RECONHECIMENTO DE SINAIS DINÂMICOS DA LÍNGUA BRASILEIRA DE SINAIS POR MEIO DE REDES NEURAIS CONVOLUCIONAIS

João Paulo Moreira Rodrigues[1], Marlon de Matos de Oliveira[2]

Resumo: O reconhecimento automático da Língua Brasileira de Sinais representa um desafio relevante para a promoção da inclusão e acessibilidade de pessoas surdas na sociedade. Em especial, o reconhecimento de sinais dinâmicos, que envolvem movimentos temporais, ainda demanda soluções computacionais mais eficazes. Este trabalho tem como objetivo desenvolver um modelo eficiente para o reconhecimento das letras representadas por sinais dinâmicos. Foram avaliadas cinco arquiteturas de Redes Neurais Convolucionais: AlexNet, DenseNet121, InceptionV3, ResNet18 e VGG16Net, para identificar a mais eficaz na detecção desses sinais. A ResNet18 apresentou o melhor desempenho geral, com uma acurácia média de 74,29%, destacando-se na captura de características complexas, enquanto a AlexNet e a VGG16Net também mostraram boa eficácia. Embora outros estudos tenham adotado modelos mais sofisticados, como redes LSTM ou 3D-CNNs, obtendo acurácias superiores a 90%, este trabalho optou por uma abordagem mais simples. Ainda assim, os resultados obtidos demonstraram um desempenho relevante, especialmente diante das limitações do conjunto de dados utilizado. A pesquisa contribui para o avanço no reconhecimento de sinais dinâmicos, reforçando seu potencial impacto na promoção da inclusão e acessibilidade de pessoas surdas.

Palavras-chave: visão computacional; língua brasileira de sinais; redes neurais convolucionais; aprendizado de máquina; reconhecimento de sinais.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), joaopaulomoreirarodrigues@unesc.net
[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), marlon.oliveira@unesc.net

ABSTRACT: Automatic recognition of Brazilian Sign Language (LIBRAS) represents a significant challenge in promoting the inclusion and accessibility of deaf individuals in society. In particular, the recognition of dynamic signs, which involve temporal movements, still demands more effective computational solutions. This study aims to develop an efficient model for recognizing letters represented by dynamic signs. Five Convolutional Neural Network (CNN) architectures were evaluated—AlexNet, DenseNet121, InceptionV3, ResNet18, and VGG16Net—to identify the most effective in detecting these signals. ResNet18 achieved the best overall performance, with an average accuracy of 74.29%, standing out in capturing complex features, while AlexNet and VGG16Net also demonstrated good effectiveness. Although other studies have employed more sophisticated models, such as LSTM networks or 3D-CNNs, achieving accuracies above 90%, this work adopted a simpler approach. Even so, the results obtained demonstrated relevant performance, especially given the limitations of the dataset used. This research contributes to advances in dynamic sign recognition, reinforcing its potential impact on promoting the inclusion and accessibility of deaf individuals.

Keywords: computer vision; brazilian sign language; convolutional neural networks; machine learning; sign language recognition.

1 INTRODUÇÃO

A inclusão de pessoas com deficiência é um desafio global e uma prioridade para sociedades que buscam maior equidade e acessibilidade. Para pessoas com deficiência auditiva, a comunicação é uma das barreiras mais significativas, pois muitas dependem de uma linguagem visual, como a Língua de Sinais, para interagir (Neto, 2018).

A escassez de fluência em Língua de Sinais entre a população em geral torna a inclusão mais difícil, pois limita a comunicação e o aprendizado, especialmente em fases iniciais do desenvolvimento (Moeller, 2000). Essa limitação reforça a necessidade de soluções tecnológicas que facilitem a comunicação entre surdos e ouvintes.

As línguas de sinais variam entre culturas, assim como as línguas orais e possuem estruturas gramaticais próprias. No Brasil, a Língua Brasileira de Sinais (LIBRAS) foi oficialmente reconhecida como meio de comunicação e expressão pela Lei nº 10.436 de 2002 (Brasil, 2002). Esse reconhecimento marca um avanço importante para a comunidade surda

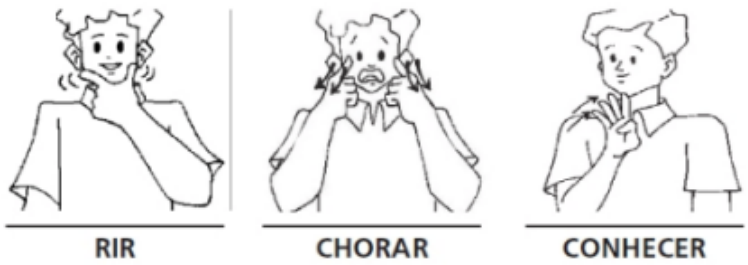
brasileira, consolidando o direito ao uso de LIBRAS e sua inclusão em contextos sociais e institucionais. No entanto, o acesso aos serviços de tradução em LIBRAS ainda é limitado, uma vez que depende principalmente de intérpretes humanos, tornando-se um serviço caro e nem sempre disponível.

Segundo Oliveira (2018), uma alternativa para tornar a interpretação mais acessível seria através da utilização de um sistema de baixo custo, por exemplo, um sistema que utilize uma câmera comum. Para que esse sistema seja eficaz, é fundamental compreender que os sinais em LIBRAS se dividem em duas categorias: sinais estáticos, sem movimento e com posição fixa das mãos (Figura 1) e sinais dinâmicos, que envolvem movimentos das mãos com variações de forma, orientação e localização ao longo do tempo (Figura 2) (Cristiano, 2024).

Figura 1 - Representação dos sinais sem movimento.

Fonte: Felipe e Monteiro (2007).

Figura 2 - Representação dos sinais com movimento.



Fonte: Felipe e Monteiro (2007).

O reconhecimento preciso dos sinais dinâmicos representa um desafio técnico significativo, pois requer a análise constante das variações dos gestos.

A pesquisa sobre o reconhecimento de sinais tem avançado com o uso de aprendizado profundo. Amaral et al. (2019) exploraram combinações de Redes Neurais Convolucionais e Recorrentes (LRCNs) e Re-

des Neurais Convolucionais 3D (3D-CNNs) para reconhecer sinais dinâmicos em LIBRAS, utilizando imagens de profundidade. O estudo alcançou alta precisão, indicando que tarefas mais complexas poderiam se beneficiar destes modelos.

