Machine learning approaches for efficient recognition of Brazilian Sign Language

Letícia M. G. de Morais
Computational Intelligence Laboratory - CiLab
Universidade Federal Rural do Semi-Árido, Brasil
leticia.morais30463@alunos.ufersa.edu.br

Welizangela M. Almeida
Univerisdade Estadual do Rio Grande do Norte
Mossoró, Brasil
welizangela_almeida@outlook.com

Rosana C. B. Rego
Departamento de Engenharias e Tecnologia
Universidade Federal Rural do Semi-Árido, Brasil
rosana.rego@ufersa.edu.br

ABSTRACT

Context: The Brazilian Sign Language (LIBRAS) is the primary language for the Brazilian Deaf community. However, the lack of LIBRAS interpreters presents challenges for the inclusion of individuals with hearing impairments in society. Additionally, learning a sign language can be as difficult for hearing individuals as learning a foreign language. Problem: There is no official translator between Brazilian Portuguese and LIBRAS, making communication between hearing and Deaf individuals challenging. Solution: This study presents machine learning algorithms that can recognize 44 LIBRAS signs and translate them in real time to Brazilian Portuguese. IS Theory: This work is grounded in the Diffusion of Innovations Theory, focusing on how new technologies are adopted within communities. The proposed LIBRAS recognition system serves as an innovative solution to enhance communication between hearing and Deaf individuals. Method: This research adopts an experimental approach with quantitative and descriptive analyses. Data were collected from three different datasets, representing a total of 44 distinct LIBRAS signs. MediaPipe Hands was used to extract the spatial coordinates of hand movements. Then, some machine learning models were employed to recognize the signs, comparing their performance based on accuracy, F1 Score, precision, and recall metrics. Summary of Results: The results indicated that the ExtraTree algorithm was the best choice for the 44 signs dataset, achieving an accuracy of 98.01%, with an F1 Score of 98.01%, precision of 98.02%, and recall of 98.01%. Contributions and Impacts in IS Area: This paper contributes a low-cost approach for sign language recognition that requires only a camera to detect hands and recognize signs.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; *Computer vision*; • **Human-centered computing** → *Accessibility systems and tools*; • **Information systems** → *Information systems applications.*

KEYWORDS

Artificial Intelligence, Neural Networks, Machine Learning, Brazilian Sign Language.

1 INTRODUÇÃO

De acordo com a Federação Mundial de Surdos (*World Federation of the Deaf* - WFD), a surdez acarreta barreiras na comunicação que frequentemente limitam a plena participação dos indivíduos surdos na sociedade [19]. Para a comunidade surda brasileira, a Língua Brasileira de Sinais (LIBRAS) é a primeira língua. Embora tenha sido reconhecida como língua nacional pela Lei nº 10.436 em 2002, a comunicação entre ouvintes e pessoas com deficiência auditiva ainda enfrenta desafios significativos, inserindo os surdos no grupo linguístico-cultural minoritário [19].

De acordo com o IBGE, cerca de 5% da população brasileira apresenta algum grau de deficiência auditiva, somando aproximadamente 10 milhões de pessoas. Dessas, quase três milhões têm surdez completa [12]. Embora não existam dados exatos sobre o número de conhecedores da LIBRAS, estima-se que eles representem uma porcentagem muito pequena em relação aos mais de 200 milhões de habitantes do país.

Conforme Phillipson et al. [21], a privação dos direitos humanos linguísticos não apenas limita a comunicação, mas também pode resultar na privação de outros direitos fundamentais, como o acesso à educação, saúde e participação política. A luta por direitos linguísticos é, portanto, uma luta pela plena cidadania e pelo reconhecimento da identidade cultural e social das pessoas surdas.

A inteligência artificial (IA) pode contribuir para o reconhecimento e tradução da Língua de sinais, tornando a comunicação mais acessível [3]. Modelos de aprendizado de máquina (*Machine Learning* - ML) podem ser treinados para identificar e interpretar gestos e expressões faciais, transformando sinais em texto ou fala em tempo real.

Diante disso, este trabalho investiga modelos de aprendizado de máquina para o reconhecimento automático de sinais de LIBRAS utilizando o *framework MediaPipe*. Essa ferramenta permite extrair coordenadas espaciais das mãos, viabilizando uma abordagem com baixo custo computacional para facilitar a comunicação entre surdos e ouvintes.

Dessa forma, as principais contribuições deste artigo são:

i) Desenvolvimento de um modelo ML para o reconhecimento de sinais em LIBRAS, utilizando apenas uma câmera para capturar dados espaciais 3D das mãos, o que facilita a implementação em ambientes com recursos limitados e sem a necessidade de equipamentos caros ou infraestrutura complexa.
ii) Integração de três *datasets* abertos e diversos de sinais LIBRAS, resultando em um conjunto de dados com 44 sinais, o que amplia as possibilidades de pesquisa para desenvolvedores e pesquisadores interessados em avançar o reconhecimento de sinais em LIBRAS, contribuindo para a melhoria de sistemas de tradução e interação humano-computador.
iii) Aplicação de modelos de ML como o *Extra Trees*, para a classificação de sinais, representando um avanço nos métodos

utilizados na área de sistemas da informação para reconhecimento de gestos, com foco em soluções escaláveis e de baixo custo, o que facilita a adoção de tais sistemas em diferentes plataformas e dispositivos.

O restante deste artigo está dividido em quatro seções, como descrito a seguir. A Seção 2 aborda a fundamentação teórica e os trabalhos relacionados. Em seguida, a Seção 3 apresenta a metodologia do trabalho. Na seção 5 são discutidos os resultados obtidos. Por fim, a seção 6 apresenta as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

A Teoria de Difusão de Inovações, de Rogers [25], descreve como novas tecnologias são adotadas dentro de comunidades, considerando fatores como vantagens percebidas, compatibilidade e facilidade de uso. Nesse contexto, essa teoria contribuiu para esta pesquisa ao permitir uma análise da aceitação de um sistema de reconhecimento de sinais em LIBRAS baseado em ML.

Nesta seção, são apresentados os trabalhos relacionados, destacando as pesquisas existentes na literatura que aplicam aprendizado de máquina para o reconhecimento da LIBRAS.

