CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

INTERPRETAÇÃO DE GESTOS DINÂMICOS DE LIBRAS USANDO K-MEANS E RANDOM FOREST

Eros Gabriel de Abreu Caiafa

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientador(a): Gabriel Matos Araujo
Coorientador(a):

Rio de Janeiro,

Janeiro 2025

INTERPRETAÇÃO DE GESTOS DINÂMICOS DE LIBRAS USANDO K-MEANS E RANDOM FOREST

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Eros Gabriel de Abreu Caiafa

Banca Examinadora:

Presidente, Professor D.Sc. Gabriel Matos Araujo (CEFET/RJ) (Orientador(a))

Professor Ph.D. Amaro Azevedo de Lima (CEFET/RJ)

Professor D.Sc. Milena Faria Pinto (CEFET/RJ)

Professor D. Sc. Michel Pompeu Tcheou (UERJ)

Rio de Janeiro,

Janeiro 2025

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C133 Caiafa, Eros Gabriel de Abreu
Interpretação de gestos dinâmicos de libras usando K-Means e Random Forest / Eros Gabriel de Abreu Caiafa – 2025.
77f. : il. color. + apêndice, enc.

Dissertação (Mestrado). Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2025.
Bibliografia : f. 75-77.
Orientador: Gabriel Matos Araujo.

1. Framework (Arquivo de computador). 2. Tecnologia assistiva. 3. Língua brasileira de sinais. 4. Engenharia elétrica. I. Araujo, Gabriel Matos (Orient.). II. Título.
II. Título.

CDD 004.6

Elaborada pelo bibliotecário Leandro Mota de Menezes – CRB/7 nº 5281

DEDICATÓRIA

Dedico esse trabalho a Deus por ter me ajudado a chegar aqui e todas oportunidades que me deu ao longo dessa jornada. Também dedico à minha mãe que esteve junto comigo e me apoiando todos esses anos, me tornando a pessoa que sou hoje. Além disso, gostaria de dedicar a minha avó que também me deu grande apoio, principalmente durante a graduação e a memória do meu tio César, que infelizmente não pode acompanhar essa etapa da minha vida, mas que me apoiou muito, principalmente durante a minha infância.

AGRADECIMENTOS

O presente trabalho foi desenvolvido com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES - Código de Financiamento 001).

Agradeço também ao meu orientador Gabriel, que desde o começo da graduação já encontrou em mim potencial e me estimulou a seguir adiante, me permitindo crescer acadêmica e profissionalmente. Também ao professor Amaro, que sempre trouxe observações pertinentes ao projeto. Também agradeço ao Laboratório de Sinais do CEFET/NI que me permitiu executar os testes desse trabalho na sua estrutura, possibilitando a sua execução.

RESUMO

Interpretação de gestos dinâmicos de Libras usando *k-means* e *Random Forest*

Pessoas com perdas mais graves de audição costumam utilizar uma língua de sinais para se comunicar. Contudo, a maioria da população não domina a comunicação em línguas de sinais, o que pode gerar barreiras de inclusão social e comunicação efetiva. Nesse contexto, a implementação e utilização de tecnologias assistivas que auxiliem na tradução e interpretação de línguas de sinais torna-se essencial para promover maior acessibilidade e inclusão. Explorando uma sub-área da visão computacional conhecida como *Signal Language Recognition (SLR)*, propomos um *framework* de baixo custo para classificar sinais da Língua Brasileira de Sinais (Libras) em vídeo. O método proposto utiliza *MediaPipe*, para extração de pontos-chaves de usuários, *K-Means* para agrupar esses pontos e classificadores KNN e *Random Forest* para classificar os agrupamentos. Os resultados experimentais mostram que o modelo proposto alcançou uma acurácia de 94,72% utilizando a base de dados MINDS-Libras com o classificador *Random Forest*, destacando o potencial do framework em aplicações práticas. Além disso, o uso de ferramentas acessíveis e algoritmos consagrados reforça a viabilidade do sistema para ser implementado em larga escala, promovendo o desenvolvimento de tecnologias inclusivas e acessíveis voltadas para a comunicação entre deficientes e ouvintes.

Palavras-chave: Surdez; Libras; Inclusão; Reconhecimento de Linguagem de Sinais; SLR; Aprendizado de Máquina

ABSTRACT

Interpreting dynamic gestures of Libras using k-means and Random Forest

People with more severe hearing loss often use sign language to communicate. However, the majority of the population is not proficient in sign language communication, which can create barriers to social inclusion and effective communication. In this context, the implementation and use of assistive technologies to help translate and interpret sign languages is essential to promote greater accessibility and inclusion.. Exploring a sub-area of computer vision known as SLR, we propose a low-cost framework to classify Libras signals in videos. The proposed method uses MediaPipe to extract user key points, K-Means to cluster these points, and KNN classifiers and Random Forest to classify the clusters. The experimental results show that the proposed model achieved an accuracy of 94.72% using the MINDS-Libras database with the Random Forest classifier, highlighting the framework's potential in practical applications. In addition, the use of accessible tools and established algorithms reinforces the system's viability for large-scale implementation, promoting the development of inclusive and accessible technologies aimed at communication between disabled people and listeners.

Keywords: Deafblindness; Libras; Inclusion; Sign Language Recognition; SLR; Machine Learning

LISTA DE ILUSTRAÇÕES

Figura 1 – Ilustração dos gestos da base Minds. Fonte: (REZENDE; ALMEIDA; GUIMARÃES, 2021). 24

Figura 2 – Representação do funcionamento de uma rede neural. Fonte: (COSTA; SENEFONTE, s.d.). 27

Figura 3 – Representação do funcionamento do *K-Means*. Fonte: Elaboração Própria. 35

Figura 4 – Representação do funcionamento do KNN. Fonte: (CORREA et al., 2023). 37

Figura 5 – Representação do funcionamento do Random Forest. Fonte: Adaptado pelo autor. 39

Figura 6 – Ilustração dos pontos-chaves detectados pelo *MediaPipe*. Fonte: (LUGARESI et al., 2019) 43

Figura 7 – Ilustração do *pipeline* do trabalho. Fonte: Elaboração Própria. 51

Figura 8 – Ilustração da execução do modelo. Fonte: Elaboração própria 54

Figura 9 – 3 diferentes usos implementadas dos T-SNE, as tabelas em azul representam os dados antes da clusterizção, em lilás representam os dados após a clusterização, as dimensões em verde representam a dimensão reduzida pelo T-SNE Fonte: Elaboração própria 79

LISTA DE TABELAS

Tabela 1 – Sinais escolhidos para compor a base de dados Minds-Libras. 23

Tabela 2 – Hiperparâmetros otimizados. 59

Tabela 3 – Comparação entre os modelos em termos de acurácia e tempo de classificação 64

Tabela 4 – Hiperparâmetros escolhidos para o K-nearest Neighbors Algorithm (KNN). 65

Tabela 5 – Hiperparâmetros escolhidos para o *Random Forest*. 66

Tabela 6 – Hiperparâmetros escolhidos para a Rede Neural. 66

LISTA DE ALGORITMOS

Algoritmo 1 – Algoritmo Adam 31

Algoritmo 2 – Algoritmo do modelo 56

LISTA DE ABREVIATURAS E SIGLAS

ADAM	Adaptive Moment Estimation
API	Interface De Programação De Aplicações
ASL	Língua Americana De Sinais
CNN	Rede Neural Convolucional
CPU	Unidade De Processamento Central
DGAN	Deep Generative Adversarial Network
ELU	Exponential Linear Unit
EMA	Média Móvel Exponencial
GPU	Unidade De Processamento Gráfico
GRU	Gated Recurrent Unit
IBGE	Instituto Brasileiro De Geografia E Estatística
ISL	Língua Indiana De Sinais
KNN	K-nearest Neighbors Algorithm
KSL	Língua Coreana De Sinais
LIBRAS	Língua Brasileira De Sinais
LOPO	Leave One Person Out
LSTM	Long Short-term Memory
MLP	Multi Layer Perceptron
OPENCV	Visão Computacional Open Source
PDI	Processamento Digital De Imagens
POO	Programação Orientada A Objetos
RELU	Rectified Linear Unit
RMSPROP	Root Mean Square Propagation
RNN	Rede Neural Recorrente
SELU	Scaled Exponential Linear Unit
SGD	Stochastic Gradient Descent
SLR	Signal Language Recognition

SUMÁRIO

Introdução **14**

1 Revisão Bibliográfica **19**

2 Fundamentação Teórica **25**

2.1 Aprendizado de Máquina 26

2.1.1 Aprendizado profundo 27

2.1.2 K-Means 35

2.1.3 KNN 37

2.1.4 Random Forest 39

2.1.5 Naïve Bayes 41

2.2 MediaPipe 42

2.3 Scikit-Learn 44

2.3.1 API Scikit-Learn 44

2.4 Tensorflow e Keras 45

2.5 OpenCV 46

2.6 Python 47

2.7 Joblib 48

3 Metodologia **50**

3.1 Pré-processamento 50

3.1.1 Separação de treino e teste 52

3.1.2 Redução de dimensionalidade 53

3.2 Classificação 55

3.3 Bayes Search 57

3.3.1 Hiperparâmetros otimizados 59

4 Resultados e discussão **63**

4.1 Resultados 63
4.1.1 Melhores parâmetros 65
4.2 Discussão 69

Considerações finais 71

5 Considerações finais 72
5.1 Trabalhos Futuros 73

Referências 74

Apêndice A 78

Introdução

Motivação

A deficiência auditiva é uma condição que afeta uma parcela considerável da sociedade em todo o mundo. No Brasil, de acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE), existem 9,7 milhões de pessoas com algum grau dessa deficiência (CAIAFA et al., 2020). Essa condição pode trazer diversos desafios na comunicação e na integração social desses indivíduos, além dos desafios que a própria deficiência já traz normalmente.

Com o objetivo de promover a inclusão e garantir os direitos das pessoas com deficiência auditiva, a Libras foi criada e reconhecida como a segunda Língua oficial do Brasil em 2002 (BRASIL, 2002). A Libras é uma língua que se baseia na comunicação visual e gestual, e que, com isso, permite a interação entre pessoas surdas e mudas entre si, ou até com não deficientes.

No entanto, apesar do reconhecimento oficial da Libras, ainda é bem comum encontrar barreiras e dificuldades na comunicação entre deficientes e não deficientes auditivos. Muitas vezes, a falta de familiaridade, ou até completo desconhecimento da Libras por parte da maior parcela da população acaba gerando exclusão e marginalização dos indivíduos com deficiência auditiva.

Com o avanço da tecnologia, foram surgindo novas oportunidades para desenvolver soluções que possam auxiliar na superação desses abismos comunicacionais, como o presente trabalho ou os trabalhos relacionados como (CAIAFA et al., 2020, 2023; ABDALLAH et al., 2023; SUNDAR B. ANDBAGYAMMAL, 2022; CASTRO et al., 2019; REZENDE; ALMEIDA; GUIMARÃES, 2021; SURABHI et al., 2012; RAO et al., 2018; CASTRO; GUERRA; GUIMARÃES, 2023). Uma área que merece destaque especial nesse contexto é o *Signal Language Recognition* (SLR), que consiste em desenvolver sistemas capazes de reconhecer e interpretar gestos e sinais utilizados na comunicação de sinais.

Para a resolução do problema de SLR, existem atualmente duas abordagens

principais, a baseada em equipamentos vestíveis e os baseados em Processamento digital de imagens (PDI) (CAIAFA et al., 2020). Os modelos vestíveis, geralmente compostos por sensores e dispositivos que capturam o movimento e posição do corpo, oferecem uma forma mais imersiva e precisa da interpretação dos gestos. No entanto, possuem limitações, como restrições físicas, necessidade de utilização de dispositivos físicos e um custo mais elevado. Por outro lado, os modelos baseados em PDI, como o proposto neste projeto, utilizam técnicas de processamento de imagens para extrair informações relevantes dos gestos, permitindo a interpretação e classificação dos mesmo. Embora possam apresentar uma precisão ligeiramente inferior em comparação aos modelos vestíveis, eles têm a vantagem de serem mais acessíveis, flexíveis e menos onerosos, tendo como único dado necessário o video que pode ser capturado por dispositivos comuns, como *smartphones* (CAIAFA et al., 2020). Essa abordagem baseada em PDI visa atingir um equilíbrio entre eficiência, acessibilidade e precisão, ampliando a possibilidade de utilização da Libras e contribuindo para a inclusão e comunicação efetiva entre pessoas com e sem deficiência auditiva ou de fala.

Objetivo

Nesa pesquisa, propõe-se a implementação de um modelo de SLR para Libras, utilizando um modelo de PDI com baixo custo computacional, mas ainda assim, capaz de superar o estado da arte. Essa abordagem tem como objetivo criar uma solução acessível e eficiente para reconhecer e interpretar os gestos dinâmicos de Libras, visando contribuir para a inclusão e comunicação efetiva entre pessoas com e sem deficiência auditiva ou de fala.

Para essa implementação, optou-se pelo uso do *framework MediaPipe* (LUGARESI et al., 2019), cuja escolha se justifica devido à sua flexibilidade e capacidade de processamento em tempo real, mesmo em *smartphones* mais antigos. Isso permite uma implementação mais eficiente do modelo de SLR para a Libras. Além disso, o *framework* oferece recursos e ferramentas que facilitam o desenvolvimento de sistemas de reconhecimento de sinais, fornecendo informações sobre a posição de cada ponto-chave do corpo humano, similar a um sistema vestível, mas sem as limitações impostas por esse

tipo de tecnologia.

A fim de alcançar o reconhecimento adequado dos gestos da Libras, utilizou-se o modelo K-Means (ZHOU, 2021) para agrupar os quadros do vídeo em *clusters*, reduzindo a complexidade do modelo na etapa de classificação. Essa abordagem busca minimizar a quantidade de dados a serem processados e contribuindo para um modelo mais leve e com baixo custo, ideal para ser executado em dispositivos móveis.

Após a redução de dimensionalidade com o K-Means, foram testados três algoritmos de classificação: *KNN*, *Random Forest* e *Multi Layer Perceptron (MLP)* (ZHOU, 2021). Dentre eles, o *Random Forest* obteve a melhor acurácia, alcançando 94,72%, além de um resultado de $59,44\% \pm 10,63\%$ de acurácia utilizando a metodologia *Leave One Person Out (LOPO)*. Esse resultado se deve à capacidade desse algoritmo em aprender padrões complexos mesmo com poucos dados disponíveis. Para o treinamento do modelo, utilizou-se o otimizador *Bayes Search* (ALIBRAHIM; LUDWIG, 2021), um algoritmo de busca de melhores hiperparâmetros para modelos de aprendizado de máquina, que se baseia no algoritmo Naïve Bayes para otimizar a acurácia de validação.

O treino e a avaliação de desempenho do modelo serão realizados utilizando a base de dados MINDS-Libras (CASTRO et al., 2019). Será considerada métrica de desempenho, como a acurácia, para mensurar a capacidade do modelo em reconhecer os gestos da Libras. Espera-se que este projeto contribua para reduzir o distanciamento social e promover a inclusão de pessoas com deficiência auditiva e de fala por meio da tecnologia. Através do desenvolvimento de um modelo de SLR para a Libras, utilizando o *framework MediaPipe* e os modelos *K-Means* e *Random Forest*, busca-se melhorar a comunicação e o entendimento entre surdos, mudos e ouvintes, possibilitando uma interação mais inclusiva e equitativa na sociedade.

Contribuições

Esse projeto apresenta valiosas contribuições, ele utiliza o algoritmo *K-Means* para efetuar o pré-processamento dos dados, o que traz uma grande eficiência inerente do modelo. O *K-Means* além de ser robusto à grande variação de quadros nos vídeos, podendo lidar com ela, também apresenta uma saída de dimensão constante, o que

permite uma normalização temporal o que é crucial, não só para a interpretação de Libras, mas para o processamento de uma vasta lista de sinais. A seleção estratégica dos centróides pelo *K-Means* leva em conta a natureza dos sinais, evidenciando e ressaltando os itens léxicos, que são representados por muitas observações semelhantes, ao mesmo passo que miniminiza o impacto de transições e quadros discrepantes, dando foco aos quadros relevantes.

Além disso, a escolha de classificadores de baixo custo, como KNN e *Random Forest*, amplia ainda mais as contribuições desse trabalho, se afastando um pouco da tendência da comunidade científica, que para os problemas de SLR vem cada vez mais utilizando modelos baseados em Rede Neural Convolucional (CNN). Os modelos utilizados, embora simples, demonstram uma performance notável na classificação de gestos de Libras, superando os desafios específicos relacionados à linguagem gestual, tendo um resultado semelhante aos modelos baseados em CNN. A abordagem de baixo custo não só facilita a implementação mais prática em dispositivos mais acessíveis, como celulares, mas também destaca a viabilidade de soluções econômicas para auxiliar pessoas com deficiência de fala e audição.

Em última análise, as contribuições desse projeto transcendem o âmbito técnico, pois oferecem uma perspectiva inovadora e mais prática para melhorar a qualidade de vida das pessoas com deficiência de fala e audição. Ao empregar tecnologias acessíveis e eficazes na interpretação de gestos de Libras, o trabalho não apenas aprimora a comunicação, mas também promove a inclusão, dando voz a indivíduos outrora marginalizados, se tornando uma ferramenta valiosa para expressar as suas ideias e se conectar mais plenamente com a sociedade.

Produção científica

As seguintes publicações estão diretamente relacionadas a essa dissertação:

- **Eros Caiafa**, Fabiana Fonseca, Amaro de Lima, Gabriel Araujo e Eduardo A. B. da Silva; Aprendizado profundo no reconhecimento de sinais estáticos de Libras; XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2020) (CAIAFA et al., 2020)

Nela foram explorados métodos para interpretação de gestos estáticos de Libras, utilizando redes neurais convolucionais e os resultados foram comparados com outros métodos da literatura, tendo resultados acima dos até então relatados.

Além disso, essa publicação está relacionada e foi derivada dessa dissertação:

- **Eros Caiafa**, Allan Basilio, Amaro de Lima e Gabriel Araujo; Interpretação de gestos de Libras usando k-means e Random Forest; XLI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2023) (CAIAFA et al., 2023)

Nessa publicação foi abordado o método de classificação de gestos de Libras utilizando o *K-Means* para fazer a extração de *features*, e foram comparados diversos classificadores, como o *Random Forest* e KNN, sendo o primeiro, o que obteve o melhor desempenho, e tendo superado o estado da arte.