Outra pesquisa foi conduzida por Das, Biswas e Purkayastha (2023), que aplicaram uma abordagem híbrida, combinando CNNs e Redes Neurais de Memória de Longo e Curto Prazo (LSTM) bidirecional para o reconhecimento de palavras da Língua de Sinais Indiana (ISL). O modelo apresentou alta eficiência ao lidar com características espaciais e temporais dos gestos, mas enfrentou dificuldades com gestos similares, indicando a necessidade de melhorias no reconhecimento de sinais mais complexos.

No estudo de Buttar et al. (2023), voltado para a Língua de Sinais Americana (ASL), foi adotada uma combinação do modelo *You Only Look Once* – versão 6 (YOLOv6) para sinais estáticos e LSTM para sinais dinâmicos. Embora a técnica tenha mostrado bons resultados na detecção de sinais estáticos, os sinais dinâmicos apresentaram variações na precisão, o que destacou os desafios em garantir previsões consistentes em gestos com maior variabilidade temporal.

O trabalho de Grossi e Ferreira (2024) focou na tradução de sinais de LIBRAS para texto, utilizando uma abordagem híbrida de CNNs e Redes Neurais Recorrentes (RNNs), que apresentou bons resultados, com *transformers* alcançando o melhor desempenho. No entanto, houve dificuldades na seleção de quadros-chave para sinais dinâmicos, o que causou inconsistências na classificação entre sinalizadores diferentes, sugerindo que técnicas de extração mais avançadas poderiam melhorar a precisão do modelo.

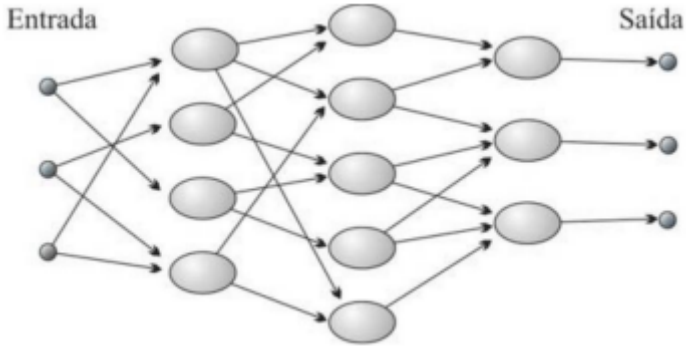
Diante desse panorama, o presente trabalho busca contribuir para o avanço das pesquisas ao focar especificamente no reconhecimento dos sinais mais complexos do alfabeto de LIBRAS. O estudo propõe o uso de CNNs para capturar as variações desses gestos. Por meio de técnicas de pré-processamento e otimizações na arquitetura, espera-se desenvolver um modelo eficaz no reconhecimento desses sinais, trazendo uma contribuição relevante para as pesquisas neste tema.

Com os avanços da inteligência artificial, especialmente no campo do aprendizado de máquina (ML), novas possibilidades surgem para a interpretação de LIBRAS. O uso de Redes Neurais Artificiais (ANNs) permite que sistemas identifiquem padrões complexos em dados visuais, sendo as redes neurais convolucionais uma das abordagens mais eficazes para re-

conhecimento de imagens (Rezende, 2021).

As ANNs são inspiradas no cérebro humano (Figura 3), com neurônios interligados que permitem o processamento de informações e o aprendizado (Haykin, 2001). Dentro das ANNs, as Redes Neurais Convolucionais (CNNs) são eficazes no reconhecimento de padrões visuais, sendo aplicadas em tarefas como classificação de imagens (LeCun; Kavukcuoglu; Farabet, 2010). Para dados temporais, como gestos dinâmicos, utilizam-se as RNNs e as LSTMs, que capturam a sequência e a continuidade das informações

Figura 3 - Representação simplificada de uma rede neural artificial.



Fonte: Ferneda (2006).

As CNNs emergem como uma solução promissora para a interpretação da linguagem de sinais, possibilitando o reconhecimento eficaz de padrões visuais em imagens e vídeos, fator essencial para a captação de sinais dinâmicos.

No entanto, ao lidar com gestos dinâmicos, como os sinais em LIBRAS, as CNNs enfrentam desafios adicionais, pois esses gestos exigem não apenas a análise de características visuais em quadros individuais, mas também a interpretação de sequências temporais para capturar movimentos contínuos e complexos. A necessidade de captar tanto a continuidade espacial quanto temporal dos gestos é fundamental para uma identificação precisa (Das; Biswas; Purkayastha, 2023).

Embora existam pesquisas focadas em gestos dinâmicos, a maioria dos estudos até o momento se concentra predominantemente na interpretação de gestos estáticos, por serem menos complexos. Em comparação, gestos dinâmicos, que envolvem movimentos complexos e sequências temporais, têm sido explorados em menor escala. Desenvolver uma interpretação precisa para esses gestos dinâmicos pode ser um avanço crucial para a comunicação em LIBRAS, ampliando as oportunidades de inclusão e acessibilidade para pessoas surdas.

Este trabalho foi organizado da seguinte forma: inicialmente, são apresentados os recursos e metodologias adotadas para o reconhecimento dos sinais dinâmicos da Língua Brasileira de Sinais. Em seguida, os resultados obtidos são discutidos, analisando o desempenho dos modelos e suas limitações. O trabalho é concluído com uma síntese dos principais achados e sugestões para futuras pesquisas, visando o aprimoramento da inclusão e acessibilidade.

2 MATERIAIS E MÉTODOS

O uso de Redes Neurais Convolucionais (CNNs) para interpretar sinais dinâmicos apresenta desafios significativos, uma vez que esses modelos precisam identificar padrões visuais em cada quadro dos sinais. Embora exista uma vasta quantidade de pesquisas focadas no reconhecimento de sinais estáticos, o estudo de sinais dinâmicos ainda enfrenta obstáculos que exigem maior investigação.

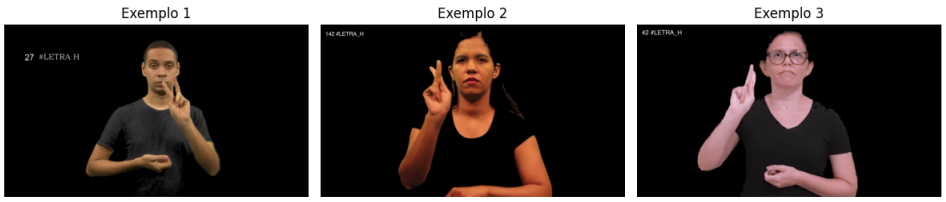
Este estudo explora uma abordagem baseada em CNNs, com o objetivo de avaliar sua capacidade de reconhecer sinais dinâmicos. São investigadas diferentes arquiteturas e técnicas de processamento para mitigar os desafios associados a esse tipo de reconhecimento.