2.1 TRABALHOS RELACIONADOS

A Tabela 1 apresenta um resumo dos trabalhos relevantes que usaram modelos de aprendizado de máquina para o reconhecimento dos sinais LIBRAS.

Table 1: Trabalhos relacionados

Artigo	Modelo	Acurácia	Tamanho	Classes
[3]	CNN	96%	8400	30
[16]	CNN	99%	224.000	26
[9]	KNN	95%	3.881	36
[26]	CNN	98,57%	4.000	10
[23]	CNN	93,3%	1.200	20
[6]	Random Forest	94,72%	1.200	20

O trabalho [3], com uma acurácia de 96%, utiliza um *dataset* de 8.400 imagens para reconhecimento de 30 sinais do alfabeto LIBRAS. Este estudo destaca-se pelo bom desempenho. Da mesma maneira, [16], realiza o reconhecimento dos sinais do alfabeto com uma CNN, obtendo uma melhor acurácia de 99%. Os autores utilizaram um *dataset* maior com 224.000 imagens.

O trabalho [23], embora tenha uma acurácia um pouco mais baixa (93,3%), oferece uma contribuição significativa ao disponibilizar um banco de dados público com 1.200 amostras de vídeos para o reconhecimento de 20 sinais. Assim como os trabalhos [3] e [16], o trabalho [9] utiliza um *dataset* de 36 classes de sinais estáticos, sendo eles 26 do alfabeto e os 10 restantes números (0-9). O conjunto de dados fora feito pelos próprios autores e distingue-se ao apresentar a construção de uma luva para a extração de dados por meio de sensores. Além disso, utiliza o modelo KNN (*K-nearest neighbors*) para classificar os dados, o qual atingiu uma acurácia de 95%.

O trabalho [26] apresenta uma CNN capaz de reconhecer os sinais dos números 0 a 9 da LIBRAS. Para isso, os autores criaram o próprio banco de dados com 4.000 imagens, onde o modelo de CNN desenvolvido obteve uma acurácia de 98,57%. Já o trabalho [6] utiliza o modelo de classificação Random Forest e o conjunto de dados disponibilizado pelo trabalho [23], MINDS-Libras [1]. Apesar de utilizarem o mesmo *dataset*, o modelo Random Forest apresentou uma acurácia maior (94,72%) do que a dos autores do conjunto de dados (93,3%). Este Projeto destaca-se devido ao método de extração de dados utilizado, o modelo *Pose* do *framework MediaPipe*.

Diferente dos trabalhos existentes na literatura, o estudo proposto visa contribuir para a comunicação inclusiva aplicando abordagens de ML no reconhecimento eficiente de sinais em LIBRAS. Foram integrados três *datasets*, incluindo os de Carneiro et al. [7], Almeida et al. [2], e Almeida et al. [1], resultando em um conjunto de dados com 44 sinais. Para a extração dos dados espaciais 3D das mãos, foi utilizado o *framework MediaPipe*, seguido pela aplicação de modelos de classificação como *Extra Trees* (ET), LightGBM, *Random Forest* (RF), Análise Discriminante Quadrática (QDA - *Quadratic Discriminant Analysis*), Classificador de *Gradient Boosting* (GBC - Gradient Boosting Classifier), KNN, Árvore de Decisão (DT - *Decision Tree*), Análise Discriminante Linear (LDA - *Linear Discriminant Analysis*), Regressão Logística (LR - *Logistic Regression*), Regressão Ridge (*Ridge Regression*), Máquina de Vetores de Suporte (SVM - *Support Vector Machine*), Naive Bayes (NB) e AdaBoost. Ao contrário dos estudos [6, 9, 23, 26], este trabalho apresenta a implementação de um modelo que alcançou uma acurácia de 98% na classificação de 44 classes.

A abordagem proposta destaca-se por ser de baixo custo, necessitando apenas de uma câmera para a captura dos pontos das mãos, sem depender de equipamentos adicionais ou do alto processamento computacional exigido por métodos como o de Rezende et al. [23], que utiliza redes neurais convolucionais 3D (CNN-3D), ou o de dos Santos et al. [9], que requer a construção de uma luva tradutora.

3 METODOLOGIA

Esta pesquisa adotou uma abordagem experimental com análises quantitativas e descritivas. O objetivo foi desenvolver e avaliar um modelo de aprendizado de máquina para o reconhecimento de sinais de LIBRAS a partir de coordenadas espaciais extraídas de imagens. Para isso, o método de pesquisa foi estruturado em etapas, conforme descrito a seguir.

3.1 LEVANTAMENTO DO CONJUNTO DE DADOS

Devido à limitação de dados temporais para sinais LIBRAS, a escolha de mais de uma base de dados foi necessária para que o modelo pudesse reconhecer um maior quantitativo de sinais. Este estudo reuniu três conjunto de dados, sendo eles: *Brazilian Sign Language - Words Recognition*[7], *Libras-10*[2] e *MINDS-Libras*[1]. Os três *datasets* foram extraídos de plataformas públicas para os quais foram doados, sendo o primeiro da *Kaggle* e os dois últimos da *Zenodo*.

A Tabela 2 apresenta o total de classes e a quantidade de dados em vídeo que cada um dos três *datasets* utilizados possuem.

É importante mencionar que o conjunto de dados *Brazilian Sign Language - Words Recognition* e o *MINDS-Libras* possuem uma classe em comum, o sinal de "Banco". Portanto, o banco de dados final deste trabalho possui um total de 44 classes.

Table 2: Dados dos *Datasets* utilizados

Dataset	Classes	Quantidade de Dados
Brazilian Sign Language - Words Recognition	15	140 vídeos
Libras-10	10	100 vídeos
MINDS-Libras	20	1.200 vídeos

3.1.1 Extração de atributos. O módulo *Hands* do *framework MediaPipe* foi utilizado neste projeto para extrair os atributos necessários. Este módulo utiliza uma *Pipeline* de *Machine Learning* que faz os uso de dois modelos combinados: um de detecção de palma de mão e um outro que extrai 21 pontos de referência tridimensionais (3D) da mão detectada a partir de um único *frame*[29], como mostra a Figura 1.