Organização do Trabalho

Após essa introdução, o presente trabalho terá o Capítulo 1 onde serão expostos trabalhos que possuem semelhanças e influenciaram essa pesquisa, seguindo, o Capítulo 2 se dedicará a explicar as bases conceituais desse trabalho e trará a revisão bibliográfica que embasa o desenvolvimento desse projeto. Após isso tem-se o Capítulo 3 que abordará em mais detalhes como esse projeto foi desenvolvido, bem como as ferramentas foram utilizadas e o motivo de suas escolhas. Por conseguinte, tem-se o Capítulo 4 que tratará dos resultados obtidos nesse projeto e será discutido acerca deles. Por fim, temos o Capítulo 5 que irá concluir esse trabalho e apresentará passos que podem ser tomados a seguir baseando-se nesse projeto e seus resultados.

1- Revisão Bibliográfica

A Libras, como uma forma de comunicação utilizada por pessoas com deficiência auditiva, tem recebido crescente atenção na área de processamento de sinais e aprendizado de máquina. Nesse contexto, a classificação de gestos dinâmicos de vídeos se apresenta como um desafio complexo, devido à variabilidade dos sinais e à natureza temporal dos dados. A presente revisão bibliográfica tem como objetivo investigar as pesquisas recentes relacionadas à classificação de sinais da Libras, bem como a classificação de gestos e técnicas de pré-processamento com algoritmos de clusterização, como o *K-Means*, bem como identificar tendências e lacunas nas pesquisas existentes nessa área.

Caiafa et al. (CAIAFA et al., 2020) utilizou duas bases diferentes de Libras com gestos estáticos, tendo imagens onde a mão do locutor já estava previamente seccionada. Em uma, o fundo era completamente branco e, na outra, as capturas eram feitas por mapa de profundidade. Os autores alcançaram 99,17% de acurácia usando um modelo baseado em CNN no conjunto de dados introduzido em (BASTOS; ANGELO; LOULA, 2015) e 99,13% no conjunto em (COSTA FILHO et al., 2017).

Rao et al. (RAO et al., 2018) apresenta mais uma abordagem baseada em CNN para o SLR. Os autores utilizaram uma base de dados da Língua Indiana de Sinais (ISL), composta por 200 sinais diferentes, executados por 50 locutores, e com troca de plano de fundo. Os vídeos dessa base possuem 60 quadros, e os resultados obtidos pelos autores alcançaram uma acurácia de 92,88%, ao aplicar a CNN. Esta abordagem se diferencia da aplicada por Caiafa et al. por lidar com gestos dinâmicos. O estudo também destaca a eficácia das CNNs no SLR, fornecendo valiosas informações para a área de processamento de sinais e aprendizado de máquina aplicado ao SLR.

Sundar et al. (SUNDAR B. ANDBAGYAMMAL, 2022) apresentou uma abordagem inovadora utilizando o *framework Mediapipe* em conjunto com *Long short-term memory (LSTM)* para fazer o reconhecimento de vídeos contendo as 26 letras do alfabeto da Língua Americana de Sinais (ASL). Os gestos analisados nesse contexto são todos estáticos. Os resultados obtidos por essa técnica foram de 99%, o que evidencia a eficácia e o potencial da utilização do *framework Mediapipe* no processo de SLR. A alta

acurácia alcançada por essa abordagem reforça a importância de explorar ferramentas avançadas de processamento de sinais, como o *MediaPipe*, para melhorar o desempenho de SLR.

Abdallah et al. (ABDALLAH et al., 2023) utilizou dois tipos de redes, uma *Gated Recurrent Unit (GRU)* e uma CNN, em conjunto com o *framework MediaPipe*, para classificar sinais. Foi criado um conjunto de dados para o experimento, chamado de DSL-46. Esse conjunto possui 2910 vídeos, gravados por 9 voluntários diferentes. A duração dos vídeos da base DSL-46 é de 1 segundo, gravados em resolução 640×480 a 30 fps. Além desse conjunto, também foram usadas as bases LSA64 e LIBRAS-BSL nos experimentos. Utilizando essa abordagem, os autores conseguiram atingir uma acurácia de 98,8%, 99,84% e 88,40% nos conjuntos de dados DSL-46, LSA64 e LIBRAS-BSL, respectivamente.

Surabhi et al. (SURABHI et al., 2012) apresentou uma proposta inovadora de reconhecimento facial utilizando o algoritmo K-Means como técnica de pré-processamento das imagens. O estudo envolveu a coleta de fotos com 13 posições diferentes de cada um dos 28 agentes e com 64 condições de iluminação diferentes e comparou o desempenho do modelo proposto com modelos mais convencionais de reconhecimento facial. O modelo utilizando K-Means teve um resultado notavelmente superior de 97,9% de acurácia, significativamente melhor do que o melhor resultado alcançado pelos modelos convencionais, que atingiu 86,14% de acurácia. Esses resultados ressaltam a eficácia do K-Means como uma técnica de pré-processamento promissora para o processamento de sinais.

Rezende et al. (REZENDE; ALMEIDA; GUIMARÃES, 2021) elaborou o conjunto de dados MINDS-Libras usado neste trabalho. Seu sistema de reconhecimento de sinais aplica uma CNN 3D de modo semelhante ao utilizado em (CASTRO et al., 2019): quadros-chave são extraídos dos vídeos e, em seguida, redimensionados e utilizados como entradas da CNN. Entretanto, o trabalho em (REZENDE; ALMEIDA; GUIMARÃES, 2021) também avaliou diferentes técnicas de pré-processamento tais como conversão em tons de cinza, HOG descritores, fluxo óptico, normalização, aumento de dados e o uso individual e combinado de dados RGB e RGB-D. O melhor modelo alcançou uma acurácia média de 93,3% usando apenas dados RGB convertidos em tons de cinza como entrada e executando um *data augmentation* similar ao aplicado em (CASTRO et al., 2019).

Castro et al. (CASTRO; GUERRA; GUIMARÃES, 2023) apresentou uma metodo-

logia onde são utilizados mapas de profundidade artificiais gerados por *Deep Generative Adversarial Network (DGAN)* para fazer a tradução automática de gestos em Libras, ISL e Língua Coreana de Sinais (KSL) sem a utilização de sensores especiais, como câmeras com mapa de profundidade ou luvas especiais. Para alcançar o resultado esperado, foi utilizada uma combinação de diferentes tipos de informações, que incluem segmentação das mãos, rostos, distâncias e velocidades dos pontos das articulações, além dos quadros em cores e dos mapas de profundidade artificiais. O modelo foi testado na base de (REZENDE; ALMEIDA; GUIMARÃES, 2021) onde alcançou uma acurácia de $91\% \pm 7\%$. Um aspecto relevante nesse trabalho foi a aplicação do método LOPO, onde se obteve uma acurácia de $57\% \pm 12\%$.

Os estudos mencionados demonstram a eficácia de diversos classificadores em reconhecer sinais de Libras com altas acurácias, sobretudo em conjuntos de sinais estáticos (CAIAFA et al., 2020; SUNDAR B. ANDBAGYAMMAL, 2022). Nos trabalhos envolvendo sinais dinâmicos (REZENDE; ALMEIDA; GUIMARÃES, 2021; RAO et al., 2018; ABDALLAH et al., 2023; CASTRO et al., 2019; CASTRO; GUERRA; GUIMARÃES, 2023), espera-se uma queda de acurácia, sobretudo quando se aumenta o vocabulário (número de classes). No entanto, ainda existe muito trabalho a ser feito na criação de sistemas mais robustos e eficientes que podem lidar com um grande conjunto de sinais (para aplicações mais realistas), variações na iluminação, ângulos de câmera e planos de fundo. Esses desenvolvimentos têm o potencial de melhorar significativamente a vida daqueles que usam a língua de sinais como sua principal forma de comunicação.

As principais diferenças entre o trabalho aqui descrito e os artigos apresentados nesta seção reside no método de pré-processamento dos dados e no método de classificação. O uso do *MediaPipe* para a extração de pontos-chave do signatário permite reduzir a carga computacional necessária em comparação com a abordagem convencional de processar as imagens completas. Entretanto, diferente dos trabalhos descritos em (SUNDAR B. ANDBAGYAMMAL, 2022; ABDALLAH et al., 2023), nós aplicamos técnicas de agrupamento nos pontos-chave e classificamos apenas os centroides dos grupos. Isso permite reduzir ainda mais o custo computacional. Por outro lado, redes convolucionais (sobretudo as 3D) e Rede Neural Recorrente (RNN) também podem exigir um poder computacional maior do que o *Random Forest*. Portanto, diferente dos trabalhos descritos em (REZENDE; ALMEIDA; GUIMARÃES, 2021; RAO et al., 2018; SUNDAR B. ANDBAGYAMMAL, 2022; ABDALLAH et al., 2023; CASTRO et al., 2019; CASTRO;

GUERRA; GUIMARÃES, 2023) o método de classificação empregado também permite uma redução significativa no custo computacional.

MINDS-Libras

No contexto do presente projeto, foi utilizado o conjunto de dados Minds-Libras, elaborado por (REZENDE; ALMEIDA; GUIMARÃES, 2021), que consiste em 1200 vídeos contendo 20 sinais de Libras repetidos 5 vezes, realizados por 12 intérpretes distintos se diferenciando em diversas características, como idade, cor de pele, cor de roupa e níveis de fluência. Os gestos foram gravados por câmeras Canon EOS Rebel t5i e Kinect RGB-D v2, proporcionando informações de profundidade e cores. A seleção dos sinais foi cuidadosamente feita por especialistas para garantir a diversidade em relação aos parâmetros fonológicos. Embora algumas filmagens tenham sido perdidas, a versão final da base de dados contém 1155 sinais diferentes, sendo a maioria deles dinâmicos, com exceção dos gestos “América” e “cinco”.

O modelo de SLR proposto pelos autores baseou-se em uma arquitetura CNN 3D, seguindo uma abordagem semelhante à utilizada em outros trabalhos, onde quadros-chave são extraídos dos vídeos, redimensionados e utilizados como entrada para outra CNN. Além disso, a pesquisa explorou diversas técnicas de pré-processamento, como conversão em tons de cinza, descritores HOG, fluxo óptico, normalização e aumento de dados. A combinação de RGB e RGB-D também foi avaliada. O modelo mais eficiente alcançou a acurácia média de 93,3%, utilizando apenas dados RGB convertidos para tons de cinza como entrada e empregando um processo de aumento de dados similar ao utilizado em estudos anteriores. No entanto, é importante destacar que os autores não se atentaram a utilização de um conjunto de testes separado, realizando a avaliação do resultado com base na média da acurácia no conjunto de validação.

Apesar disso, deve-se ressaltar que a base de dados Minds-Libras representa uma valiosa contribuição para o campo de SLR de Libras, fornecendo um conjunto diversificado de gestos dinâmicos e estáticos, capturados pelos mais distintos intérpretes. O modelo de SLR proposto demonstrou resultados promissores, alcançando uma acurácia bem significativa, principalmente ao utilizar dados RGB convertidos em tons de cinza,

Tabela 1 – Sinais escolhidos para compor a base de dados Minds-Libras.

Palavras	
Acontecer	Conhecer
Aluno	Espelho
Amarelo	Esquina
América	Filho
Aproveitar	Maçã
Bala	Medo
Banco	Ruim
Banheiro	Sapo
Barulho	Vacina
Cinco	Vontade

evidenciando o potencial dessa abordagem de pré-processamento.

Na Tabela 1, são apresentados os 20 tipos distintos de sinais de Libras que compõem o conjunto de dados Minds-Libras, assim como na Figura 1, que além disso representa como se gesticulam. Esses sinais foram criteriosamente selecionados por especialistas, buscando garantir a diversidade em relação aos parâmetros fonológicos e, assim, proporcionar uma ampla representação dos gestos utilizados na comunicação em Libras.

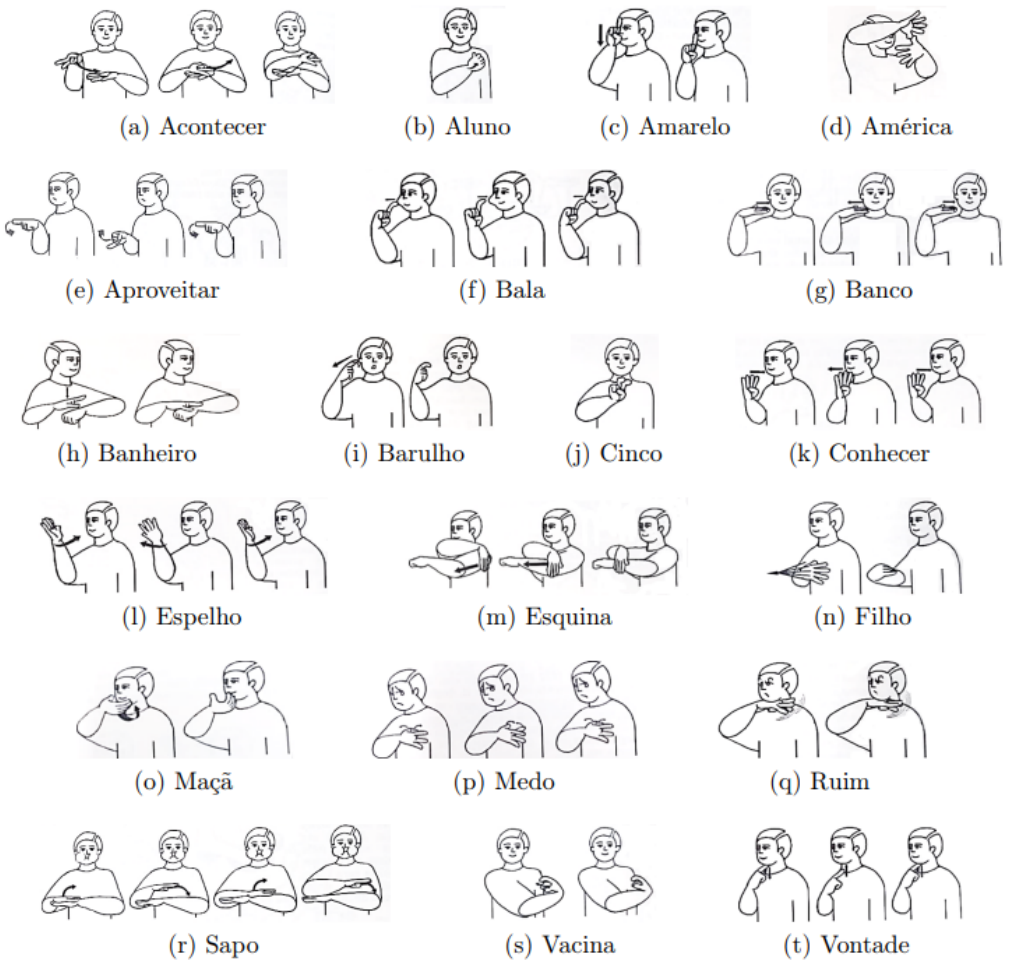


Figura 1 – Ilustração dos gestos da base Minds.
Fonte: (REZENDE; ALMEIDA; GUIMARÃES, 2021).

2- Fundamentação Teórica

O presente capítulo tem como objetivo oferecer uma base sólida para a compreensão das principais metodologias e tecnologias que moldam a implementação do sistema proposto. De forma abrangente, esta seção visa fornecer o leitor uma visão clara de como os conceitos fundamentais de aprendizado de máquina, redes neurais, e técnicas de pré-processamento se entrelaçam para criar uma solução coesa para o reconhecimento de gestos de Libras.

Foram utilizadas ferramentas eficientes para o processamento eficiente de dados, como o *OpenCV*, que foi utilizada para efetuar o processamento de imagens juntamente com o *MediaPipe*, que teve o papel de efetuar a detecção de pontos-chave dos gestos, oferecendo uma compreensão profunda de como a visão computacional impulsiona a interpretação de sinais de Libras.

No contexto de aprendizado de máquina, exploraremos a aplicação de algoritmos fundamentais, como o *K-Means*, utilizado no pré-processamento, bem como o KNN e *Random Forest*, utilizados para efetuar uma classificação eficiente dos padrões gestuais. Dentro desse escopo, também foram utilizadas as redes neurais convolucionais, que são implementadas utilizando o *Tensorflow* e Keras, voltados a desenvolver e treinar modelos eficazes de aprendizado profundo.

Por fim, utilizaremos a busca de hiperparâmetros, especificamente através do *Bayes Search*, se destacando por sua contribuição crucial para o refinamento do modelo, garantindo a maximização de desempenho. Tendo todas essas ferramentas a sua implementação e utilização sendo feitas com a linguagem de programação *Python*.

Em suma, este capítulo serve como um guia introdutório das ferramentas que foram utilizadas e como podemos esperar que seja feita a sua implementação e resultados mais a frente, fornecendo uma visão panorâmica e delineando como cada componente, desde algoritmos de aprendizado de máquina, até as bibliotecas mais específicas desempenham um papel integral na realização desse trabalho.

2.1- Aprendizado de Máquina

O aprendizado de máquina (também comumente chamado de "*Machine learning*") é uma aproximação da inteligência artificial [1] que se dedica ao estudo e desenvolvimento de algoritmos de modelos computacionais capazes de aprender a partir de dados. Esses algoritmos são projetados para identificar padrões e relações nos dados fornecidos, com o objetivo de realizar tarefas específicas sem a necessidade de programação explícita.

Algoritmos de aprendizado de máquina ajustam os seus parâmetros de acordo com os dados de treinamento fornecidos. Durante o treino, o algoritmo, procura se ajustar de forma a generalizar e conseguir fazer a tomada de decisão com base nos padrões identificados nos dados. A quantidade de dados disponibilizados na etapa de treino impacta positivamente na qualidade do aprendizado e na precisão ao ser apresentado a novos dados.

Existem três abordagens de aprendizado de máquina, sendo elas, o aprendizado supervisionado, no qual o algoritmo é treinado com exemplos de entradas e saídas, podendo fazer regressões ou classificações; o aprendizado não supervisionado, onde o algoritmo busca identificar padrões ou estruturas ocultas nos dados sem necessitar de rótulos; e por fim, o aprendizado por reforço, nessa abordagem, o algoritmo é exposto ao ambiente, onde ele tem capacidade de agir e deve aprender como reagir a ele com base nas suas observações por tentativa e erro e experiências que vai acumulando com base nas suas ações anteriores e o resultado que elas trouxeram.

O aprendizado de máquina tem sido amplamente aplicado nas mais diversas áreas, como reconhecimento de padrões e processamento de linguagem natural, visão computacional, análise de dados, entre outros. E são aplicados em diversas gamas de aplicações; sistemas de recomendações, análise de fraudes, segurança e em SLR.