2.1 BASE DE DADOS

Foram utilizadas duas bases de dados, selecionadas de acordo com sua qualidade e diversidade. A base Pernambuco (2024) inclui vídeos na resolução de 1920x1080 *pixels*, com taxa de 30 quadros por segundo e uma duração média de 5 segundos por vídeo, no formato MP4, representando todas as letras do alfabeto da LIBRAS que são expressas por sinais dinâmicos, sendo elas: J, H, K, X, Y e Z. Para cada letra, são disponibilizadas três amostras de vídeos, sendo que, em cada uma dessas amostras, o sinal é executado pelas mesmas três pessoas distintas, como pode ser observado na Figura 4, contabilizando dezoito vídeos no total.

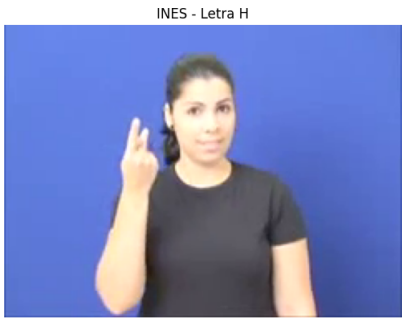
A base Surdos (2024) apresenta uma qualidade de captura inferior à da base Pernambuco (2024), como pode ser observado na Figura 5, contendo vídeos na resolução de 240x276 *pixels*, com taxa de 12 quadros por segundo e uma duração média de 3 segundos por vídeo, também no formato MP4. Esta base abrange todas as letras do alfabeto da LIBRAS mencionadas anteriormente, exceto a letra Y, com apenas uma amostra de cada letra, executada pela mesma pessoa, totalizando cinco vídeos.

Figura 4 - *Frames* das três amostras da representação da letra H em LIBRAS.



Fonte: Pernambuco (2024).

Figura 5 - *Frame* da representação da letra H em LIBRAS.



Fonte: Surdos (2024).

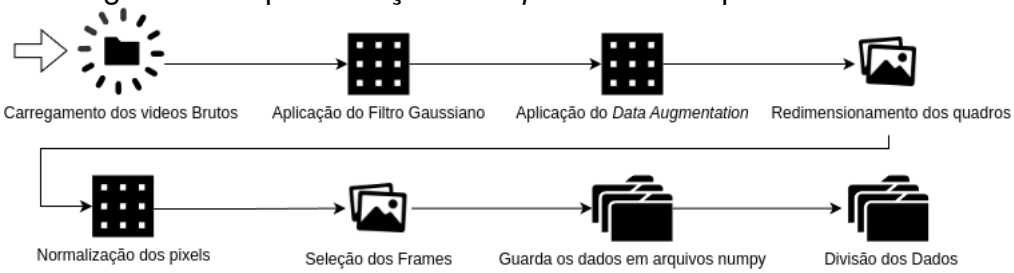
Quando as duas bases são combinadas, obtem-se um total de vinte e três amostras, sendo cinco vídeos na resolução de 240x276 e dezoito vídeos na resolução de 1920x1080 *pixels*.

2.2 PRÉ-PROCESSAMENTO DE DADOS

Foi realizado um tratamento inicial nos vídeos brutos com o objetivo de isolar os sinais a serem analisados. Esse processo envolveu o corte dos vídeos, feito com a ferramenta Shotcut (2025), removendo trechos irrelevantes, de modo a preservar apenas o intervalo correspondente à sinalização.

Após o tratamento inicial, os vídeos foram submetidos no *pipeline* de pré-processamento com o objetivo de padronizar os dados e prepará-los para as etapas subsequentes da análise, conforme ilustrado na Figura 6. Esse *pipeline* foi implementado em Python, utilizando, as bibliotecas OpenCV, NumPy e PyTorch.

Figura 6 - Representação do *Pipeline* de Pré-processamento.



Fonte: Elaborado pelo autor.

Inicialmente, cada vídeo foi carregado e convertido para a escala de cinza (*grayscale*) para reduzir a complexidade computacional do modelo. Em seguida, os vídeos foram redimensionados para a resolução de 224x224 pixels e convertidos em sequências de quadros (*frames*), com a extração de 20 quadros por sinal. Um filtro Gaussiano foi aplicado em cada quadro para suavizar ruídos e realçar bordas e regiões de interesse. A Figura 7 ilustra alguns destes quadros extraídos.

Figura 7 - *Frames* da representação da letra H em LIBRAS.

Fonte: Pernambuco (2024).

Na etapa seguinte, os valores dos *pixels* foram normalizados para o intervalo [0, 1], dividindo-se todos por duzentos e cinquenta e cinco, a fim de estabilizar o treinamento da rede neural e evitar valores extremos.

Para vídeos com menos de vinte quadros, foram adicionados quadros artificiais em preto (matrizes de zeros). Nos casos em que a quantidade de quadros excedeu vinte, foi realizada uma amostragem espaçada ao longo da sequência, de modo a preservar a distribuição temporal e a dinâmica do movimento representado pelo sinal.

A técnica de *data augmentation* foi aplicada (Figura 8) com o intuito de ampliar a diversidade do conjunto de dados e, consequentemente, melhorar a capacidade de generalização do modelo. As transformações realizadas introduziram variações nos dados, preservando as características essenciais dos sinais, o que permitiu ao modelo aprender a reconhecê-los sob diferentes condições e perspectivas. A partir de um conjunto inicial de

vinte e três amostras, foram geradas dez variações para cada uma, totalizando duzentas e trinta amostras utilizadas no treinamento.

As sequências dos quadros processados foram armazenadas como arrays no formato NumPy, organizadas de forma a manter a estrutura temporal necessária para o reconhecimento dos sinais.

Figura 8 – *Frames* da representação da letra H em LIBRAS após a aplicação da técnica de *data augmentation*.

Fonte: Pernambuco (2024).

Após esse processo, os arquivos NumPy foram organizados em três diretórios distintos, correspondentes aos conjuntos de treino, validação e teste.

