analisar fora da amostra, a □(□) pode ter apenas "decorado" os rótulos previamente estabelecidos, o que causa resultados muito ruins. Isso acontece pelo aumento da complexidade da função sem que exista uma compensação no acréscimo da quantidade total de dados [1]. Observa-se um exemplo deste comportamento na **Figura 5**.

Figura 4 - Exemplo de *overfitting* de dados



Fonte: ABU-MOSTAFA (2012)

Figura 5 - Relação entre o aumento da complexidade do modelo e o erro esperado



Fonte: KAGGLE (2010)

Os hiperparâmetros são variáveis que controlam o funcionamento interno de um modelo. Para ajustá-los em cada um dos algoritmos, dentre os dados do conjunto de treino, separa-se uma nova parcela chamada “conjunto de validação”. Esses são utilizados pelos algoritmos para conseguirem ajustar métricas “internamente” e assim obter hiperparâmetros que melhor se adequam ao problema. Este ajuste também pode ser utilizado para evitar problemas como *overfitting*.

A separação do conjunto de treino e do conjunto de validação é feita através do uso de diferentes técnicas, dentre elas o *KFold* [41], que faz a divisão aleatória dos dados em K conjuntos diferentes. Ela é utilizada por apresentar melhor aproveitamento dos dados que já existem, principalmente quando existem poucos. Em cada iteração, um desses conjuntos é utilizado para a validação, enquanto os outros K-1 são utilizados como treino para a criação do modelo.

Dessa forma, com a mesma configuração do algoritmo é possível criar K diferentes modelos, porém será utilizado somente aquele que tiver o melhor desempenho. Pode-se também utilizar a média dos resultados para estimar se esta abordagem é ou não eficiente. Esta técnica é utilizada para fazer a validação cruzada do conjunto de dados, a fim de que o resultado consiga extrair o máximo de características através da otimização dos hiperparâmetros.

3.2 EXPRESSÃO FACIAL

Os itens a seguir são referentes ao conhecimento necessário para a elaboração do sistema de classificação de expressões faciais. Os dados de entrada (□) para todos esses algoritmos no

projeto é uma imagem que contêm a face de uma pessoa. A saída esperada é a expressão facial correspondente (□) da pessoa presente na imagem.

Para um tratamento preliminar da imagem que alimenta esses algoritmos é utilizado o "*HaarCascade*" que têm por objetivo selecionar somente a face de uma pessoa perante toda a imagem. Este tratamento foi feito de forma automática no sistema final, que redireciona a saída deste módulo para a entrada dos algoritmos de IA. São necessários vários algoritmos para obter a função de classificação que melhor se adapta aos dados de entrada. Para isto, são utilizados os algoritmos: "KNN", "Regressão Logística", "XGBoost", "*Randon Forest*" e "Redes Neurais Convolucionais".

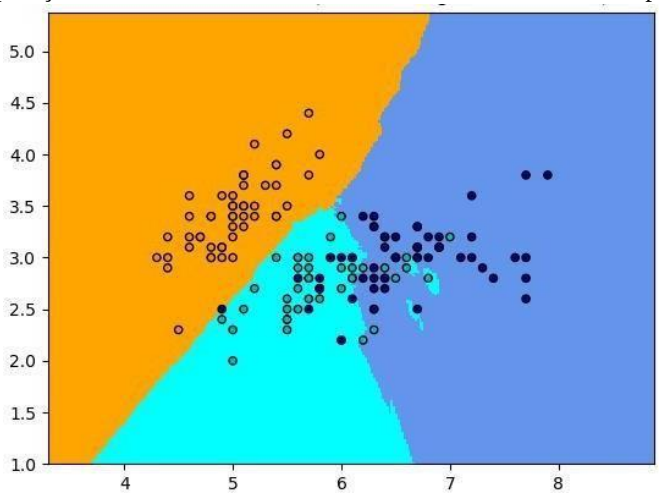
3.2.1 KNN

Segundo Gongde [22], o KNN (*K-Nearest Neighborhood*) é um método de classificação não paramétrico simples e efetivo em muitos casos. Esse tem como objetivo buscar os □ vizinhos de um determinado dado, baseado na distância entre ele e as outras amostras no espaço das *features*, sendo geralmente aplicada a distância euclidiana como métrica. Com um sistema de votação, classifica-se esta determinada amostra com base na classe que estiver mais presente dentre os seus vizinhos.

Um ponto negativo do funcionamento deste algoritmo é que a escolha do valor de □ pode enviesar o resultado encontrado para a classificação de padrões. Outro problema é o fato dele não criar de verdadeiramente um modelo (uma equação). Ele apenas guarda as instâncias de treino e calcula suas distâncias em relação ao ponto em questão para encontrar os □ pontos mais próximos. Logo, para cada classificação, calcula-se a distância entre seu dado e todos os outros já classificados, obtendo uma ordem de complexidade □(□), enquanto um algoritmo composto por um modelo em forma de função obtém a classificação em □(1).

Uma das formas de solução para este problema de enviesamento é utilizar uma busca a fim de otimizar o valor de □ encontrado para o seu *dataset* utilizando o conjunto de validação. Desta forma, seu modelo é preparado para ter a maior compatibilidade com seu conjunto de treino. Na **Figura 6** observa-se um exemplo de como é feita essa separação do espaço das *features* utilizando o *KNN* tiradas do site do *sklearn* [37]. Para cada cor de região é atribuída uma classe, de acordo com os dados de treino do modelo.

Figura 6 - Separação de dados utilizando KNN com 3 classes distintas no plano cartesiano



Fonte: PEDREGOSA, et. al. (2011)

3.2.2 Regressão logística

Regressão logística [8] é uma técnica estatística que tem por objetivo classificar de forma categórica os dados de entrada. Neste algoritmo, é construído um vetor de pesos □ que multiplica cada entrada □ de seu conjunto de dados. Seu resultado final é uma probabilidade (entre 0 e 1) de um dado pertencer a uma classe de maneira binária. Com esta técnica, diferentemente de uma regressão linear comum, é possível avaliar o quão provável um dado pode pertencer a uma determinada classe, o que na abordagem anterior não era possível.

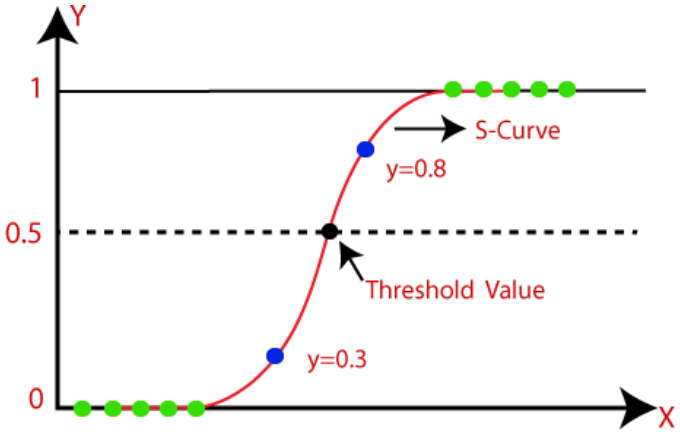
Para criar essa função de probabilidade de forma suave, utiliza-se a função sigmoide □ = $\frac{1}{1+□^{-□(□)}}$ em que □(□) é a função de pesos e variáveis. Para se obter os pesos ideais para a classificação do algoritmo, este método calcula iterativamente a função de erro (*loss function*), representada pela seguinte equação *loss* = $^{1}*\sum_{□}^{□}\ln(1+□^{-□□*□(□□)})$. Tal que □ e □ são referentes ao n-ésimo dado de seu *dataset*.

Para calcular a minimização do erro de forma iterativa, utiliza-se a função de gradiente descendente. Essa avalia a função atual em relação ao seu erro e então desloca seus pesos na direção oposta do gradiente da função, a fim de se obter o mínimo. O gradiente é aplicado ao ponto até que a norma da diferença entre os pesos seja menor do que um erro □.

Com a obtenção dos pesos, basta aplicar a função sigmoide para ter como resultado a probabilidade de um dado estar na classe definida pelo problema. Para se dividir em 2 diferentes classes, utiliza-se uma função constante no valor de 0.5, chamada *threshold*, que servirá de divisor entre dados classificados como classe 1 e 0. Na **Figura 7** é exemplificada a forma com

que a regressão logística modela sua curva com relação aos dados e ilustra como é definido o ponto de separação entre uma classe e outra.

Figura 7 - Exemplo de criação de um modelo utilizando a regressão logística



Fonte: JAVATPOINT (2020)

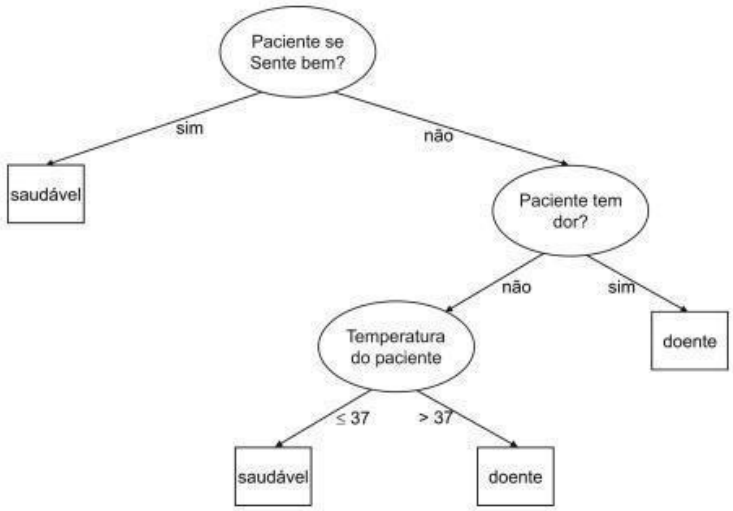
3.2.3 Árvores de decisão

Para desenvolver os conceitos envolvidos nos algoritmos "*XGBoost*" e no "*Random Forest*", é necessário introduzir o conceito de árvores de decisão. Segundo Monard [38, p.60]:

> Uma árvore de decisão é uma estrutura de dados definida recursivamente como: um nó folha que corresponde a uma classe ou um nó de decisão que contém algum teste sobre algum atributo. Para cada resultado do teste existe uma aresta para uma subárvore. Cada subárvore tem a mesma estrutura que a árvore.

Exemplifica-se esta definição na **Figura 8,** construída por [32], em que é retratado um problema de classificação para se fazer o diagnóstico de um paciente de forma simplificada. Nela, observa-se que os nós-folha representam apenas a classe do problema, enquanto os nós intermediários e raiz representam algum tipo de pergunta que são substituídos por *features* dos dados em casos reais.

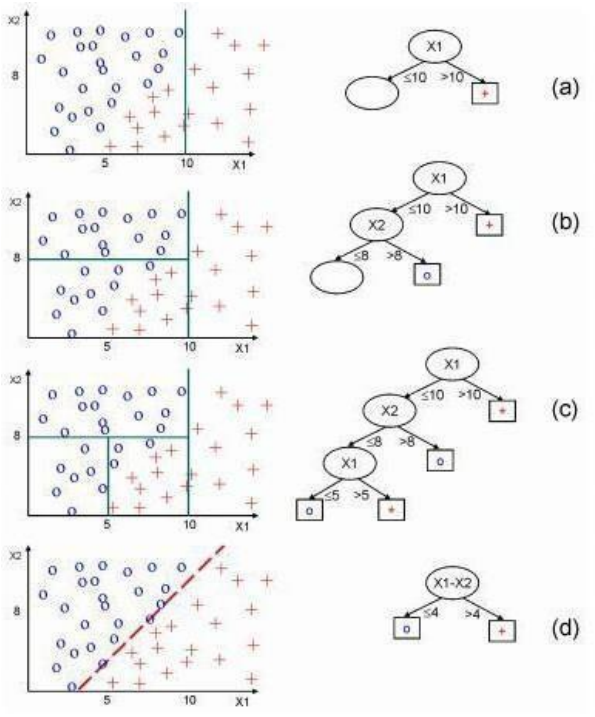
Figura 8 - Exemplo de árvore de decisão simples para a classificação de pacientes doentes ou saudáveis



Fonte: MONARD (2003)

Analisando de forma analítica este problema, o que é realizado retrata a divisão do espaço em hiperplanos a cada nó intermediário. Cada tomada de decisão feita nos nós intermediários divide o espaço amostral em duas regiões. No final do processo, é designada cada uma dessas regiões a uma determinada classe. Este comportamento pode ser ilustrado pela **Figura 9** de forma iterativa.

Figura 9 - Divisão geométrica do espaço com as árvores de decisão



Fonte: MONARD (2003)

3.2.4 XGBoost

O XGBoost ("*extreme gradient boost*") é um algoritmo baseado em árvores de decisão que utiliza comitê de *boosting* para aprendizado, além de reduzir o erro dentro da amostra através do gradiente descendente [18]. Este método foi publicado em 2016 no artigo [13] pelo Tianqi Chen e em seu próprio site [12] é citado que o algoritmo possui uma série de soluções vencedoras das competições do Kaggle [25], assim como da KDDCup.

Com essas conquistas, o algoritmo se destaca muito pela sua capacidade de classificação eficiente de dados, além de conseguir trabalhar de forma muito eficiente para conjuntos de dados pequenos. Uma explicação mais detalhada sobre as técnicas utilizadas pelo algoritmo é presente em [34] 'Why Does xgboost win "every" machine learning competition'.

