Dynamic Sign Recognition in Brazilian Sign Language Through Convolutional Neural Networks

João Paulo Moreira Rodrigues[1], **Ana Claudia Barbosa**[1]
Luis Ricardo Fiera[1], **Marlon Oliveira**[1]

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc)
Av. Universitária, 1105 – Universitário – Criciúma – SC – Brazil

{joaopaulomoreirarodrigues,agb,ricardo.fiera,marlon.oliveira}@unesc.net

Abstract. *Automatic recognition of Brazilian Sign Language (LIBRAS) is crucial for promoting inclusion and accessibility for deaf individuals. This study focuses on recognizing letters represented by dynamic signs, which involve temporal movements and remain a challenge for computational models. Five Convolutional Neural Network (CNN) architectures—AlexNet, DenseNet121, InceptionV3, ResNet18, and VGG16Net—were evaluated, with ResNet18 achieving the highest accuracy (74.29%) by effectively capturing complex features. Despite using simpler models compared to others in the literature, the results were relevant given the dataset limitations.*

Resumo. *O reconhecimento automático da Língua Brasileira de Sinais (LIBRAS) é fundamental para promover a inclusão e a acessibilidade de pessoas surdas. Este estudo foca no reconhecimento de letras representadas por sinais dinâmicos, que envolvem movimentos temporais e ainda representam um desafio para os modelos computacionais. Foram avaliadas cinco arquiteturas de Redes Neurais Convolucionais (CNNs) — AlexNet, DenseNet121, InceptionV3, ResNet18 e VGG16Net — sendo que a ResNet18 obteve a maior acurácia (74,29%), destacando-se na captura de características complexas. Apesar do uso de modelos mais simples em comparação a outros presentes na literatura, os resultados foram relevantes, considerando as limitações do conjunto de dados.*

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. [Moeller 2000].

As línguas de sinais variam entre culturas, 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].No entanto, o acesso aos serviços de tradução em LIBRAS ainda é limitado, uma vez que depende principalmente de intérpretes humanos.

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 e sinais dinâmicos, que envolvem movimentos das mãos com variações de forma, orientação e localização ao longo do tempo [Cristiano 2024].

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 Redes Neurais Convolucionais 3D (3D-CNNs) para reconhecer sinais dinâmicos em LIBRAS, utilizando imagens de profundidade.

Outra pesquisa foi conduzida por [Das et al. 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 resultados satisfatórios 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 and Ferreira 2024] focou na tradução de sinais de LIBRAS para texto, utilizando uma abordagem híbrida de CNNs e Redes Neurais Recorrentes (RNNs), que apresentou resultados satisfatórios, 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 cenário, este trabalho foca no reconhecimento dos sinais mais complexos do alfabeto em LIBRAS, utilizando CNNs para capturar suas variações. Busca-se desenvolver um modelo eficaz e contribuir para o avanço das pesquisas na área.

Com os avanços da inteligência artificial, especialmente no 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. [Rezende 2021].

As ANNs, inspiradas no cérebro humano, permitem o processamento e aprendizado por meio de neurônios interligados [Haykin 2001]. As CNNs são eficazes no reconhecimento de padrões visuais, aplicadas em tarefas como classificação de imagens [LeCun et al. 2010]. Para dados temporais, como gestos dinâmicos, utilizam-se RNNs e LSTMs, que capturam a sequência e a continuidade das informações.

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.

Embora os gestos dinâmicos estejam ganhando cada vez mais espaço nas pesquisas, muitos estudos ainda focam em gestos estáticos devido à menor complexidade de processamento. Por envolverem movimentos e sequências temporais, os sinais dinâmicos apresentam desafios técnicos adicionais, mas seu reconhecimento é fundamental para ampliar a inclusão e a acessibilidade na comunicação.

Este trabalho apresenta os métodos utilizados para reconhecer sinais dinâmicos da LIBRAS, discute os resultados obtidos e conclui com os principais achados e sugestões para pesquisas futuras voltadas à inclusão e acessibilidade.

2. Materiais e Métodos

Este estudo explora uma abordagem baseada em CNNs, com o objetivo de avaliar sua capacidade de reconhecer sinais dinâmicos. São investigadas diferentes arquiteturas.

2.1. Base de Dados

Foram utilizadas duas bases de dados, selecionadas de acordo com sua qualidade e diversidade. A base [de 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, contabilizando dezoito vídeos no total.