Para que possa ser devidamente implementado o aprendizado de máquina, faz-se necessário um conjunto de etapas, para que se obtenham resultados adequados, que partem da coleta, seleção e preparação dos dados, a escolha do modelo mais indicado para os dados que pretende utilizar, tendo também como parâmetro para a seleção, a quantidade de dados e a complexidade do modelo esperado, o treinamento, validação,

[1] A inteligência artificial é uma ciência que busca criar algoritmos que consigam aprender sozinhos, sem a necessidade de serem propriamente programados para executar a tarefa que se destinam

ajuste dos hiperparâmetros e avaliação do modelo.

Também é importante ter uma boa compreensão dos conceitos estatísticos e matemáticos que envolvem o aprendizado de máquina, bem como habilidade em programação e manipulação de dados e ter uma compreensão dos dados que estão sendo utilizados, esses fatores podem ajudar no direcionamento do modelo. O aprendizado de máquina é um campo em constante evolução e vem mostrando resultados cada vez mais impactantes, tendo o potencial de transformar diversas áreas da sociedade e impulsionar avanços significativos (ZHOU, 2021).

2.1.1 Aprendizado profundo

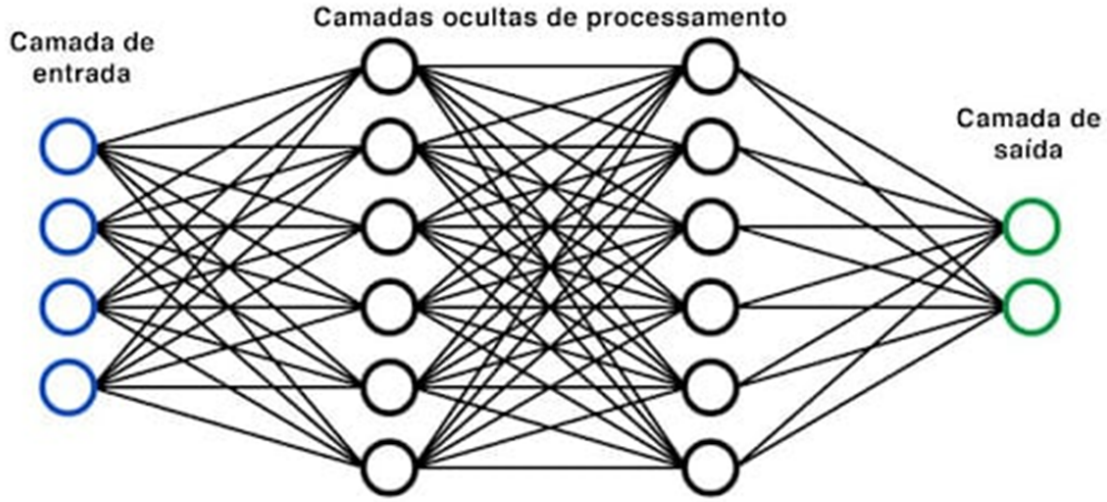


Figura 2 – Representação do funcionamento de uma rede neural.
Fonte: (COSTA; SENEFONTE, s.d.).

O aprendizado profundo (comumente chamado também de *"deep learning"*), é um subcampo do aprendizado de máquina, que se baseia em Redes Neurais para realizar análise e processamento de dados. As Redes Neurais são compostas por várias unidades (Também conhecidas como neurônios) de processamento interconectadas (conhecidas como MLP), sendo as unidades representadas pelo círculos e as conexões pelos traços como observado na Figura 2. Esse modelo multi-camadas permite que o sistema aprenda

representações complexas dos dados de entrada.

As unidades são os blocos que constroem as Redes Neurais. Cada unidade recebe um conjunto de entradas, realiza uma operação matemática e então produz uma saída, e essa saída é propagada para as camadas seguintes da rede neural. As unidades podem ter diferentes arquiteturas e funções de ativação. Sendo a função de ativação, um cálculo matemático aplicado às saídas das unidades. A função de ativação, deve ser uma função não linear, para introduzir a não linearidade ao modelo, permitindo que a rede modele padrões complexos.

Nesse projeto, foram aplicadas diversas funções de ativação, sendo elas:

1. ***Rectified Linear Unit (ReLu)***, uma das funções mais amplamente utilizadas em Redes Neurais. Ela se caracteriza pela Equação 1

$$ReLu(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}. \tag{1}$$

   É uma função simples, eficiente computacionalmente e resolve alguns problemas comuns nas Redes Neurais, no entanto pode gerar um problema chamado de neurônio “morto”, quando um neurônio sempre retorna zero, tendo praticamente nenhuma utilidade no modelo.

2. ***Exponential Linear Unit (ELU)*** é uma variação da ReLu que permite valores negativos, se caracterizando pela Equação 2

$$ELU(x) = \begin{cases} \alpha \times (e^x - 1), x < 0 \\ x, x \geq 0 \end{cases}, \tag{2}$$

   sendo α um parâmetro obrigatório da função. A ELU é uma função suave em todo o seu domínio, o que pode ajudar no treinamento das Redes Neurais. Além disso, ela evita o problema dos neurônios “mortos” e, com isso, pode em alguns casos trazer resultados melhores que a ReLu.

3. ***Leaky ReLu*** é outra variação da função ReLu, que se caracteriza pela Equação 3

$$LeakyReLu(x) = \begin{cases} \alpha \times x, x < 0 \\ x, x \geq 0 \end{cases}, \tag{3}$$

sendo α um parâmetro obrigatório da função, geralmente assumindo valores no intervalo $0 \leq \alpha \ll 1$, sendo o caso $\alpha = 0$ um caso particular onde a LeakyRelu se iguala à ReLu. A introdução desse coeficiente angular α, gera uma pequena inclinação para valores negativos, o que também ajuda a resolver o problema dos neurônios "mortos" e em alguns casos pode trazer melhores resultados que a ReLu.

4. ***Scaled Exponential Linear Unit (SELU)*** é uma variação da função ELU auto-normalizada que visa manter a média e a variância aproximadamente constantes, se caracterizando pela Equação 4

$$SELU(x) = \begin{cases} \alpha \times scale(e^x - 1), x < 0 \\ scale \times x, x \geq 0 \end{cases}, \tag{4}$$

sendo α e $scale$ constantes com valores $\alpha \approx 1.67326324$ e $scale \approx 1.05070098$. A SELU é projetada para redes com muitas camadas, aplicando uma escala e deslocamento aos valores de entrada, de modo que a média e variância seja preservada em cada camada, estabilizando o treinamento e melhorando o desempenho da rede neural.

5. **Sigmoid**, também conhecida com função logística, é caracterizada pela Equação 5

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

mapeando cada valor de x para uma saída no intervalo $\{0 \leq sigmoid(x) \leq 1 | x \in \mathbb{R}\}$, é utilizada principalmente para saída das Redes Neurais em problemas de classificação. Ela possui uma saída no formato de "S" e é diferenciável em todo o seu domínio, o que permite técnicas de otimização baseadas em gradiente, no entanto ela pode ter problema de saturação em regiões distantes do zero.

6. **Softmax** é utilizada, principalmente, em saídas de Redes Neurais em problemas de classificação, onde a saída representa a probabilidade de pertencer a uma classe, tendo como particularidade essa função o fato dela ser aplicada a todo o vetor de saída da camada, e não só à saída do neurônio. Ela é definida pela Equação 6

$$softmax[X] = \left\{ \frac{e^{X_n}}{\sum_{i=1}^{i \leq N} e^{X_i}} \mid n \in \mathbb{N}, n \in [1\ N] \right\}, \tag{6}$$

sendo X_n o n-ésimo valor do vetor X e N o tamanho vetor X. Ela normaliza o valor das saídas de forma que $\sum softmax[X] = 1$, para qualquer $X \in \mathbb{R}_n$, e por isso é muito utilizada em problemas de classificação multi-classe.

Ao construir uma rede neural, se tem diversas camadas de neurônios, sendo a primeira, onde se recebem as entradas da rede, designada como a camada de entrada, sendo representada pelos círculos azuis na Figura 2, e a última, onde se entregam o resultado da rede, a camada de saída, sendo representada pelos círculos verdes, e entre essas duas camadas, possuem diversas camadas, designadas camadas ocultas, representadas pelos círculos pretos. As camadas ocultas são camadas intermediárias, chamadas assim por não terem uma ligação direta com a entrada e saída. Elas são responsáveis por generalizar os dados e extrair padrões a medida que o dado percorre as camadas. A quantidade de camadas ocultas e o tamanho delas variam de acordo com o problema.

Os otimizadores são algoritmos que durante o treinamento da rede, ajustam os pesos e vieses das unidades, minimizando a função perda. Eles possuem um papel fundamental na atualização dos parâmetros da rede, permitindo que ela aprenda de fato com o treinamento. Existem diversos otimizadores, e nesse projeto foram utilizados os seguintes:

1. ***Adaptive Moment Estimation (Adam)*** se baseia em outros otimizadores, utilizando uma estimativa adaptativa dos momentos de primeira ordem (a média móvel dos gradientes) e os de segunda ordem (média móvel dos gradientes ao quadrado) para atualizar os pesos da rede neural. Ele se baseia no Algoritmo 1.

   Onde os parâmetros β_1 e β_2 controlam as taxas de decaimento desses momentos. Além disso, ele possui um parâmetro ϵ que serve para evitar divisões por zero. Ele também possui uma opção chamada *amsgrad*, que é uma variação que modifica o cálculo dos momentos de segunda ordem para evitar degradações de desempenho (DIEDERIK; B., 2014).

2. ***Stochastic Gradient Descent (SGD)*** é uma abordagem básica, se assemelhando ao método de Newton-Raphson, mas voltado para vetores. O SGD originalmente é definido pela Equação 7 onde θ representa os pesos, α, a taxa de aprendizado, e, L, a função perda.

Algoritmo 1 – Algoritmo Adam

Input: α: Taxa de aprendizado
$\beta_1, \beta_2 \in [0, 1)$: Taxas de decaimento exponencial para estimativa dos momentos
$f(\theta)$: Função estocástica objetivo com parâmetro θ
Output: θ_t: Parâmetros resultantes

begin
  $m_0 \leftarrow 0$ (Inicializa o vetor do primeiro momento)
  $v_0 \leftarrow 0$ (Inicializa o vetor do segundo momento)
  $t \leftarrow 0$ (Inicializa o *timestamp*)
  while θ_t não convergir **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_t - 1)$ (Gradiente da função objetivo)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_i). \tag{7}$$

Ele atualiza os pesos usando o gradiente negativo da função perda. Além disso, ele tem o parâmetro momento, que adiciona uma fração do gradiente anterior ao atual, o que pode ajudar a acelerar o processo de otimização, definido pelas Equações 8, 9 onde m é o momento.

$$update_{t+1} = m_t * update_t - \epsilon \nabla f(\theta_t) \tag{8}$$

$$\theta_{t+1} = \theta_t + update_{t+1}. \tag{9}$$

Também é possível utilizar a correção de momento de Nesterov que pode melhorar ainda mais a convergência, sendo definida pelas Equações 10, 11 (SUTSKEVER et al., 2013).

$$update_{t+1} = m_t * update_t - \epsilon \nabla f(\theta_t + m_t * update_t) \tag{10}$$

$$\theta_{t+1} = \theta_t + update_{t+1}. \tag{11}$$

3. ***Root Mean Square Propagation (RMSprop)*** também utiliza uma estimativa adaptativa dos momentos de primeira ordem, mas se difere do Adam na forma de calcular e atualizar esses momentos. Ele possui um parâmetro ρ, que controla a taxa de decaimento do momento de primeira ordem. Possui também um parâmetro momento, que funciona como no SGD, adicionando frações do momento anterior ao atual. Uma forma comum de se utilizar o RMSprop é centralizando os gradientes subtraindo a média geral do gradiente, o que, na maioria dos casos, melhora o desempenho e a estabilidade da rede neural, resolvendo problemas de convergência e acelerando o processo de otimização. O otimizador pode ser definido pelas Equações 12, 13.

$$E[g^2]_t = \rho * E[g^2]_{t-1} + (1-\rho)\nabla L(\theta_{t-1})^2, \tag{12}$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{E[g^2]}}\nabla L(\theta_{t-1}). \tag{13}$$

Além dos otimizadores, existem alguns parâmetros utilizados por eles que merecem destaque, que foram utilizados nesse projeto, sendo eles (ZHOU, 2021):

1. **Taxa de aprendizado** é um parâmetro fundamental. Ele determina o tamanho dos passos de atualização dos pesos durante o treinamento. Uma taxa muito alta, resulta em atualizações mais rápidas, mas gera oscilações e até divergências. Já uma taxa pequena pode prolongar muito o tempo de convergência. É importante uma taxa de aprendizado adequada para obter bom desempenho na rede, ou até técnicas que reduzam a taxa de aprendizado no decorrer do tempo, que foram utilizadas no presente projeto.

2. **Decaimento do peso**, também conhecido como “regularização L2”, é uma técnica utilizada para auxiliar na generalização das Redes Neurais, evitando que decore as entradas e saídas. Adicionando uma penalidade proporcional ao quadrado dos pesos durante o processo de atualização, incentivando os pesos a permanecerem pequenos e evitando coeficientes excessivamente grandes, o que geralmente caracteriza uma falta de generalização. O Decaimento do peso pode ser definido pela equação 14.

$$L_{l2} = l2 * \sum \theta^2. \tag{14}$$

3. **Média Móvel Exponencial (EMA)** é uma técnica que suaviza os valores ao longo do tempo, atribuindo maior relevância aos valores recentes. Costuma ser utilizada em otimizadores como uma forma de acompanhar tendências dos parâmetros de forma mais suave. O momento do EMA controla a taxa de decaimento dos valores anteriores na média móvel exponencial e a frequência de sobre-escrita determina com que frequência os valores da EMA são atualizados durante o treino.

4. ***Clip*** é um conjunto de parâmetros, que agrega técnicas de controle de gradiente, limitando o tamanho dos gradientes durante o processo de atualização dos pesos. O *clipnorm* define um limite para a norma do gradiente, enquanto o *clipvalue* define um limite absoluto para os valores do gradiente. Essas técnicas são úteis para evitar o desvanecimento do gradiente e podem ajudar a estabilizar o treinamento da rede neural.

O tamanho do lote é um parâmetro importante no aprendizado profundo, visto que ele pode afetar tanto a eficiência computacional quanto a acurácia do modelo. Tamanhos de lotes menores levam a mais atualizações dos pesos, o que pode contribuir para uma convergência mais acelerada, principalmente em grandes conjuntos de dados. Todavia tamanhos de lotes excessivamente pequenos contribuem negativamente para a variabilidade dos gradientes, o que dificulta a convergência e pode reduzir a acurácia.

Lotes grandes podem fornecer estimativas mais precisas do gradiente, pois têm mais amostras para calcular a média. Todavia, para ter grandes lotes é necessário mais memória e poder computacional, o que pode ser proibitivo dependendo do problema, então é necessário achar um equilíbrio entre eficiência e acurácia ao definir o tamanho do lote.

Outro parâmetro muito importante são as épocas, que permitem que o modelo seja exposto diversas vezes a todo o conjunto de treinamento, permitindo que ele ajuste os pesos e minimize a fórmula de perda. O número de épocas pode influenciar na capacidade do modelo generalizar os padrões quando escolhido adequadamente, mas quando mal dimensionado, pode impactar tanto em um subajuste, quando o modelo não aprende nada, como no sobreajuste, quando o modelo para de aprender e começa a decorar os dados.

No PDI, o aprendizado profundo vem se mostrando extremamente eficaz, permitindo a extração automática de características e o reconhecimento de padrões complexos

em imagens. As Redes Neurais Profundas podem aprender a identificar objetos, faces e outros elementos em imagens, com precisão inclusive superior a média humana (SZEGEDY et al., 2015; RUSSAKOVSKY et al., 2014).

Redes Neurais Rasas

Dentre os diversos tipos de Redes Neurais que existem, uma em especial merece destaque nesse trabalho, as Redes Neurais Rasas. Elas consistem em apenas uma camada de neurônios artificiais conectados diretamente a uma camada de entrada. Esses neurônios na camada de saída são independentes e não há conexões entre neurônios na mesma camada.

Na rede neural rasa, os neurônios na camada de saída realizam a operação de combinação linear dos valores de entrada ponderada com seus respectivos pesos, seguidos então por uma função de ativação não linear. A função de ativação é o que permite que a rede neural aprenda a representar relações não lineares nos dados de entrada.

As Redes Neurais Rasas possuem a limitação de ter uma representação restrita. Por sua estrutura simples, elas são eficazes em problemas com padrões mais lineares, onde as fronteiras de decisões podem ser representadas por linhas retas. Todavia, para problemas mais complexos que envolvem relacionamentos não lineares, as Redes Neurais Rasas podem ter dificuldade em aprender padrões e obter resultados precisos.

Devido às razões apresentadas, Redes Neurais Rasas são mais comuns em problemas de classificação binária simples ou como uma primeira tentativa para entender os dados antes de adotar redes mais profundas e complexas. Apesar de serem limitadas, as Redes Neurais Rasas têm valor como ferramenta de aprendizado de máquina e podem fornecer informações úteis em certos cenários (ZHOU, 2021).

2.1.2 K-Means

O algoritmo *K-Means* é um algoritmo de aprendizado de máquina não supervisionado, baseado em clusterização [2] amplamente utilizado no campo da análise de dados. O algoritmo faz o agrupamento dos dados em k clusters, onde k é um hiperparâmetro. Ele se baseia no princípio de minimização da distância entre os dados e os centroides dos clusters correspondentes. Geralmente, a distância utilizada é a euclidiana.