Um dos motivos para sua ampla utilização é o fato dele já implementar várias técnicas que melhoram o desempenho da classificação, como a regularização, o *boosting*, o gradiente descendente e outros métodos que previnem *overfitting* como o *Shrinkage* e o *Column subsampling*. Além de ser rápido e otimizado para computação *out-of-core*, ele tambémpossui outros aprimoramentos no nível de performance para dados de grande escala, porém estes não são tão relevantes para este projeto. Basicamente é um conjunto de técnicas que já existiam, porém, combinadas em sua implementação.

O algoritmo já possui biblioteca implementada em *python* [52]. Em sua forma padrão é possível avaliar seu desempenho sem grandes modificações em seus hiperparâmetros, otimizados pelo próprio algoritmo e desta forma é possível observar a flexibilidade deste modelo em relação aos outros.

3.2.5 Random forest

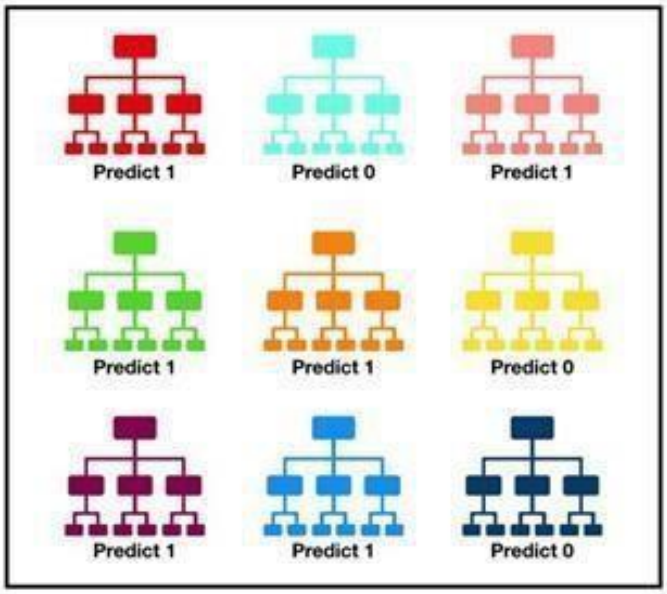
O *Random Forest* (RF) [11], assim como o *XGBoost*, é um algoritmo baseado em árvores de decisão, porém, ao invés de *boosting*, ele utiliza outra técnica de comitê para conseguir bons resultados. Neste algoritmo é utilizado um sistema de votação de classe feito por □ diferentes árvores geradas pelo sistema de treinamento, sendo eleita a classe que foi mais votada dentre as árvores em relação a uma dada amostra, chamada de *bagging*. Para explicar de forma simplificada esse conceito, separa-se o nome do método em duas partes: a etapa de randomização e a etapa de criação da "floresta".

Primeiramente, para a construção de diferentes árvores, seleciona-se conjuntos de dados de forma randômica , de modo que todos sejam completamente independentes dos selecionados anteriores e posteriores, utiliza-se da mesma distribuição. É feito então o treinamento de cada uma das árvores com os dados dos conjuntos de forma independente.

Com isso é criada uma "floresta" de diferentes árvores que podem ser escritas na forma de modelo como *h* (,), em que X representa a matriz de entrada. Com a construção das árvores que compõem o modelo, o sistema de classificação é feito mediante o voto destas em relação à amostra inserida, sendo escolhido o mais votado dentre eles. Pode-se ver um exemplo deste comportamento na **Figura 10**. Nesse exemplo, o resultado em seis predições é de classe 1 e três predições de classe 0.

Este método é muito utilizado dentre os algoritmos de classificação baseados em árvore de decisão, pois ele possui uma capacidade de generalização muito sofisticada. Tendo em vista que diferentes árvores classificam uma única amostra, os "vícios" de uma determinada árvore são anulados pela classificação em conjunto das outras restantes, gerando uma grande camada de generalização.

Figura 10 - Exemplo da predição das k diferentes árvores no *random forest*



Fonte: YIU (2019)

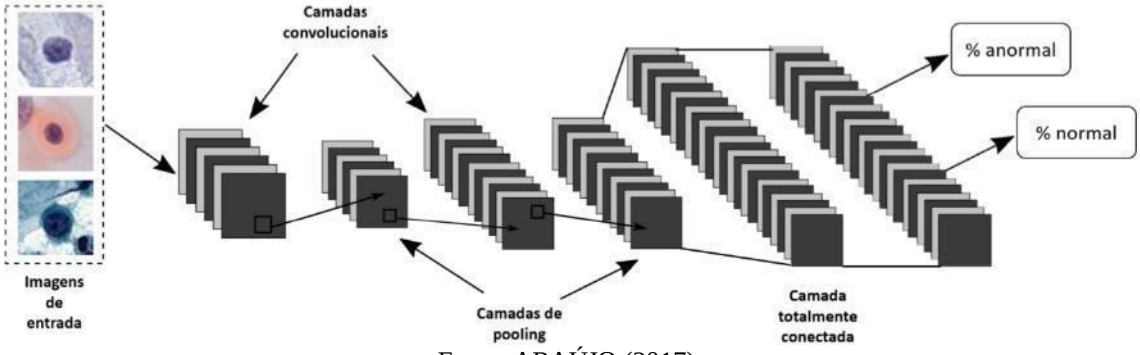
3.2.6 Redes neurais convolucionais

Quando se fala sobre *deep learning*, um dos principais algoritmos de classificação para esse tema são as redes neurais convolucionais (CNN) [51]. Por mais que essas tenham uma

estrutura semelhante a uma rede neural convencional, elas possuem também camadas de convolução, *pooling* e *flatten*, tornando seu modelo mais sofisticado para a classificação de padrões.

Dada uma arquitetura de redes neurais, cada uma das células nas camadas tem como objetivo fazer a propagação de um sinal que funcionam como um classificador simples. Esses têm como entrada um vetor que evolui conforme o decorrer da rede. Dentre as estruturas mais simples de CNN, a *LeNet*, segue a estrutura como mostrado na **Figura 11**.

Figura 11 - Arquitetura de uma rede *LeNet*



Fonte: ARAÚJO (2017)

Em sua camada de convolução, a rede utiliza filtros de redução de dimensionalidade em seus dados, de forma a minimizar a perda de informação que ocorre nessa iteração. Contudo o modelo se torna mais genérico e reduz a complexidade do sinal propagado, diminuindo a quantidade de atributos que são passadas para a próxima camada.

Em sua camada de *pooling*, também é feita uma redução de dimensionalidade dos dados. Todos atributos de certa região são substituídos pelo valor do maior atributo ou a média dentre eles de forma aleatória. Esta técnica também é utilizada para a prevenção de *overfitting* e redução de complexidade.

Na camada de *flatten* é utilizada como conectora da camada convolucional da rede eda camada totalmente conectada. É operada uma transformação na matriz de saída da primeira região que resulta em um vetor de uma dimensão com toda a informação que serve de entrada para a segunda região.

Em adição, existem as camadas completamente (ou parcialmente) conectadas que também estão presentes nas redes neurais convencionais com suas funções de ativação. Com a composição sequencial destas técnicas é possível utilizar esta estrutura juntamente com o algoritmo de *backpropagation*. Esse faz com que a rede passe a “aprender” de acordo com os

dados que são processados nesses neurônios, ajustando assim o peso de cada um deles para os novos dados de entrada [15].

Devido a sua alta popularidade como algoritmo de classificação e por já possuir estudos na área de classificação de imagens [39], ele é um bom candidato para o conjunto de hipóteses . Outra vantagem que ele possui é o fato de ser muito flexível para qualquer tipo de dado, desde que seja utilizado da forma correta.

3.2.7 Haar Cascade

Para obter bons resultados a partir da captura da imagem de uma pessoa é necessário um pré-processamento, enviando para os algoritmos de classificação apenas a face do usuário, sem o seu plano de fundo. Para tal tarefa, foi escolhido o *Haar cascade* [47], também conhecido como "algoritmo de Viola-Jones", que é uma técnica de aprendizado de máquina focada para a detecção de objetos visuais. Ele é capaz de processar imagens rapidamente, em tempo real, e de alcançar bons resultados de detecção. Para entender o funcionamento deste método, nos próximos parágrafos são explicados como funcionam as "*haar features*", a imagem integral (responsável pela eficiência do algoritmo) e a função de classificação em cascata.

Diferentemente de outros algoritmos de classificação de imagens, foi observado pelos criadores que trabalhar diretamente com os *pixels* da imagem possuía resultados menos efetivos do que a criação de *features* em relação a eles. Essas são formadas conforme a **Figura 12**, em que o valor final de cada um desses atributos é igual ao valor somado dos *pixels* presentes no retângulo branco subtraídos da soma deles no retângulo cinza.

Faz-se diversas combinações exaustivas desses retângulos ao longo de toda a imagem, mapeia-se todos os atributos presentes nesta imagem em relação a este tipo de abordagem. Nota-se que a execução de todas as combinações possíveis de retângulos pode ser muito custosa. Em função disso, foi introduzido o conceito de imagem integral.

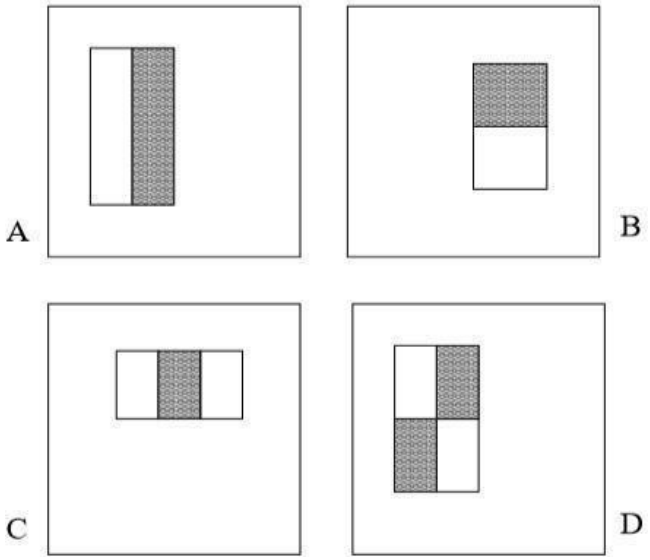
A imagem integral traz uma forma fácil de calcular a soma dos valores dos *pixels* dentro de cada um destes retângulos. Para isto, cada um destes valores será substituído pela seguinte fórmula:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y}^{x, y} i(x', y')$$

Em que $ii(x, y)$ é o valor da imagem integral no ponto (x, y) equivalente a soma de todos os valores (tons de cinza, entre 0 e 255) dos pontos de coordenadas x e y até ele (com o

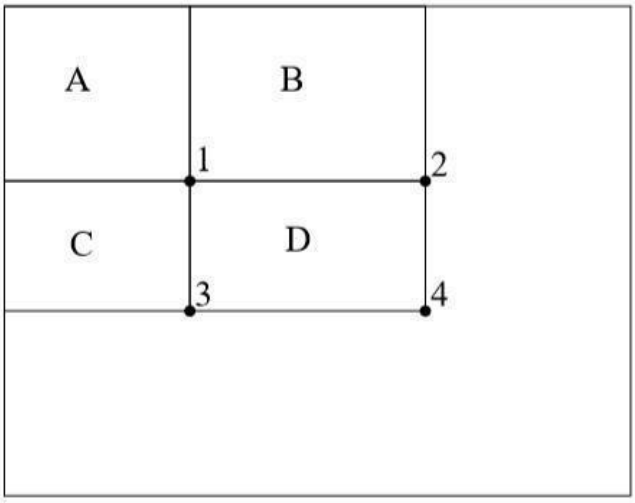
próprio incluso). Para computar o valor de cada retângulo, aplica-se a formula □ = □4 + □1 − (□2 + □3), em que os números são os pontos □□ da **Figura 13**.

Figura 12 - Exemplo de formação de *haar features*



Fonte: VIOLA (2001)

Figura 13 - exemplo de conta rápida para obtenção dos valores de D a partir da imagem integral

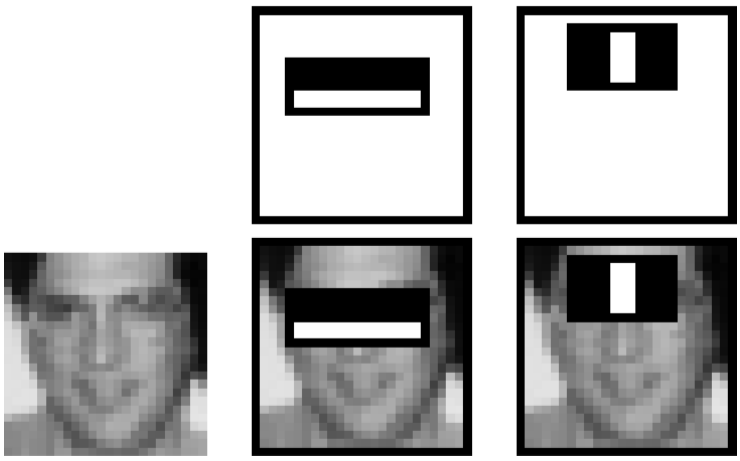


Fonte: VIOLA (2001)

Para selecionar as melhores *features* que compõem o modelo, é utilizado o método de aprendizado por *boosting*, o *AdaBoost*. Como são muitas combinações possíveis de atributos, este método é encarregado de selecionar apenas as mais relevantes dentre todas, criando o espaço de representatividade.

Esse modelo é criado por diversos classificadores “fracos” e, para cada um deles, o algoritmo faz o ajuste de seus *thresholds* de forma a minimizar o número de exemplos classificados de maneira incorreta. O objetivo de cada um destes classificadores fracos é selecionar uma única *feature* que melhor separa os exemplos positivos e negativos. Todavia, por hipótese do autor [47], são necessários poucos atributos para, de fato, conseguir classificar a imagem corretamente. Algumas *features* selecionadas são exemplificadas na **Figura 14**.