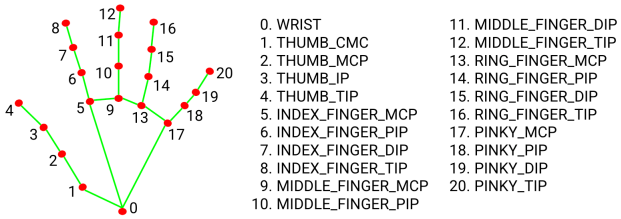


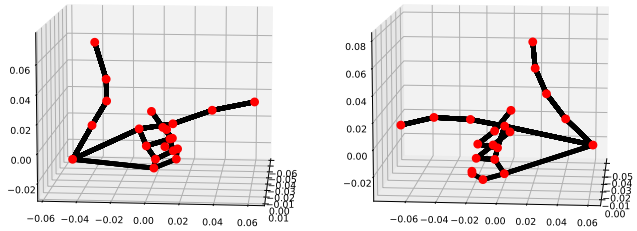
Figura 1: Pontos de referência de uma mão detectada no módulo *Hands* [10].

Dessa forma, as funções do *Hands* permitiram a criação de um algoritmo o qual fora utilizado para coletar os principais atributos dos vídeos dos três *datasets* anteriormente mencionados. Este algoritmo de extração armazena as coordenadas espaciais (X, Y e Z) dos 21 pontos-chaves de cada mão detectada em um quadro. Além disso, o mesmo algoritmo classifica as mãos em direita ou esquerda, para que possa ser feita uma diferenciação entre as duas.

A Figura 2 apresenta o sinal "Sortudo" do *dataset Libras-10* com os pontos-chaves detectados pelo modelo de extração de do *MediaPipe* de cada mão destacados.

Figura 2: Pontos-chave detectados em um quadro do sinal "Sortudo" do *dataset Libras-10*[2].

A Figura 3 mostra as coordenadas espaciais dos pontos-chaves extraídos do sinal "Sortudo" da Figura 2 para cada mão: direita (a) e esquerda (b).



(a) **(b)**

Figura 3: Coordenadas espaciais dos pontos de referência da Figura 2 : (a) Coordenadas espaciais da mão esquerda e (b) Coordenadas espaciais da mão direita.

É importante destacar que o algoritmo de extração apresenta comportamentos distintos dependendo da quantidade de mãos detectadas no quadro. Quando são detectadas duas mãos de classificações diferentes (uma esquerda e uma direita), as coordenadas dos pontos-chave de ambas são extraídas normalmente. Caso seja detectada apenas uma mão, o algoritmo classifica e extrai suas coordenadas, atribuindo as coordenadas da mão ausente como 0. Se o modelo de detecção de palma não encontrar nenhuma mão ou identificar duas mãos com a mesma classificação, o quadro é descartado e seus dados não são utilizados.

3.1.2 Divisão e balanceamento de Dados. Após o processo de extração dos atributos, as coordenadas foram armazenadas em um *Dataframe* com a biblioteca *Pandas*. Este *Dataframe* possui 127 colunas e 40.347 linhas. Das colunas, 63 delas correspondem aos atributos da mão direta, as outras 63 correspondem aos atributo da mão esquerda e a última equivale à classe ao qual aquele conjunto de mãos pertence. As linhas representam os dados extraídos de cada quadro dos vídeos dos três *datasets*. Ao fim da construção do *dataset* de 44 classes, seus dados foram divididos em 80% para o conjunto de treino e 20% para o conjunto de teste.

Como diferentes bases de dados foram utilizadas para a criação do *dataset* final, é normal que ele possua um desbalanceamento entre suas classes. A Figura 4 exibe a quantidade de dados (grupo de mãos) correspondentes a cada classe no conjunto de treino. É possível notar que a classe "Formação" é a que possui um maior quantitativo de dados, enquanto que a classe "Eu" possui o menor.

Para resolver o problema de desbalanceamento de classes, foi aplicada a técnica SMOTE (*Synthetic Minority Oversampling Technique*), que gera amostras sintéticas para as classes minoritárias, de modo a equilibrar a distribuição dos dados em relação à classe majoritária [5]. A Figura 5 ilustra a distribuição das classes após a aplicação do SMOTE. Com o uso dessa técnica, as classes com menor número de amostras receberam dados sintéticos, permitindo que alcançassem a mesma quantidade de dados da classe majoritária.

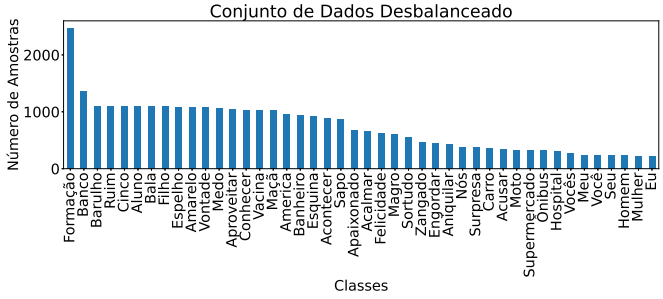


Figura 4: Conjunto de treino desbalanceado.

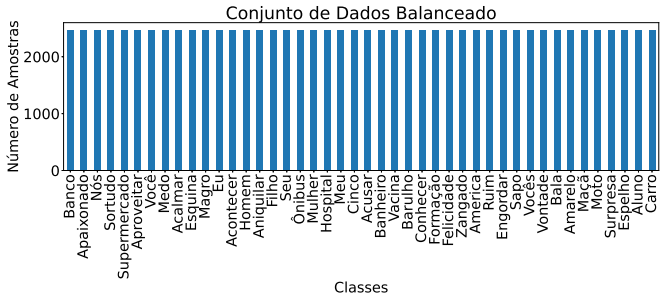


Figura 5: Conjunto de treino balanceado com o uso da técnica SMOTE.

3.2 Estudo sobre Modelos de *Machine Learning*

Nesta seção, alguns modelos de ML, incluindo árvore de decisão, *Extra Trees*, *LightGBM*, *Random Forest* e KNN são apresentados.