A divisão dos dados foi realizada de forma que 80% dos dados foram destinados ao treinamento, 10% à validação e 10% ao teste, utilizando a técnica de estratificação para garantir que as classes fossem distribuídas de maneira balanceada em cada conjunto. No contexto do aprendizado de máquina, é comum adotar uma divisão semelhante (Salazar et al., 2022).

2.3 ARQUITETURAS ESCOLHIDAS

Foram selecionadas cinco arquiteturas de redes neurais convolucionais, cada uma com características únicas que oferecem vantagens distintas para diferentes aspectos do problema de reconhecimento de sinais em LIBRAS.

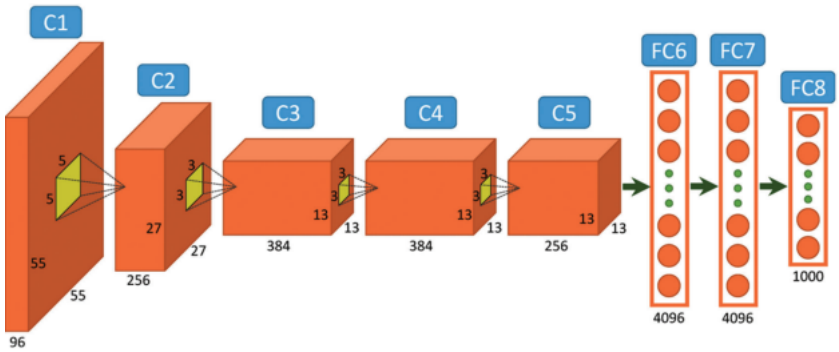
2.3.1 AlexNet

A AlexNet é conhecida por sua eficácia em problemas de classificação de imagens, especialmente quando o objetivo é detectar padrões em dados complexos.

A arquitetura original (Figura 9) consiste em cinco camadas convolucionais, seguidas de camadas totalmente conectadas e utiliza funções de ativação *ReLU* e técnicas como *dropout* para evitar *overfitting*. Para este

caso, a arquitetura foi adaptada para lidar com imagens em escala de cinza, com a camada inicial sendo ajustada para aceitar entradas com apenas um canal, convertidas internamente para três canais.

Figura 9 - Arquitetura geral da AlexNet.



Fonte: Collet (2017).

As camadas totalmente conectadas da arquitetura original foram ajustadas para uma maior eficiência. O número de unidades foi reduzido, começando com 512, seguida por uma camada intermediária com 256 unidades, o que ajuda a equilibrar a capacidade do modelo com a complexidade computacional e ao mesmo tempo evita o *overfitting*.

Para estabilizar e acelerar o treinamento, foi adicionada uma camada de *Batch Normalization* após a primeira camada totalmente conectada. As camadas finais consistem em três camadas totalmente conectadas, com *dropout* de 0.5 e 0.3 para regularização, visando reduzir o risco de *overfitting* durante o treinamento.

Esta arquitetura utilizou-se do otimizador *AdamW*, uma versão aprimorada do algoritmo *Adaptive Moment Estimation* (*Adam*). *Adam* é um método de otimização baseado em gradiente, amplamente utilizado no treinamento de redes neurais. Ele ajusta automaticamente a taxa de aprendizado de cada parâmetro com base na média móvel dos gradientes e de seus quadrados, o que contribui para uma convergência mais eficiente e estável (Kingma; Ba, 2017).

A variante *AdamW* corrige uma limitação do algoritmo original ao separar corretamente o termo de regularização, melhorando ainda mais a estabilidade do treinamento (Loshchilov; Hutter, 2019).

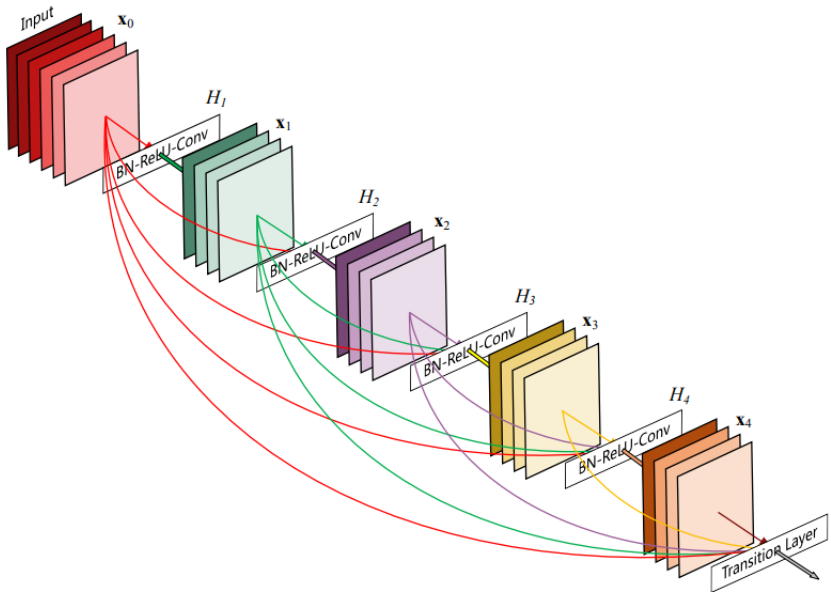
O modelo foi treinado com taxa de aprendizado de $1e-3$, proporcionando um bom desempenho durante o treinamento.

2.3.2 DenseNet121

A DenseNet121 adota uma arquitetura onde cada camada recebe entradas de todas as camadas anteriores, promovendo maior reutilização de características e facilitando a propagação do gradiente. Essa abordagem permite que o modelo aproveite melhor as informações extraídas nas camadas iniciais, melhorando o aprendizado e a capacidade de capturar detalhes importantes.

A Figura 10 ilustra esse princípio de conectividade densa, mostrando como cada camada dentro de um bloco recebe como entrada a saída de todas as camadas anteriores.

Figura 10 – Bloco denso da DenseNet, com conexões entre todas as camadas.



Fonte: Huang et al. (2018).

A arquitetura foi configurada com as camadas convolucionais padrão da DenseNet121, que foram mantidas congeladas durante o treinamento para reduzir o custo computacional. A saída dessas camadas é ajustada por meio de uma camada de *pooling* adaptativo, que garante um tamanho fixo de saída, independentemente da resolução da imagem de entrada.

Utilizou-se o ativador *ReLU* (*Rectified Linear Unit*) na etapa de classificação, uma função que zera os valores negativos e mantém os positivos, permitindo que a rede aprenda representações mais complexas de forma rápida e estável (Goodfellow; Bengio; Courville, 2016).