Figura 14 - Exemplo de seleção de atributos

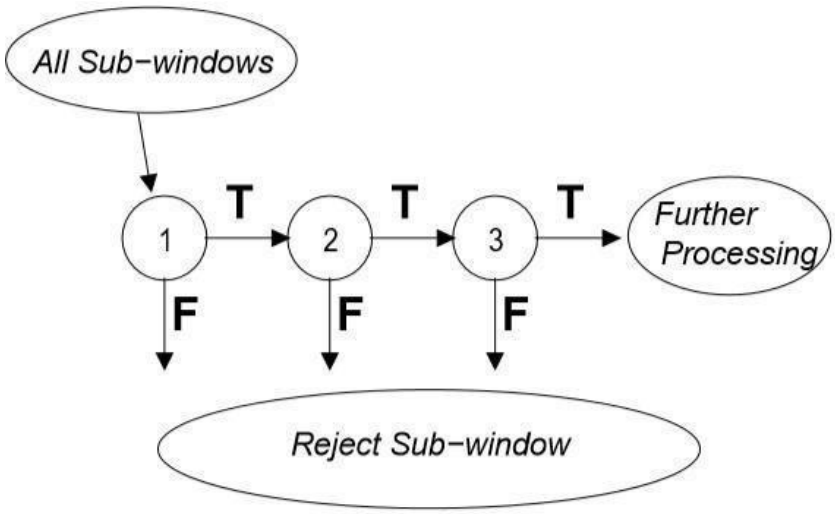


Fonte: VIOLA (2001)

Finalmente, para o processo de classificação final, o autor desenvolveu um processo de cascata, no qual aumenta a performance de detecção e diminui o tempo de processamento. Os *thresholds* dos classificadores fracos são ajustados de tal forma que a quantidade de falsos negativos fique muito próxima de zero.

Em contrapartida, isso aumenta a quantidade de falsos positivos. Logo a sequência de classificadores rejeita em poucas iterações a maioria das sub-janelas que foram classificadas de forma negativa, enquanto os mais complexos nesta sequência só classificam as sub-janelas que têm mais tendência de serem as imagens desejadas. Observa-se este comportamento na **Figura 15**.

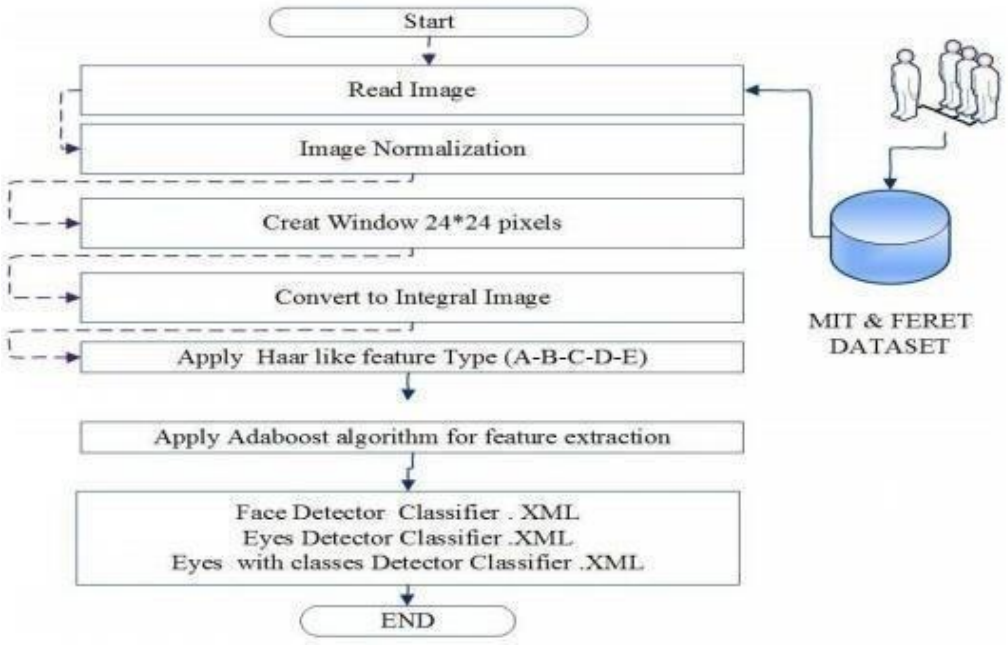
Figura 15 - Processo de classificação de sub-janelas do *haar cascade*



Fonte: VIOLA (2001)

Em resumo, no fluxograma presente na **Figura 16** são indicados todos os procedimentos feitos pelo algoritmo.

Figura 16 - Fluxograma completo do funcionamento do *haar cascade*



Fonte: OGLA (2017)

3.3 LUVA SENSORIAL

Os itens a seguir são referentes ao conhecimento necessário para a elaboração do sistema de classificação de sinais realizados com as mãos. Os dados de entrada (□) para todos esses

algoritmos no projeto é um conjunto de dados capturados pelos sensores do microcontrolador presentes na luva. A saída esperada é o sinal correspondente (□) que o usuário realizou no intervalo de tempo.

São necessários vários algoritmos para obter a função de classificação que melhor se adapta aos dados de entrada. Para isto, são utilizados os algoritmos: “KNN”, “XGBoost” e “SVM”. Os dois primeiros métodos já foram explicados na seção anterior, na aplicação a única modificação necessária está relacionada aos dados de entrada, neste caso da luva sensorial. O último será introduzido a seguir.

Em adição aos algoritmos, para a elaboração da luva sensorial foi necessário a utilização de uma estrutura física. Os componentes escolhidos para realizar tal tarefa são explicitados nesta seção, ressaltando o modelo bem como o básico de seu funcionamento.

3.3.1 SVM

Segundo [28], “As Máquinas de Vetores de Suporte (SVM, do Inglês *Support Vector Machines*) constituem uma técnica de aprendizado que vem recebendo crescente atenção da comunidade de Aprendizado de Máquina”. Apesar da grande tendência das pesquisas em inteligência artificial (IA) usarem cada vez mais redes neurais, o SVM ainda é um algoritmo amplamente utilizado em diversas aplicações. Ele consegue apresentar bons resultados neste segmento, conforme comprovados nos experimentos realizados por [28]. Um lado negativo desta técnica se encontra no seu tempo de treinamento que pode ser muito mais custoso do que outros algoritmos citados neste estudo.

Criado por Vapnik em seu artigo original [16], o SVM tem como principal objetivo mapear vetores de suporte para criar a separação dos dados. Para a identificação de tais vetores, são aplicadas diversas transformações algébricas nos dados, além de elevar a dimensão do problema para realizar a busca em dimensão mais alta. Nesta nova dimensão, que não possui um acesso direto, os dados ficam mais esparsos, sendo possível localizar pontos que definem a margem entre uma classe e outra.

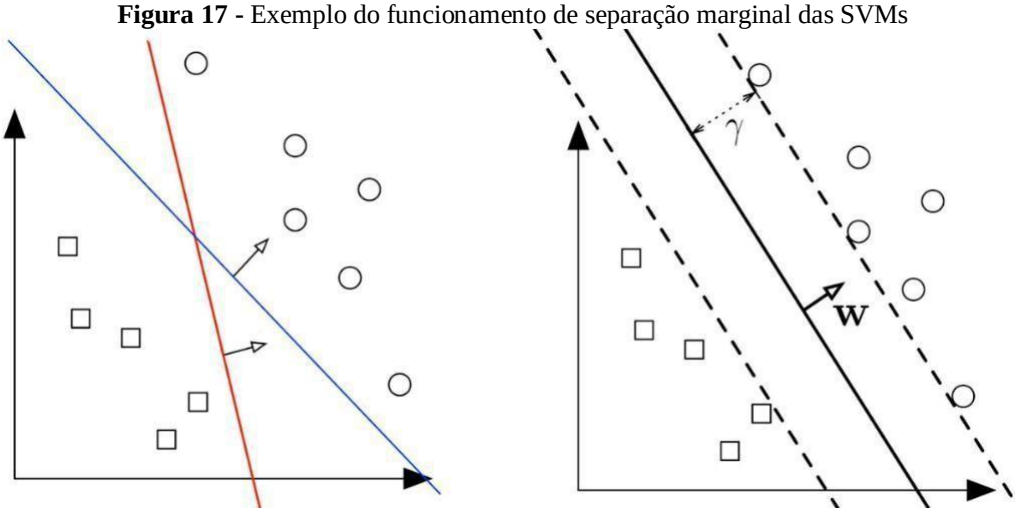
Essa margem pode ser formada de diversas formas distintas, sendo modelada por diferentes funções, também chamadas de “*kernels*”. Por exemplo, define-se que a função separadora seja uma reta ou até mesmo um polinômio de grau N, este será então o *kernel* do problema. A partir do mapeamento dos vetores de suporte, é construída a função que utiliza os

pontos e a margem. Então é realizada a separação dos dados na dimensão alta, maximizando o espaço entre as classes.

Na **Figura 17** é possível visualizar a forma em que este processo ocorre de maneira simplificada. É possível observar que a reta vermelha e a reta azul fazem uma separação dos dados de forma correta, de acordo com o problema exemplificado, a reta azul consegue fazer a separação de forma a maximizar a distância das suas margens entre os vetores de suporte.

Após a realização da separação, é feita a transformação inversa para se retornar a dimensão de origem do problema. Então, mesmo que seja utilizado uma função linear para fazer a separação em dimensão mais alta no SVM, ao se retornar, a função separadora pode possuir curvas. Para a classificação, este modelo trabalha como um sistema de pesos, assim como outros já descritos, e uma aplicação do vetor de atributos nele resulta na classificação imediata .

Figura 17 - Exemplo do funcionamento de separação marginal das SVMs



Fonte: MUBARIS (2017)

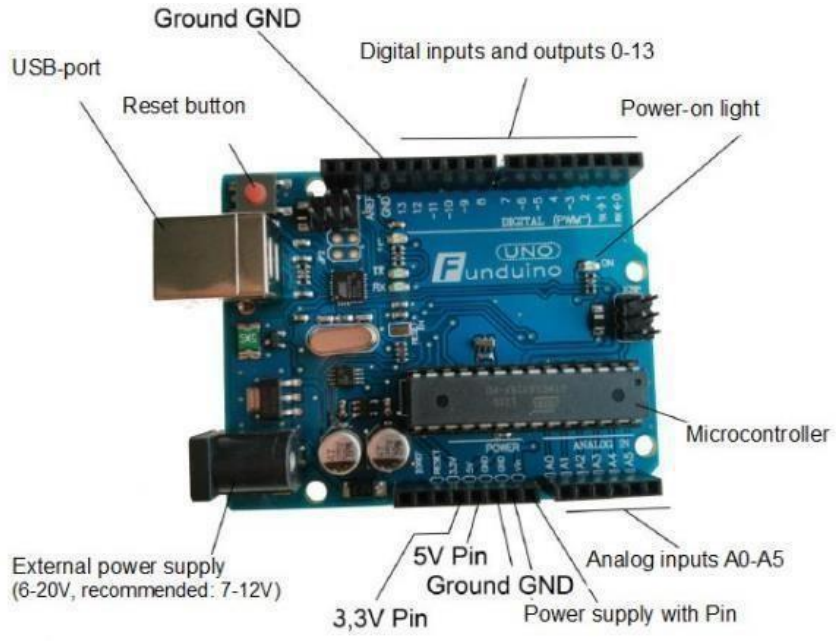
3.3.2 Arduino

O Arduino é um microcontrolador utilizado como núcleo de processamento, capaz de executar instruções computacionais, captar e transmitir dados com o auxílio de suas componentes. Um modelo similar a este, é a placa Funduino UNO [19] que é compatível com toda a tecnologia do Arduino UNO, e foi utilizado no projeto. Todos os detalhes do hardware da placa podem ser vistos na **Figura 18**.

Essa placa possui 6 entradas analógicas capazes de receber componentes extras, como sensores. Além disto, possui uma saída de alimentação de energia para o funcionamento dos mesmos. Uma visão mais detalhada dos recursos da placa é acessível pelo site [19], em que

também é possível encontrar um tutorial para aplicações simples [19]. Toda a programação desta placa pode ser feita utilizando a própria IDE do Arduino [7], o que torna simples a sua utilização.

Figura 18 - placa do funduino UNO



Fonte: FUNDUINO (2020)

3.3.3 Sensor Flexível

Este tipo de componente é responsável pela captação de dados através da mudança na resistência de seu circuito com a medição da sua dobradura (flexão) [3]. No modelo "*Adafruit long flex sensor ADA182*", esse sensor possui um tamanho de 112 mm, e sua resistência pode variar entre 10□Ω e 20□Ω. Este modelo de sensor também foi utilizado para a realização do projeto de controle PowerGlove [49], da Nintendo para *video games*. Na **Figura 19** é possível ver a aparência do sensor.

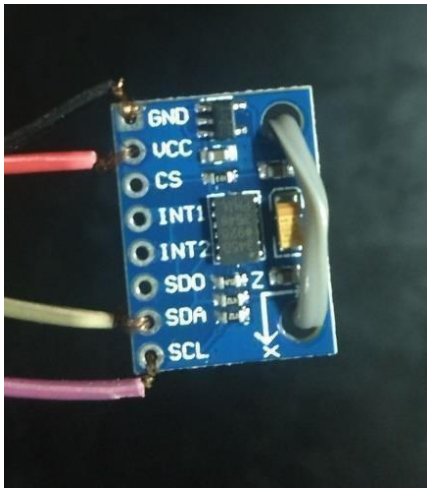
Figura 19 - Sensor flexível

Fonte: Reprodução do autor

3.3.4 Acelerômetro

Um acelerômetro é responsável pela medição da inclinação da superfície na qual ele se encontra [5]. No modelo "*ADXL345*" existe uma componente "*3-axis sensor*" que permite medir a variação entre a capacitância de placas paralelas energizadas em conjunto com seu módulo de "*sense eletronics*", como descrito no manual. Isto permite que ele consiga captar variações de angulação menores do que 1 grau que o torna muito preciso em relação aos dados coletados [5]. Na **Figura 20** é possível ver a aparência do sensor.