A base [de Educação de Surdos 2024] apresenta uma qualidade de captura inferior à da base [de Pernambuco 2024], 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.

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 para isolar os sinais, cortando trechos irrelevantes com a ferramenta [Shotcut 2025] e preservando apenas o intervalo da 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 1. Esse *pipeline* foi implementado em Python, utilizando, as bibliotecas OpenCV, NumPy e PyTorch.

Cada vídeo foi convertido para escala de cinza para reduzir a complexidade computacional, considerando os recursos disponíveis. Apesar de não haver comparação com vídeos em RGB, essa escolha priorizou a eficiência no processamento. Em seguida, os vídeos foram redimensionados para a resolução de 224x224 pixels e convertidos em

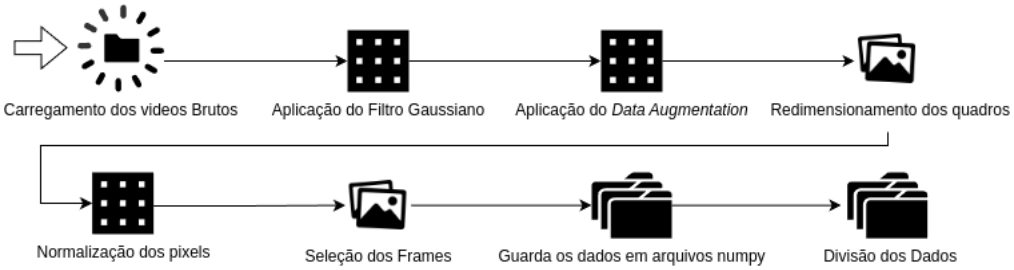

Figura 1. Representação do *Pipeline* de Pré-processamento.

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 2 ilustra alguns destes quadros extraídos.

Figura 2. *Frames* da representação da letra H em LIBRAS.

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 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, como rotação em pequenos ângulos, adição de ruído gaussiano e variações leves de brilho. Isso 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 vetores no formato NumPy, organizadas de forma a manter a estrutura temporal necessária para o reconhecimento dos sinais.

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

Após esse processo, a divisão dos dados foi realizada de forma que 80% das amostras foram destinadas 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].

Além disso, buscou-se evitar que os mesmos participantes (sinalizadores) estivessem presentes simultaneamente nos conjuntos de treinamento e teste, o que poderia enviesar os resultados.

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.

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 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.

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 contribui 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 and 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 and Hutter 2019].

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 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 et al. 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.

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.

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.

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*.

2.3. Treinamento das Arquiteturas

O treinamento das 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 épocas (*epochs*), a taxa de aprendizado e a função de perda. Foram utilizadas arquiteturas pré-treinadas, como ResNet18, VGG16Net e InceptionV3.

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 nos conjuntos de treino, validação e teste. Algumas arquiteturas utilizaram *scheduler* para ajuste automático da taxa de aprendizado. Esse número de *epochs* foi definido com base em testes preliminares, onde se observou estabilização de *overfitting* a partir da *epoch* 20. Ainda assim, reconhece-se que esse limite pode ter impactado o desempenho final.

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.

2.4. 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. Os resultados de cada avaliação foram registrados em um relatório.

Os códigos fontes deste trabalho encontram-se no GitHub[1]

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

[1] https://github.com/joaorodriguesz/dynamic-signs-recognition-libras-alphabet-cnn

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 corretas, 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 apresentou desempenho inferior 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 arquitetura 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.

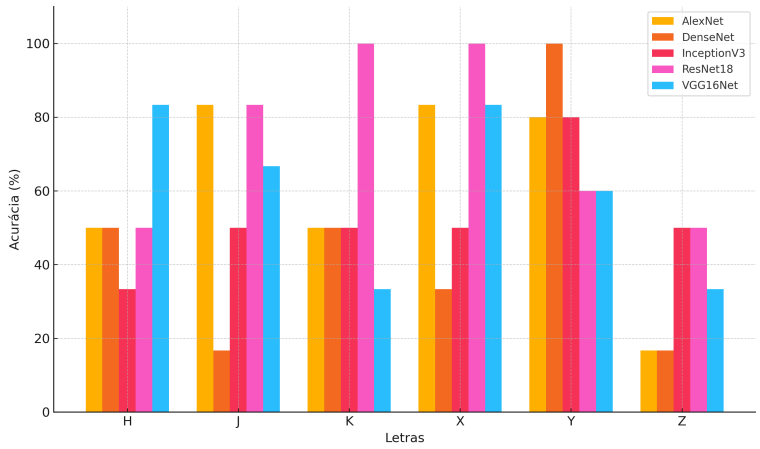

Figura 3. Acurácia (%) final das arquiteturas de CNNs por letra.