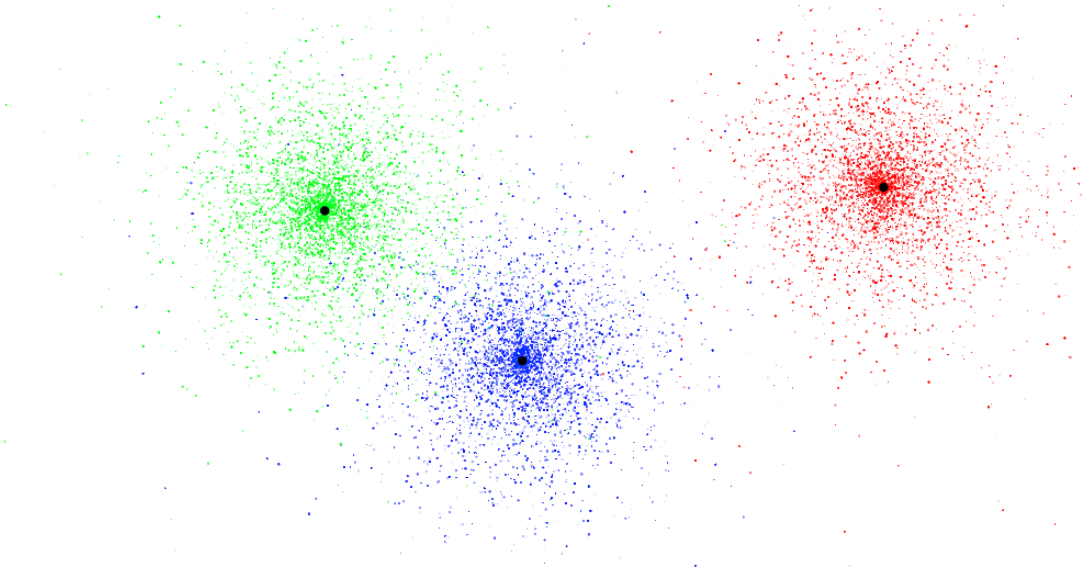


Figura 3 – Representação do funcionamento do *K-Means*.
Fonte: Elaboração Própria.

O funcionamento do *K-Means* é representado na Figura 3, nela podemos ver 3 clusteres formados, representados pelas cores verde, azul e amarelo. Podemos ver também o centroide de cada um dos clusters, representados pelos pontos pretos, que são o resultado da média das coordenadas de cada um dos elementos do cluster.

O *K-Means* inicialmente seleciona aleatoriamente k centroides, que representam os centros iniciais de cada cluster. Feito isso, ele atribui os dados aos clusters, de forma que o cluster a que o dado pertence, é o que tem o centroide mais próximo a ele.

Após ter feito a atribuição inicial, o algoritmo iterativamente recalcula os centroides dos clusters, calculando a média dos elementos de cada clusters, e verificando se o

[2] Um cluster é um grupo de objetos ou dados similares que são agrupados em conjuntos com base em suas características semelhantes, e a clusterização é o processo de agrupar esses objetos ou dados de acordo com as suas similaridades.

centroide mais próximo deles era o que havia sido anteriormente definido, ou se deve ser efetuada uma troca. Esse processo é executado até que ocorra convergência, ou seja, até que não ocorra mais troca de clusters entre os dados.

O algoritmo *K-Means* é conhecido por sua eficiência, baixo custo computacional e simplicidade, tornando uma escolha popular para problemas de clusterização. No entanto, é importante ressaltar que para que ele tenha um bom resultado é fundamental escolher um valor k adequado, além disso, o algoritmo também depende da escolha inicial de centroides. Ele também pode ter dificuldade para lidar com dados de alta dimensionalidade ou com dados muito dispersos.

As aplicações do *K-Means* são diversas, desde análise de mercado, processamento de imagens, bioinformática e muitas outras (PRADANA; HA, 2021; SURABHI et al., 2012). Devido a sua capacidade de agrupar dados de forma automática e não supervisionada, permite a identificação de estruturas intrínsecas nos dados, auxiliando na compreensão e organização das informações, podendo muitas vezes ser utilizado até em uma pré-análise.

Uma outra aplicação do *K-Means* é a redução de parâmetros em conjuntos de dados com alta dimensionalidade (FENG et al., 2020). Agrupando os dados em k grupos, representados pelos seus centroides, de forma que cada grupo é representado por uma média das observações contidas neles. A redução de parâmetros ocorre ao substituirmos as observações individuais pelos centroides correspondentes, reduzindo e tornando constante a dimensão do conjunto de dados para k.

Essa redução pode ser útil em diversas situações. Por exemplo, em problemas de classificação, a redução de dimensionalidade pode ajudar a mitigar o problema da "maldição da dimensionalidade", que faz com que a eficácia de grande parte dos algoritmos de aprendizado de máquina diminua com o aumento expressivo de características e reduzindo a sua capacidade de generalizar. Essa redução também aumenta a simplicidade da representação dos dados, tornando mais inteligíveis e possivelmente facilitando a visualização.

A abordagem de utilizar o *K-Means* para realizar a redução de dimensionalidade antes de aplicar outros modelos de aprendizado de máquina é comumente utilizado para aumentar a eficiência computacional, remover ruídos e características irrelevantes dos dados, focando nas informações mais significativas para a tarefa. Também pode ser aplicado em dados com dimensões variáveis, forçando com que o dado tenha uma

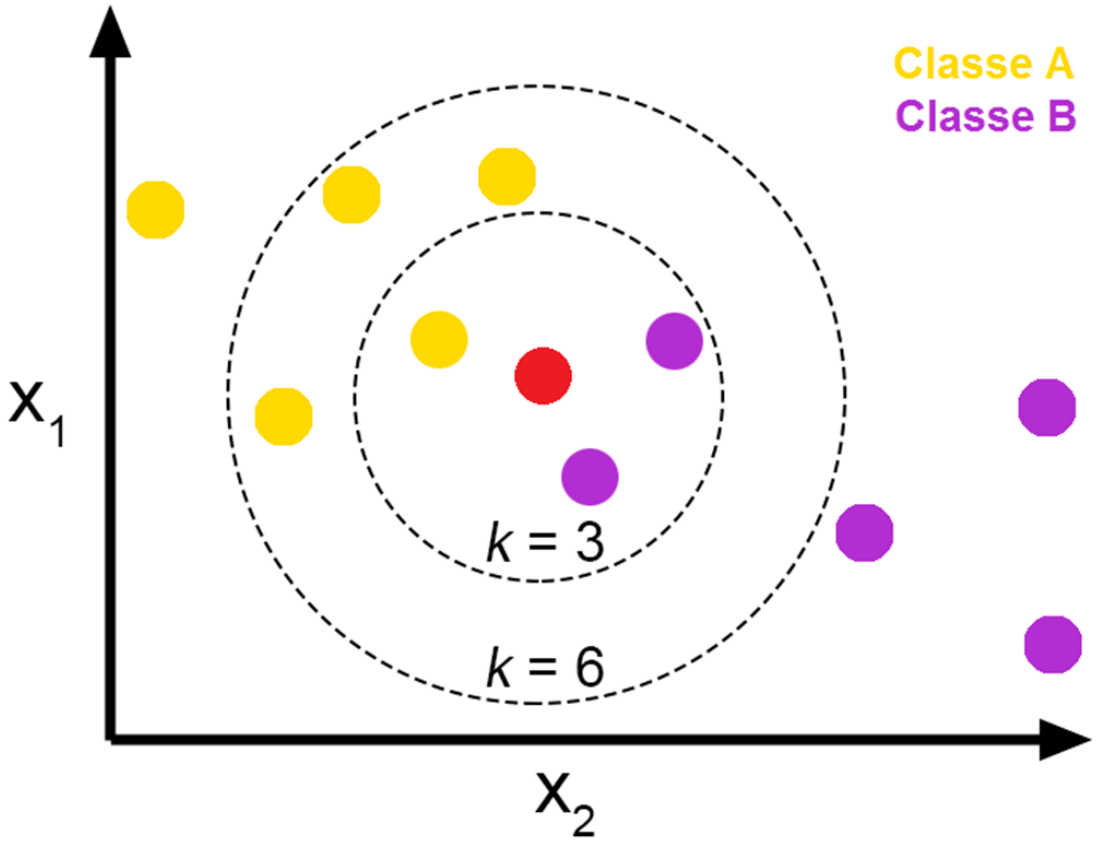


Figura 4 – Representação do funcionamento do KNN.
Fonte: (CORREA et al., 2023).

quantidade fixa k de dimensões (ZHOU, 2021).

2.1.3 KNN

O KNN é um modelo de aprendizado supervisionado utilizado para classificar observações com base na proximidade dos seus vizinhos mais próximos no conjunto de treinamento, sendo assim considerado um modelo “preguiçoso”, pois na fase de treinamento ele só copia os dados de exemplo, tendo ação somente na classificação, onde ele busca proximidades com os dados copiados (ZHOU, 2021).

Ele funciona calculando a distância entre os exemplos, representados na Figura 4 pelos círculos amarelos e violeta, e a observação atual, representada pelo círculo vermelho, e seleciona os k vizinhos mais próximos, sendo k um parâmetro do modelo. No funcionamento padrão, a classe que a maioria dos k vizinhos mais próximos pertencerem

é atribuída à observação. Além disso, como podemos ver na figura de exemplo, a escolha do valor de k tem impacto direto no resultado da classificação, quando selecionamos $k = 3$ no exemplo, temos como resultado Classe B, e quando $k = 6$, temos Classe A, por isso é importante utilizar algum algoritmo otimizador para poder fazer a escolha do melhor valor de k para cada caso.

O parâmetro k define o número de vizinhos que serão considerados para a classificação. Valores maiores de k tendem a tornar a classificação mais suave, enquanto k menores tendem a tornar a classificação mais sensível a pequenas variações e ruídos nos dados. Todavia, em espaços com alta dimensionalidade, valores de k mais baixos tendem a ser mais utilizados, pois o aumento da dimensionalidade, tende a uma maior dispersão nos dados.

Além do funcionamento padrão, o KNN também pode ponderar os vizinhos com base na sua distância. Nessa forma de funcionamento, os vizinhos mais próximos têm um peso maior na classificação das observações. Essa é uma forma de funcionamento interessante quando a distância é relevante para a classificação.

Em termos de desempenho, o treinamento o KNN é extremamente rápido, haja vista, que, como dito anteriormente, consiste numa simples cópia de dados, mas ao classificar pode ser computacionalmente intensivo para conjuntos de dados grandes, visto que ele é um modelo considerado “preguiçoso” (ZHANG; ZHOU, 2007), pois ele requer o cálculo de distâncias entre todos os exemplos de treinamento com a observação. No entanto, ao utilizar um índice espacial, como a árvore *KD-tree*, é possível acelerar consideravelmente a velocidade de busca pelos vizinhos mais próximos e melhorar a eficiência do algoritmo, deixando de ser um custo $O(n^2)$ e se tornando um custo $O(n \times log(n))$, tendo como penalidade a redução da velocidade de treino, não sendo mais uma simples cópia, mas também efetuando uma indexação (HOU et al., 2018).

Comparado a outros modelos de aprendizado de máquina, o KNN tem a vantagem de ser um modelo não paramétrico [3] e não fazer suposições sobre a distribuição dos dados, no entanto, ele pode ser afetado por ruídos e dados discrepantes, além de ter dificuldade para generalizar em dados com alta dimensionalidade. É importante que seja feita uma etapa de ajuste nos dados, contando com estratégias para lidar com valores discrepantes e redução de dimensionalidade, além de um ajuste fino dos parâmetros do KNN para obter os melhores resultados (ZHOU, 2021).

[3] Uma técnica não paramétrica não faz suposições estatística sobre a distribuição dos dados e vai ajustando o modelo conforme eles são apresentados.

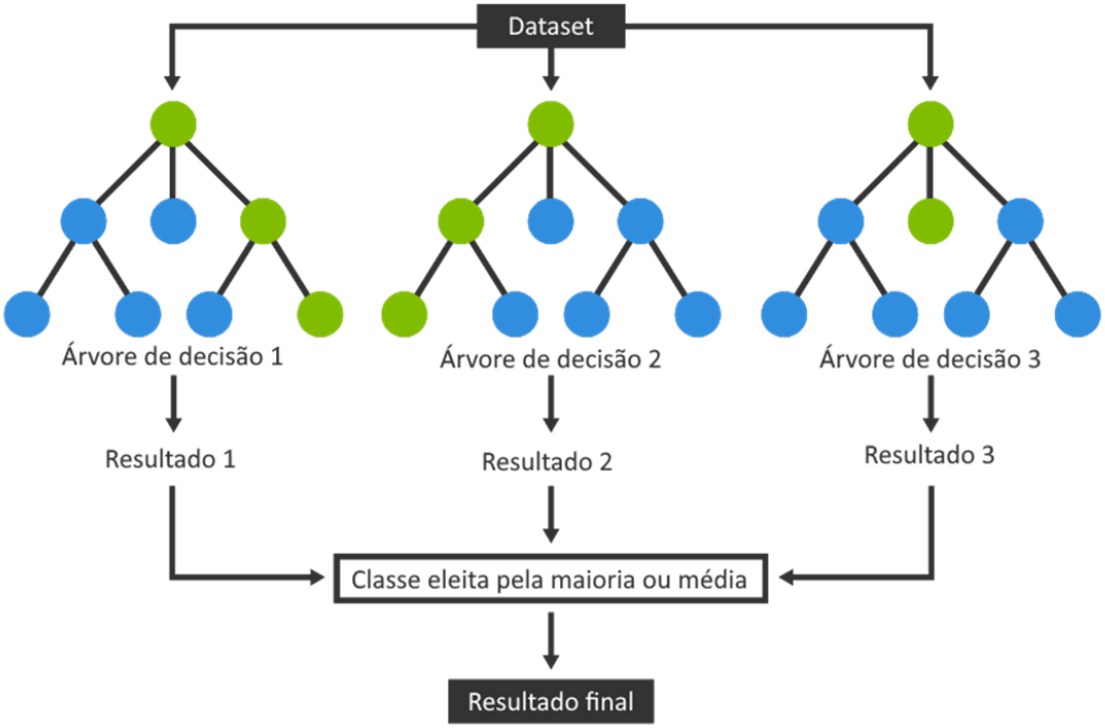


Figura 5 – Representação do funcionamento do Random Forest.
Fonte: Adaptado pelo autor.

2.1.4 Random Forest

O *Random Forest* é um algoritmo de aprendizado de máquina que se baseia em conjuntos, combinando a previsão de vários classificadores individuais, conhecidos como árvores de decisão, para realizar a classificação ou regressão. Quando se fala de classificação, cada árvore de decisão na floresta, faz uma votação para determinar a classe final da observação, e então o *Random Forest* combina todas as previsões de todas as árvores e seleciona a classe com mais votos sendo essa a predição final (ZHOU, 2021). Tal comportamento pode ser visto na Figura 5 onde uma observação passa por 3 árvores distintas, cada uma produzindo um resultado diferente, e o resultado final é definido com base na classe atribuída pela maioria das árvores.

Uma característica importante do *Random Forest* é a sua capacidade de usar *bootstrap*, uma técnica de amostragem não paramétrica com reposição para construir

cada árvore de decisão. Implicando em amostrar aleatoriamente um subconjunto dos dados de treinamento com reposição, fazendo com que cada árvore seja treinada com diferentes subconjuntos, mas possivelmente possuindo interseções entre eles. Essa abordagem, aumenta a diversidade de árvores, tornando o modelo mais robusto e menos suscetível ao sobreajuste.

Também podemos controlar a profundidade máxima, que controla o nível de complexidade das árvores de decisão no *Random Forest*. Ao definir uma profundidade máxima, limitamos o número de divisões e impedimos que tenham árvores demasiadamente profundas, evitando sobreajuste. Um ponto a se destacar é que é possível definir uma profundidade infinita, mas isso pode levar ao sobreajuste, visto que as árvores podem se ajustar perfeitamente aos dados de treinamento. No caso de profundidade infinita, as árvores se expandem até que todas as folhas só possuam amostras de uma única classe.

Outro parâmetro importante é o número máximo de características, que define a quantidade máxima de características que são consideradas para cada divisão na árvore. Ele é importante para controlar a diversidade de árvores, pois limita a quantidade disponível para cada árvore. O estudo do valor desse parâmetro é fundamental, pois ele reduz a correlação entre as árvores e intra-árvore, o que melhora o desempenho do modelo.

Os parâmetros de amostras mínimas por folhas e por divisão definem o número mínimo de amostras necessárias para formar uma folha (o nó final do modelo) ou uma divisão, respectivamente. Esses parâmetros evitam divisões em nós com poucas amostras, o que pode levar a um sobreajuste.

Também possui o parâmetro que define o número de classificadores, que define quantas árvores de decisão vão ser construídas na floresta. Geralmente, quanto mais árvores, maior a capacidade de generalização, mas também aumenta o custo computacional. Esse número pode ser influenciado pela quantidade de dados, complexidade do problema e recursos computacionais disponíveis.

Um ponto de destaque é que o *Random Forest* é capaz de precisar o quanto uma determinada característica é importante para a classificação. Ao selecionar aleatoriamente os subconjuntos de características, ele acaba reduzindo a influência de características irrelevantes, o que o torna capaz de lidar com problemas de alta dimensionalidade.

Em suma, o *Random Forest* é um poderoso algoritmo de aprendizado de máquina,

capaz de realizar tanto classificação, quanto regressão. Ele combina múltiplas árvores de decisão, podendo ou não aplicar *bootstrap* para as construir e fazendo votações para a classificação final, o que além de prever a classe, pode dar a probabilidade de pertencimento da observação a ela (ZHOU, 2021).

2.1.5 Naïve Bayes

Naïve Bayes é um algoritmo de aprendizado de máquina supervisionado, que se utiliza de classificação probabilística baseada no Teorema de Bayes (ZHOU, 2021). Ele assume uma independência condicional entre os atributos, o que geralmente não é verdade, isso significa que cada atributo é considerado independente dos outros, daí se dá o uso da expressão “Naïve” (que significa ingênuo). Essa suposição permite uma simplificação e uma implementação eficiente e rápida do algoritmo.

O algoritmo é ideal para lidar com fluxos de dados contínuos e em tempo real. Ele atualiza e ajusta suas probabilidades a cada nova instância de treinamento, adaptando-se de forma incremental às novas mudanças no fluxo de dados. Isso torna o Naïve Bayes uma escolha adequada em cenários onde os dados estão em constante evolução.

O teorema de Bayes utiliza duas fórmulas para efetuar as previsões, sendo elas o Teorema de Bayes, definido pela Equação 15.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}. \tag{15}$$

E a fórmula de probabilidade condicional definida pela Equação 16.

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \Rightarrow P(A \cap B) = P(A) \times P(B|A). \tag{16}$$

O algoritmo é feito de forma que o Teorema de Bayes calcula a probabilidade de uma classe dado um conjunto de atributos, enquanto a probabilidade condicional é usada para estimar a probabilidade de um atributo pertencer a uma determinada classe.

O Naïve Bayes também pode ser utilizado como um componente de técnicas de otimização, entre elas a otimização de hiperparâmetros de outros modelos de aprendizado de máquina. Nesse caso, o Naïve Bayes é denominado Bayes *Search* e é utilizado para

estimar a probabilidade de acurácia para diferentes configurações de hiperparâmetros, permitindo que a busca pelos melhores hiperparâmetros seja mais eficiente e possa ser feita em espaços contínuos (ZHOU, 2021).