Figura 20 - Acelerômetro



Fonte: Reprodução do autor

4 IMPLEMENTAÇÃO

Para tornar possível a execução do projeto, foi necessário estudar especificamente LIBRAS para, posteriormente, buscar alternativas que contemplassem as suas características morfológicas. Além disto, encontrar uma solução definitiva e geral não era viável, tanto pelos recursos financeiros de criar uma arquitetura mais completa quanto pelo prazo do projeto.

Gerar dados específicos que contemplem um grande conjunto de termos da língua demanda muito tempo e mão de obra. Entretanto, para uma solução ideal, estas características são necessárias no aprendizado por máquina. Então, foi definido um escopo de testes que contemplasse um estudo de casos sobre o problema.

O projeto computacional foi dividido em dois grandes procedimentos: a identificação de expressões faciais e os sinais/movimentação com a mão. Além da criação de sistemas capazes de solucionar os problemas acima, também foi feito um código para conectar esses dois grandes desafios. Neste capítulo é apresentando o funcionamento e criação do STILL, uma ferramenta capaz de fazer a tradução de um conjunto de termos em LIBRAS para a língua portuguesa.

4.1 IDENTIFICAÇÃO DE REQUISITOS DA APLICAÇÃO

Para o melhor entendimento do cenário de atuação realizou-se um estudo mais específico da LIBRAS com o auxílio do curso na duração de 6 meses no departamento de Letras da Universidade Federal do Rio de Janeiro (UFRJ). Neste período, com a maior aproximação da comunidade com deficiência auditiva, foi possível perceber com mais clareza suas dificuldades de comunicação com pessoas sem conhecimento da língua de sinais. Também foi possível identificar limitações, necessidades e requisitos para o desenvolvimento do sistema de tradução.

Subsequentemente, deu-se início a fase de planejamento e definição do escopo do projeto. Nesta etapa, trabalhos compatíveis com a meta deste projeto foram analisados, de modo que fosse possível avaliar os pontos positivos e negativos em cada um daqueles para que tais aspectos fossem tratados aqui. Alguns testes foram realizados utilizando abordagens consideradas relevantes para o escopo deste projeto e, ao final dessa etapa, foi definida uma arquitetura que melhor resolvia o problema proposto.

Como LIBRAS não é composta apenas por sinais de mão, para aumentar a compatibilidade de um tradutor com a língua em questão, notou-se a necessidade de um sistema de reconhecimento facial em conjunto com um reconhecimento dos referidos gestos. Esse projeto tem como público alvo a população surda do Estado do Rio de Janeiro, então foi necessária a utilização da variação carioca desta língua.

Em busca de aplicabilidade em diferentes locais, a sua arquitetura foi desenvolvida visando futuras possibilidades do projeto ser completamente móvel e utilizável tanto em ambientes fechados como em abertos. Ainda nesta linha, para a ampliação do modelo, foi criada uma plataforma simples que fosse possível inserir novas palavras para aumentar o vocabulário entendido pelo sistema.

Para início da prototipação do sistema, com relação ao *hardware* utilizado, foram levantados e adquiridos todos os componentes necessários para o desenvolvimento do projeto. Um dos problemas encontrados estava relacionado à aquisição de sensores flexíveis que não eram encontrados nacionalmente e tiveram que ser importados. Para a programação do sistema, como o aprendizado por máquina é dependente de dados, buscou-se por *datasets* que melhor representassem as expressões faciais e a composição de sinais com as mãos de LIBRAS. Porém, com a não existência do último, foi necessário a sua criação com escopo reduzido.

Por fim, foram realizados experimentos com cada algoritmo, segundo a **Tabela 1,** para os sistemas de classificação do projeto. Seus resultados foram todos analisados e constatou-se qual seria o melhor modelo a ser utilizado no sistema atual. Posteriormente foi testado com sucesso o funcionamento do protótipo com luva e câmera desenvolvido com o modelo que obteve o melhor desempenho.

Tabela 1 - Modelos utilizados em cada sistema

Sistema	**Modelos**
Identificação facial	*KNN* *XGBoost* *Random Forest* *CNN* *Regressão Logística*
Luva sensorial	*KNN* *XGBoost* *SVM*

Fonte: Reprodução do autor

Resumidamente, o sistema completo é composto por:

1. Um *dataset* de expressões faciais.
2. Um *dataset* de sinais e movimentos de LIBRAS.
3. Uma arquitetura física de luva sensorial para captação e transmissão de dados.
4. Uma câmera frontal para a captura da face.
5. Um sistema de cadastro para ampliação do *dataset* de sinais e movimentos de LIBRAS.
6. Um sistema de identificação facial.
7. Um sistema de aprendizado e classificação de expressões faciais.
8. Um sistema de aprendizado e classificação de sinais e movimentos de LIBRAS.
9. Um sistema de conexão entre os modelos obtidos e execução da aplicação.

4.2 ESCOPO DO PROJETO

Apesar de toda capacidade, existem diversos fatores limitantes com relação aos recursos. Logo, foi necessário definir um escopo para que uma versão protótipo fosse concluída. Dentre as limitações, pode-se citar equipamentos, tempo do projeto e, principalmente, a quantidade de dados disponíveis para a realização dos modelos.

No que se refere ao sistema de identificação de expressões faciais, não existe um conjunto de dados específico de LIBRAS e a produção desse seria inviável no prazo estipulado. Foi necessária a utilização de um *dataset* genérico de expressões [25], com 7 diferentes classes e uma qualidade inferior ao utilizado em estudos mais recentes [38]. Este não contempla de forma integral todas as variações que observamos em LIBRAS, porém era o mais próximo possível no momento.

Essa decisão foi tomada pela baixa capacidade de processamento da máquina utilizada que não conseguia tratar e classificar de forma eficiente as imagens com maior resolução. Essas necessitavam de um pré-processamento mais elaborado que dificultava ainda mais a sua utilização com os recursos disponíveis.

Com relação ao sistema de detecção de sinais e movimentos com a mão, seria necessária a confecção de duas luvas sensoriais para mapear todos os movimentos e configurações de todas as palavras existentes em LIBRAS. Todavia, devido ao difícil acesso às componentes e pelo custo inicial elevado dos sensores flexíveis [4] (aproximadamente 52 dólares a unidade, devido às altas taxas de importação), só foi possível produzir uma luva, o que restringe inicialmente o projeto a sinais que necessitam apenas de uma mão para serem diferenciados.

Ainda neste sistema, era necessária a fabricação de dados que contemplassem a língua e que não eram disponíveis ou não existiam anteriormente. Foi definida a construção de um *dataset* com um escopo reduzido de 8 diferentes classes, uma vez que gerar um número maior demandava-se mais tempo. Esse foi produzido com a ajuda de voluntários que com a supervisão do autor, produziram, com o auxílio do sistema de cadastro, o conjunto de palavras. Porém, com esse mecanismo, é possível uma maior flexibilização do modelo, sendo mais fácil adicionar mais classes e palavras ao vocabulário do projeto futuramente.

4.3 RECONHECIMENTO FACIAL EM IMAGEM

Para a realização do reconhecimento de expressão facial, dois desafios diferentes foram encontrados. O primeiro foi: como que a partir de uma imagem capturada por uma câmera seria possível identificar todas as faces presentes na imagem ou no vídeo? E o segundo: depois que as identificar, como será possível classificá-las de acordo com o tipo de expressão facial presente? A primeira pergunta é respondida neste capítulo e tem como resultado o dado de entrada para responder a segunda pergunta no capitulo subsequente. Na **Figura 21**, há uma visão global do sistema de identificação facial genérico.

Figura 21 - Fluxograma do sistema de identificação facial

Fonte: Reprodução do autor

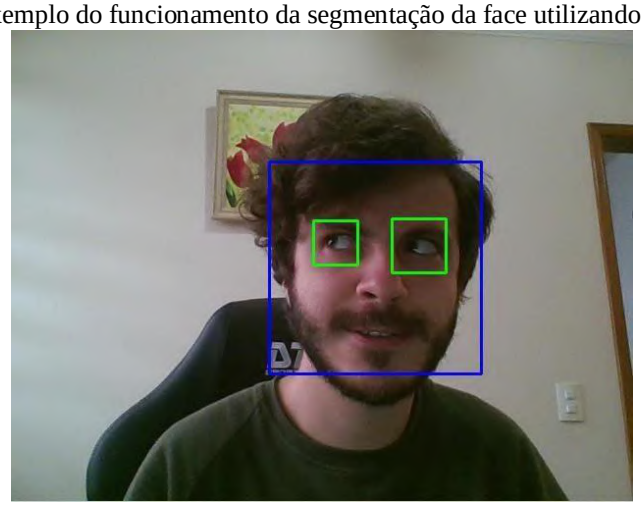
Como primeira hipótese de solução do problema, foi testada a segmentação da face do usuário em diferentes partes (olhos, boca e utilização inteira da face). Ao analisar cada parte separadamente, esperava-se um resultado mais preciso a partir da composição destes. Utilizou-se a câmera do computador para a captura do vídeo e aplicou-se um algoritmo de identificação facial escrito em *Python*. Foi escolhido o uso do *Haar Cascade* [47] pela sua simplicidade e alta taxa de acerto para a realização desta tarefa. Com isso, foi possível obter a identificação da face, boca e olhos separadamente em uma imagem.

Na **Figura 22** observa-se o experimento da execução deste algoritmo, identificando apenas a face e os olhos de uma imagem com um rosto. O fato de nestes primeiros testes não terem reconhecido a boca, demonstrou que o algoritmo não estava tendo bons resultados nesta

segmentação. Uma das hipóteses levantadas para tal observação foi que o bigode pudesse interferir.

Por não ser genérico o suficiente para ser utilizado por pessoas com muita diversidade, foi necessária outra abordagem. No caso do Brasil, pela grande diferença de fisionomias e miscigenação da população, este acontecimento é muito recorrente. *Datasets* específicos compostos de faces prototípicas [31] não puderam ser utilizados, pois não estão disponíveis com rótulos de classes a serem classificadas e não são referentes a LIBRAS.

Figura 22 - Exemplo do funcionamento da segmentação da face utilizando o *haar cascade*



Fonte: Reprodução do autor

Como visto anteriormente, é preciso utilizar um *dataset* para o treinamento do modelo do *Haar Cascade* ou empregar um já pronto e treinado. No experimento, foi utilizado um modelo pronto e disponível em [40]. Elementos faciais usados nos primeiros experimentos como os olhos, boca e nariz também são encontrados neste mesmo repositório.

Outro problema encontrado em função da abordagem de divisão das partes do rosto foi a necessidade de utilizar um algoritmo de aprendizado de máquina após a aplicação do *Haar Cascade.* Para classificar a expressão presente no recorte produzido pelo algoritmo é necessário um modelo das diferentes partes, como observa-se na **Figura 22,** destacado em verde.

Para criar este segundo modelo, seria essencial um *dataset* preparado e dividido por rótulos de expressões faciais dos elementos contidos no rosto (olhos, boca, nariz), porém este conjunto de dados em específico é difícil de encontrar, ou não são divulgados publicamente. Logo, mesmo ao fazer a separação dos elementos faciais, não seria possível fazer a classificação dos mesmos.

Portanto, as soluções propostas para a resolução deste problema foi a utilização apenas da face como um todo (representada pelo retângulo azul da **Figura 22**). Desta forma, a análise facial tornar-se-ia menos específica, e, em contrapartida, mais *datasets* de expressões faciais já rotulados de cortes faciais inteiros como este estariam disponíveis. Logo, obtendo-se este conjunto de dados, seria possível criar um modelo a partir de diferentes algoritmos, para avaliar qual possuiria o melhor desempenho na identificação das expressões faciais.

4.4 RECONHECIMENTO DE EXPRESSÃO FACIAL

A partir das imagens obtidas na extração de faces, é necessária a construção de um modelo para o reconhecimento das expressões faciais. Para tal, foi imprescindível a seleção de um conjunto de dados, o treinamento do modelo de predição e, finalmente, a execução do algoritmo em tempo real.

4.4.1 Conjunto de dados

Para realizar a criação dos modelos, foi feito um estudo dentre diferentes *datasets* disponíveis e rotulados pelas expressões faciais [29] [21] [25] [48] [38]. Foram levados em consideração a qualidade e relevância destes para pesquisas neste ramo, a quantidade de *pixels* presente em cada imagem e também o nicho experimental (diferentes tipos de pele, nacionalidade, etc.).

Em relação à quantidade de *pixels*, quanto maior a quantidade, melhor a resolução da imagem, o que pode resultar em um modelo com mais taxa de acerto. Em contrapartida, devido às limitações do *hardware* utilizado, ao invés de se optar por faces com muita resolução, foi selecionado um conjunto com uma resolução mais baixa. Essas imagens também possuem relevância na pesquisa sobre esta área e são mais genéricas em relação ao seu nicho experimental [25].

Por exemplo, um dos *datasets* pesquisados, o JAFFE [29] é composto apenas por expressões faciais de mulheres japonesas, o que não seria interessante neste caso, visto que é muito específico. Foi também observado um conjunto de dados artificial, com caras animadas, o FERG-DB [21], mas que parece muito distante da realidade humana pela construção das feições não realistas de pessoas.