3.2.1 Árvore de decisão. A árvore de decisão é um modelo que pode ser descrito como uma série de decisões baseadas em características dos dados [11]. A árvore de decisão escolhe a melhor característica para dividir os dados com base em métricas de impureza [22]. Existem duas métricas comuns para isso, neste trabalho o índice de Gini foi utilizado [30]. O índice de Gini mede a impureza de um conjunto de dados e é dado por

$$\text{Gini}(D) = 1 - \sum_{c} p(c|D)^2 \tag{1}$$

onde $p(c|D)$ é a probabilidade de classe c no conjunto D. Após a divisão com a característica X_i, o índice de Gini para o conjunto D é dado por:

$$\text{Gini}(D|X_i) = \sum_{v \in \{0,1\}} \frac{|D_v|}{|D|} \text{Gini}(D_v) \tag{2}$$

O objetivo da árvore de decisão é escolher a característica e o limiar t que minimizam o índice de Gini ou maximizam o ganho de informação. Na simulação, o parâmetro *splitter* foi definido como '*best*', o que significa que o algoritmo escolhe a melhor divisão em cada nó. O parâmetro de profundidade máxima foi definido como *None*, ou seja, não há limite de profundidade, permitindo que a árvore expanda até que todas as folhas sejam puras ou contenham menos do que o mínimo de amostras necessário para dividir, que foi definido como 2.

3.2.2 Extra Trees. *Extra Trees* (*Extreme Randomized Trees*) é um modelo de ML baseado em um conjunto de árvores de decisão, mas que introduz maior aleatoriedade durante o processo de treinamento [27]. Para cada árvore T_i, os dados de treinamento D são divididos aleatoriamente em subamostras. Em cada divisão do nó, em vez de selecionar a melhor divisão com base em uma métrica como a entropia ou o índice de Gini, o *Extra Trees* escolhe a divisão de forma aleatória, o que acelera o processo e aumenta a diversidade entre as árvores [18].

$$X_j = \arg\min(\delta) \tag{3}$$

em que δ é o critério aleatório. Cada árvore de decisão no *ensemble* faz uma predição $\hat{y}_i$ para uma dada entrada x. As previsões de todas as árvores são então combinadas, dadas por

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i \tag{4}$$

onde N é o número total de árvores no modelo. Na simulação, o parâmetro *n-estimators* foi definido como 100, indicando que o *ensemble* contém 100 árvores de decisão. O critério de qualidade da divisão, *criterion*, foi definido como 'gini', o que faz com que o algoritmo utilize o índice de Gini para avaliar a pureza das divisões em cada nó. Assim como na implementação da árvore de decisão, a profundidade máxima das árvores no *extra tree* foi definida como *None*. O parâmetro *min-samples-split* foi estabelecido como 2. O número mínimo de amostras para que um nó seja considerado folha foi definido como 1 por meio do parâmetro *min-samples-leaf*.

3.2.3 LightGBM. *LightGBM* (*Light Gradient Boosting Machine*) é um algoritmo de *boosting* baseado em árvores de decisão [14]. Uma série de árvores de decisão são construídas de forma sequencial para minimizar uma função de perda L através de uma otimização de gradiente [14]. A função de perda L para um modelo $F(x)$ com um conjunto de dados $\{(x_i, y_i)\}$ é dada por

$$L = \sum_{i=1}^{n} \ell(y_i, F(x_i)) \tag{5}$$

em que n é o número de amostras, y_i é o rótulo verdadeiro para a amostra i, $F(x_i)$ é a previsão do modelo para a amostra x_i, ℓ é uma função de perda, como o erro quadrático para regressão ou a entropia cruzada para classificação. Para cada árvore t no modelo, o objetivo é minimizar a função de perda em relação ao gradiente da iteração anterior $F_{t-1}(x)$.

Para cada árvore adicionada, *LightGBM* ajusta o modelo de acordo com o gradiente da função de perda. A atualização do modelo $F_t(x)$ na iteração t é dada por:

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x) \tag{6}$$

em que η é a taxa de aprendizado e $h_t(x)$ representa a nova árvore de decisão ajustada em t. Na simulação, o número de árvores foi definido como 100, o que significa que o modelo usará 100 iterações de árvores para fazer previsões. A profundidade máxima das árvores foi definida como -1, indicando que não há limite de profundidade. O número de folhas em cada árvore foi configurado como 31. A taxa de aprendizado foi definida como 0.1, o que reduz a contribuição de cada árvore sucessiva para o modelo, ajudando a melhorar a generalização.

3.2.4 Random Forest. O *Random Forest* é um algoritmo de ML de *ensemble* que cria um conjunto de árvores de decisão[20]. Ele utiliza o princípio de *bagging* (*bootstrap aggregating*) para aumentar a precisão e reduzir o sobreajuste, combinando os resultados de múltiplas árvores de decisão independentes [17, 24]. No *Random Forest*, cada árvore T_i é treinada em uma amostra D_i do conjunto de dados D, obtida por *bootstrap sampling*. A amostra D_i é gerada aleatoriamente com reposição e pode conter duplicatas.

$$D_i = \{(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{in}, y_{in})\} \tag{7}$$

Cada árvore gera uma previsão $\hat{y}_i$ independente. No caso de um problema de classificação, a previsão final $\hat{y}$ é obtida pelo voto majoritário das previsões individuais das k árvores T_i:

$$\hat{y} = \text{mode}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k) \tag{8}$$

onde mode representa a classe mais frequente entre as previsões individuais. Foram utilizadas 100 árvores de decisão para simulação. Além disso, o índice de Gini foi utilizado para avaliar a qualidade das divisões em cada nó. Já o parâmetro *bootstrap* foi definido como *True*, o que significa que as amostras para cada árvore são retiradas com reposição, garantindo diversidade entre as árvores do *ensemble*.

3.2.5 KNN. O *K-Nearest Neighbors* (KNN) é um algoritmo utilizado em problemas de classificação e regressão [13, 15]. O algoritmo identifica os k vizinhos mais próximos de um novo ponto de entrada, com base em uma medida de distância, e utiliza esses vizinhos para determinar a previsão final [8, 28].