Como pode ser observado na Figura 3, 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.

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

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.

Na Tabela 1, 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 4, que mostra a distribuição dos acertos entre as classes, com destaque para o reconhecimento preciso das letras K, X e J. Por outro lado, as letras Y e, especialmente, Z apresentaram maior índice de erros, possivelmente devido à baixa variabilidade nos sinais, semelhança com gestos de outras classes ou à pouca representação nos dados.

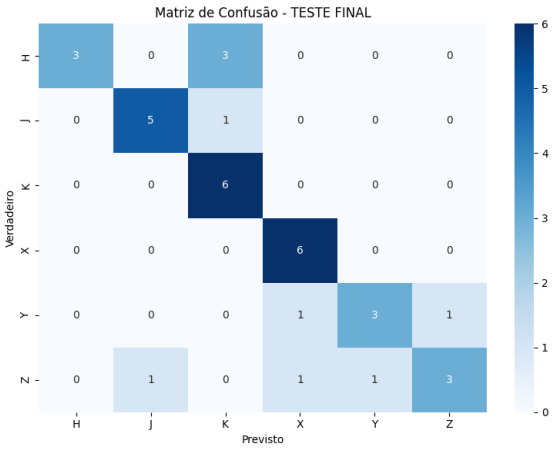

Figura 4. Matriz de confusão Resnet - Teste Final

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 et al. 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 and 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.

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.

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., Júnior, G. L. N., Vieira, T., and Vieira, T. (2019). *Evaluating Deep Models for Dynamic Brazilian Sign Language Recognition*. Springer International Publishing,

Cham.

Brasil (2002). Lei nº 10.436, de 24 de abril de 2002. Diário Oficial [da] República Federativa do Brasil. Acesso em: 05 de maio de 2024.

Buttar, A., Ahmad, U., Gumaei, A., Assiri, A., Akbar, M., and Alkhamees, B. (2023). Deep learning in sign language recognition: A hybrid approach for the recognition of static and dynamic signs. Acesso em: 05 de maio de 2024.

Cristiano, A. (2024). Os cinco parâmetros da libras. Acesso em: 13 de maio de 2025.

Das, S., Biswas, S. K., and Purkayastha, B. (2023). A deep sign language recognition system for indian sign language. Acesso em: 01 de jun. de 2025.

de Educação de Surdos, I. N. (2024). Dicionário de libras. Acesso em: 21 de out. de 2024.

de Pernambuco, U. F. (2024). Libras ufpe: Sistema de busca por sinais. Acesso em: 21 de out. de 2024.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grossi, V. S. and Ferreira, B. S. (2024). Aplicação de técnicas de reconhecimento de imagens na classificação de sinais em libras (linguagem brasileira de sinais) para tradução em texto. Acesso em: 13 de maio de 2025.

Haykin, S. (2001). *Redes Neurais: Princípios e Prática*. Bookman, Porto Alegre.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. Acesso em: 26 de maio de 2025.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). *Convolutional Networks and Applications in Vision*. IEEE, Nano-Bio Circuit Fabr. Syst.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization. Acesso em: 26 de maio de 2025.

Moeller, M. P. (2000). Early intervention and language development in children who are deaf and hard of hearing. Acesso em 1 jun. 2025.

Neto, G. M. R. (2018). Reconhecimento de língua de sinais baseado em redes neurais convolucionais 3d.

Oliveira, M. (2018). Handshape recognition using principal component analysis and convolutional neural networks applied to sign language.

PyTorch (2025). Crossentropyloss. Acesso em: 05 de maio de 2025.

Rezende, T. M. (2021). Reconhecimento automático de sinais da libras: Desenvolvimento da base de dados minds-libras e modelos de redes convolucionais. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte. Acesso em: 05 de maio de 2024.

Salazar, J. J., Garland, L., Ochoa, J., and Pyrcz, M. J. (2022). Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy. *Journal of Petroleum Science and Engineering*, 209:109885. Acesso em: 22 de maio de 2025.

Shotcut (2025). Shotcut video editor. Acesso em: 14 de abr. de 2025.