Com isso, a camada de classificação foi redesenhada com três camadas totalmente conectadas: a primeira com 1024 unidades, seguida

por normalização em lote (*Batch Normalization*), ativação *ReLU* e *dropout* de 0.5; a segunda com 512 unidades, também seguida por *ReLU* e *dropout* de 0.5; e, por fim, uma camada de saída com número de unidades correspondente ao número de classes.

O modelo foi treinado com taxa de aprendizado de $1e-5$, utilizando o otimizador *AdamW*. Para otimizar o uso de memória e acelerar o treinamento, foi empregada a técnica de *mixed precision*, permitindo maior eficiência sem comprometer a precisão do modelo.

2.3.3 InceptionV3

A InceptionV3 é uma rede profunda que utiliza múltiplos filtros convolucionais de diferentes tamanhos na mesma camada, permitindo a captura de informações em várias escalas. Sua arquitetura também incorpora convoluções 1x1 (*bottleneck*), reduzindo o número de parâmetros e melhorando a eficiência computacional.

A InceptionV3 combina camadas convolucionais com filtros de diferentes tamanhos para capturar informações em diferentes escalas espaciais. A saída dessas camadas é reduzida por um *pooling* adaptativo, que diminui a dimensionalidade mantendo as informações mais relevantes.

A camada de classificação final consiste em uma camada totalmente conectada com 512 unidades, seguida por *Batch Normalization*, ativação *ReLU*, *dropout* de 0.5 e uma camada de saída com o número de classes correspondente.

O treinamento foi realizado utilizando o otimizador *AdamW*, com taxa de aprendizado diferenciada: $1e-4$ para as camadas de classificação e $1e-5$ para os blocos finais da Inception (Mixed_7), que foram liberados para *fine-tuning*, que é um ajuste mais fino dessas camadas já pré-treinadas. A técnica de *mixed precision* foi utilizada para melhorar a eficiência do treinamento sem comprometer a precisão.

2.3.4 ResNet

A ResNet adota a ideia de conexões residuais, permitindo que a rede se aprofunde sem enfrentar o problema de *vanishing gradients*. Essa arquitetura é composta por blocos residuais que ajudam a rede a aprender representações mais complexas e a preservar informações importantes durante o treinamento.

A arquitetura foi adaptada para processar imagens em escala de cinza, com a primeira camada convolucional modificada para aceitar entradas com 1 canal. A rede possui 18 camadas, organizadas em 4 blocos

residuais. Após o último bloco, a saída passa por uma camada de *pooling* adaptativo para gerar uma saída fixa.

A camada de classificação é composta por uma camada totalmente conectada com 256 unidades, seguida por normalização em lote (*Batch Normalization*), ativação *ReLU* e uma camada de *dropout* de 0.3 para regularização, evitando o *overfitting*.

O treinamento foi realizado utilizando uma taxa de aprendizado de $1e-4$, com o otimizador *AdamW* e com uso de *mixed precision* para maior eficiência computacional.

2.3.5 VGG16Net

A VGG16 é uma arquitetura profunda e simples, caracterizada pelo uso de pequenas convoluções (3x3) e camadas de *pooling máximas* (2x2). Sua estrutura permite uma boa capacidade de extração de características com um número considerável de camadas convolucionais seguidas de camadas totalmente conectadas.

O modelo VGG16 foi adaptado para entradas de 1 canal (escala de cinza), mantendo as camadas convolucionais pequenas que são uma característica principal da arquitetura. As últimas camadas convolucionais da rede foram descongeladas para permitir o *fine-tuning*, ajustando o modelo às necessidades específicas do problema.

Foi utilizada uma camada de *pooling* adaptativo para ajustar a saída da rede para um tamanho fixo, adequado para a classificação. A camada de classificação é composta por uma camada densa com 1024 unidades, seguida por *Batch Normalization*, ativação *ReLU*, *dropout* de 0.5, uma segunda camada com 512 unidades, ativação *ReLU*, *dropout* de 0.3 e uma camada final com o número de classes.

O treinamento foi realizado utilizando uma taxa de aprendizado de $1e-4$ e o otimizador *AdamW*.

2.4 TREINAMENTO DAS ARQUITETURAS

O treinamento das redes neurais convolucionais (CNNs) foi realizado utilizando o conjunto de dados pré-processado, dividido em treinamento, validação e teste. Para garantir a comparabilidade entre as diferentes arquiteturas, foram definidos hiperparâmetros consistentes, como o número de passagens completas pelo conjunto de dados (*epochs*), a taxa de aprendizado e a função de perda.

Durante o treinamento, os modelos foram otimizados utilizando

variantes do otimizador *Adam*, reconhecido por sua capacidade de convergência e robustez. A função de perda adotada foi a *CrossEntropyLoss* (PyTorch, 2025), que é comumente utilizada em tarefas de classificação multiclasse. Essa função de perda ajusta os pesos com base na frequência das classes no conjunto de treinamento, ajudando a lidar com o desbalanceamento entre as classes.

O treinamento foi realizado ao longo de 25 *epochs*, com monitoramento contínuo da acurácia e da perda tanto no conjunto de treino quanto no conjunto de validação. A cada *epoch*, o modelo era avaliado no conjunto de teste e em algumas arquiteturas, como a VGG16, o ajuste era auxiliado por técnicas como o *scheduler*, que é a redução automática da taxa de aprendizado e ajusta essa taxa conforme o desempenho do modelo, ajudando a melhorar os resultados ao longo do treinamento.

Ao final de cada *epoch*, as métricas de desempenho, como acurácia geral e acurácia por classe, foram calculadas e registradas em um arquivo de relatório para análise posterior.

Essa abordagem de treinamento foi executada com os dados organizados em *data loaders* para otimizar o processo de alimentação do modelo. Utilizou-se o embaralhamento dos dados para o conjunto de treinamento, assegurando a aleatoriedade na seleção de amostras a cada *epoch*, Já os conjuntos de validação e teste foram mantidos sem embaralhamento, garantindo consistência nas avaliações e medindo de forma precisa o desempenho do modelo em dados não vistos.

2.5 AVALIAÇÃO DOS MODELOS

A avaliação incluiu a acurácia geral do modelo, que representa a razão entre o número total de acertos (A) e o número total de exemplos avaliados (T), ou seja, a proporção de predições corretas em relação ao total de amostras no conjunto de teste.

$$Acc = \frac{A}{T}$$

Além disso, foi realizada a análise da acurácia por classe, fornecendo uma visão mais detalhada do desempenho de cada arquitetura.