A distância entre o ponto de entrada x e um ponto de treinamento x_i foi calculada usando a distância euclidiana. Para dois pontos em um espaço n-dimensional, a distância euclidiana $d(x, x_i)$ é dada por

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{ij})^2} \tag{9}$$

em que $x = (x_1, x_2, \ldots, x_n)$ é o vetor de entrada, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ é o vetor de um ponto de treinamento. Para classificação dos sinais, o algoritmo KNN conta o número de vizinhos pertencentes a cada classe entre os k vizinhos mais próximos do ponto x. A classe atribuída é aquela que possui a maioria dos vizinhos (voto majoritário). Logo, seja C o conjunto de classes e N_c o número de vizinhos pertencentes à classe c, então a previsão $\hat{y}$ é

$$\hat{y} = \arg\max_{c \in C} N_c \tag{10}$$

onde N_c é o número de vizinhos entre os k mais próximos que pertencem à classe c.

Na simulação, foi considerado os 5 vizinhos mais próximos para fazer a previsão. Também foi considerado que todos os vizinhos têm o mesmo peso na decisão final.

4 RESULTADOS E DISCUSSÕES

Os modelos de classificação foram desenvolvidos em um ambiente computacional composto por um processador *Intel ® Core™* I5-9300H (2.40GHz), uma GPU *NVIDIA GeForce GTX* 1650, 12GB de memória RAM e sistema operacional Windows 10. Quanto às tecnologias utilizadas, adotaram-se a linguagem *Python 3* e o *framework Scikit-learn*, versão 1.4.2.

Foram utilizadas as seguintes métricas de desempenho para avaliar os modelos: acurácia, área sob a curva (AUC), *recall*, precisão e *F1 Score*. Em cada uma dessas métricas, valores mais altos indicam uma melhor performance, refletindo o desempenho do modelo em classificar corretamente as diferentes classes. Além disso, também fora analisado o tempo de treinamento dos modelos, onde quanto menor o resultado, melhor a eficiência computacional e escalabilidade do modelo.

A Tabela 3 mostra a performance dos modelos quanto às 15 classes advindas do conjunto de dados *Brazilian Sign Language - Words Recognition*[7]. Dentre os modelos treinados, o modelo *Extra Trees* (ET) apresentou o melhor desempenho em suas métricas. Os modelos *Light Gradient Boosting Machine* (LightGBM) e *Random Forest* (RF) foram os que conseguiram chegar mais próximo dos resultados do *Extra Trees* na métricas de acurácia, AUC, *Recall*, precisão e *F1 Score*.

Table 3: Métricas dos modelos com o conjunto de dados *dataset Brazilian Sign Language - Words Recognition*[7].

Modelo	Acurácia	AUC	Recall	Precisão	F1	Tempo
ET	0.9978	1.0000	0.9978	0.9978	0.9978	1.7660
LightGBM	0.9973	1.0000	0.9973	0.9973	0.9973	15.0350
RF	0.9972	1.0000	0.9972	0.9972	0.9972	17.1390
QDA	0.9970	0.0000	0.9970	0.9973	0.9971	0.9300
GBC	0.9945	0.0000	0.9945	0.9945	0.9945	1262.6060
KNN	0.9923	0.9991	0.9923	0.9924	0.9924	2.2990
DT	0.9906	0.9994	0.9906	0.9909	0.9909	1.3200
LDA	0.9778	0.0000	0.9778	0.9778	0.9778	0.2090
LR	0.9451	1.0000	0.9451	0.9448	0.9444	3.9620
Ridge	0.9451	1.0000	0.9451	0.9452	0.9445	0.3090
SVM	0.8957	0.0000	0.8957	0.8957	0.8908	0.7800
NB	0.8810	0.9929	0.8810	0.8836	0.8801	0.4320
AdaBoost	0.1456	0.1456	0.1456	0.1012	0.0912	18.5600

A Tabela 4 apresenta os resultados quando foram adicionadas as classes do conjunto de dados *Libras-10*[2] ao primeiro *dataset*, totalizando 25 classes. Houve um aumento no tempo de treinamento dos modelos devido ao acréscimo de mais exemplos, assim como houve uma pequena redução nas outras métricas. Porém, assim como no conjunto de 15 classes, o modelo *Extra Trees* (ET) foi o que obteve as melhores métricas, demonstrando uma generalização entre as classes, um baixo nível de erro e um tempo de treinamento rápido em relação ao demais modelos.

Na Tabela 5 são exibidos as performances do modelo quando o último *dataset* é adicionado, totalizando 44 classes. Nota-se que os modelos obtiveram resultados menores devido ao aumento do vocabulário. O que é um desempenho esperado, visto que com a adição de mais classes, os modelos precisam distinguir entre nuances e características específicas de mais categorias. Ainda assim, o modelo *Extra Trees* continua sendo o que possui as melhores métricas em relação aos demais, demonstrando estabilidade e robustez ao possuir métricas de avaliação com valores próximos.

Também foi medido o tempo de inferência dos três melhores modelos com 44 classes (*Extra Trees*, *Random Forest* e *K-Nearest Neighbors*) para avaliar sua eficiência. O tempo de inferência médio (em segundos) foi calculado com os mesmos 100 exemplos. A Tabela 6 apresenta os resultados desse cálculo, evidenciando que o modelo

Table 4: Métricas dos modelos com os conjuntos de dados *Brazilian Sign Language - Words Recognition*[7] e *Libras-10*[2].

Modelo	Acurácia	AUC	Recall	Precisão	F1	Tempo
ET	0.9977	1.0000	0.9977	0.9977	0.9977	4.1620
LightGBM	0.9973	1.0000	0.9973	0.9974	0.9973	37.4090
RF	0.9971	1.0000	0.9971	0.9971	0.9970	22.7340
KNN	0.9937	0.9993	0.9937	0.9938	0.9936	4.8000
GBC	0.9923	0.0000	0.9923	0.9924	0.9923	2149.1620
DT	0.9865	0.9929	0.9865	0.9864	0.9864	5.5990
LDA	0.9478	0.0000	0.9478	0.9495	0.9479	2.7070
LR	0.9145	0.0000	0.9145	0.9152	0.9136	3.9400
Ridge	0.9048	0.0000	0.9048	0.9087	0.9019	0.2740
QDA	0.8666	0.0000	0.8666	0.8183	0.8326	13550
SVM	0.8657	0.0000	0.8657	0.8704	0.8577	1.1980
NB	0.8313	0.9828	0.8313	0.8460	0.8293	0.7030
Ada	0.1217	0.0000	0.1217	0.0897	0.0591	23.3970

Table 5: Métricas dos modelos utilizando o conjunto de teste final utilizando *Brazilian Sign Language - Words Recognition*[7], *Libras-10*[2] e *MINDS-Libras*[1].