Dentre os *datasets*, foi escolhida a utilização do FER, disponibilizada pelo site *kaggle* [25], rotuladas pelos seguintes 7 tipos de expressões faciais distintas: “irritado”, “nojo”, “medo”, “feliz”, “neutro”, “triste” e “surpreso”. Além disso, possui 28709 exemplos de treino, 3589 exemplos para teste e suas imagens contém uma resolução de 48x48 com *pixels* em tons de cinza (valores entre 0 e 255).

Com a separação das expressões desta forma, existe um mínimo de chance de obter certa semelhança com algumas expressões presentes em LIBRAS. Vale ressaltar que este conjunto de dados não é específico para a identificação de expressões feitas nesta língua, porém, não existe um disponível e que faça esta divisão. Gerar um *dataset* específico para este desafio seria muito custoso, além de necessitar de muito mais recurso. Na **Figura 23** existem alguns exemplos de imagens que foram utilizadas.

Figura 23 - Exemplo de imagens presentes no conjunto de dados FER

Fonte: KAGGLE (2010)

4.4.2 Treinamento do modelo de aprendizado de máquina

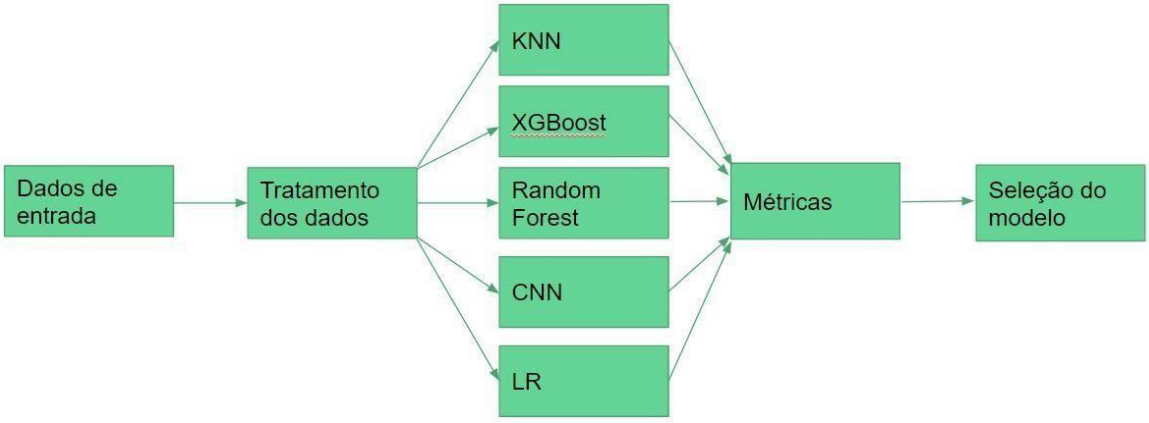
Para a criação de uma aplicação de tempo real para a classificação de expressão facial presente na imagem, foi necessário um modelo com base nas imagens do *dataset* escolhido. Foi proposta uma sequência de experimentos que utiliza diferentes algoritmos de aprendizado de máquina e também executa o chamado “ajuste fino” de parâmetros para cada um deles. Esses são capazes de ajustar os hiperparâmetros para obter melhores resultados em relação a acurácia dos rótulos das imagens. Em seguida, selecionou-se o modelo que obteve o melhor desempenho dentre os testados.

Foram escolhidos cinco diferentes algoritmos que compuseram o conjunto de hipóteses, dentre eles: o *KNN*, o *XGBoost*, o *Random Forest*, o *CNN* e a Regressão Logística. Esses

algoritmos foram escolhidos pela sua relevância e compatibilidade com o problema a ser resolvido na área de classificação de padrões [22] [51] [8]. Além disso, buscou-se utilizar os algoritmos de árvore de decisão [13] [11] a fim de alcançar mais possibilidades de melhores resultados, conforme descrito no **capítulo 3.1.2,** que pudessem ser comparadas à estudos anteriores.

Foi intencional manter o máximo de pluralidade em relação às suas formas de separação de dados e classificação para abranger o maior número de possibilidades de resultados relevantes, como por exemplo, o *XGBoost* e o *Random Forest*, que são separadores com estrutura em árvores de *ensembles* diferentes. Em conjunto, são feitos experimentos com o algoritmo CNN que utiliza redes neurais. Na **Figura 24** é possível observar o fluxo proposto de seleção dos classificadores.

Figura 24 - Fluxo do processo de classificação de expressões faciais utilizados neste experimento



Fonte: Reprodução do autor

Na etapa de seleção de dados, a cada execução do algoritmo é feita a separação de forma dinâmica e aleatória do *dataset*. Esses são separados entre o conjunto de treino e o conjunto de teste, na proporção 0,9 e 0,1 respectivamente. Como existem muitas amostras neste conjunto, essa divisão pôde ser feita sem a preocupação de faltar alguma classe em uma das partes.

Além disso, o conjunto de treino foi separado em 10 partes utilizando *K-fold* para a criação do conjunto de validação. Como os dados são padronizados em relação a sua resolução e todos estão em escala de cinza, não foi necessário um pré-processamento extra.

O “ajuste fino” dos parâmetros de cada um dos algoritmos listados acima, com exceção da CNN, foi feito com o método de busca em grade (*GridSearch*) da biblioteca do *Sklearn* [37] do *Python*. Esta função testa exaustivamente cada combinação de hiperparâmetros que é recebida, retornando no final apenas o modelo que apresentou o melhor resultado, com base na

acurácia do conjunto de validação. Cada um desses algoritmos possui um conjunto diferente de hiperparâmetros. Abaixo são listados quais deles foram explorados:

- KNN: Quantidade de vizinhos “k”.
- XGBoost: Uso padrão da biblioteca do *Sklearn* (que já faz as otimizações internamente).
- Random Forest: Estimador (número de arvores na floresta).
- LR: Parâmetro c (força do regularizador).

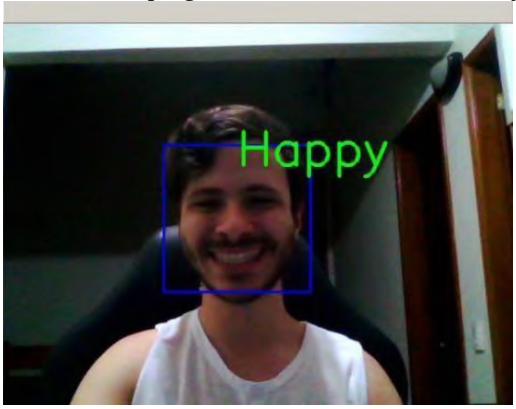
A CNN foi formada por 6 camadas de convolução, com respectivamente 32, 64, 128, 128, 7, 7 neurônios em cada camada utilizando função de ativação “*relu*” ($\square = \max(0, \square)$) e uma camada de normalização com ativação em “*softmax*” (função *sigmoid*). Esta estrutura foi escolhida pelo modelo dos experimentos de Priyanka Dwivedi [39] que lidavam a mesma base de dados. Ele foi empregado para uma comparação direta de resultados entre os outros modelos propostos em [39] e nesta pesquisa. Esses são apresentados no capítulo 5.1 referente aos resultados.

4.4.3 Identificação das expressões faciais

Na execução do melhor modelo obtido é exibida na tela, em tempo real, a identificação das faces presentes na *webcam* e sua respectiva expressão facial classificada pelo algoritmo. Para cada *frame* obtido pela câmera do computador, é feita a conversão de cor para tons de cinza, que são processados pelo algoritmo de identificação facial do *Haar Cascade*.

Para cada face identificada, é realizada uma redimensionalização da imagem para o tamanho 48 x 48, como nos dados de treino, que é enviada para a classificação pelo modelo. No fim, é escrita na tela sua expressão facial, assim como no exemplo a seguir na **Figura 25**.

Figura 25 - Exemplo de resultado do programa executável de classificação de expressões faciais



Fonte: Reprodução do autor

4.5 RECONHECIMENTO DE SINAIS E MOVIMENTAÇÃO DA MÃO

Na elaboração da resolução deste problema, foram utilizados alguns projetos como inspiração, presentes no capítulo de trabalhos relacionados [27] [46] [30] [26] [2] [9] [36] [42] [44]. Dentre as soluções propostas pelos autores, essas podem ser divididas principalmente em duas aproximações diferentes: a primeira utiliza apenas uma câmera para a captação dos movimentos da mão e para o mapeamento espacial dos pontos de interesse para a formação dos sinais. A segunda alternativa usa uma luva sensorial para fazer o mapeamento da posição da mão, dos dedos e de sua movimentação.

Apesar dos estudos, a abordagem pela utilização apenas da câmera para a captação de movimentos possui algumas desvantagens: seria necessário um ambiente controlado para fazer a captura dos vídeos, e isso não pode ser considerado como usual no dia-a-dia de um usuário. Outra dificuldade consiste na necessidade de maior poder computacional para o processamento dessas imagens. Finalmente, seria necessário um conjunto de dados específico de LIBRAS para imagens de configurações de mãos e de movimento. Entretanto, não foi encontrado publicamente no momento em que este trabalho foi realizado.

Tendo em vista o futuro do projeto e a praticidade do usuário, foi escolhido o método de reconhecimento com uma luva sensorial. Desta forma, com a evolução do protótipo, o usuário poderá facilmente utilizar este acessório em qualquer lugar, até mesmo andar na rua com uma luva tradutora, assim como já é feito com alguns dispositivos de amplificação de áudio para pessoas com grandes dificuldades auditivas [17]. Nesse sentido, parte do *software* também pode ser incorporado nos *smartphones* dos usuários.

Nos subcapítulos a seguir é discutida a forma como foi elaborada a arquitetura do protótipo, a solução encontrada para gerar dados para o treinamento do modelo que classifica os símbolos feitos pela luva, a seleção e o treinamento dos modelos. Finalmente, é realizada em tempo real a tradução feita pela luva em conjunto com o modelo treinado.

4.5.1 Arquitetura da luva sensorial

Dentre os modelos relacionados já citados, observa-se uma gama de alternativas para a construção deste equipamento. Com objetivo de possuir a melhor representação possível neste protótipo, foi escolhida a utilização de sensores flexíveis [3] para o mapeamento dos dedos em conjunto com um acelerômetro [5] para o referenciamento espacial. Essa composição de

sensores nos permite ter flexibilidade para criar um modelo que atenda de forma inicial a demanda da língua brasileira de sinais.

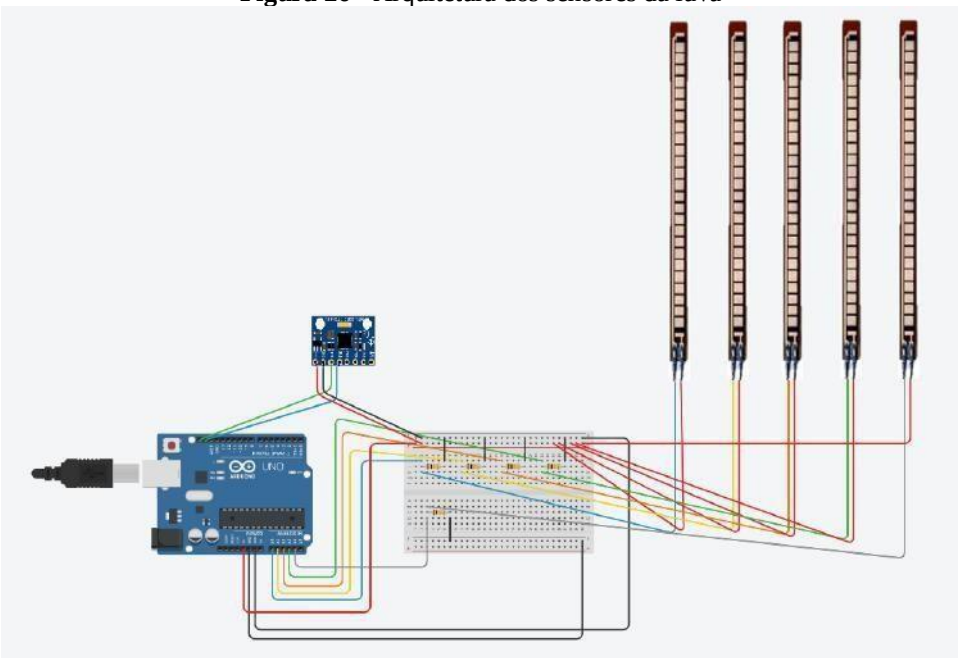
Para uma abordagem mais completa de todos os requisitos necessários, deve-se levar em consideração os pontos de contato, porém o microcontrolador não seria capaz de possuir mais sensores vinculados em sua estrutura básica, devido à limitação de entradas analógicas do *hardware*.

Primeiramente, foi feita uma reprodução caseira de um sensor flexível, porém depois de diversas falhas, optou-se pela compra do modelo "*Adafruit long flex sensor ADA182*". Em relação à eficiência e baixo custo, escolheu-se o acelerômetro de modelo "*ADXL345*", capaz de medir variações de inclinação de até um grau [5].

Esses componentes foram ligados a um Funduino UNO para fazer a programação necessária para transformar os sinais elétricos provindos das componentes alimentadas pelo controlador em dados que são utilizados pelo sistema. As saídas analógicas do microcontrolador servem para receber os dados dos sensores, obtendo um modelo de arquitetura semelhante ao demonstrado na **Figura 26**. Nota-se que todas as resistências na imagem são de 10 kΩ para o funcionamento adequado dos sensores flexíveis.

Dada esta arquitetura de sensores, foi produzida uma luva física, funcional, para que esses fossem dispostos da maneira correta para a leitura. Os sensores flexíveis foram postos sobre os dedos da luva base e o módulo do acelerômetro foi disposto perto do pulso.