2.2- MediaPipe

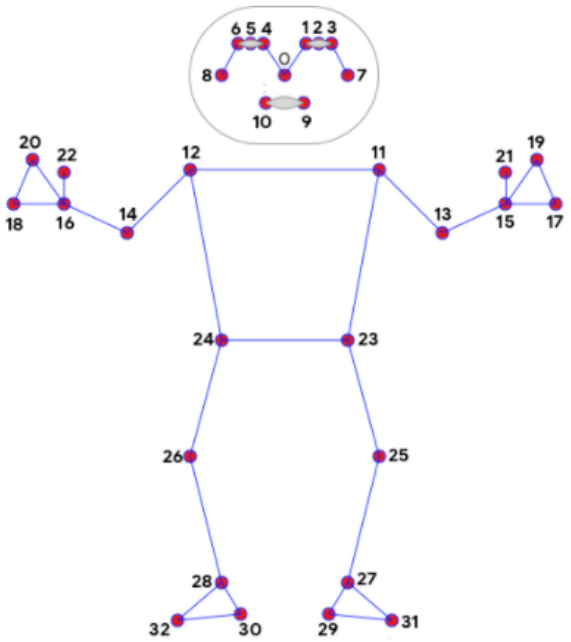
MediaPipe é um *framework* desenvolvido pela *Google Inc.* que traz uma estrutura flexível, eficiente, de baixo custo computacional e de código aberto para o desenvolvimento de aplicações multimídia, multiplataforma e em tempo real (LUGARESI et al., 2019). Ele foi projetado para simplificar a implementação de *pipelines* de processamento de dados multimídia, podendo fazer desde o processamento de imagens e vídeos, até a análise de movimento e reconhecimento de gestos.

Um dos principais módulos do *MediaPipe* é o de pose, que tem como objetivo estimar no espaço tridimensional a posição e orientação das articulações humanas em imagens, vídeos, inclusive em tempo real, utilizando um dispositivo de captura de vídeo simples (como uma *webcam* ou a câmera de um celular). Esse módulo utiliza técnicas de visão computacional e aprendizado de máquina para identificar e rastrear as poses humanas, permitindo a detecção de movimentos e gestos.

Esse módulo utiliza um modelo de aprendizado profundo para realizar a estimativa das poses, treinado usando grandes bases de dados com imagens rotuladas com as posições das articulações humanas e tendo imagens nas mais diversas condições de iluminação, pontos de vista, rotações, entre outros aspectos, a fim de tornar o modelo o mais robusto possível a esses fatores. Os pontos-chave, bem como seu índice no vetor gerado pela biblioteca podem ser conferidos na Figura 6.

O módulo de pose é capaz de lidar também com poses múltiplas e complexas, podendo detectar e rastrear várias pessoas na mesma imagem ou vídeo, sendo assim, capaz de estimar as poses individualmente ou pode ser utilizado para detectar interações entre várias pessoas. Outra finalidade é permitir que na cena possa ter mais de uma pessoa além da que seja a que estejamos interessados em identificar o gesto sem interferir no resultado do *framework*.

Além de poder estimar as poses, o módulo consegue fornecer outras informações



0. Nariz
1. Parte interna do olho esquedo
2. Olho esquerdo
3. Parte externa do olho esquerdo
4. Parte interna do olho direito
5. Olho direito
6. Parte externa do olho direito
7. Orelha esquerda
8. Orelha direita
9. Canto esquerdo da boca
10. Canto direito da boca
11. Ombro esquedo
12 Ombro direito
13. Cotovelo esquerdo
14. Cotovelo direito
15. Pulso esquerdo
16. Pulso direito
17. Mindinho esquerdo
18. Mindinho direito
19. Indicador esquerdo
20. Indicador direito
21. Polegar esquerdo
22. Polegar direito
23. Canto esquerdo da cintura
24. Canto direito da cintura
25. Joelho esquerdo
26. Joelho direito
27. Tornozelo esquerdo
28. Tornozelo direito
29. Calcanhar esquerdo
30. Calcanhar direito
31. Indicador do pé esquerdo
32. Indicador do pé direito

Figura 6 – Ilustração dos pontos-chaves detectados pelo *MediaPipe*.
Fonte: (LUGARESI et al., 2019)

adicionais, a orientação das articulações, verificar variações na pose independentemente do ambiente ao redor, a visibilidade da articulação e confiabilidade das estimativas, podendo inclusive estimar a posição de articulações que não estejam visíveis na cena com base nas disponíveis. Todo esse conjunto de informações pode ser utilizado para que sejam feitas análises avançadas e conseguir fazer classificações e tomada de decisões com base nas poses estimadas.

Apesar de robusto e tendo resultados próximos ao estado da arte, o *MediaPipe* apresenta uma Interface de programação de aplicações (API) simples e intuitiva, desenvolvida em diversas linguagens e plataformas e um baixo custo computacional. Isso permite que os desenvolvedores integrem facilmente o *framework* em suas aplicações, sem a necessidade de conhecimentos avançados em PDI ou aprendizado de máquina (LUGARESI et al., 2019).

2.3- Scikit-Learn

Scikit-Learn, popularmente chamado também de sklearn, é uma biblioteca de aprendizado de máquina em *Python*, que oferece uma gama de algoritmos para a construção, treinamento, avaliação, otimização e pré-processamento de aprendizado de máquina (PEDREGOSA et al., 2011). Ela foi planejada para ser simples e eficiente, fornecendo uma interface consistente para a aplicação de técnicas de aprendizado de máquina em problemas do mundo real. Ela é amplamente utilizada pela sua documentação abrangente, por possuir uma comunidade ativa de desenvolvedores e ter integração com diversas outras bibliotecas que possuem esses mesmos aspectos, como Pandas e NumPy (BUITINCK et al., 2013).

Uma das principais vantagens do *Scikit-Learn* é a sua simplicidade, praticidade e a sua usabilidade. A biblioteca é desenhada para ser fácil de usar, necessitando poucas linhas de código para construir complexos modelos de aprendizado de máquina, com diversas camadas de pré-processamento e múltiplas métricas de validação.

O *Scikit-Learn* possui uma grande quantidade de algoritmos de aprendizado de máquina prontos para uso de forma simples. Tendo suporte para algoritmos de classificação, regressão, agrupamento, redução de dimensionalidade, seleção de características e muitos outros. Também possui uma grande gama de ferramentas de pré-processamento de dados, como normalização, padronização, imputação e outros. Também possui recursos para validação cruzada, métricas de avaliação e seleção de modelos (PEDREGOSA et al., 2011).

2.3.1 API Scikit-Learn

O *Scikit-Learn* é conhecido por sua API bem projetada, oferecendo uma interface consistente e intuitiva para a construção e utilização de modelos de aprendizado de máquina, e é importante uma boa interface de API para a usabilidade e extensibilidade de uma biblioteca de aprendizado de máquina, principalmente considerando que o *Scikit-Learn* foi planejado com princípios em sua simplicidade, coerência e flexibilidade.

O *Scikit-Learn* permite que os usuários criem pipelines de transformação de dados, realizem o treinamento de modelos complexos e façam previsões com novos dados de forma clara e concisa. Além disso, a API facilita a experimentação com diferentes algoritmos, tornando o processo e a seleção de modelos mais eficientes.

O projeto da API do *Scikit-Learn* foi influenciado por várias diretrizes, incluindo a preferência por interfaces simples de programação, a utilização do paradigma de Programação Orientada a Objetos (POO) para fazer a organização das funcionalidades da biblioteca e um uso consistente de nomenclatura e convenções. Por fim, o Scikit-Learn possui um conceito de “estimadores”, que são objetos que podem ser ajustados aos dados de treino e usados para efetuar previsões. Com essa abordagem de projeto modular e coesa, torna-se possível que os usuários criem seus fluxos de trabalho complexos e reutilizem os componentes de maneira eficaz (BUITINCK et al., 2013).

2.4- Tensorflow e Keras

O TensorFlow é uma biblioteca que por muito tempo foi mantida fechada pela empresa *Google Inc.*, mas decidiram abrir seu código em 2015, sendo hoje uma das bibliotecas de aprendizado profundo mais populares (HIN, 2020). Ela possui uma gama de ferramentas e recursos para construir e treinar modelos de aprendizado profundo. Ela é construída sobre o paradigma de computação em grafos, onde cada operação é representada como nós nos grafos e as conexões entre os nós são os fluxos nos dados. Essa arquitetura permite que o TensorFlow otimize e execute eficientemente as operações, tanto em Unidade de Processamento Central (CPU) quanto em Unidade de Processamento Gráfico (GPU).

A fim de facilitar o desenvolvimento de modelos de aprendizado profundo, o TensorFlow fornece o Keras como uma API de alto nível. O Keras é uma API que permite construir modelos de forma rápida e intuitiva, abstraindo diversos detalhes da implementação, fornecendo uma interface simples, mas consistente para a construção de Redes Neurais, que muito se assemelha com a API do *Scikit-Learn*, permitindo a criação de camadas, funções de ativação, funções de perda, otimizadores, entre outras opções com poucas linhas de código.

O Keras é muito flexível e extensível, suportando os mais diversos tipos de arquiteturas de Redes Neurais, com camadas de diferentes tamanhos, múltiplas entradas e saídas, RNN, com camadas compartilhadas entre outros. Além disso, o Keras é totalmente compatível com o TensorFlow, podendo ser integrado a outros componentes da biblioteca.

O Keras já vem com muitas implementações prontas para uso, tanto em CPU quanto em GPU, como funções de ativação, por exemplo, ReLu, sigmoid, softmax e SELU, que podem ser facilmente aplicadas às camadas da rede neural. Essas funções de ativação desempenham um papel muito importante no processo de aprendizado, permitindo que a rede neural introduza não-linearidades e capture relações complexas nos dados.

Por fim, o Keras suporta nativamente o treinamento em lote e em épocas, permitindo o ajuste dos hiperparâmetros do modelo a fim de melhorar os resultados. Essas estratégias de treinamento permitem uma otimização mais eficiente do modelo e podem melhorar a sua acurácia (ABADI et al., 2015; CHOLLET, 2015).

2.5- OpenCV

A biblioteca Visão Computacional Open Source (OpenCV) é uma biblioteca de código aberto amplamente utilizada para processamento de imagens e visão computacional. Foi desenvolvida inicialmente pela empresa *Intel Inc.* em 1999 e teve seu código aberto no ano 2000. A partir disso, a comunidade de desenvolvedores vem contribuindo para seu aprimoramento e expansão, o que tornou-a uma das bibliotecas mais populares na área de visão computacional.

A biblioteca possui diversas funções e algoritmos para realizar desde tarefas comuns em preprocessamento de imagens, como leitura e gravação de imagens, filtragens e até tarefas mais complexas como detecção de bordas, segmentação, reconhecimento de padrões e execução de classificadores *cascade*. Ele tem suporte a operações em tempo real, o que torna o OpenCV adequado para aplicações como visão computacional em robótica, detecção de movimentos, segurança entre outros.

O OpenCV possui API para *Python*, o que é uma grande vantagem, visto que é

uma linguagem popular, principalmente na ciência de dados. A API oferece uma interface de programação simples e intuitiva, o que permite que os desenvolvedores aproveitem os recursos do OpenCV com facilidade, e possuindo integração com a biblioteca NumPy, que se integra com diversas bibliotecas como Tensorflow, Keras e *Scikit-Learn*. A API para *Python* porta funções e classes, abstraindo a complexidade subjacente dos algoritmos de visão computacional, o que permite que os desenvolvedores se concentrem na lógica dos aplicativos.

Por fim, o OpenCV possui uma gama de funções mais avançadas, como reconhecimento facial, reconhecimento de objetos, rastreamento de movimentos, calibração de câmeras estéreo e muito mais, adicionando diversas capacidades ao OpenCV e permitindo a construção de sistemas ainda mais complexos com visão computacional (BRADSKI, 2000).

2.6- Python

O *Python* é uma linguagem de programação de alto nível, interpretada, de código aberto e de propósito geral, que foi desenvolvida por Guido van Rossum e lançada pela primeira vez em 1991. Ela foi projetada para ser fácil de ler e escrever, com uma sintaxe clara e intuitiva. O *Python* ao longo dos anos ganhou muita popularidade e se tornou uma das linguagens mais utilizadas no mundo da programação.

A linguagem causou um grande impacto na computação científica, e se tornou uma escolha bem popular entre pesquisadores, cientistas de dados e engenheiros. Por possuir uma grande simplicidade e legibilidade permite rápida prototipagem e desenvolvimento ágil de soluções, e por possuir código aberto, não há necessidade de licenciamentos. Além disso, ao redor da linguagem, se formou uma extensa comunidade, que contribui com uma ampla gama de bibliotecas e pacotes voltados à computação científica, como Numpy, SciPy, Pandas e Holoviews, que fornecem ferramentas poderosas para análise de dados, processamento numérico, visualização de dados e muito mais.

No campo do aprendizado de máquina, *Python* se destacou como uma das principais linguagens. Novamente a sua facilidade de uso é um ponto de destaque, além de possuir toda a comunidade existente, gerando bibliotecas que contribuem para essas

tarefas, como o *Scikit-Learn*, Tensorflow e Keras, que permitiram o desenvolvimento e implementações eficientes de algoritmos de aprendizado de máquina com simplicidade. A comunidade do *Python* contribui para além da extensa quantidade de recursos, mas também com tutoriais e exemplos que disponibilizam para quem deseja aprender e trabalhar com aprendizado de máquina.

Quando falado de adoção, *Python* tem um alto nível nesse quesito dentro da comunidade acadêmica e industrial para quem faz o desenvolvimento de algoritmos de aprendizado de máquina. De acordo com uma pesquisa realizada em 2020 pelo Kaggle, uma plataforma muito popular para cientistas de dados e aprendizado de máquina, ferramentas baseadas em *Python* são líderes absolutas na escolha dos profissionais e acadêmicos da área.

Por fim, *Python* possui uma ampla gama de aplicações de aprendizado de máquina, como por exemplo, classificação de textos, reconhecimento de imagens, processamento de linguagem natural e muitos mais. Devido a toda a sua capacidade de integração com outras tecnologias existe essa contribuição para o rápido avanço do campo de aprendizado de máquina e para a sua aplicação nas mais diversas áreas, como finanças, IoT, automação industrial, robótica e muito mais (VAN ROSSUM; DRAKE, 2009).

2.7- Joblib

Joblib é uma biblioteca em *Python* que fornece avançados recursos de serialização de objetos, permitindo que eles sejam armazenados em disco e posteriormente recuperados de forma eficiente. Ela é muito útil quando se está trabalhando com objetos grandes ou demasiadamente complexos, como modelos de aprendizado de máquina treinados (o que, se for tentar recriar pode levar muito tempo, e até levar a resultados diferentes). Utilizando a biblioteca é possível salvar esses objetos no disco, compartilhar e carregá-los de forma rápida, simples e eficiente e até em outros equipamentos, economizando recursos computacionais e tempo.

Uma grande vantagem do joblib é a sua integração perfeita com outras bibliotecas bem populares do *Python*, como NumPy e *Scikit-Learn*, oferecendo suporte nativo à

serialização de arrays do Numpy e modelos do *Scikit-Learn*, o que permite que se preserve as estruturas de dados e garanta a sua compatibilidade, além de também poder serializar objetos personalizados.

Por fim, a serialização com o Joblib pode utilizar formatos especificos, como o *pickle*[4], que é o padrão, mas também pode ser comprimido, salvando espaço em disco, o que proporciona mais flexibilidade quando escolhe o método de armazenamento mais adequado às necessidades do projeto (ESTÈVE, 2015; JOBLIB DEVELOPMENT TEAM, 2020).

[4] *Pickle* é o formato padrão de serialização de objetos do *Python*, ou seja, conversão de objetos complexos da linguagem em texto binário.

3- Metodologia

Nesse capítulo, será delineado como foi desenvolvido esse projeto, particularidades no treinamento do modelo e na sua modelagem. Será detalhado como foram tratados os dados, o pré-processamento, a otimização dos hiperparâmetros, e como foram feitos os testes e validações, etapas essas fundamentais para obter um bom resultado.

Na Figura 7, temos uma visão abrangente de como funciona o modelo, e ao decorrer desse capítulo será detalhada cada etapa representada.

3.1- Pré-processamento

A base Minds Libras contém 1155 vídeos distribuídos em diversas pastas seguindo um formato padronizado, sendo este

Minds/Sinalizador*[número sinalizador]*/Canon/*[ID da palavra] [Palavra]*

Sinalizador*[número sinalizador]*-*[número da repetição]*.mp4 .

Com o objetivo de extrair informações necessárias desses vídeos, um programa em *Python* foi desenvolvido para ler os nomes de arquivos e pastas, bem como salvar os metadados, como a palavra mencionada em cada vídeo, em um arquivo CSV. Em seguida, utilizando a biblioteca *OpenCV*, os vídeos foram lidos e pré-processados.

Após iniciar a leitura dos vídeos usando o *OpenCV*, a biblioteca *MediaPipe* foi empregada para fazer a detecção dos pontos chave dos sinalizadores em cada vídeo, por meio do módulo Pose. Os pontos-chave são pontos de interesse, que podem representar as características distintivas dos sinais em Libras. Os resultados da detecção foram aplicados na construção de um tensor contendo as posições dos pontos-chave, no formato $M \times 33 \times 3$, em que M é o número de quadros lidos do vídeos, 33 é a quantidade de pontos-chave detectados pelo *MediaPipe* e 3 representa as coordenadas tridimensionais estimadas para cada ponto-chave no espaço.