Modelo	Acurácia	AUC	Recall	Precisão	F1	Tempo
ET	0.9801	0.9995	0.9801	0.9802	0.9801	12.4980
RF	0.9761	0.9994	0.9761	0.9762	0.9761	58.7820
KNN	0.9478	0.9954	0.9478	0.9481	0.9477	7.7130
DT	0.8937	0.9456	0.8937	0.8938	0.8935	13.6200
LightGBM	0.8886	0.9622	0.8886	0.8949	0.8877	98.9300
QDA	0.8042	0.0000	0.8042	0.8002	0.7911	1.4410
LDA	0.6869	0.0000	0.6869	0.7440	0.6979	1.3500
LR	0.6293	0.0000	0.6293	0.6610	0.6269	4.9200
Ridge	0.5960	0.0000	0.5960	0.5792	0.5606	0.3300
SVM	0.5597	0.0000	0.5597	0.6960	0.5428	6.5800
NB	0.5576	0.9214	0.5576	0.6305	0.5585	1.0080
Ada	0.1000	0.0000	0.1000	0.1148	0.0659	39.7990

ET também se sobressai sobre os demais ao apresentar o menor tempo de inferência entre os modelos avaliados.

Table 6: Tempo de inferência médio dos 3 melhores modelos com 44 classes para 100 exemplos.

Modelo	Inferência (s)
ET	0.002730
RF	0.004525
KNN	0.012364

A Figura 6 apresenta as predições do modelo *Extra Trees* para sinais do *dataset* completo. Além disso, na Figura 6 também são mostradas os pontos-chaves detectados pelo modelo *MediaPipe*. No sinal de "Esquina", percebe-se que uma das mãos não são detectadas pelo modelo de extração de pontos-chaves. Como citado anteriormente, isso ocorre devido a esta mão estar parcialmente fora do campo de visão da câmera.

Na Figura 7 são expostas as predições do modelo no algoritmo de reconhecimento de sinais da LIBRAS em tempo real para os sinais de: "Carro" (a), "Ônibus" (b), "Moto" (c) e "Sapo"(d). Para o usuário, as predições dos sinais feitos são exibidas no canto superior esquerdo.

Figura 6: Pontos-chaves capturados pelo modelo do *MediaPipe* e predições feitas pelo modelo *Extra Trees*.

Além disso, também são expostas os pontos-chaves das mãos que estão presentes no quadro.

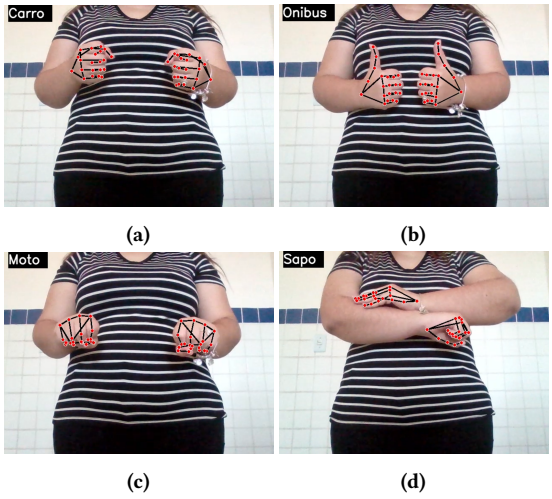


Figura 7: Predições do modelo em tempo real: (a) sinal de "Carro", (b) sinal de "Ônibus", (c) sinal de "Moto" e (d) sinal de "Sapo".

A Figura 8 representa a matriz de confusão normalizada do modelo *Extra Trees* quanto ao *dataset* completo de 44 classes. A normalização foi feita de acordo com as predições do modelo, permitindo avaliar a precisão de cada classe predita. Dessa forma, o quanto maior a taxa de acerto na diagonal principal, maior é a precisão do modelo naquela classe. Já os valores fora da diagonal principal indicam erros cometidos pelo modelo.

Com isso, os resultados apontam um bom desempenho, tendo em vista a quantidade de previsões na diagonal principal em relação aos outros itens da matriz. Das 44 classes, 25 delas são preditas com mais de 90% de precisão pelo modelo proposto. Classes que possuem necessidade de expressões ou posições específicas do braço tendem a possuir uma taxa de assertividade menor, devido ao modelo capturar somente as posições das mãos. Um exemplo desse caso são os resultados das palavras "Maçã", "Felicidade", "Zangado" e "Sapo".