Figura 26 - Arquitetura dos sensores da luva

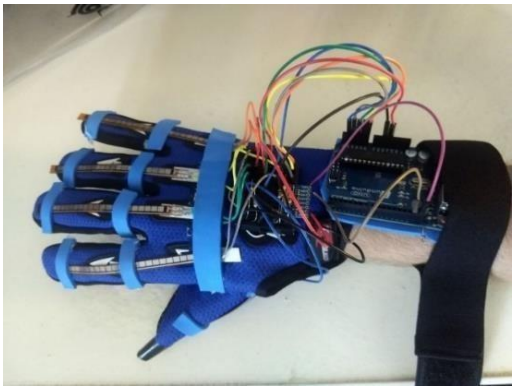


Fonte: Reprodução do autor

Como a finalidade era montar um modelo protótipo que pudesse ser facilmente alterado para testes, as componentes não foram fixadas de forma “permanente” (o que seria difícil para remodelar caso houvesse algum problema). Foram utilizados, em conjunto com a luva base, emborrachados de EVA para fazer a fixação de todos os sensores, sendo facilmente manipulável, caso precise alguma adaptação.

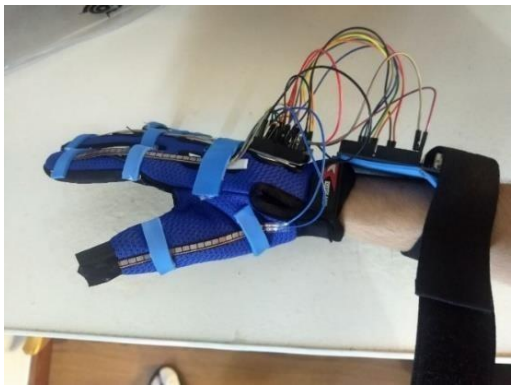
O protótipo inicial não utiliza uma comunicação sem fio (*wifi* ou *bluetooth*), e todos os dados da referida luva são transmitidos serialmente para o servidor pela saída USB (*Universal Serial Bus*) do Funduino. Em projetos futuros é discutido a possibilidade da utilização completamente sem fio. Nas **Figuras 27 e 28** é possível ver o modelo final do protótipo em uma visão frontal e lateral.

Figura 27 - Visão frontal do protótipo final de luva sensorial



Fonte: Reprodução do autor

Figura 28 - Visão lateral do protótipo final de luva sensorial



Fonte: Reprodução do autor

4.5.2 Conjunto de dados

Tendo em vista que a luva foi produzida especificamente para esta pesquisa, mesmo que inspirada em outros trabalhos, a estrutura da luva é ligeiramente diferente. Logo, não seria possível reaproveitar os dados provindos de outros projetos (inclusive pela sua indisponibilidade). Isso se dá pelo fato da maior parte dos trabalhos não ser referente a LIBRAS, mas sim a ASL, sendo essas bem distintas entre si.

Conjuntamente, foi pensada em outra maneira de mapear os dados, levando-se em consideração diferentes instantes do movimento, para um maior aproveitamento dos componentes, agregando mais valor às predições. Logo, foi necessária a criação de um novo conjunto de dados específico para este problema. Devido a esse fator, a quantidade de dados é bastante limitada, já que demandava muito tempo e variedade de pessoas para se construir

um *dataset* robusto. Criou-se, dessa forma, um sistema para fazer o cadastro de novos dados que também possibilitasse a ampliação do vocabulário futuramente.

Foi definido que o modelo busca predizer movimentos completos de uma palavra em LIBRAS. Logo, por mais que um sinal possa ser composto por uma quantidade variada de movimentos intermediários, não é lidado individualmente com estes pequenos estados, apenas com movimentações completas.

Ainda que fosse interessante gerar experimentos com o mapeamento de movimentos intermediários, seria necessário muito mais tempo de produção. Essa abordagem poderia gerar um ganho no nível de precisão do modelo para palavras mais complexas. Um exemplo disso pode ser observado na **figura 29**, em que um movimento é composto por 2 estados intermediários que estão enumerados na ordem de movimentação.

Figura 29 - Representação da palavra "marido" em LIBRAS

Fonte: CLAC (2019)

A fim de elaborar um bom mapeamento dos dados, foram experimentadas diferentes conjecturas com os sensores. Inicialmente, para a criação dos atributos, foi avaliado utilizar apenas uma iteração do laço de repetição da transmissão de dados dos sensores para o programa. Porém, isso faria com que o modelo tratasse os dados "estaticamente" ou em um momento específico dentro de toda a composição do movimento dos gestos.

Para obter uma percepção de movimentação e sua evolução conforme o tempo, ao invés de se capturar apenas um estado da luva, foi utilizado dois tempos: um próximo do início do movimento, e um próximo do final do movimento. Desta forma, conforme o vocabulário da luva aumenta, previne-se o risco de classificar igualmente as palavras que tivessem o início semelhante, porém o final do movimento é distinto.

Uma observação pertinente é que a quantidade de capturas de um movimento ao longo do tempo poderia ser maior, entretanto, quanto mais capturas forem feitas, maior será a quantidade de atributos do modelo (uma vez que cresce linearmente de acordo com a quantidade de capturas). A escolha de apenas duas capturas é justificada pela maior simplicidade do modelo e pela menor quantidade de ruídos nos dados. Dessa forma a classificação é mais rápida e pode ser utilizada em tempo real por um usuário no contexto atual. Conforme eram transmitidos mais capturas em um intervalo de tempo, a quantidade de dados defeituosos aumentava muito, e isso prejudicava o desempenho dos algoritmos de aprendizado de máquina.

Para esclarecer que tipos de atributos estão presentes em cada uma destas capturas em relação ao tempo, na **Tabela 2** há uma correlação entre a variável capturada pelo microcontrolador e a origem (sensor) proveniente de tal dado.

Os dados dos sensores flexíveis foram todos calibrados e normalizados entre valores de 0 a 100 inteiros, com uma pequena margem de segurança, em que 0 representa o sensor sem qualquer tipo de flexão e 100 representa o sensor completamente flexionado. Esta margem de segurança foi feita para não existir o caso de valores que não estejam entre 0 e 100. Em experimentos práticos, os dedos esticados apresentaram valores em torno de 30 e dobrados em torno de 80.

Para os dados do acelerômetro, obtêm-se um ponto flutuante na unidade m/s², em que os valores ficaram entre +/- 10 m/s², baseados na sua disposição espacial em relação ao solo. No modelo, foram utilizados dois instantes para a classificação, o *dataset* final possui um total de 16 atributos por amostra observada (x) mais a classe referente ao movimento em questão (y).

Tabela 2 - Tabela de atributos da luva sensorial

variável (atributo)	**origem**
flex_0	Sensor flexível do dedo mínimo
flex_1	Sensor flexível do dedo anelar
flex_2	Sensor flexível do dedo médio
flex_3	Sensor flexível do dedo indicador
flex_4	Sensor flexível do polegar
accel_x	Componente x do acelerômetro
accel_y	Componente y do acelerômetro
accel_z	Componente z do acelerômetro

Fonte: Reprodução do autor

Em relação à captura dos dados para o *dataset*, foi elaborado um programa onde é atribuído um rótulo de palavra ao movimento feito pela luva. Isso foi feito com apoio de uma interface simples em terminal Linux. Foi estipulado que o usuário faça o movimento dentro de um intervalo de tempo de 1 segundo quando sinalizado o início da iteração do programa. Após a realização do movimento é necessário indicar a palavra referente ao movimento efetuado.

Este procedimento foi realizado várias vezes pelo usuário, com a finalidade de adicionar diferentes palavras e repetições de exemplos da mesma. Ao final, todos esses dados foram salvos em um documento utilizado pelo programa de criação do modelo de classificação.

Para fazer o povoamento do conjunto de dados, o projeto contou com a ajuda de cinco voluntários, que foram ensinados a fazer os sinais de forma correta. Cada pessoa realizou um total de 10 repetições por palavra diferente, para ser adquirida uma quantidade suficiente de dados. Foram estipulados oito termos distintos para a realização dos testes, logo, foram criadas oito classes para este modelo.

Dentre os sinais, foram representados os termos: “oi”, “meu”, “nome”, “Rennan”, “aluno”, “computação”, “ajuda” e “neutro”. A classe neutra foi utilizada com objetivo de criar um estado em que a luva estivesse em repouso, sendo um mecanismo feito para a luva não reproduzir sons sem necessidade. No total, foram obtidas 319 amostras de todas as 8 classes em conjunto. Esse número se deu pelo fato de ter dados perdidos completamente por conta de ruídos e precisaram ser eliminados da quantidade inicialmente estipulada.

Esses ruídos eram causados pela transmissão entre o Funduino e o servidor que estava executando o programa do cadastramento em *Python*. Os dados foram transmitidos de forma bruta e um problema na limpeza do *buffer* de recepção causou a má formação dos dados enviados. A separação dos dados pelo programa em *Python* (alocação do *buffer* em variáveis de ponto flutuante) é altamente sensível a qualquer ruído de *bit* extra ou faltante ou invertido, resultando em variáveis com dados ruidosos.

Para avaliar o impacto dessa observação, foram feitos dois experimentos independentes. Em um deles foi utilizando os dados com ruídos, assim como eram recebidos normalmente. No segundo, foi feita uma limpeza no arquivo removendo todos os exemplos que possuíam algum ruído.

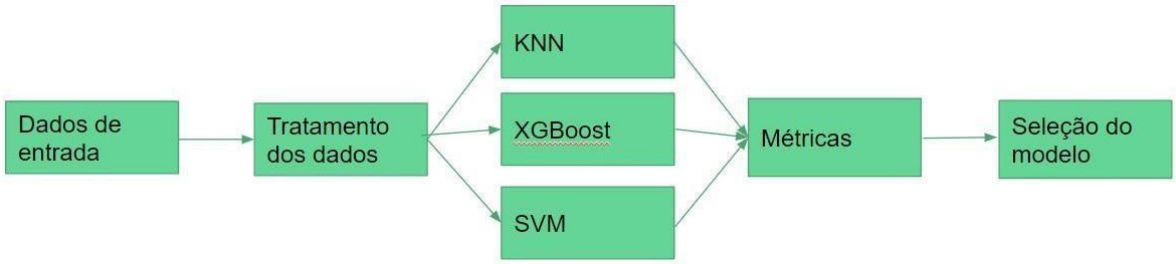
4.5.3 Treinamento do modelo de aprendizado de máquina

Foi necessário criar um modelo com base nos dados obtidos, para então executar uma aplicação que, em tempo real, classificasse os movimentos executados pelo usuário que utiliza a luva. Foi feita uma sequência de testes, com diferentes algoritmos de aprendizado de máquina, e, em seguida, executado o "ajuste fino" de parâmetros para cada um deles. Estes foram capazes de ajustar os hiperparâmetros de forma a obter os melhores resultados em relação à acurácia dos rótulos dos movimentos. Posteriormente, foi selecionado o modelo que tivesse o melhor desempenho dentre os testados.

Foram escolhidos três diferentes algoritmos que compuseram o conjunto de hipóteses. Dentre eles: o *KNN*, o *XGBoost* e o *SVM*. Esses algoritmos foram definidos pela sua relevância na área de classificação de padrões [22] [13] [16], além de se buscar por novas alternativas como descrito no **capítulo 4.4.2**.

Foi intencional manter o máximo de pluralidade em relação ao seu funcionamento para avaliar diversos contextos, de modo que seja possível obter o que melhor resolve o problema desta classificação. Na **Figura 30** é possível observar o fluxo de seleção dos classificadores.

Figura 30 - Fluxograma de seleção do modelo classificador



Fonte: Reprodução do autor

Para se separar as amostras necessárias para a criação e validação do modelo, foi utilizado o mesmo procedimento, descrito no **capítulo 4.4.2** de treinamento do modelo de classificação de expressões faciais. Foi feito desta forma, pois este conjunto de dados não possui uma característica que necessita de um tratamento especial. Suas classes são balanceadas, seus dados são tabulares e não houve problema na separação dos dados, apesar de possuir menos amostras.

A realização do "ajuste fino" de parâmetros de cada um desses algoritmos listados acima foi feita com o método de busca em grade (*GridSearch*) da biblioteca do *Sklearn* [37]

do *Python*. Essa função testa exaustivamente cada combinação de hiperparâmetros que é recebida, retornando no final apenas o modelo que apresentou o melhor resultado com base na acurácia do conjunto de validação. Cada um desses algoritmos possui um conjunto diferente de hiperparâmetros, estando abaixo os que foram explorados:

- *KNN*: Quantidade de vizinhos "k".
- *XGBoost*: Uso *default* da biblioteca do *Sklearn* (que faz as otimizações internamente).
- *SVM*: Parâmetro C, de -5 até 15. Parâmetro G, de 3 até -15, com passos de 2 em 2. *Kernels* dos tipos, "*rbf*", "*poly*" e "*sigmoid*". Função de decisão "*ovo*" e "*ovr*".

Os resultados e métricas de todos esses modelos são apresentados no **capítulo 5.2**.

4.5.4 Identificação dos sinais gestuais

Foi criada uma aplicação em *Python* para a execução do melhor modelo de classificação encontrado. Com o mesmo código de obtenção dos dados do Funduino, a luva continua a enviar os dados para o sistema. Porém, ao invés de adicionar ao *dataset* as palavras salvas, o mesmo lê, continuamente, todos os dados que são alocados em um vetor com estrutura igual ao conjunto de treino e enviados para o classificador.

