REDES NEURAIS CONVOLUCIONAIS NA DETECÇÃO E INTERPRETAÇÃO DA LINGUAGEM DE SINAIS PARA A ASSISTÊNCIA A INDIVÍDUOS COM DEFICIÊNCIA AUDITIVA

Gustavo Antonio dos Santos Fontana[1], Sérgio Coral[2]

Resumo: A comunicação entre surdos e ouvintes ainda enfrenta barreiras, especialmente no reconhecimento automático da Língua Brasileira de Sinais (LIBRAS). Este trabalho apresenta o desenvolvimento de um protótipo web para identificar e classificar sinais do alfabeto manual da LIBRAS, utilizando redes neurais convolucionais treinadas com aprendizado supervisionado e otimizadas com o algoritmo Stochastic Gradient Descent. A solução integra TensorFlow, Keras e MediaPipe para detecção e análise de gestos em tempo real, processando imagens RGB de mãos com 64x64 píxeis. O modelo atingiu acurácia de 98,67% na validação e desempenho semelhante no teste, demonstrando boa generalização em ambientes controlados. A análise por classe indicou 100% de precisão em 12 das 21 letras, com erros concentrados entre sinais semelhantes. Em testes com webcam, a acurácia variou entre 50% e 100% conforme o cenário, destacando limitações em ambientes com ruído visual ou ângulos desfavoráveis. Comparado a estudos anteriores, o modelo apresentou desempenho superior ao aplicar técnicas como *data augmentation* e normalização. Os resultados demonstram o potencial do sistema para aplicações acessíveis e interativas no navegador.

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

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), gustavoasf77@unesc.net
[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), sergiocoral@unesc.net

ABSTRACT: Communication between deaf and hearing individuals still faces barriers, especially in the automatic recognition of Brazilian Sign Language (LIBRAS). This study presents the development of a web-based prototype for identifying and classifying manual alphabet signs from LIBRAS using convolutional neural networks trained with supervised learning and optimized via the Stochastic Gradient Descent algorithm. The solution integrates TensorFlow, Keras, and MediaPipe for real-time gesture detection and analysis, processing 64x64 pixel RGB hand images. The model achieved 98.67% accuracy on the validation set, with similar performance on the test set, indicating good generalization in controlled environments. Class-wise analysis showed 100% precision in 12 out of 21 letters, with errors concentrated among visually similar signs. In real-time webcam tests, accuracy ranged from 50% to 100% depending on the scenario, revealing limitations in conditions with background noise or unfavorable hand angles. Compared to previous studies, the model achieved superior performance by applying techniques such as data augmentation and normalization. Results highlight the system's potential for accessible, interactive browser-based applications.

Keywords: brazilian sign language; convolutional neural networks; deep learning; accessibility; computer vision.

1 INTRODUÇÃO

A deficiência auditiva representa um desafio significativo no Brasil, onde mais de 10 milhões de pessoas convivem com algum grau de surdez, dos quais aproximadamente 2,7 milhões possuem surdez profunda (Junior; Santos; Penteado, 2022). A Língua Brasileira de Sinais (LIBRAS) desempenha um papel essencial nesse contexto, funcionando como o principal meio de comunicação para essa população e como ferramenta indispensável para a promoção da inclusão social (Barral; Rumjanek, 2018). No entanto, estudos como o de Silva (2020) ressaltam que o aprendizado de LIBRAS enfrenta barreiras significativas, especialmente entre ouvintes, como a escassez de profissionais qualificados, a limitação de materiais didáticos e a baixa oferta de recursos tecnológicos acessíveis e interativos (Rodrigues-Moura; Desidério, 2023).

Com o avanço da tecnologia, a área da computação desempenha um papel fundamental no desenvolvimento de soluções inclusivas. Nesse cenário, a inteligência artificial tem-se destacado como uma aliada

promissora no enfrentamento de desafios sociais, oferecendo recursos inovadores para o reconhecimento de padrões, análise de dados e interpretação de imagens e vídeos em tempo real (Li et al., 2020). A aplicação de Inteligência Artificial (IA) em tecnologias assistivas possibilita o surgimento de ferramentas capazes de melhorar a qualidade de vida de pessoas com deficiência, por meio de soluções mais adaptáveis e eficientes (Silva et al., 2020).

Este estudo surge da necessidade de promover a inclusão digital e suprir a carência de recursos tecnológicos voltados ao ensino e à prática da LIBRAS. Diante das projeções da Organização das Nações Unidas, que estimam um crescimento significativo da população com deficiência auditiva até 2050, torna-se urgente o investimento em soluções tecnológicas que facilitem a comunicação e promovam a acessibilidade (Souza et al., 2023).