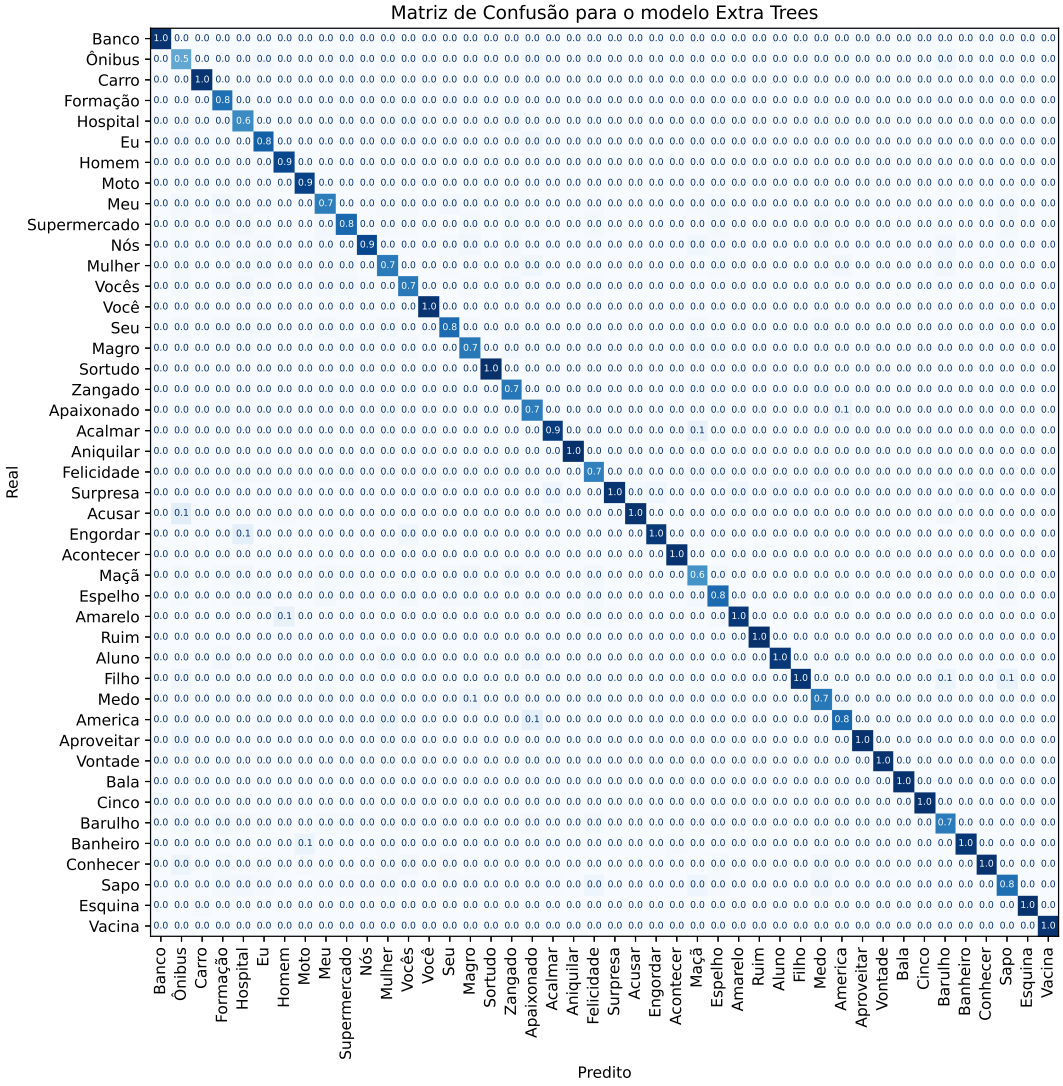


Figura 8: Matriz de Confusão do Modelo *Extra Trees*

A precisão do modelo também decaí devido à classes em que os sinais são feitos por movimentos que deixam partes da mão fora do campo de visão da câmera. Isso acontece devido ao modelo do *MediaPipe* fazer previsões com base em ambientes bidimensionais para gerar os pontos-chave em três dimensões. O sinal de "Mulher", por exemplo, faz com que o polegar e o indicador fiquem fora do campo de visão da câmera, perdendo a precisão da captura de pontos-chave.

5 CONCLUSÃO

A partir dos resultados obtidos, é possível concluir que o modelo *Extra Trees* apresentou um desempenho geral bom, com alto desempenho nas previsões, segundo as métricas de avaliação adotadas, especialmente para a maioria das classes. A matriz de confusão normalizada revelou que 25 das 44 classes foram preditas com 98% de precisão, o que demonstra a eficácia do modelo em capturar padrões de movimento das mãos para a maioria dos sinais. No entanto, as classes que envolvem movimentos complexos ou que exigem expressões faciais e posições específicas do braço, como as palavras "Maçã", "Felicidade", "Zangado" e "Sapo", apresentaram taxas de acerto mais baixas, devido à dificuldade do modelo em identificar esses sinais com precisão.

O trabalho demonstrou que com o uso de tecnologias como visão computacional e aprendizado de máquina, os sistemas de informação podem reconhecer sinais em LIBRAS e transformá-los em texto, ou mesmo realizar a tradução reversa, permitindo que mensagens em português sejam interpretadas e apresentadas em LIBRAS.

Dessa forma, o presente artigo alinha-se ao quarto desafio dos Grandes Desafios da Pesquisa em Sistemas de Informação no Brasil

2016-2026 [4], que enfatiza a necessidade de uma abordagem sociotécnica no desenvolvimento de Sistemas de Informação. Seguindo essa visão, esta pesquisa contribui para a inclusão digital e social da comunidade surda, destacando o papel da tecnologia na redução de barreiras de comunicação.

Para trabalho futuros, modelos de linguagem serão integrados para estruturação das frases geradas na tradução LIBRAS-Português. Essa integração tem o potencial de beneficiar setores como a educação, atendimento ao cliente, saúde e até o entretenimento, promovendo uma comunicação mais inclusiva e eliminando barreiras. Os materiais e modelos desenvolvidos neste trabalho estão disponíveis em https://github.com/cilab-ufersa/ml_libras.

AGRADECIMENTOS

Ao Laboratório de Inteligência Computacional (CILab) pelo suporte à pesquisa e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro por meio da bolsa de Iniciação Científica.

REFERÊNCIAS

[1] Sílvia G. M. Almeida, Tamires M. Rezende, Gabriela T. B. Almeida, Andreia C. R Toffolo, and Frederico G. Guimarães. 2019. *MINDS-Libras Dataset.* https://doi.org/10.5281/zenodo.2667329

[2] Silvia Grasiella Moreira Almeida, Tamires Martins Rezende, Andreia Chagas Rocha Toffolo, and Cristiano Leite de Castro. 2019. *Libras-10 Dataset.* https://doi.org/10.5281/zenodo.3229958

[3] N. F. Andrade Junior, A. A. B. F. Pinto, W. M. Almeida, and Rosana C. B. Rego. 2024. Brazilian Sign Language Translation: AI for the Inclusion of Deaf People. In *Anais do V Congresso Brasileiro Interdisciplinar em Ciência e Tecnologia.* V Congresso Brasileiro Interdisciplinar em Ciência e Tecnologia.