Também foram utilizadas outras métricas de avaliação, como *precision*, definida como a razão entre o número de predições corretas para uma determinada classe (PC) e o total de predições feitas para essa classe (TP):

$$Prec = \frac{PC}{TP}$$

A métrica *recall* considera a razão entre as predições corretas (PC) e o total de exemplos reais da classe (TR):

$$Rec = \frac{PC}{TR}$$

Já o *F1-score* é calculado como a média harmônica entre *precision* e *recall*:

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec}$$

As métricas *precision*, *recall* e *F1-score* foram calculadas utilizando a média *macro*, que considera o desempenho de cada classe de forma independente, atribuindo pesos iguais a todas elas na média final.

Também foi utilizada a matriz de confusão para o modelo com melhor desempenho como ferramenta de apoio. A matriz permite visualizar, de forma compacta, quantas vezes cada classe foi corretamente reconhecida e em quais situações ocorreram erros de classificação. Com isso, torna-se possível identificar padrões de confusão entre as classes e avaliar o comportamento do modelo.

Os resultados de cada avaliação foram registrados em um relatório, que incluiu todas as métricas mencionadas anteriormente, assim como as taxas de acerto para cada sinal.

3 DISCUSSÃO E RESULTADOS

Cada arquitetura apresentou um desempenho distinto, refletindo suas características específicas e a maneira como lidaram com a complexidade dos sinais. As análises revelaram tanto as forças quanto as limitações de cada modelo, proporcionando uma compreensão sobre como cada arquitetura reconhece os sinais dinâmicos do alfabeto de LIBRAS.

A AlexNet obteve uma acurácia geral de 45,71% na validação e 60% no teste final. No teste final, o desempenho foi superior nas letras J, X e Y enquanto as letras H e K apresentaram algumas dificuldades, a letra Z foi a que mais enfrentou problemas de reconhecimento.

Em relação às outras métricas, a *precision* de 59% indica que, entre todas as vezes em que o modelo classificou uma amostra como pertencente a uma determinada classe, 59% dessas predições estavam corre-

tas, evidenciando uma taxa considerável de erros. O *recall* de 60% revela que o modelo foi capaz de identificar corretamente 60% dos exemplos reais de cada classe. Já o *F1-score* apresentou 58% do desempenho ideal no equilíbrio entre precisão e revocação, sugerindo que o modelo teve dificuldade em manter regularidade no reconhecimento das diferentes classes.

A DenseNet121 alcançou uma acurácia geral de 34.29% na validação e 42.86% no teste final. No teste final, o modelo teve um bom desempenho em algumas letras, como H, K e Y. No entanto, as letras J, X e Z apresentaram dificuldades, com taxas de acerto mais baixas.

Já na análise das demais métricas, a *precision* de 42,31%, o *recall* de 44,44% e o *F1-score* de 40,47% indicam que o modelo enfrentou dificuldades em manter consistência na identificação e na cobertura das classes. Esses resultados sugerem limitações na capacidade de generalização da rede, comprometendo a regularidade no reconhecimento entre diferentes categorias. A DenseNet121 pode se beneficiar de mais dados ou ajustes no processo de treinamento.

A InceptionV3 obteve uma acurácia geral de 45.71% na validação e 51.43% no teste final. No teste final, o modelo teve um bom desempenho nas letras J, K, X, Y e Z, mas a letra H apresentou dificuldades, com uma taxa de acerto mais baixa.

As métricas *precision* de 56,49%, *recall* de 52,22% e *F1-score* 53,30% reforçam a performance equilibrada da InceptionV3. Esses resultados indicam que o modelo foi capaz de identificar corretamente uma boa parte dos exemplos reais, mantendo também um nível consistente de acertos em suas predições. Ainda assim, há espaço para melhorias, especialmente no refinamento do modelo para lidar com sinais mais sutis ou com maior variação.

A ResNet18, com sua arquitetura de conexões residuais, destacou-se pela eficácia apresentando 62.86% de acurácia na validação e 74.29% no teste final. No teste final, o modelo teve um desempenho excelente nas letras K e X. Além disso, obteve boas taxas nas letras J e Y, mas a letra Z apresentou dificuldades.

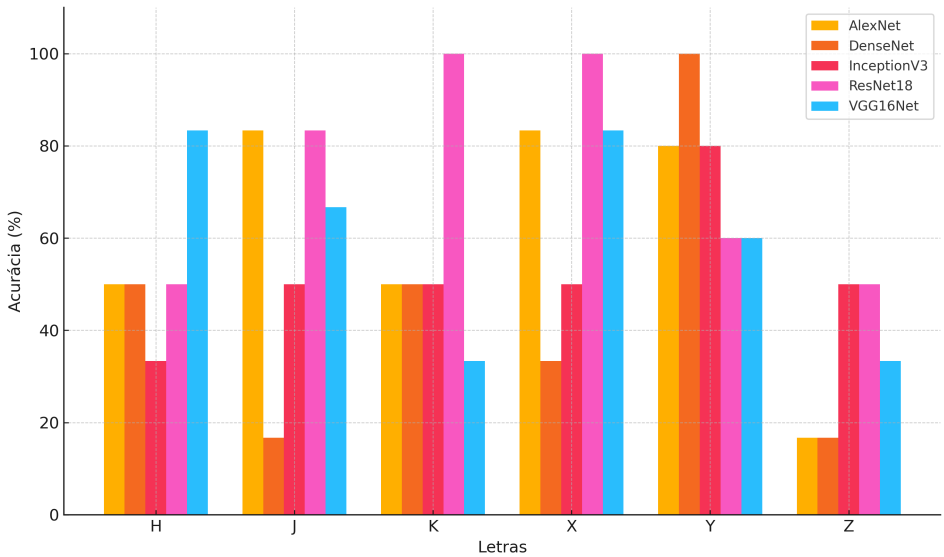
As métricas *precision* 78,06%, *recall* de 73,89% e *F1-score* de 72,90% reforçam o bom desempenho da ResNet18, evidenciando sua capacidade de manter consistência tanto na identificação correta dos exemplos quanto na precisão das predições. Esses resultados indicam que o modelo conseguiu generalizar bem os padrões dos sinais utilizados.