O objetivo deste trabalho é desenvolver um protótipo de aplicação web para identificação e classificação de sinais da LIBRAS, utilizando redes neurais convolucionais treinadas com aprendizado supervisionado e o otimizador Stochastic Gradient Descent. Para atingir esse objetivo, foram definidos os seguintes objetivos específicos: desenvolver um sistema de reconhecimento de sinais a partir de imagens, integrar o modelo gerado com uma página web e verificar a eficácia da aplicação quanto à classificação correta dos sinais, a partir de testes experimentais.

Diversos estudos demonstram o potencial da inteligência artificial no reconhecimento e tradução de sinais em Libras. Albino (2022) apresentou uma aplicação web baseada no algoritmo Dynamic Time Warping, obtendo mais de 90% de acurácia na tradução de sinais. Já Souza et al. (2023) desenvolveu a ferramenta CAMLIBRAS, que utiliza MediaPipe para capturar gestos e a biblioteca FANN para classificá-los, atingindo 75,91% de acurácia no reconhecimento do alfabeto por meio de jogos educativos. Semelhantemente, Araújo (2023) propôs uma rede neural convolucional 2D para reconhecer gestos estáticos em tempo real, alcançando 97,93% de acurácia nas letras do alfabeto e mais de 91% para números e palavras específicas. Complementando esses avanços, Fan (2023) aprimorou o uso do MediaPipe ao considerar variações de distância e rotação da mão, utilizando parâmetros fixos como o comprimento da palma para garantir maior estabilidade e precisão na interação humano-computador. Adicionalmente, Leacina (2021) propôs uma luva vestível equipada com sensores, capaz de reconhecer letras e números em Libras, integrada a um aplicativo com fins educacionais. O sistema alcançou acurácia de 94,11%.

O levantamento bibliográfico evidencia que a Inteligência Artificial tem se consolidado como um pilar central na inovação tecnológica, especialmente nas áreas de acessibilidade e inclusão social. No contexto da comunicação entre surdos e ouvintes, o uso de Redes Neurais Convolucionais (do inglês, Convolutional Neural Networks – CNN) e visão computacional tem mostrado potencial para a tradução em tempo real da Língua Brasileira de Sinais (LIBRAS), promovendo a inclusão digital (Kuzenkov, 2024). De acordo com Szeliski (2022), a visão computacional permite que máquinas interpretem dados visuais, utilizando técnicas como segmentação e extração de características essenciais para identificar padrões em imagens. As redes neurais convolucionais, inspiradas no córtex visual humano, são compostas por camadas que aprendem bordas, formas e texturas, sendo eficazes em tarefas de visão computacional (Goodfellow; Bengio; Courville, 2016). As redes neurais convolucionais, aplicadas ao reconhecimento de sinais em LIBRAS, permitem interpretar variações contextuais dos gestos ao extrair padrões espaciais em imagens. Aliadas à visão computacional, viabilizam soluções automatizadas de tradução em tempo real (Goodfellow; Bengio; Courville, 2016). Nesse cenário, destaca-se o algoritmo Stochastic Gradient Descent (SGD), amplamente utilizado na otimização de modelos supervisionados, por promover convergência rápida em tarefas de classificação (Asritha; Soundari, 2023).

O artigo está organizado em seções, iniciando pela introdução, que apresenta o tema, os objetivos, a justificativa, trabalhos correlatos e a fundamentação teórica. Em seguida, a seção de materiais e métodos descreve as tecnologias utilizadas, a arquitetura da solução proposta e a metodologia adotada para o desenvolvimento. A terceira seção apresenta e discute os resultados obtidos, com destaque para o desempenho do sistema em diferentes cenários de teste, a análise da acurácia alcançada e a comparação com resultados de trabalhos correlatos. Por fim, a conclusão retoma os principais pontos do estudo, destacando suas contribuições, limitações e possíveis direções para pesquisas futuras. Ao final, são listadas as referências bibliográficas que embasaram teoricamente o desenvolvimento do trabalho e sustentam as análises realizadas.

2 MATERIAIS E MÉTODOS

Esta pesquisa é de natureza aplicada, com foco tecnológico, e de caráter descritivo, pois visa desenvolver uma aplicação web responsiva para a detecção e tradução de sinais em Libras. A abordagem adotada en-

volveu a condução de experimentos com técnicas de visão computacional e aprendizado de máquina, permitindo a interpretação de sinais manuais diretamente no navegador de dispositivos móveis ou computadores.