O resultado obtido pelo classificador é analisado de forma que, se ele tiver uma probabilidade inferior a 70% de pertencer a uma classe ou se for obtido o rótulo "neutro", o programa não retorna a palavra falada. Caso contrário, o sistema emite uma sintetização do som do gesto.

Esta probabilidade foi escolhida com base nos testes realizados pelos experimentos. Atualmente, para se reproduzir o som referente à palavra, é utilizado o comando de terminal Linux "espeak", usando uma chamada de sistema no código *Python*. Essa solução foi escolhida, uma vez que, desta maneira, não seria necessário o acesso à internet para a execução desta aplicação.

5 RESULTADOS

Após a observação do funcionamento geral do projeto, este capítulo tem como objetivo apresentar a performance de cada uma das partes, com comentários e discussões a partir dos fatos examinados. Em relação ao treinamento dos modelos vistos nos capítulos anteriores, são obtidas as métricas de acurácia e tempo consumido de cada um dos classificadores. Em relação à aplicação final, é demonstrado com mais detalhes o resultado com exemplos práticos.

5.1 RECONHECIMENTO DE EXPRESSÕES FACIAIS

Com base nos testes propostos no **capítulo 4.4.2** referentes à implementação, foi escolhido o modelo que apresentasse o melhor desempenho. Na **tabela 3** são observados os resultados provenientes da estrutura de otimização e "ajuste fino" de hiperparâmetros realizado em relação aos dados. Os tempos da CNN não são mostrados, assim como a acurácia máxima, pois esse modelo foi treinado pelos estudos de Priyanka [39], utilizados com finalidade de comparação.

Tabela 3 - Tabela de resultados do modelo de reconhecimento facial

modelo	Acurácia média	Tempo médio (hh:mm:ss)	Acurácia máxima
LR	0,366	00:10:24.2	0,382
KNN	0,388	01:42:19	0,400
RF	**0,401**	**00:01:27.6**	**0,412**
XGBoost	0,397	00:05:35.6	0,410
CNN	0,319	-	-

Fonte: Reprodução do autor

Como visto nos resultados, o classificador *Random Forest* obteve a melhor performance, com base na acurácia dentre os modelos treinados no conjunto de teste, o que foi, de certa forma, surpreendente, visto que era esperado que as redes neurais, muito conhecidas para a realização deste tipo de tarefa ou até mesmo o *XGBoost*, obtivessem resultados superiores.

No caso da *CNN*, essa era de fato bem simples, como descrita no capítulo de implementação. Mas devido à complexidade das imagens utilizadas, esta deveria obter resultados mais relevantes dos que apresentados acima. Para este tipo de problema seria necessário buscar uma estrutura diferente de redes neurais que se adaptasse melhor aos dados.

Em relação ao resultado do *XGBoost*, notou-se que ele ficou muito próximo do obtido pelo *Random Forest*. É interessante observar que ambos os métodos são classificadores baseados em árvore de decisão, porém, o primeiro se baseia em técnicas de *boosting*, enquanto o segundo se baseia em técnicas de *bagging*. O resultado foi inesperado, pois o *XGBoost* apresenta várias técnicas de regularização intrínsecas no método que aumentam a capacidade de generalização, todavia foi observado o contrário.

Apesar de tudo, verifica-se que os resultados nesta etapa de reconhecimento de expressões faciais ainda não são bons. Para tornar este método mais sólido, seria necessária uma grande melhoria nesta etapa do processo. Vale ressaltar que existem sete classes possíveis para a classificação, logo, este algoritmo é relativamente melhor que uma escolha aleatória de uma classe que seria equivalente a uma acurácia de 14.2%.

Na **Figura 31** é possível ver exemplos dos resultados obtidos na execução em tempo real do algoritmo de classificação de expressões faciais com o modelo que obteve o melhor resultado, o *random forest*. Na primeira e terceira foto observam-se exemplos de classificação correta pela função. Na segunda foto, verifica-se um caso em que o resultado apresentado não corresponde à classe esperada (“angry”).

Na aplicação final, foi avaliado que a classificação das expressões faciais ainda apresentava resultados abaixo do esperado. Em testes práticos, foi apreciado que classes como “*surprise*”, “*angry*” e “*disgust*” foram raramente ou nunca exibidas de fato. Isto demonstra que o modelo não está retratando o ambiente real de forma similar aos dados utilizados para treino.

Figura 31 - Fotos classificadas pelo algoritmo de reconhecimento de expressões

Fonte: Reprodução do autor

5.2 RECONHECIMENTO DE SINAIS DA LUVA

Neste capítulo são apresentados os resultados provenientes do **capítulo 4.5.3** de implementação dos modelos da luva. Com base nos testes propostos, foi escolhido o modelo com melhor performance dentre os testados. Na **tabela 4** contém os resultados provenientes da estrutura de otimização e "ajuste fino" de hiperparâmetros, realizado nos dados com ruído. Como o *dataset* é limitado a 319 exemplos, o que é relativamente pouco, nesta tabela é apresentado a acurácia tanto dentro da amostra de treino (conjunto de validação), quanto fora da amostra de treino (conjunto de teste).

Tabela 4 - Tabela de resultados do modelo da luva com ruído

Modelos	Acurácia média dentro da amostra	Acurácia máxima dentro da amostra	Acurácia média fora da amostra	Acurácia máxima fora da amostra	Tempo médio de treinamento (segundos)
KNN	0,919	**1**	0,9	0,937	0,21
XGBOOST	**0,955**	**1**	**0,968**	**0,968**	**0,13**
SVM	0,912	0,965	0,934	0,937	11,93

Fonte: Reprodução do autor

Como visto na tabela de resultados, o classificador do *XGBoost* obteve o melhor rendimento, com base na acurácia em relação ao conjunto de teste. Além disso, o mesmo possui o menor tempo de treino dentre os demais. Pode-se também ressaltar o valor das acurácias máximas dos modelos *KNN* e *XGBoost* no conjunto de validação. Isso mostra que esses podem ter se ajustado aos dados de treino, porém, ao executarem em observações fora da amostra, eles obtiveram uma acurácia de respectivamente 0,906 e 0,968.

Esse é um tipo de fenômeno que se quer evitar, pois o *overfitting* é extremamente prejudicial ao modelo quando se trata de dados fora da amostra, que, na prática, são os que realmente importam. Este incidente pode ter acontecido com mais facilidade neste experimento principalmente pela baixa quantidade de amostras, por mais que tenham sido utilizados reguladores nas buscas em grade do *SVM* e do *XGBoost*.

Outra observação relevante se refere à quantidade de amostras de teste (correspondente a 32 amostras). Ao se errar uma classificação, impacta-se diretamente em 0,032 na acurácia do modelo (que é exatamente a diferença das acurácias máximas entre o melhor e os demais). Logo, é difícil afirmar, pela melhor execução, qual deles se ajusta melhor aos dados de forma

geral. Porém, em relação à constância dos resultados, o modelo de *boosting* possui uma vantagem significativa.

Por este motivo, em projetos futuros, existe a intenção de aumentar a quantidade de amostras, assim como criar novas classes (adicionar mais palavras ao vocabulário). Desta forma, é possível ter resultados que diferenciam mais a eficácia dos modelos escolhidos, sendo possível propor qual possui o melhor desempenho.

O modelo final encontrado foi capaz de ter uma taxa de acerto muito interessante, de 96,8%, que de fato se reflete na realidade dos testes da luva em modo de tradução. Quando os sinais são realizados, o sistema consegue obter uma alta precisão de acerto, mesmo sendo feito por pessoas que não fizeram parte da construção do *dataset* referente a este estudo.

Outro fato que vale ser ressaltado está presente no *dataset* deste problema. Como este conjunto foi elaborado especificamente para este tipo de sistema, observa-se que os resultados ficaram muito superiores aos encontrados nos modelos de reconhecimento de expressões faciais. Logo, a construção deste foi imprescindível para os bons resultados que foram encontrados.

No segundo experimento foi proposta a utilização do conjunto sem ruído. Seu propósito visa um caso que no futuro seja possível trafegar os dados sem nenhum tipo de falha ou criar um mecanismo de limpeza sem prejudicar o tempo de resposta do sistema. Atualmente, estas amostras foram selecionadas manualmente com base nas coletas. Na **tabela 5** é possível observar o resultado deste experimento, da mesma forma que foi feito anteriormente.

Tabela 5 - Tabela de resultados do modelo da luva sem ruído

Modelos	Acurácia média dentro da amostra	Acurácia máxima dentro da amostra	Acurácia média fora da amostra	Acurácia máxima fora da amostra	Tempo médio de treinamento (segundos)
KNN	0,958	**1**	0,960	0,966	0,20
XGBOOST	0,958	**1**	**1**	**1**	**0,12**
SVM	**0,965**	**1**	0,950	**1**	10,02

Fonte: Reprodução do autor

Ao observar os dados da **tabela 5**, nota-se que, ao fazer a limpeza do *dataset*, os resultados são muito superiores. Inclusive foi possível obter modelos de *XGBoost* e *SVM* que conseguiram uma taxa de acerto de 100% com relação ao conjunto de teste. Porém, uma acurácia de 100% neste caso não significa que este modelo na prática acertará sempre a classe das palavras. Isto ocorreu devido à quantidade limitada de dados para este espaço

amostral. Logo, em projetos futuros, é indicado a necessidade de gerar mais dados desta forma, para que os modelos sejam ainda mais estáveis do que os presentes.

Por mais que os dados tentem representar o universo de palavras definidos pelo *dataset*, podem existir variações que não tenham sido captadas pelas amostras. Logo esses resultados mostram que no conjunto sem ruídos é observado um mapeamento e classificação superiores ao primeiro experimento. Por intuição, com a remoção de dados ruidosos, existem menos inconsistências no posicionamento espacial das amostras, diminuindo possíveis *outliers* que poderiam prejudicar a classificação.

Mesmo que todos os classificadores tenham obtido ótimos resultados neste caso, em especial, vale ressaltar que o *XGBoost* conseguiu uma acurácia média fora da amostra de 100%. Isto significa que todos os 10 modelos criados pelos *k-fold* do classificador conseguiram acertar a classe de todos os dados do conjunto de teste. Dessa forma, este classificador conseguiu fazer a melhor generalização possível.

5.3 COMPOSIÇÃO DOS PROCEDIMENTOS

A composição dos reconhecimentos para a aplicação final é feita a partir dos dois classificadores obtidos nos dois processos descritos anteriormente (reconhecimento de expressões faciais e reconhecimento de sinais com a luva). É feito um mapeamento do sinal e da expressão obtida, ligando uma palavra em LIBRAS a uma combinação dos dois resultados. Esse processo ainda é manual, na forma de condicional no programa final.

Como as palavras utilizadas nos testes não são muito dependentes da expressão facial em si, sua utilização é limitada a uma possível entonação na fala. Por exemplo, uma composição entre a expressão de surpresa, juntamente com o sinal de aluno, indicaria uma pergunta em LIBRAS com o significado: “você é um aluno?”.

Para o exemplo do funcionamento da aplicação, foi gerado um vídeo [20], que exemplifica como foi feita a comunicação com a utilização deste sistema. Nele, é formada a frase “Oi meu nome Rennan, aluno computação” que indica um exemplo de execução perfeita.

6 CONCLUSÃO E PROJETOS FUTUROS

Vale ressaltar que este projeto foi um estudo inicial sobre o problema, e foram testados vários métodos para a solução do mesmo como descritos no capítulo de implementação. Não foram especificadas todas as tentativas neste documento, uma vez que muitas delas não tiveram um ganho mínimo que pudesse agregar algum valor ao trabalho.

No estado atual, foram estudadas e implementadas diversas ferramentas para criar um método de tradução eficiente. Ao final do projeto, foi possível concluir que é viável construir um modelo capaz de realizar tal tarefa. Apesar de todas as dificuldades, alguns experimentos possuíram resultados muito melhores do que os esperados em primeiro momento. Entretanto, ainda há espaço para muitas melhorias, inclusive de resultados que são essenciais para a proposta inicial.

Sua mobilidade ainda é restrita pela utilização de um *notebook* e de uma conexão com fio. Mas sua arquitetura abre espaço para o aprimoramento de uma solução ainda mais móvel, com a utilização de um *smartphone* e luvas sem fio. Estes não foram possíveis de implementar até o momento, mas existem planos para sua realização.

Apesar dos esforços, com relação à identificação de expressões faciais, existem muitos fatores que resultaram em uma performance abaixo do esperado. Primeiramente, para construir um sistema especifico para LIBRAS, seriam necessários os dados específicos deste nicho, com todos os tipos de intensidade e variação de expressões faciais. Porém, não estão disponíveis e a reprodução de um trabalho como este pode resultar até em outra tese completamente nova.

Outro fator foi o baixo desempenho da máquina utilizada para se fazer os testes. Com ela, não foi possível utilizar *datasets* que são conhecidos por terem mais eficácia e isso contribuiu de forma negativa nos resultados encontrados.

Entretanto, o sucesso da construção de uma luva sensorial eficiente foi uma grande surpresa. Como no início do projeto não era de conhecimento a manipulação de microcontroladores, era esperado que ocorressem diversas falhas neste processo. Mesmo com problemas relacionados à obtenção das componentes para a criação do protótipo, os resultados foram superiores a 95% inclusive com ruídos. Esses resultados indicam que a abordagem e a arquitetura proposta são muito eficientes para a resolução deste problema.

Foi observado que todos os esforços para a criação de um conjunto de dados de sinaise movimentações realizados pela luva sensorial foram recompensados. Este mapeamento de dados que possibilitou essa taxa de acerto muito elevada. Esse se contrapõe com o resultado

obtido pelo sistema de reconhecimento de expressões faciais, em que foram utilizados dados já disponíveis.