Por fim, a VGG16Net, com convoluções pequenas e uma arqui-

tetura relativamente simples, obteve 48.57% de acurácia na validação e 60.00% no teste final. No teste final, o modelo apresentou bom desempenho nas letras H, J, X e Y No entanto, a letra Z apresentou dificuldades, com taxas de acerto mais baixas.

As métricas *precision* de 61,01%, *recall* de 60,00% e *F1-score* de 59,06% reforçam a consistência apresentada pela VGG16Net. Esses valores indicam que, mesmo com um volume limitado de dados de treinamento, o modelo conseguiu manter um bom equilíbrio entre acerto e cobertura, demonstrando estabilidade no reconhecimento.

Figura 11 - Acurácia (%) final das arquiteturas de CNNs por letra.

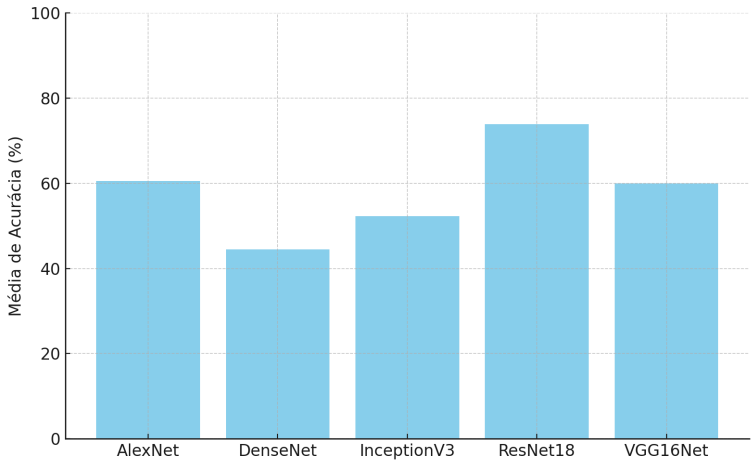


Fonte: Elaborado pelo autor.

Como pode ser observado na Figura 11, a ResNet18 se destacou com a maior acurácia geral entre as arquiteturas avaliadas. O modelo demonstrou robustez e consistência no reconhecimento dos sinais, evidenciando sua eficácia no manuseio de gradientes.

As arquiteturas AlexNet e VGG16Net, mesmo com estruturas mais simples, apresentaram desempenho estável e resultados consistentes na maioria dos sinais. Apesar de algumas oscilações, ambas demonstraram bom desempenho geral, destacando-se positivamente dentro do conjunto de modelos avaliados.

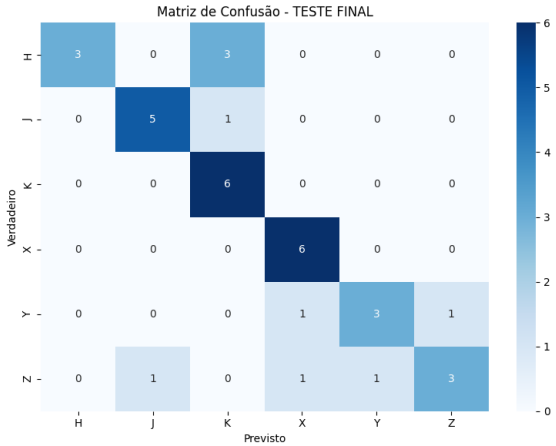
Figura 12 - Média geral (%) de acurácia por arquitetura.



Fonte: Elaborado pelo autor.

Já a DenseNet121 apresentou um desempenho mais irregular, com acertos pontuais, porém sem manter consistência entre os diferentes sinais. A InceptionV3, embora tenha demonstrado um comportamento mais equilibrado, registrou acurácia moderada e não se destacou em nenhum caso específico. Ambas as arquiteturas revelaram limitações na extração de características mais complexas, especialmente em sinais com maior variação espacial ou menor representatividade no conjunto de dados.

Figura 13 - Matriz de confusão Resnet - Teste Final



Fonte: Elaborado pelo autor.

Na Figura 12, observa-se que a ResNet18 obteve a maior média de acurácia entre as arquiteturas avaliadas, atingindo 74,29%. Informações importantes podem ser visualizadas na matriz de confusão apresentada na Figura 13, que mostra a distribuição dos acertos entre as classes, com

destaque para o reconhecimento preciso das letras K, X e J. No entanto, também é possível observar a dificuldade nas classes Y e Z, com erros mais distribuídos, indicando limitações na distinção de sinais com menor representatividade ou maior semelhança gestual.

As arquiteturas AlexNet e VGG16Net também apresentaram resultados expressivos, ambas com média de 60%, superando, inclusive, modelos mais complexos como a DenseNet121 e a InceptionV3.

Tabela 1 – Desempenho das arquiteturas no teste final.

Arquiteturas	Acurácia (%)	Precision (%)	Recall (%)	F1-score (%)
AlexNet	60,00	59,00	60,00	58,00
DenseNet121	42,86	42,31	44,44	40,47
InceptionV3	51,43	56,49	52,22	53,30
ResNet18	74,29	78,06	73,89	72,90
VGG16Net	60,00	61,01	60,00	59,06

Fonte: Elaborado pelo autor.

Os trabalhos estudados para a realização desta pesquisa, como Amaral et al. (2019), alcançaram acurácias superiores a 90% com o uso de redes 3D-CNN e redes convolucionais recorrentes de longo prazo (LRCNs). Da mesma forma, Das, Biswas e Purkayastha (2023), ao combinarem CNNs com redes neurais recorrentes com memória de curto e longo prazo (LSTM) bidirecional para o reconhecimento de palavras da Língua de Sinais Indiana (ISL), relataram uma acurácia média em torno de 87%.

Por outro lado, Grossi e Ferreira (2024), ao aplicar arquiteturas como CNNs e redes neurais recorrentes (RNNs) em sinais da LIBRAS, relataram acurácias na casa dos 90%, com variações significativas conforme o tipo de sinal e a base de dados utilizada. Em comparação, os modelos avaliados nesta pesquisa, conforme apresentado na Tabela 1, como a ResNet18 (74,29% de acurácia no teste final), assim como a VGG16Net e a InceptionV3 (60%), demonstraram desempenho relevante, especialmente diante das limitações do conjunto de dados.

Dessa forma, os resultados alcançados reforçam que, mesmo com abordagens mais simples e conjuntos de dados reduzidos, é possível obter desempenhos comparáveis aos de estudos mais complexos. As CNNs testadas mostraram-se eficazes na tarefa proposta e podem servir como base promissora para investigações futuras, com possíveis avanços mediante o uso de arquiteturas temporais ou dados mais ricos.