Partindo desse passo, os pontos-chave de todos os vídeos foram então serializados em um arquivo utilizando a biblioteca *Joblib*. Essa serialização permite armazenar e

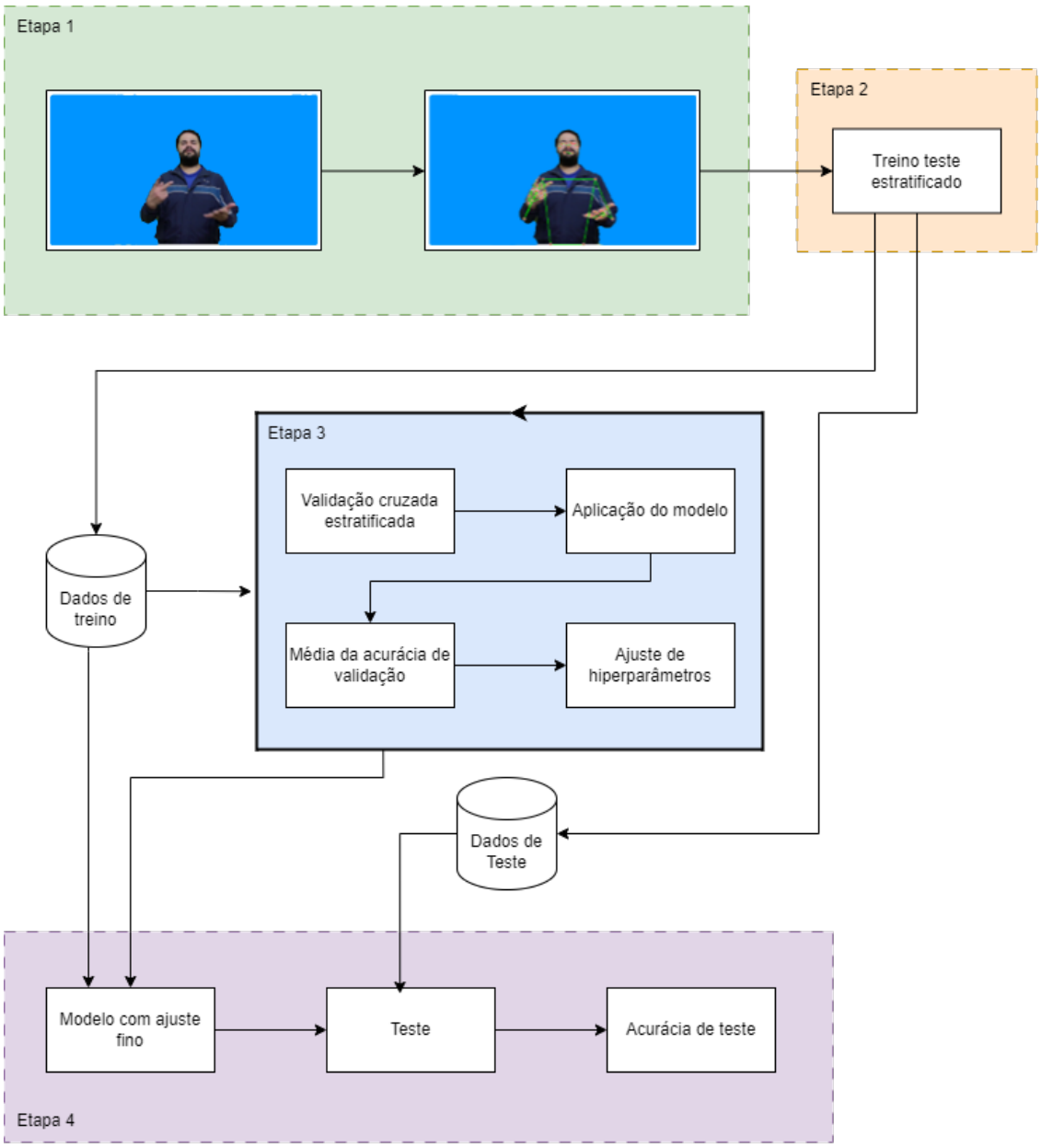


Figura 7 – Ilustração do *pipeline* do trabalho.
Fonte: Elaboração Própria.

recuperar eficientemente os dados dos pontos-chave, preservando as suas estruturas e atributos. Ao utilizar a serialização, os pontos-chave podem ser armazenados em disco e posteriormente, podem ser utilizados para diferentes análises e processamento, inclusive em outros computadores, sem a necessidade de reprocessamento. Esse processo é representado na Etapa 1 da Figura 7.

Essa abordagem de leitura dos vídeos, detecção dos pontos-chave e serialização dos dados é fundamental para o desenvolvimento do modelo proposto de reconhecimento de sinais de Libras. Os dados dos pontos-chave são utilizados como entrada para os algoritmos que serão utilizados na redução de dimensionalidade e classificação, permitindo a criação do presente modelo capaz de reconhecer e interpretar gestos da Libras. A combinação das bibliotecas *OpenCV* e *MediaPipe*, por meio do módulo Pose proporciona uma solução eficiente, poderosa e robusta para o pré-processamento de vídeos de sinais da Libras e a extração dos pontos-chave e a biblioteca *Joblib* oferece uma alternativa simples e eficiente de serialização para esses dados.

3.1.1 Separação de treino e teste

Utilizando arquivo CSV gerado na fase de pré-processamento, foi empregada a separação de treino e teste estratificada, foi selecionado esse método, pois ele preserva a distribuição das classes nos conjuntos de treino e teste. Inicialmente, os dados foram agrupados de acordo com suas respectivas classes. Em seguida, foi aplicada a técnica de amostragem estratificada, selecionando uma proporção de 75% dos dados de cada classe para o conjunto de treino, e os 25% restantes para o conjunto de teste. Dessa forma, garante-se que as proporções de cada classe fossem mantidas nos dois conjuntos, evitando viés na avaliação do modelo.

A separação estratificada é utilizada para evitar desequilíbrio entre as classes nos conjuntos de treino e teste. É uma abordagem de muita importância quando se trabalha com conjuntos de dados desbalanceados, nos quais uma ou mais classes são representadas por um número menor de amostras. Ao manter a proporção das classes nos conjuntos de treino e teste, garante-se que o modelo seja treinado e avaliado de forma equilibrada, o que permite uma avaliação mais precisa da sua capacidade de

generalização.

Quando utiliza-se uma separação de treino e teste estratificada, mantendo uma proporção de 75% para treino e 25% para teste, obtêm-se conjuntos representativos e equilibrados para todas as classes. É uma abordagem especialmente relevante em tarefas de aprendizado de máquina, onde a avaliação correta do desempenho do modelo depende da representatividade das amostras de treino e teste. Quando utilizamos essa técnica, temos uma contribuição para resultados mais confiáveis e robustos, permitindo que tenhamos uma capacidade de avaliação precisa e do modelo de generalizar para novos dados, independentemente da classe. Esse processo é representado na Etapa 2 da Figura 7.

Além da separação convencional de treino e teste, também é feita uma nova rodada de teste, efetuando validação cruzada, onde cada subconjunto é composto integralmente por vídeos de uma única pessoa, ou seja é feito um subconjunto específico para cada pessoa, totalizando 12 subconjuntos. Tal metodologia se denomina LOPO.

3.1.2 Redução de dimensionalidade

Tendo todos os vídeos da base pré-processados com o módulo Pose do MediaPipe e serializados com o Joblib em um arquivo, foi feita a leitura destes dados, o que nos retorna o tensor gerado anteriormente, de tamanho $M \times 33 \times 3$, sendo M a quantidade de quadros no vídeo. É importante destacar que essa representação de dados possui uma dimensão que varia em cada vídeo, o que cria um novo desafio, uma vez que a maioria dos modelos de aprendizado de máquina não são capazes de lidar com dados de dimensões variáveis.

Torna-se necessário realizar um tratamento adequado nos dados para que possam ser utilizados. Além disso, a dimensão $M \times 33 \times 3$ representa uma grande quantidade de dados, visto que o menor vídeo da base Minds-Libras possui $M = 71$ quadros. Assim a menor amostra da base possui 7029 características, muitas das quais podem ser consideradas desnecessárias para o problema de classificação.

A fim de reduzir a quantidade de amostras e obter uma dimensão constante, foi proposta a adoção do algoritmo *K-Means*, um algoritmo de aprendizado de máquina

de baixo custo computacional, não supervisionado, baseado em clusterização. Com o *K-Means*, agrupamos os quadros em uma quantidade parametrizável de clusters, o que permite que grandes quantidades de quadros semelhantes tenham uma maior influência sobre a localização dos centroides, enquanto quadros atípicos, que geralmente representam transições entre os gestos, sem valor para a classificação, tenham menor influência. Essa abordagem nos permite reduzir a dimensionalidade dos dados e obter uma representação compacta e de dimensão constante.

Ao definir os centroides, também é necessário ajustar a ordem deles, uma vez que a ordem dos sinais pode influenciar no significado das expressões da Libras. Para isso, foi realizado o ordenamento dos centroides verificando a qual cluster cada quadro pertence. Em seguida, calculamos a mediana do índice dos quadros que pertencem a cada cluster e utilizamos essa métrica para ordenar os centroides, do menor valor de mediana até o maior.

Uma observação que merece destaque é que a implementação do *K-Means* utilizada, do *Scikit-Learn*, suporta somente entradas unidimensionais. Essa limitação exigiu que fosse feita a reorganização da entrada, substituindo o tensor $M \times 33 \times 3$ por uma matriz $M \times 99$. Aplicando os dados nesse formato, obtivemos os centroides na saída no formato $N \times 99$, onde N representa a quantidade de clusters definida como parâmetro k no K-Means.

Com essa abordagem, desenvolveu-se um algoritmo de redução de dimensionalidade robusto a valores discrepantes. Sendo esse algoritmo capaz de receber uma entrada com dimensões variáveis e produzir como saída os centroides que definem os clusters das entradas, os quais possuem dimensões constantes, sendo a sua quantidade definida como parâmetro. Essa técnica permite obter uma representação mais compacta e adequada para o problema de classificação dos sinais de Libras. Esse processo é representado na Figura 8.

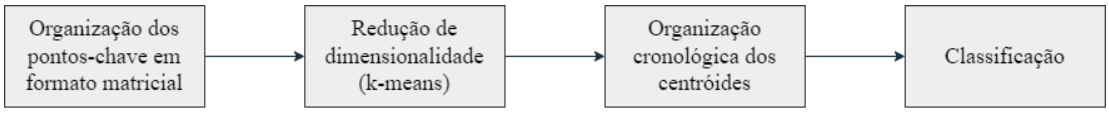


Figura 8 – Ilustração da execução do modelo.
Fonte: Elaboração própria

3.2- Classificação

Para a classificação dos gestos, foram escolhidos os algoritmos KNN, *Random Forest* e Redes Neurais. Com essa escolha, há uma diversidade de abordagens utilizadas para a classificação de gestos dinâmicos de Libras. Essa combinação permite explorar diferentes estratégias e métodos para melhor compreender e capturar as características dos gestos, tanto no aspecto temporal quanto espacial. Além disso, esses algoritmos têm uma ampla base de pesquisa e implementações disponíveis, o que facilita a aplicação e experimentação com diferentes configurações e hiperparâmetros.

O algoritmo KNN é uma escolha intuitiva para a classificação de gestos dinâmicos de Libras, pois é um método simples, rápido na implementação utilizada com *Scikit-Learn*, que utiliza vários algoritmos de indexação e intuitivo. Ele se baseia na proximidade de vizinhos para atribuir rótulos às amostras de teste. Quando utilizado no caso dos gestos dinâmicos de Libras, em que a sequência temporal é um fator importante, o KNN pode levar em consideração a estrutura temporal dos gestos ao avaliar a proximidade entre as amostras. Além disso, o KNN é robusto a ruídos e pode lidar com a variabilidade inerente aos gestos dinâmicos de locutores de diferentes tamanhos e que se expressam de formas diferentes.

O algoritmo *Random Forest* é uma poderosa escolha para a classificação de gestos dinâmicos de Libras, pois é capaz de lidar com alta dimensionalidade e características complexas. Com a sua capacidade de combinar várias árvores de decisão permite capturar interações não lineares características dos gestos. Além disso, ele possui uma capacidade intrínseca de lidar com sobreajuste e valores discrepantes, o que é importante ao trabalhar com dados de gestos dinâmicos que podem apresentar variações consideráveis. O *Random Forest* também é capaz de fornecer informações de probabilidade de pertencimento a cada classe, o que pode ser útil num próximo momento quando for feita a tradução de frases, para ser empregada essa informação num modelo linguagem natural.

As Redes Neurais são uma escolha natural para a classificação de gestos dinâmicos da Libras, pois são capazes de aprender representações complexas dos dados. Ao utilizar Redes Neurais, é possível capturar a estrutura temporal dos gestos por meio de camadas ocultas e arquiteturas específicas. As Redes Neurais podem aprender

características discriminativas a partir dos pontos-chave. As Redes Neurais também são capazes de lidar com alta dimensionalidade e também são capazes de informar a probabilidade de pertencimento de cada classe.

A combinação do *MediaPipe* para a extração de pontos-chave dos gestos da Libras e o algoritmo *K-Means* para a redução de dimensionalidade oferece uma abordagem eficiente e eficaz para lidar com a natureza dinâmica e dimensionalidade variável dos gestos. Sendo o *MediaPipe* uma forma confiável de extrair os pontos-chave e o *K-Means* capaz de reduzir a dimensionalidade e representar os gestos em uma forma mais compacta e significativa. Ao juntar essa poderosa combinação com algoritmos distintos e complementares, capazes de lidar com a complexidade dos gestos de Libras, são apresentadas tanto abordagens de proximidade entre as amostras, quanto relações mais profundas. Tem-se assim uma combinação de técnicas robustas e eficientes, capazes de interpretar os gestos de forma precisa e contribuir para a inclusão e comunicação efetiva das pessoas com deficiência auditiva e de fala. Podemos ver um pseudo-algoritmo representando o funcionamento do modelo no Algoritmo 2.

Algoritmo 2 – Algoritmo do modelo

Input:
$vídeo$: Vídeo de entrada
$n_centroides \in (0, TamanhoDoMenorVídeo]$: Quantidade de centroides gerados pelo K-Means
Output: L: Classe do vídeo
begin
  $PontosChave \leftarrow ExtratorDePontosChave(vídeo)$ Começamos extraindo os pontos chave do vídeo
  $KMeans \leftarrow TreinaKMeans(PontosChave, n_centroides)$ Treinamos o *K-Means* com os pontos chave de modo a ter $n_centroides$ centroides
  $Centroides \leftarrow ObtémCentoides(KMeans)$ Obtém os centroides gerados pelo treinamento do *K-Means*
  $Clusters \leftarrow ObtémClusters(KMeans)$ Obtém o cluster que cada quadro pertence
  $Medianas \leftarrow \{\}$ Inicializa o vetor de medianas com um vetor vazio
  for $Centroide \leftarrow 1$ até $n_centroides$ **do**
    $Medianas[Cluster] \leftarrow Mediana(j \mid Clusters[j] = i)$ Calcula a mediana do número do quadro de cada cluster
  $CentroidesOrdenados \leftarrow OrdenaPor(Centroides, Medianas)$ Ordena os centroides baseado na ordem das medianas de forma que os primeiros centroides se refiram aos quadros do começo do vídeo e os últimos aos quadros do final do vídeo
  $L \leftarrow Classificador(CentroidesOrdenados)$ Obtém a classe do vídeo com base nos centroides ordenados

3.3- Bayes Search

A utilização do algoritmo Naïve Bayes para a escolha adequada de hiperparâmetros denominada Bayes Search, possui uma estrutura simples e eficiente. Naïve Bayes é um classificador probabilístico baseado no teorema de Bayes, que assume independência condicional entre as variáveis. Essa característica permite que o algoritmo seja rápido e adequado para a otimização de hiperparâmetros em um espaço de busca.

A técnica Bayes Search utiliza a inferência bayesiana para realizar a busca dos melhores hiperparâmetros do modelo. Ela considera a informação prévia disponível sobre os hiperparâmetros, atualizando essa informação à medida que novos resultados são observados. Dessa forma, o Bayes Search pode direcionar a busca para as regiões do espaço de hiperparâmetros que têm maior probabilidade de levar a melhores resultados.

Bayes Search oferece vantagens significativas em relação à busca por grade. Quando se trata de eficiência de tempo e recursos computacionais a busca por grade examina todas as combinações disponíveis de parâmetros em uma grade pré-definida, enquanto o Bayes Search usa informações prévias e atualizações iterativas para direcionar a busca para as regiões de hiperparâmetros mais promissoras. Dessa forma, o Bayes Search é capaz de explorar de maneira mais inteligente e profunda o espaço de hiperparâmetros, descartando as combinações menos promissoras e concentrando-se nas áreas de maior potencial, podendo ir além de valores pré-definidos. Isso resulta em uma redução significativa no tempo necessário para encontrar uma boa configuração de hiperparâmetros e tendo valores ainda mais precisos.

Além da eficiência de tempo, o Bayes Search também se destaca em termos de precisão quando comparado com uma busca de parâmetros aleatória. Enquanto uma busca aleatória não possui uma estratégia bem definida para explorar o espaço de hiperparâmetros, o Bayes Search utiliza informações prévias e atualizações iterativas para selecionar as combinações de parâmetros com maior probabilidade de levar a melhores resultados. Isso resulta em uma exploração mais focada e direcionada do espaço de hiperparâmetros, permitindo uma otimização mais eficaz. Portanto o Bayes Search é capaz de encontrar configurações de hiperparâmetros que geralmente superam as encontradas por uma busca aleatória, resultando em modelos mais eficazes e com melhor desempenho.

A validação cruzada estratificada é uma abordagem importante para avaliar a contribuição da mudança para a otimização da acurácia ao usar a técnica Bayes Search. A validação cruzada estratificada permite avaliar o desempenho do modelo de forma mais robusta, considerando a distribuição das classes nos conjuntos de treinamento e teste. Isso é especialmente relevante quando se trabalha com conjuntos de dados desbalanceados.

Ao se utilizar a validação cruzada estratificada, pode-se verificar como a mudança para a otimização da acurácia com a técnica Bayes Search afeta o desempenho do modelo em diferentes cenários. Essa abordagem permite uma avaliação mais precisa e confiável do modelo, considerando a variabilidade dos dados e a distribuição das classes. Além disso, a validação cruzada estratificada ajuda a mitigar o viés que pode surgir devido à divisão arbitrária dos dados em conjuntos de treinamento e validação.

A busca de hiperparâmetros utilizando a técnica Bayes Search pode contribuir para a otimização mais eficiente e eficaz do modelo. O algoritmo Naïve Bayes é conhecido por sua eficiência computacional e baixa sensibilidade a hiperparâmetros, o que facilita a busca em um espaço de hiperparâmetros mais amplo. A técnica Bayes Search, combinada com a validação cruzada estratificada, permite explorar de forma mais estratégica as diferentes combinações de hiperparâmetros, melhorando a acurácia do modelo e reduzindo o risco de sobreajuste.

A otimização da acurácia por meio da busca de hiperparâmetros é um passo crucial no desenvolvimento de modelos de aprendizado de máquina. O uso da técnica Bayes Search em conjunto com o algoritmo Naïve Bayes permite uma exploração sistemática e eficiente do espaço de hiperparâmetros, em busca de configurações que maximizem o desempenho do modelo. A validação cruzada estratificada, por sua vez, garante uma avaliação mais precisa dos resultados e ajuda a generalizar o desempenho do modelo em diferentes conjuntos de dados.