Com o rendimento abaixo do esperado do sistema de expressões faciais, a integração das partes ficou muito prejudicada. Para que esta parte seja melhor desenvolvida, se faz necessária a criação de um modelo específico em LIBRAS, composto por dados medidos a partir das variações de expressão que uma pessoa pode ter nesta língua. Desta forma, será possível obter o resultado esperado que funcione no dia-a-dia de todos os usuários.

Em adição, pelo fato de não existir previamente um *dataset* relacionado à movimentação das mãos e por ser necessário mais investimento para a produção de uma segunda luva, para captar os sinais das duas mãos, não foi obtida uma arquitetura completa. Sobre as melhorias, existem dois principais caminhos a seguir. O primeiro deles se refere à melhoria nos modelos em geral, o segundo se refere a estrutura física e de arquitetura do problema.

No que tange ao modelo da luva, até o momento não foi possível produzir um vasto banco de dados que pudesse ser utilizado, limitando também a quantidade de classes a 8. Desta forma, ampliar a quantidade de dados e criar mais classes com mais vocabulário irá enriquecer muito o trabalho e gerar novos desafios ainda não respondidos, como por exemplo: “O modelo gerado consegue ser escalável com o aumento de classes?”, “Existirá um problema com o tempo de resposta?”. Essas questões irão aparecer conforme o modelo se tornar mais complexo.

Sobre a implementação do reconhecimento de expressões faciais, existe uma grande melhoria a ser feita nos modelos produzidos. Já existem conjuntos de dados que melhor representam a realidade para este desafio e, com mais recursos de *hardware,* será possível melhorar consideravelmente a acurácia encontrada.

No mesmo segmento, podem ser explorados também diferentes modelos, como redes neurais mais complexas para a solução do problema, além da possibilidade de utilização de séries temporais como modelos baseados em cadeias de Markov, já observados em trabalhos relacionados na área (tanto para os modelos de expressão facial, quanto para os da luva).

Em relação à arquitetura proposta, um projeto futuro imediato é a implementação de uma segunda luva que capte também sinais da outra mão. Atualmente, o projeto é limitado a movimentos com apenas uma mão, por mais que seja possível fazer o cadastro de novas palavras pelo *software* criado.

Felizmente, para cumprir este desafio basta adquirir os componentes presentes no modelo de arquitetura do projeto e replicar a segunda luva. As mudanças devem ser feitas de

forma pontual a partir deste ponto. Outra melhoria estrutural imediata é a transmissão de dados via *wifi* ou *bluetooth*, ao invés da utilização de um cabo USB.

Esta foi uma sugestão para a solução do problema com base em outros estudos, entretanto, nada impede a criação de um modelo ainda mais completo e eficiente, com base em todas as possibilidades que os sinais da LIBRAS fornecem. Até que o objetivo final de construção de um sistema capaz de realizar a tradução de forma perfeita seja realizado, esta área ainda pode ser amplamente explorada.

REFERÊNCIAS

[1] ABU-MOSTAFA, Y. **Learning from data**: machine learning course. [S. l.]: Caltech, 2012. Disponível em: https://work.caltech.edu/lectures.html. Acesso em: 20 maio 2020.

[2] ABHISHEK, K. S.; QUBELEY, L. C. K.; HO, D. Glove-based hand gesture recognition sign language translator using capacitive touch sensor. *In:* IEEE INTERNATIONAL CONFERENCE ON ELECTRON DEVICES AND SOLID-STATE CIRCUITS, 2016, Hong Kong. **Proceedings** [...]. Hong Kong: IEEE, 2016, p. 334-337.

[3] ADAFRUIT. **Long flex sensor**. [2010?]. Disponível em: https://www.adafruit.com/product/182. Acesso em: 12 nov. 2019.

[4] AMAZON. **Adafruit Long Flex sensor [ADA182]**. [2017?]. Disponível em: https://www.amazon.com/Adafruit-Long-Flex-sensor-ADA182/dp/B01BNNNS5Q. Acesso em: 15 jun. 2020.

[5] ANALOG DEVICES. **ADXL345 overview**. [S. l.], [2010?]. Disponível em: https://www.analog.com/en/products/adxl345.html#. Acesso em: 26 nov. 2019.

[6] ARAÚJO, F. H. D. et al. **Redes neurais convolucionais com tensorflow: teoria e prática**. *In*: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos, v.1, 2017, p. 382-406.

[7] ARDUINO. **Arduino IDE**. [S. l.], 2005. Disponível em: https://www.arduino.cc/en/main/software. Acesso em: 18 set. 2019.

[8] BISHOP, C. M. Linear Models for Classification. In:_____. **Pattern Recognition and Machine Learning**. New York: Springer, 2006, p. 179-220.

[9] BRASHEAR, H., et al. Using multiple sensors for mobile sign language recognition. In: IEEE INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, 7., 2003, White Plains. **IEEE** [...]. White Plains, New York: IEEE, 2003, p. 45-52.

[10] BRASIL. **Lei nº 13.146, de 6 de julho de 2015**. Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência. Estatuto da Pessoa com Deficiência. Brasília, 2015.

[11] BREIMAN, L. Random forests. **Machine learning**, Boston, v. 45, n. 1, p. 5-32, 2001.

[12] CHEN, T. **Story and lessons behind the evolution of xgboost**. 2016. Disponível em: https://homes.cs.washington.edu/~tqchen/2016/03/10/story-and-lessons-behind-the-evolution-of-xgboost.html. Acesso em: 10 jan. 2020.

[13] CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. In: International Conference on Knowledge Discovery and Data Mining, 22., 2016. **Proceedings** [...]. San Francisco: ACM, 2016, p. 785–794. Disponível em: http://doi.acm.org/10.1145/2939672.2939785. Acesso em: 29 jul. 2019.

[14] UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. Faculdade de Letras. CLAC. **Lista de exercícios de LIBRAS**. 2019. 1 fotografia.

[15] COOLEN, A. C. C. A beginner's guide to the mathematics of neural networks. In: LANDAU, L. J.; TAYLOR, J. G. (eds.). **Concepts for Neural Networks.** London: Springer, 1998, p. 13-70.

[16] CORTES, C.; VAPNIK, V. N**.** Support-vector networks. **Machine Learning**, Boston, v. 20, n. 3, 1995.

[17] COTANET. **Aparelho de Surdez Discreto**. 2018. Disponível em: http://saudebem-estar.cotanet.com.br/aparelhos-de-surdez/aparelho-de-surdez-discreto. Acesso em: 21 jan. 2020.

[19] FUNDUINO. **Funduino website**. 2017. Disponível em: https://funduino.de/. Acesso em: 16 jul. 2019.

[20] GAIO, R. **Protótipo do projeto STILL**. 2020. Disponível em: https://www.youtube.com/watch?v=iZr5drbHE7k. Acesso em: 21 jun. 2020.

[21] GRAIL. **Deep expressions**. [2015?]. Disponível em: https://grail.cs.washington.edu/projects/deepexpr/. Acesso em: 14 nov. 2019.

[22] GUO, G., et al. KNN model-based approach in classification. *In:* MEERSMAN, R.; TARI, Z.; SCHMIDT, D. C. (orgs.). **On the move to meaningful internet systems**: CoopIS, DOA, and ODBASE. Berlin, Heidelberg: Springer, 2003. v. 2888, p. 986–996.

[23] IBGE. **Censo demográfico 2010**: características gerais da população, religião epessoas com deficiência. Rio de Janeiro: IBGE, 2012. 215 p. Disponível em: https://biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf Acesso em: 02/05/2019

[24] JAVATPOINT. **Logistic regression in machine learning**. [S.D]. Disponível em: https://www.javatpoint.com/logistic-regression-in-machine-learning. Acesso em: 20 maio 2020.

[25] KAGGLE. **Kaggle website**. 2010. Disponível em: https://www.kaggle.com/. Acesso em: 14 nov. 2019.

[26] KIM, J.; THANG, N. D.; KIM, T. 3-d hand motion tracking and gesture recognition using a data glove. *In:* IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, Seoul, Korea, 2009. **IEEE** [...]. Piscataway, New Jersey: IEEE, 2009.

[27] LEMELSON-MIT. **Thomas Pryor and Navid Azodi**. 2020. Disponível em: https://lemelson.mit.edu/winners/thomas-pryor-and-navid-azodi. Acesso em: 11 jan. 2020.

[28] LORENA, A. C.; CARVALHO, A. C. P. L. F de. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43-67, 2007.

[29] LYONS, M. et al. **The japanese Female facial expression (JAFFE) database**. 2020. Disponível em: https://zenodo.org/record/3451524#.XkbZaWhKiUk. Acesso em: 14 nov. 2019.

[30] MEHDI, S. A.Y. et. al. Sign language recognition using sensor gloves. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING, 9., 2002. **Proceedings** [...]. Singapore: IEEE, 2002. p. 2204-2206.

[31] MENDES, A. I. F.; ARRAIS, K. C.; FUKUSIMA, S. S. F. Faces prototípicas provenientes de amostras populacionais de uma região brasileira. **Psicologia: Reflexão e Crític**a, v. 22, n. 2, p. 261-268, 2009.

[32] MONARD, M. C.; BARANAUSKAS, J. A. Indução de regras e árvores de decisão. In: OLIVEIRA, S. R. (ed.). **Sistemas Inteligentes**: fundamentos e aplicações. Barueri, SP: Manole, 2003, p. 115-139.

[33] MUBARIS, N. K. **Support Vector Machines for Classification**. 2017. Disponível em: https://mubaris.com/posts/svm/. Acesso em: 20 maio 2020.

[34] NIELSEN, D. **Tree Boosting With XGBoost -** Why does Xgboost Win "Every" Machine Learning Competition? Master's Thesis, NTNU. Trondheim, Norway, 2016. 110 p. Disponível em: http://pzs.dstu.dp.ua/DataMining/boosting/bibl/Didrik.pdf. Acesso em: 20 nov. 2019.

[35] OGLA, R.; OGLA, A.; Abdul Hussien, A.; Mahmood, M. Face Detection by Using Open CV's Viola-Jones Algorithm based on coding eyes. **Iraqi Journal of Science**, v. 58, n. 2, p. 735-745, 2017.

[36] OZ, C.; LEU, M. C. Recognition of finger spelling of American sign language with artificial neural network using position/orientation sensors and data glove. *In:* INTERNATIONAL SYMPOSIUM ON NEURAL NETWORKS, 2., 2005, Chongping, China. **Proceedings** [...]. New York: Springer, 2005. p. 157-164.

[37] PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of machine learning research**, v. 12, n. 85, p. 2825-2830, 2011.

[38] LUCEY, P.; COHN, J. F.; KANADE, T.; SARAGIH, J.; AMBADAR, Z.; MATTHEWS, I. The Extended Cohn-Kanade Dataset (CK+): a complete dataset for action unit and emotion-specified expression. *In:* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, San Francisco. **Proceedings** [...]. New York: IEEE, 2010, p. 94-101.

[40] REIMONDO, A. **Haar Cascades**. Disponível em: http://alereimondo.no-ip.org/OpenCV/34/. Acesso em: 14 nov. 2019.

[41] RODRIGUEZ, J. D.; PEREZ, A.; LOZANO, J. A. Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE transactions on pattern analysis and machine intelligence, v. 32, n. 3, p.569-575. 2009.

[42] SANTOS, J.; COSTA, M.; COSTA FILHO, C. Reconhecimento das configurações de mão de libras baseado na análise de discriminante de fisher bidimensional, utilizando imagens de profundidade. In: ANAIS PRINCIPAIS DO SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE (SBCAS), 15., 2015, Recife. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2015. p. 21-30.

[43] SHAFFI, N. & Kumar, G & Shivakumara, P. 2006. (2D)^2 LDA: an efficient approach for face recognition. **Pattern Recognition**, v.39, n. 7, p. 1396-1400, 2006.

[44] SILVANO, F. et al. **ALFALUVA**: Luva detectora de sinais em LIBRAS para auxílio na alfabetização**.** Disponível em: https://alfaluva.wordpress.com/. Acesso em: 14 jan. 2020.

[45] SOFIATO, C. G.; REILY, L. H. Dicionarização da língua brasileira de sinais: estudo comparativo iconográfico e lexical. **Educação e Pesquisa**, São Paulo, v. 40, n. 1, p. 109-126, 2014. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1517-97022014000100008&lng=en&nrm=iso. Acesso em: 18 jun.2020.

[46] SU, M. A fuzzy rule-based approach to spatio-temporal hand gesture recognition. **IEEE Transactions on Systems, Man, and Cybernetics**, part C (Applications and Reviews), v. 30, n. 2, p.276-281, 2000.

[47] VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE computer society conference on computer vision and pattern recognition, 2001, Kauai, Hawaii. **Proceedings** [...] Los Alamitos, Calif.: IEEE Computer Society, 2001.

[48] VISGRAF. **FacesDB:** VISGRAF faces database. 2008. Disponível em: http://app.visgraf.impa.br/database/faces/. Acesso em: 14 nov. 2019.

[49] WIKIPEDIA. **Power Glove**. 2000. Disponível em: https://en.wikipedia.org/wiki/Power_Glove. Acesso em: 26 maio 2020.

[50] WIKIPEDIA. **Supervised learning**. [S. l.], 2020. Disponível em: https://en.wikipedia.org/wiki/Supervised_learning. Acesso em: 10 jan. 2020. CONFERIR

[52] XGBOOST DEVELOPERS. **XGBoost Documentation**. 2020. Disponível em: https://xgboost.readthedocs.io/en/latest/. Acesso em: 10 jan. 2020. Conferir data documento

[53] YIU, T. **Understanding Random Forest**. 2019. Disponível em: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. Acesso em: 11 dez. 2019.