4 CONCLUSÃO

Este trabalho avaliou o desempenho de diferentes arquiteturas de redes neurais convolucionais no reconhecimento das letras H, J, K, X, Y e Z da Língua Brasileira de Sinais, visando desenvolver um modelo eficiente para a detecção desses sinais dinâmicos.

A alta qualidade dos dados foi um ponto forte ao longo do trabalho, permitindo que os modelos apresentassem resultados promissores. No entanto, a principal limitação foi a escassez de dados, o que prejudicou a extração de características mais robustas dos sinais. Para contornar essa dificuldade, foi utilizada a técnica de *data augmentation*, que aumentou a diversidade dos dados e possibilitou a continuidade do trabalho de maneira eficaz.

Outra dificuldade enfrentada foi a limitação computacional da máquina utilizada para os testes, o que restringiu a complexidade dos modelos e aumentou o tempo necessário para o treinamento de algumas arquiteturas mais.

Sugestões para trabalhos futuros incluem: explorar o uso de redes neurais temporais, como RNNs ou LSTMs, para uma captura mais eficaz das informações temporais dos sinais; expandir a amostragem de dados por meio da integração de novas bases de dados; Testar as arquiteturas avaliadas com diferentes sinais, indo além das letras do alfabeto.

REFERÊNCIAS

AMARAL, L. et al. **Evaluating Deep Models for Dynamic Brazilian Sign Language Recognition**. Cham: Springer International Publishing, 2019. 930–937 p. ISBN 978-3-030-13469-3.

Brasil. **Lei nº 10.436, de 24 de abril de 2002**. 2002. Diário Oficial [da] República Federativa do Brasil. Acesso em: 05 de maio de 2024. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>.

BUTTAR, A. et al. **Deep Learning in Sign Language Recognition: A Hybrid Approach for the Recognition of Static and Dynamic Signs**. 2023. 3729 p. Acesso em: 05 de maio de 2024. Disponível em: <https://doi.org/10.3390/math11173729>.

COLLET, S. **Object Detection - Part 1: Overview of Object Detection Techniques**. 2017. Acesso em: 26 maio 2025. Disponível em: <https://www.saagie.com/blog/object-detection-part1/>.

CRISTIANO, A. **Os Cinco Parâmetros da Libras**. 2024. Acesso em: 13 de maio de 2025. Disponível em: <https://www.libras.com.br/os-cinco-parametros-da-libras>.

DAS, S.; BISWAS, S. K.; PURKAYASTHA, B. **A deep sign language recognition system for Indian sign language**. 2023. 1469–1481 p. Acesso em: 01 de jun. de 2025. Disponível em: <https://doi.org/10.1007/s00521-022-07840-y>.

FELIPE, T. A.; MONTEIRO, M. **Libras em Contexto: Curso Básico - Livro do Professor**. [S.l.]: Programa Nacional de Apoio à Educação dos Surdos, MEC: SEEP, 2007.

FERNEDA, E. **Redes Neurais e Sua Aplicação em Sistemas de Informação**. 2006. [online] Acesso em: 13 de maio de 2025. Disponível em: <https://www.scielo.br/j/ci/a/SQ9myjZWLxnyXfstXMgCdcH/?format=pdf&lang=pt>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016.

GROSSI, V. S.; FERREIRA, B. S. Trabalho de Conclusão de Curso, **Aplicação de técnicas de reconhecimento de imagens na classificação de sinais em LIBRAS (Linguagem Brasileira de Sinais) para tradução em texto**. João Monlevade, MG: [s.n.], 2024. Acesso em: 13 de maio de 2025. Disponível em: <http://www.monografias.ufop.br/handle/35400000/6934>.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. Porto Alegre: Bookman, 2001.

HUANG, G. et al. **Densely Connected Convolutional Networks**. 2018. Acesso em: 26 maio 2025. Disponível em: <https://arxiv.org/abs/1608.06993>.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017. Acesso em: 26 de maio de 2025. Disponível em: <https://arxiv.org/abs/1412.6980>.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. **Convolutional Networks and Applications in Vision**. Nano-Bio Circuit Fabr. Syst.: IEEE, 2010. 253–256 p.

LOSHCHILOV, I.; HUTTER, F. **Decoupled Weight Decay Regularization**. 2019. Acesso em: 26 de maio de 2025. Disponível em: <https://arxiv.org/abs/1711.05101>.

MOELLER, M. P. **Early Intervention and Language Development in Children Who Are Deaf and Hard of Hearing**. 2000. E43 p. Acesso em 1 jun. 2025. Disponível em: <https://publications.aap.org/pediatrics/article-pdf/106/3/e43/886965/pe0900000u1p.pdf>.

NETO, G. M. R. **Reconhecimento de Língua de Sinais Baseado em Redes Neurais Convolucionais 3D**. Pós-Graduação — Universidade Federal do Maranhão, São Luís - MA, 2018.

OLIVEIRA, M. **Handshape Recognition using Principal Component Analysis and Convolutional Neural Networks applied to Sign Language**. Doctoral Thesis — Dublin City University, 2018.

PERNAMBUCO, U. F. de. **LIBRAS UFPE: Sistema de Busca por Sinais**. 2024. Acesso em: 21 de out. de 2024. Disponível em: <https://libras.cin.ufpe.br/?search=letra>.

PYTORCH. **CrossEntropyLoss**. 2025. Acesso em: 05 de maio de 2025. Disponível em: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>.

REZENDE, T. M. **Reconhecimento Automático de Sinais da Libras: Desenvolvimento da Base de Dados MINDS-Libras e Modelos de Redes Convolucionais**. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, 2021. Acesso em: 05 de maio de 2024. Disponível em: <https://repositorio.ufmg.br/handle/1843/39785>.

SALAZAR, J. J. et al. **Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy**. Journal of Petroleum Science and Engineering, 2022, v. 209, p. 109885. ISSN 0920-4105. Acesso em: 22 de maio de 2025. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0920410521015023>.

SHOTCUT. **Shotcut Video Editor**. 2025. Acesso em: 14 de abr. de 2025. Disponível em: <https://www.shotcut.org>.

SURDOS, I. N. de Educação de. **Dicionário de Libras**. 2024. Acesso em: 21 de out. de 2024. Disponível em: <https://www.ines.gov.br/dicionario-de-libras/>.