A utilização do algoritmo Naïve Bayes para a busca de hiperparâmetros e a aplicação da validação cruzada estratificada proporciona uma abordagem sólida para a busca de hiperparâmetros e otimização da acurácia na classificação de gestos dinâmicos de Libras. Essa metodologia permite explorar de forma eficiente e profunda o espaço de hiperparâmetros, ajustar o modelo de acordo com as características específicas dos dados e avaliar sua capacidade de generalização em diferentes cenários, pois necessita de menos iterações e explora valores dentro de intervalos contínuo, ações essas que os

modelos mais convencionais não costumam tomar. Ao se adotar essa abordagem, espera-se obter modelos de aprendizado de máquina mais robustos e precisos na classificação de gestos de Libras. Esse processo é representado na Etapa 3 da Figura 7.

3.3.1 Hiperparâmetros otimizados

Para a otimizar o modelo proposto, foram exploradas diferentes combinações de hiperparâmetros. Esses podendo ser divididos em duas categorias: aqueles que são características inerentes ao modelo e necessários independentemente do classificador utilizado, e aqueles que são específicos do classificador, e variavam a depender se foi escolhido KNN, *Random Forest* ou Redes Neurais. Os hiperparâmetros podem assumir três tipos diferentes de espaços: inteiro, quando se trata de um número inteiro; real, quando ele é um número de ponto flutuante; e categórico, quando o valor deve ser selecionado a partir de uma coleção finita ou, em alguns casos, infinita, sendo necessário nesse caso uma amostragem adequada.

Os hiperparâmetros otimizados estão expostos na Tabela 2, sendo na primeira coluna a designação do hiperparâmetro, na segunda, se ele é inerente ao modelo ou caraterístico de um classificador, nesse caso é preenchido com o nome do classificador no qual ele é utilizado, na terceira coluna, o tipo de espaço no qual ele pertence, se é um parâmetro inteiro, real ou categórico e na quarta, em qual sub-conjunto do espaço ele foi otimizado. Também podemos ver a aplicação desse hiperparâmetros na Etapa 4 da Figura 7.

Tabela 2 – Hiperparâmetros otimizados.

Hiperparâmetro	Escopo	Tipo de espaço	Espaço
Número de centroides	Modelo	Inteiro	[1, 70]
Número de vizinhos	KNN	Inteiro	[1, 50]
Peso dos vizinhos	KNN	Categórico	{uniforme, distância}
Continua na próxima página			

Tabela 2 Continuação da página anterior

Hiperparâmetro	Escopo	Tipo de espaço	Espaço
Aplicar bootstrap	*Random Forest*	Categórico	{Sim, Não}
Profundidade máxima	Random Forest	Categórico	{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 500, ∞}
Quantidade de características	*Random Forest*	Categórico	{raiz quadrada, $\log_2$, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%}
Quantidade mínima de amostras por folha	*Random Forest*	Inteiro	[1, 20]
Quantidade mínima de amostras por ramo	*Random Forest*	Inteiro	[2, 20]
Número de estimadores	*Random Forest*	Inteiro	[100, 20.000]
Tamanho do lote	Redes Neurais	Inteiro	[5, 128]
Épocas de treino	Redes Neurais	Inteiro	[5, 1000]
Parada prévia	Redes Neurais	Categórico	{Sim, Não}
Variação mínima para parada prévia	Redes Neurais	Real	[0, 100]
Paciência para parada prévia	Redes Neurais	Inteiro	[0, 100]
Reduzir a taxa de aprendizado em estagnação	Redes Neurais	Categórico	{Sim, Não}

Continua na próxima página

Tabela 2 Continuação da página anterior

Hiperparâmetro	Escopo	Tipo de espaço	Espaço
Fator de redução da taxa de aprendizado em estagnação	Redes Neurais	Real	$[10^{-10}, \quad (1 - 10^{-10})]$
Paciência para redução de taxa de aprendizado	Redes Neurais	Inteiro	[0, 100]
Variação mínima para redução de taxa de aprendizado	Redes Neurais	Real	[0, 100]
Épocas para reativação da redução de taxa de aprendizado	Redes Neurais	Inteiro	[0, 100]
Taxa mínima de aprendizado após redução	Redes Neurais	Real	$[10^{-10}, \quad (1 - 10^{-10})]$
Unidades por camada oculta	Redes Neurais	Inteiro	[0, 1000]
Quantidade de camadas ocultas	Redes Neurais	Inteiro	[0, 100]
Função de ativação das camadas ocultas	Redes Neurais	Categórico	{ReLu, ELU, SELU, leaky relu}
Função de ativação da camada de saída	Redes Neurais	Categórico	{sigmoid, softmax}
Normalização por lote	Redes Neurais	Categórico	{Sim, Não}
Otimizador	Redes Neurais	Categórico	{Adam, SGD, RMSprop}

Continua na próxima página

Tabela 2 Continuação da página anterior

Hiperparâmetro	Escopo	Tipo de espaço	Espaço
Taxa de aprendizado do Adam	Redes Neurais	Real	$[10^{-15}, \quad 10^{-1}]$
β_1 do Adam	Redes Neurais	Real	$[0,5, \quad (1 - 10^{-15})]$
β_2 do Adam	Redes Neurais	Real	$[0,5, \quad (1 - 10^{-15})]$
ϵ do Adam	Redes Neurais	Real	$[10^{-15}, \quad 0,3]$
Utilizar gradiente AMS no Adam	Redes Neurais	Categórico	{Sim, Não}
Taxa de aprendizado do SGD	Redes Neurais	Real	$[10^{-15}, \quad 0,3]$
Momento do SGD	Redes Neurais	Real	[0, 1]
SGD com nesterov	Redes Neurais	Categórico	{Sim, Não}
Taxa de aprendizado do RMSprop	Redes Neurais	Real	$[10^{-15}, \quad 0,3]$
ρ do RMSprop	Redes Neurais	Real	$[0,5, \quad (1 - 10^{-15})]$
ϵ do RMSprop	Redes Neurais	Real	$[10^{-15}, \quad 0,3]$
RMSprop centralizado	Redes Neurais	Categórico	{Sim, Não}

4- Resultados e discussão

4.1- Resultados

Os resultados obtidos no projeto proposto são altamente promissores e destacam a eficácia dos algoritmos utilizados, mesmo com sua simplicidade. O KNN alcançou uma acurácia de 81,89% para o conjunto de teste e 100% quando verificada a acurácia com o mesmo conjunto que foi utilizado anteriormente para treinar, mostrando uma capacidade de aprendizado significativa. Por outro lado o *Random Forest* obteve resultados ainda mais impressionantes, apresentando uma acurácia de 94,72% para o conjunto de teste e 100% para o conjunto de treino, indicando uma excelente capacidade de generalização.

Enquanto isso, as Redes Neurais apresentaram uma acurácia de 74,34% para o conjunto de teste e 90,42% para o conjunto de treino, o que se justifica por uma dificuldade de generalização do modelo neural para novos dados, o que pode ser comum quando há limitações de tamanho no conjunto de treino.

Neste projeto, o *hardware* utilizado para as avaliações e classificações foi composto por um processador Intel™ i9 9900k, acompanhado por 32 GB de RAM DDR4, e uma placa de vídeo Nvidia™ RTX 2080 Ti, com *overclock* de 1720 MHz. Cada modelo foi testado com o objetivo de classificação em tempo real, considerando a natureza dos vídeos de, em média, terem 5 segundos de duração. Todos os classificadores apresentaram resultados excepcionais no que tange a tempo de classificação, demonstrando sua capacidade de realizar a tarefa de forma ágil, eficiente e em tempo real.

O KNN exibiu um tempo médio de 0,791 ms de tempo de classificação, mesmo em *single-thread*, destacando-se como uma abordagem extremamente ágil. O *Random Forest*, com uma configuração *multi-thread* utilizando 16 threads, obteve uma média de 28,55 ms de tempo de classificação. As Redes Neurais aproveitando o poder da GPU, alcançaram uma média de 10,29 ms de tempo de classificação. Esses tempos foram mensurados juntamente com o pré-processamento utilizando *K-Means* em *single-thread* e demonstram que todos os modelos são mais do que capazes de realizar a classificação em tempo real para os vídeos de, no mínimo, 2 segundos que foram utilizados, o que

reforça sua eficiência computacional.

Comparando os resultados relatados no projeto proposto com o modelo proposto por Rezende et al. (REZENDE; ALMEIDA; GUIMARÃES, 2021), com duas CNNs, sendo uma CNN 3D extremamente complexa e de difícil utilização em dispositivos mais simples, como celulares, os modelos propostos baseados em KNN e *Random Forest* demonstram desempenho próximos ou até superior em relação à acurácia geral. O modelo da literatura alcançou 93,3% de acurácia média de validação, e relatou um tempo médio de classificação de 5 segundos na configuração utilizada pelos autores, enquanto o modelo com *Random Forest* obteve 94,72% de acurácia no conjunto de teste, tendo uma velocidade 200 vezes maior. Vale ressaltar que o modelo proposto é de baixo custo computacional e facilmente embarcado em dispositivos móveis, o que torna uma opção muito mais prática e viável para aplicações em tempo real e de uso no dia-a-dia, podendo ser utilizado em celulares.

Já Castro et al. (CASTRO; GUERRA; GUIMARÃES, 2023) apresentou diversos modelos trabalhando simultaneamente, como três CNN's que recebem os quadros em RGB, além de uma obter o fluxo óptico e outra mapas de profundidade artificias gerados por DGAN como mais um canal da imagem, além de outros 2 modelos que recebem as imagens das mãos, imagens do rosto, distância das juntas e velocidade do movimento, e adicionalmente um recebe a imagem RGB completa do vídeo e os mapas de profundidade artificial, totalizando 5 modelos diferentes. Após todos esses modelos darem as suas respectivas classificações, é feito um sistema de voto para determinar a predição final da classe do gesto. Com essa metodologia, obtiveram uma acurácia de $91\% \pm 7\%$ e utilizando LOPO alcançaram uma acurácia de $57\% \pm 12\%$, com um tempo de 552,79 ms.

A comparação entre os modelos pode ser vista também na tabela 3.

Trabalho	Acurácia (%)	Acurácia LOPO (%)	Tempo de classificação (ms)
Random Forest	94,72	59,44	28,55
Rezende et al.	93,3	-	5000
Castro et al.	91	57	552,79

Tabela 3 – Comparação entre os modelos em termos de acurácia e tempo de classificação

Em conclusão, os resultados obtidos neste projeto mostram que o KNN e o *Random Forest* são abordagens eficazes para a classificação de gestos dinâmicos de Libras. O *Random Forest*, em particular, se destacou pelo desempenho superior em termos de

acurácia geral, já o KNN em custo computacional, o que permite ter a possibilidade de escolha com base em critérios, como acurácia e complexidade computacional, podendo deixar até como uma opção de escolha do usuário em um aplicativo móvel. Além disso, o modelo proposto, baseado em modelos de baixo custo computacional, revelou-se uma alternativa viável em relação ao modelo proposto pela Minds, possibilitando sua utilização em dispositivos mais simples, como celulares, sem comprometer a qualidade da classificação. Esses resultados têm um grande impacto na aplicação prática da classificação de gestos de Libras, tornando possível sua utilização em diversos cenários, e abrindo caminho para novas aplicações e aprimoramentos futuros.

4.1.1 Melhores parâmetros

Neste estudo, foram empregadas as técnicas de otimização de hiperparâmetros para três modelos de aprendizado de máquina amplamente utilizados: KNN, *Random Forest* e Redes Neurais. O objetivo foi explorar as melhores configurações de parâmetros para cada algoritmo, a fim de alcançar o melhor desempenho possível na tarefa de classificação de gestos dinâmicos de Libras. Para tal, utilizou-se a técnica denominada Bayes *Search*, que permitiu uma busca mais eficiente e assertiva na busca pelos melhores hiperparâmetros.

Neste trabalho estão presentes os melhores parâmetros encontrados pelo Bayes *Search* para o KNN, *Random Forest* e Redes Neurais, relatando os resultados de desempenho alcançado com essas configurações otimizadas na Seção 4.1. A Tabela 4 mostra os melhores parâmetros escolhidos para o KNN, já a Tabela 5 para o *Random Forest* e a Tabela 6 para as Redes Neurais, tomando como base a média da acurácia de validação ao fazer validação cruzada estratificada.

Tabela 4 – Hiperparâmetros escolhidos para o KNN.

Hiperparâmetro	**Tipo de espaço**	**Valor escolhido**
Número de centroides	Inteiro	1
Continua na próxima página		

Tabela 4 Continuação da página anterior

Hiperparâmetro	Tipo de espaço	Espaço
Número de vizinhos	Inteiro	1
Peso dos vizinhos	Categórico	uniforme

Tabela 5 – Hiperparâmetros escolhidos para o *Random Forest*.

Hiperparâmetro	Tipo de espaço	Valor escolhido
Número de centroides	Inteiro	70
Aplicar bootstrap	Categórico	Não
Profundidade máxima	Categórico	∞
Quantidade de características	Categórico	raiz quadrada
Quantidade mínima de amostras por folha	Inteiro	1
Quantidade mínima de amostras por ramo	Inteiro	2
Número de estimadores	Inteiro	18.785

Tabela 6 – Hiperparâmetros escolhidos para a Rede Neural.

Hiperparâmetro	Tipo de espaço	Valor escolhido
Número de centroides	Inteiro	70
Tamanho do lote	Inteiro	31
Épocas de treino	Inteiro	1000
Continua na próxima página		

Tabela 6 Continuação da página anterior

Hiperparâmetro	Tipo de espaço	Espaço
Parada prévia	Categórico	Não
Variação mínima para parada prévia	Real	59,363370150935054
Paciência para parada prévia	Inteiro	75
Reduzir a taxa de aprendizado em estagnação	Categórico	Sim
Fator de redução da taxa de aprendizado em estagnação	Real	0,5577047493819142
Paciência para redução de taxa de aprendizado	Inteiro	0
Variação mínima para redução de taxa de aprendizado	Real	40,9087392652479
Épocas para reativação da redução de taxa de aprendizado	Inteiro	73
Taxa mínima de aprendizado após redução	Real	0,9211766935186267
Unidades por camada oculta	Inteiro	855
Quantidade de camadas ocultas	Inteiro	0
Função de ativação das camadas ocultas	Categórico	ReLu

Continua na próxima página

Tabela 6 Continuação da página anterior

Hiperparâmetro	Tipo de espaço	Espaço
Função de ativação da camada de saída	Categórico	softmax
Normalização por batch	Categórico	Não
Otimizador	Categórico	Adam
Taxa de aprendizado do Adam	Real	0,04365510082176306
β_1 do Adam	Real	0,9085584803515259
β_2 do Adam	Real	0,7064117693708993
ϵ do Adam	Real	0,015640708235555995
Utilizar gradiente AMS no Adam	Categórico	Sim
Taxa de aprendizado do SGD	Real	0,2478239233120042
Momento do SGD	Real	0,5993324145775742
SGD com nesterov	Categórico	Não
Taxa de aprendizado do RMSprop	Real	0,012571443288409631
ρ do RMSprop	Real	0,5460736798660093
ϵ do RMSprop	Real	0,11707484392445998
RMSprop centralizado	Categórico	Sim

4.2- Discussão

No contexto do projeto proposto, a otimização dos parâmetros do *K-Means* utilizado no pré-processamento, assim como dos classificadores KNN, *Random Forest* e Redes Neurais, revelou resultados interessantes e distintos para cada modelo. Observou-se na tabela 4 que o KNN obteve melhores resultados de classificação quando o *K-Means* gerava saídas com apenas um centroide no pré-processamento. Isso pode ser explicado pelo fato de que, nessa configuração, o *K-Means* agrupa os quadros de forma mais abrangente, reduzindo a complexidade do modelo, e essa redução permite que o KNN se beneficie de uma maior generalização nos dados de treinamento.

Por outro lado o *Random Forest* e as Redes Neurais apresentam melhores resultados quando o *K-Mens* gerava 70 centroides, o limite que poderia ser gerado, dado que o menor vídeo tem 71 quadros. Essa abordagem de pré-processamento resulta em clusters mais específicos, o que pode favorecer a capacidade de capturar padrões mais complexos e sutis nos dados, tornando o modelo mais adequado para lidar com a tarefa de classificação de gestos dinâmicos de Libras.

Quanto ao número de estimadores no *Random Forest*, a necessidade de uma grande quantidade de estimadores (18785) pode ser atribuída à complexidade da base de dados e à variedade de gestos presentes. Esse alto número de estimadores permite ao modelo aprender com maior precisão os padrões mais específicos e detalhados, resultando em uma melhor acurácia.

As Redes Neurais obtiveram melhores resultados sem a utilização de camadas ocultas, como se pode ver na tabela 6, caracterizando-as como Redes Neurais Rasas. Isso pode indicar que a simplificação gerada pelo pré-processamento foi suficiente para capturar as informações relevantes dos gestos de Libras, não havendo necessidade de camadas intermediárias de processamento, também é notório que houve uma saturação na quantidade de épocas de treino, o que pode ser mais um motivo causador, visto que foi necessário um modelo mais simples para efetuar a classificação. Essa descoberta ressalta a importância de experimentar diferentes configurações de Redes Neurais para cada problema específico, a fim de encontrar a arquitetura mais adequada.

Quanto aos otimizadores utilizados nas Redes Neurais, o Adam mostrou-se mais eficaz em relação ao SGD e ao RMSprop. O Adam combina o conceito de momentum

com o RMSprop, o que permite um ajuste mais adaptativo e eficiente dos pesos durante o processo de treinamento. Essa capacidade de adaptação torna o Adam mais resiliente a problemas de otimização, o que pode explicar seu melhor desempenho em relação a outros otimizadores.

Os resultados obtidos com cada classificador revelam informações importantes sobre a complexidade do problema de classificação de gestos dinâmicos de Libras. Primeiramente, a constatação de que as Redes Neurais alcançavam o melhor resultado sem a utilização de camadas ocultas pode sugerir que o problema possui um padrão mais simples do que o inicialmente esperado. A ausência de camadas ocultas indica que a representação das informações dos gestos pode ser capturada de forma mais direta e linear pelas Redes Neurais, sem a necessidade de camadas intermediárias de processamento para a extração de características complexas.

Comparando esse resultado com a literatura, onde foi relatado o uso de uma CNN 3D, um modelo muito mais complexo e capaz de lidar com tarefas altamente desafiadoras, pode-se inferir que o problema de classificação de gestos dinâmicos de Libras não requer uma abordagem tão sofisticada e profunda. Isso sugere que os gestos de Libras podem ser bem representados e discriminados por meio de abordagens mais simples, como os modelos de baixo custo computacional utilizados neste projeto.