[4] Clodis Boscarioli, Renata M. Araujo, and Rita Suzana P. Maciel. 2017. *I GranDSI-BR – Grand Research Challenges in Information Systems in Brazil 2016-2026.* Brazilian Computer Society (SBC), Brazil. 184 pages.

[5] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. 2011. SMOTE: Synthetic Minority Over-sampling Technique. *CoRR* abs/1106.1813 (2011). arXiv:1106.1813 http://arxiv.org/abs/1106.1813

[6] Eros Caiafa, Allan Basilio, Amaro de Lima, and Gabriel Araujo. 2023. Interpretação de gestos de Libras usando K-means e Random Forest. In *XLI Simpósio Brasileiro De Telecomunicações E Processamento De Sinais (SBrT2023).* Biblioteca Da SBrT. https://doi.org/10.14209/sbrt.2023.1570923868

[7] AL Cavalcante Carneiro, L Brito Silva, and DH Pinheiro Salvadeo. 2021. Efficient sign language recognition system and dataset creation method based on deep learning and image processing. In *Thirteenth International Conference on Digital Image Processing (ICDIP 2021)*, Vol. 11878. SPIE, 11–19.

[8] Rodrigo A de Freitas Vieira and Clodoaldo A de Moraes Lima. 2018. Channel selection for EEG-based biometric recognition. In *Proceedings of the XIV Brazilian Symposium on Information Systems.* 1–8.

[9] Herdney Souza dos Santos, Leila Fabiola Ferreira, and Poliana Gonçalves Leite Alves. 2019. Luva Tradutora da Língua Brasileira de Sinais. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Centro Universitário Internacional Uninter. Orientador: Prof. Me. Charles Way Hun Fung.

[10] Google AI Edge. [n. d.]. Guia de detecção de pontos de referência manuais. https://ai.google.dev/edge/mediapipe/solutions/hands/. Acesso em: 29 de setembro de 2024.

[11] Sam Fletcher and Md Zahidul Islam. 2019. Decision tree classification with differential privacy: A survey. *ACM Computing Surveys (CSUR)* 52, 4 (2019), 1–33.

[12] Karina Freitas. 2021. Dia Internacional da Linguagem de Sinais procura promover a inclusão de pessoas surdas.

[13] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *Twenty-Second International Joint Conference on Artificial Intelligence.*

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[15] Oliver Kramer and Oliver Kramer. 2013. K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors* (2013), 13–23.

[16] D.F.L. Lima, A.S. Salvador Neto, E.N. Santos, T.M.U. Araujo, and T.G. Do Rêgo. 2019. Using convolutional neural networks for fingerspelling sign recognition in Brazilian sign language. In *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, WebMedia 2019.*

[17] Yanli Liu, Yourong Wang, and Jian Zhang. 2012. New machine learning algorithm: Random forest. In *Information Computing and Applications: Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3.* Springer, 246–252.

[18] Saulo Martiello Mastelini, Felipe Kenji Nakano, Celine Vens, André Carlos Ponce de Leon Ferreira, et al. 2022. Online extra trees regressor. *IEEE Transactions on Neural Networks and Learning Systems* 34, 10 (2022), 6755–6767.

[19] World Federation of the Deaf (WFD). 24AD. UN Enable - Promoting the rights of Persons with Disabilities - Contribution by WFD. https://www.un.org/esa/socdev/enable/rights/contrib-wfd.htm

[20] Aakash Parmar, Rakesh Katariya, and Vatsal Patel. 2019. A review on random forest: An ensemble classifier. In *International conference on intelligent data communication technologies and internet of things (ICICI) 2018.* Springer, 758–763.

[21] Robert Phillipson, M. Rannut, and Tove Skutnabb-Kangas. 1994. *Linguistic Human Rights: Overcoming Linguistic Discrimination.* Tove Skutnabb-Kangas. http://www.tove-skutnabb-kangas.org/dl/128-Phillipson-R-Rannut-M-Skutnabb-Kangas-T-1994-Intro-Linguistic-Human-Rights-Overcoming-Linguistic-Discrimination.pdf

[22] J. Ross Quinlan. 1996. Learning decision tree classifiers. *ACM Computing Surveys (CSUR)* 28, 1 (1996), 71–72.

[23] Tamires Martins Rezende, Sílvia Grasiella Moreira Almeida, and Frederico Gadelha Guimarães. 2021. Development and validation of a Brazilian sign language database for human gesture recognition. *Computação Neural e Aplicações* (2021).

[24] Steven J Rigatti. 2017. Random forest. *Journal of Insurance Medicine* 47, 1 (2017), 31–39.

[25] E.M. Rogers. 1983. *DIFFUSION OF INNOVATIONS 3RD E REV.* Free Press. https://books.google.com.br/books?id=pXRkAAAAIAAJ

[26] Adriel Santos, Iago Bacurau, Jayne de Morais Silva, Talles Viana, and Robson Feitosa. 2019. Rede Neural Artificial Convolucional Aplicada ao Reconhecimento de Configuração de Mão nos Símbolos de 0 a 9 da Língua Brasileira de Sinais (LIBRAS). 21–24. https://doi.org/10.5753/sbsi.2019.7432

[27] Aakanksha Sharaff and Harshil Gupta. 2019. Extra-tree classifier with metaheuristics approach for email classification. In *Advances in Computer Communication and Computational Sciences: Proceedings of IC4S 2018.* Springer, 189–197.

[28] Luis Alvaro de Lima Silva, Lori RF Machado, and Leonardo Emmendorfer. 2024. A Case and Cluster-Based Framework for Reuse and Prioritization in Software Testing. In *Proceedings of the 20th Brazilian Symposium on Information Systems.* 1–10.

[29] Andrey Vakunov, Chuo-Ling Chang, Fan Zhang, George Sung, Matthias Grundmann, and Valentin Bazarevsky. 2020. MediaPipe Hands: On-device Real-time Hand Tracking. https://mixedreality.cs.cornell.edu/workshop.

[30] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. 2020. Zero knowledge proofs for decision tree predictions and accuracy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.* 2039–2053.