Essa descoberta pode ter implicações significativas para a aplicação prática de solução de classificação de gestos de Libras, especialmente quando considera-se a viabilidade de implementação em dispositivos mais simples, como celulares ou outros dispositivos com recursos computacionais limitados. Ao se optar por modelos menos complexos, mas ainda altamente eficazes, pode-se obter resultados precisos em tempo real, tornando o uso dessas tecnologias mais acessível e eficiente para a comunicação e inclusão de pessoas com deficiência auditiva.

Em suma, as diferentes escolhas de pré-processamento, parâmetros e configurações dos algoritmos de aprendizado de máquina adotados no projeto demonstram a importância da experimentação sistemática e refinamento de hiperparâmetros para obter o melhor desempenho em tarefas específicas. Os resultados obtidos enfatizam a necessidade de abordagens personalizadas para cada modelo e problema, a fim de alcançar resultados mais precisos e eficientes. Outro ponto que esse projeto traz à luz é que os resultados indicam que o problema de classificação de Libras pode ser abordado de forma eficaz com modelos de baixo custo computacional, o que destaca a importância

de avaliar a complexidade necessária para cada tarefa e problema específico. Essas conclusões contribuem para uma maior compreensão da natureza desse desafio e abrem caminho para o desenvolvimento de soluções mais acessíveis e eficientes no contexto de comunicação e inclusão de pessoas com deficiência auditiva e de fala.

5- Considerações finais

A solução proposta nesse projeto apresenta uma abordagem inovadora e eficaz para a classificação de gestos dinâmicos de Libras, utilizando modelos de aprendizado de máquina de baixo custo computacional, como o KNN, *Random Forest* e Redes Neurais Rasas. Os resultados superiores alcançados em relação à literatura, mesmo com modelos mais simples e rápidos, evidencia a relevância dessa abordagem para a comunidade científica e para a sociedade em geral.

A aplicação do *K-Means* como técnica de pré-processamento para redução de dimensionalidade dos dados, resultando em centroides que representam clusters de gestos, apresenta uma contribuição inovadora para a classificação gestual. Essa abordagem pode ser aplicada não apenas para a classificação de Libras, mas também em outras tarefas de análise de vídeos e sinais como um todo, abrindo novas perspectivas e possibilidades para o campo de reconhecimento de padrões.

Os resultados desse projeto têm o potencial de impactar positivamente a comunidade científica ao fornecer valiosas perspectivas sobre a eficácia de modelos de aprendizado de máquina mais simples e de baixo custo computacional em tarefas complexas de classificação. Essa abordagem pode ser aplicada em diversos modelos adequados para diferentes cenários e maximizar a eficiência computacional sem comprometer a precisão.

Além disso, os resultados têm um impacto significativo na sociedade, especialmente no que tange à inclusão e comunicação de pessoas com deficiência auditiva. A classificação eficiente de gestos dinâmicos de Libras com modelos rápidos e acessíveis pode levar a aplicações práticas em dispositivos móveis e tecnologias assistivas, permitindo que pessoas com deficiência auditiva se comuniquem de forma mais fluida e integrada.

Isso posto, chega-se à conclusão que a solução proposta neste projeto oferece uma abordagem promissora para a classificação de gestos dinâmicos de Libras, destacando-se pelos resultados superiores em relação à literatura e pelo uso de modelos de aprendizado de baixo custo computacional. As implicações dessa abordagem vão além da classificação de Libras, trazendo novas contribuições para a utilização do

K-Means na classificação de vídeos e sinais e impactando positivamente a comunidade científica e a sociedade como um todo.

5.1- Trabalhos Futuros

Tendo os resultados expostos nesse projeto, abre-se uma grande janela de trabalhos que podem ser explorados a partir desse ponto, que podem contribuir ainda mais para a ciência e para a sociedade, entre eles, merecem destaque a exploração de outros tipos de arquiteturas de Redes Neurais, como as CNN ou RNN para classificar os centroides, a fim de verificar se modelos mais voltados a imagens e sequências temporais podem melhorar ainda mais a acurácia, explorando os padrões temporais entre os dados.

Um outro caminho que pode ser explorado é a utilização de modelos de DGAN baseados em difusão estável (*stable diffusion*) para gerar vídeos artificiais, visto que com eles se torna possível imitar os gestos utilizados na base de forma a não ser uma réplica exata, alterando levemente o movimento, mas sem executar movimentos impossíveis, além de permitir a total alteração da aparência do intérprete.

É possível aplicar também a transferência de aprendizado para investigar a aplicação dela no contexto de classificação de gestos dinâmicos de Libras. Tendo essa abordagem envolvida, pode-se reduzir a quantidade de dados necessários, o que é um ponto crítico para as Redes Neurais, e assim ser capaz de generalizar os gestos de Libras com a quantidade de dados na base atual.

Um outro caminho também que pode ser investigado é a troca do algoritmo extrator de pontos-chave (*MediaPipe*) por outro, como por exemplo, o *Detectron* ou *DensePose*, ambos da Meta.

Também é possível verificar se existe ganho ao otimizar os hiperparâmetros do algoritmo extrator de pontos-chave com o *Bayes Search*.

A investigação de outros algoritmos para o pré-processamento também é algo possível, como outros algoritmos de clusterização, ou até algoritmos mais voltados ao pré-processamento, a fim de verificar se diferentes abordagens de redução de dimensionalidade podem impactar o desempenho dos classificadores.

Também é importante avaliar a eficiência e a precisão do modelo proposto em

dispositivos móveis, visto que essa é a proposta do projeto, para garantir que o modelo é viável para aplicações práticas e reais do dia-a-dia no contexto de comunicação em Libras.

Outro caminho que o presente projeto pode se estender é o de reconhecimento e tradução de frases inteiras, expandindo o escopo para esse ponto, ao invés de se limitar à simples classificação de palavras, pode-se criar um sistema mais abrangente para a tradução de Libras e trazer uma ferramenta de uso prático e viável para os deficientes auditivos e de fala.

Por fim, é importante avaliar a robustez do modelo, testando em relação a variações de iluminação, angulação da captura do vídeo e os mais diversos fatores que possam afetar a qualidade dos dados. Garantir que o modelo seja capaz de lidar com as intempéries que podem ocorrer em cenários reais é de suma importância para um projeto como esse.

Referências

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. [S.l.: s.n.], 2015. Software available from tensorflow.org.

ABDALLAH, M. S. et al. Light-Weight Deep Learning Techniques with Advanced Processing for Real-Time Hand Gesture Recognition. **Sensors**, v. 23, n. 1, p. 1–20, 2023.

ALIBRAHIM, H; LUDWIG, S. A. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In: IEEE. 2021 IEEE Congress on Evolutionary Computation (CEC). [S.l.: s.n.], 2021. P. 1551–1559.

BASTOS, I. L. O.; ANGELO, M. F.; LOULA, A. C. Recognition of static gestures applied to brazilian sign language (Libras). In: PROC. of the 28th SIBGRAPI Conf. on Graphics, Patterns and Images. Salvador: [s.n.], ago. 2015.

BRADSKI, G. The openCV library. **Dr. Dobb's Journal: Software Tools for the Professional Programmer**, Miller Freeman Inc., v. 25, n. 11, p. 120–123, 2000.

BRASIL. Lei nº 10.436, de 24 de Abril de 2002. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 24 abr. 2002.

BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. [S.l.: s.n.], 2013. P. 108–122.

CAIAFA, E. G. A. et al. Aprendizado profundo no reconhecimento de sinais estáticos de Libras. In: XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2020). [S.l.: s.n.], 2020. DOI: 10.14209/SBRT.2020.1570660513.

CAIAFA, E. G. A. et al. Interpretação de gestos de Libras usando k-means e Random Forest. In: XLI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2023). [S.l.: s.n.], 2023. DOI: 10.14209/sbrt.2023.1570923868.

CASTRO, G. Z. de; GUERRA, R. R.; GUIMARÃES, F. G. Automatic translation of sign language with multi-stream 3D CNN and generation of artificial depth maps. **Expert Systems with Applications**, v. 215, p. 119394, 2023. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.119394.

CASTRO, G. Z. et al. Desenvolvimento de uma base de dados de sinais de Libras para aprendizado de máquina: Estudo de caso com CNN 3D. In: PROC. 14th Simpósio Brasileiro de Automação Inteligente (SBAI). [S.l.: s.n.], 2019. P. 2116–2121.

CHOLLET, F. at al. **Keras**. [S.l.: s.n.], 2015. https://keras.io.

CORREA, D. V. et al. Aprendizagem de máquina para identificação de plantas de soja sob ataque de insetos usando dados hiperespectrais. In: COLLOQUIUM EXACTARUM, V. 14, 2023.

COSTA, L. A. J. da; SENEFONTE, H. C. M. GNN em Segurança de Redes.

COSTA FILHO, C. F. F. C. et al. A fully automatic method for recognizing hand configurations of Brazilian sign language. **Research on Biomedical Engineering**, v. 33, n. 1, p. 78–89, mar. 2017.

DIEDERIK, P. K.; B., Jimmy. **Adam: A Method for Stochastic Optimization**. [S.l.: s.n.], 2014. eprint: `arXiv:1412.6980`.

ESTÈVE, L. Big data in practice: the example of nilearn for mining brain imaging data. In: SCIPY 2015. [S.l.: s.n.], 2015.

FENG, C. et al. Dimension reduction and clustering models for single-cell RNA sequencing data: a comparative study. **International journal of molecular sciences**, MDPI, v. 21, n. 6, p. 2181, 2020.

HIN, D. **StackOverflow vs Kaggle: A Study of Developer Discussions About Data Science**. [S.l.: s.n.], 2020. eprint: `arXiv:2006.08334`.

HOU, W. et al. An advanced k nearest neighbor classification algorithm based on KD-tree. In: IEEE. 2018 IEEE International Conference of Safety Produce Informatization (IICSPI). [S.l.: s.n.], 2018. P. 902–905.

JOBLIB DEVELOPMENT TEAM. **Joblib: running Python functions as pipeline jobs**. [S.l.: s.n.], 2020. https://joblib.readthedocs.io/.

LUGARESI, C. et al. MediaPipe: A Framework for Building Perception Pipelines. **CoRR**, abs/1906.08172, n. 1, p. 1–9, 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, n. 85, p. 2825–2830, 2011.

PRADANA, M. G.; HA, H. T. Maximizing strategy improvement in mall customer segmentation using k-means clustering. **Journal of Applied Data Sciences**, v. 2, n. 1, p. 19–25, 2021.

RAO, G. A. et al. Deep convolutional neural networks for sign language recognition. In: 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES). [S.l.: s.n.], 2018. P. 194–197.

REZENDE, T. M.; ALMEIDA, S. G. M.; GUIMARÃES, F. G. Development and validation of a Brazilian sign language database for human gesture recognition. **Neural Cmputing and Applications**, v. 33, n. 16, p. 10449–10467, 2021.

RUSSAKOVSKY, O. et al. **ImageNet Large Scale Visual Recognition Challenge**. [S.l.: s.n.], 2014. eprint: arXiv:1409.0575.

SUNDAR B. ANDBAGYAMMAL, T. American Sign Language Recognition for Alphabets Using MediaPipe and LSTM. **Procedia Computer Science**, v. 215, n. 1, p. 642–651, 2022. 4th International Conference on Innovative Data Communication Technology and Application.

SURABHI, A. R. et al. Background removal using k-means clustering as a preprocessing technique for DWT based Face Recognition. In: IEEE. 2012 International Conference on Communication, Information & Computing Technology (ICCICT). [S.l.: s.n.], 2012. P. 1–6.

SUTSKEVER, I. et al. On the importance of initialization and momentum in deep learning. In_____. 3. **Proceedings of the 30th International Conference on Machine Learning**. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013. v. 28. (Proceedings of Machine Learning Research, 3), p. 1139–1147.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. [S.l.: s.n.], 2015. eprint: arXiv:1512.00567.

VAN ROSSUM, G.; DRAKE, F. L. **Python**. [S.l.: s.n.], 2009.

ZHANG, M.; ZHOU, Z. ML-KNN: A lazy learning approach to multi-label learning. **Pattern recognition**, Elsevier, v. 40, n. 7, p. 2038–2048, 2007.

ZHOU, Z. **Machine learning**. [S.l.]: Springer Nature, 2021.

Apêndice A

Outras abordagens

A fim de explorar as possibilidades, foram executadas diversas outras abordagens com o intuito de exaurir as alternativas que poderiam trazer melhores resultados, mas no fim, não alcançaram melhorias significativas no resultado, entre elas, se destacam:

Ao utilizarmos LSTM, verificamos que a acurácia não estava alcançando valores superiores a 70%. Infelizmente não pudemos acompanhar os testes até o final, pois após cerca de uma semana efetuando uma bateria de testes, houve queda de energia e não foram retomados os testes desse modelo, priorizando outros modelos mais promissores e mais velozes.

Também utilizamos técnicas semelhantes à utilizada pelo SMOTE para geração de dados artificiais, interpolando dados entre os vizinhos, mas não houve melhora nos resultados quando aplicado ao *Random Forest*. Ainda explorando outras possibilidades com o mesmo modelo, foram feitos testes efetuando uma seleção de *features*, tanto manualmente, quanto automaticamente. Foram removidos pontos que julgamos desnecessários, como pernas e removemos *features* com menor significância selecionadas pelo modelo, mas o melhor resultado não superou o obtido com todas as *features*, trazendo somente uma ligeira melhora na performance.

Um outro teste utilizado foi a interpolação por splines, parábolas e polinômios de terceira ordem para geração de pontos-chave adicionais, a fim de ver se adicionando essa informação e, após isso, removendo o excesso com a seleção de *features* traria ganho, mas também não houve melhora no resultado.

Outro teste feito foi a aplicação do algoritmo T-SNE para reduzir a quantidade de elementos na entrada do classificador. Ao aplicar diretamente, a acurácia caía para valores inferiores a 10%, após isso, foram feitos outros testes, onde aplicávamos em diversas combinações, representadas na Figura 9.

As diferentes implementações representadas na Figura 9 são, redução de dimensionalidade nos pontos-chave de cada vídeo antes da clusterização, redução na quantidade

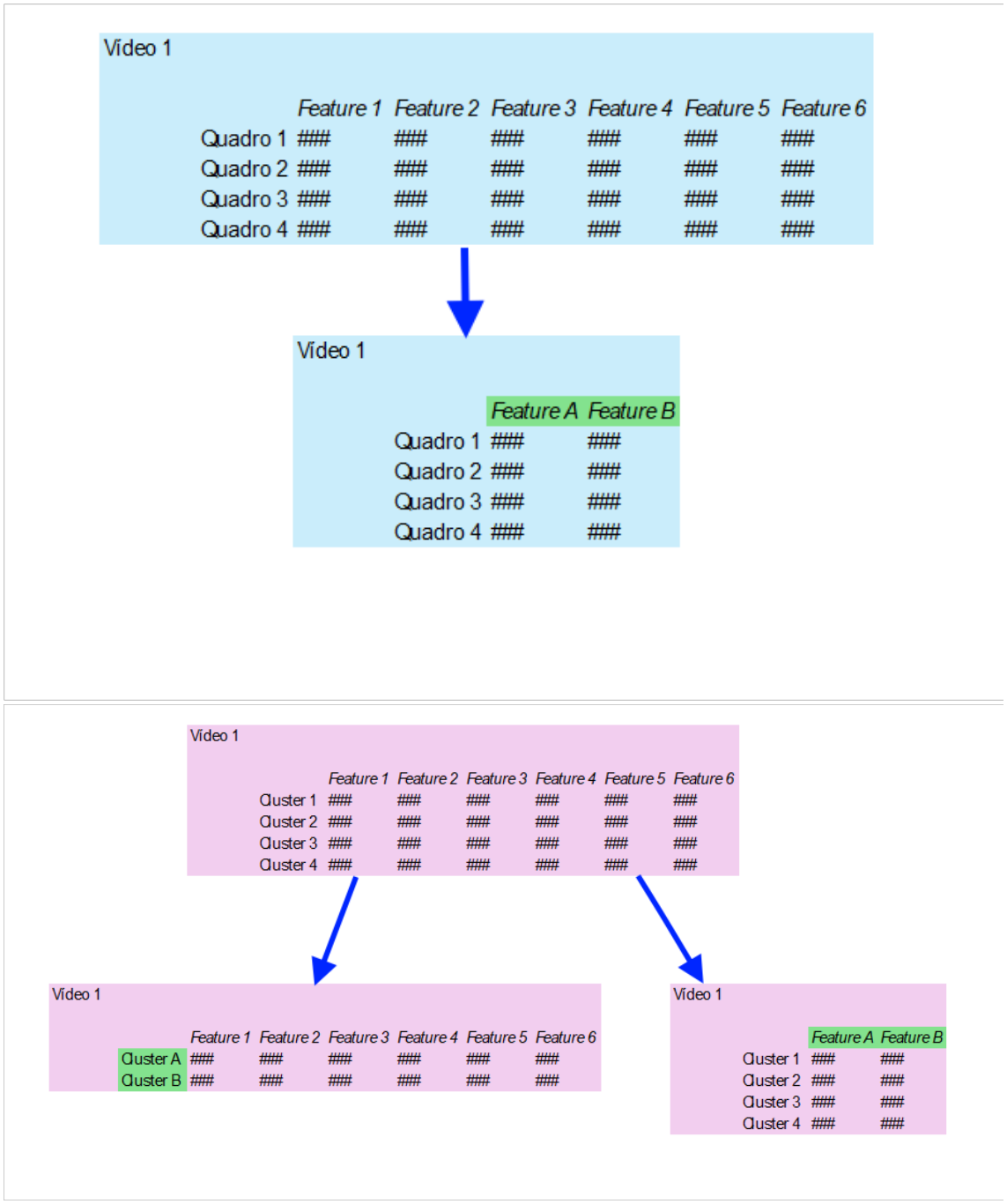


Figura 9 – 3 diferentes usos implementadas dos T-SNE, as tabelas em azul representam os dados antes da clusterização, em lilás representam os dados após a clusterização, as dimensões em verde representam a dimensão reduzida pelo T-SNE
Fonte: Elaboração própria

de *clusters* após a clusterização e redução no número de *features* dos *clusters*. O melhor resultado nessa implementação foi de 26,55% no teste e 56,68% no treino.

Por fim, foram testadas diversas implementações utilizando o modelo *XGBoost*, utilizando seleção de *features*, geração de dados artificiais e T-SNE, tendo a melhor acurácia medida sendo 88,28% no teste e 98,85% no treino.