A Figura 1 ilustra as etapas envolvidas na aplicação desenvolvida, composta por duas fases principais: Inteligência Artificial e Aplicação Web. A primeira fase compreende o treinamento de uma rede neural convolucional para reconhecer sinais de Libras a partir de imagens, utilizando TensorFlow GPU e Keras. Foram utilizadas aproximadamente 1.650 imagens por sinal, organizadas em conjuntos de dados para treinamento, validação e teste do modelo.

Figura 1 - Etapas da aplicação.

Fonte: Elaborado pelo autor.

Já a etapa de Aplicação WEB corresponde ao desenvolvimento de uma aplicação que acessa a câmera do dispositivo, realiza a extração dos pontos de referência da mão por meio do MediaPipe, processa esses dados com o modelo treinado em TensorFlow.js e exibe a tradução do sinal em tempo real ao usuário.

O ambiente de desenvolvimento contou com um computador de mesa equipado com uma placa de vídeo NVIDIA RTX 4060, processador AMD Ryzen 5 5600, 16 GB de memória RAM, webcam Logitech Brio 100 e sistema operacional Windows 10. O código-fonte foi versionado com Git e hospedado no GitHub.

2.1 DESENVOLVIMENTO DA ETAPA INTELIGÊNCIA ARTIFICIAL

Nesta etapa, foi desenvolvido um modelo de CNN utilizando as bibliotecas TensorFlow e Keras, para classificar os sinais do alfabeto manual de Libras. A arquitetura do modelo é composta por três camadas convolucionais seguidas de pooling, além de camadas totalmente conectadas com ativação softmax na saída. O treinamento foi realizado com imagens

de 64x64 píxeis, utilizando o algoritmo Stochastic Gradient Descent (SGD) para otimização, e monitorado com EarlyStopping para evitar overfitting. O desempenho foi avaliado por métricas de acurácia e perda, e o modelo final foi salvo no formato .h5. A biblioteca Keras, integrada ao TensorFlow, foi escolhida por sua simplicidade e eficiência no desenvolvimento de redes neurais, como evidenciado em aplicações de detecção de doenças em aves (Jain; Kapoor; Singh, 2024). Além disso, o uso do TensorFlow com suporte a GPU se mostrou vantajoso para acelerar o treinamento e aumentar a precisão na classificação de imagens, conforme destacado por Yeboah, Sanoussi e Agordzo (2021). Todo o desenvolvimento foi realizado em ambiente configurado com Python 3.7.9, utilizando Keras 2.10.0 e TensorFlow-GPU 2.10.0, com aceleração via CUDA 11.2, o que contribuiu para uma maior eficiência computacional durante o processo de treinamento.

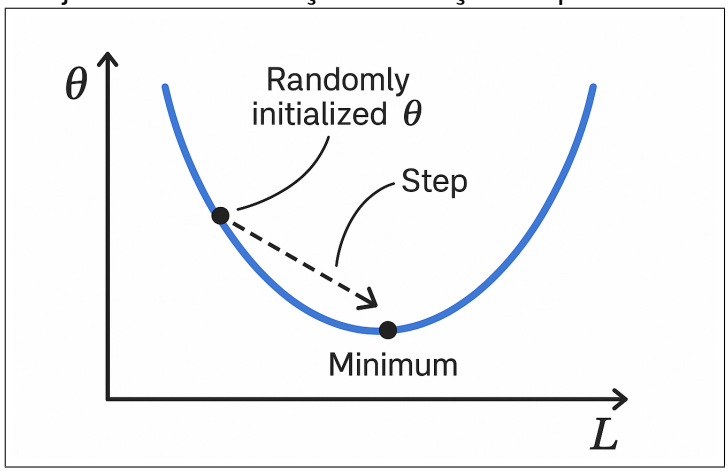
2.1.1 Stochastic Gradient Descent

O Stochastic Gradient Descent (SGD) é um algoritmo de otimização amplamente utilizado no treinamento de redes neurais, sendo aplicado neste projeto visando minimizar a função de perda durante o processo de aprendizado da rede convolucional (CNN) responsável pela tradução de sinais da LIBRAS. Para a implementação do SGD, utilizou-se a biblioteca Keras, que disponibiliza a classe SGD no módulo keras.optimizers, permitindo a configuração de diversos parâmetros, como a taxa de aprendizado, que neste projeto foi definida como 0,01.

A integração do algoritmo SGD com o restante do modelo foi realizada por meio do método compile da biblioteca Keras, o qual é responsável por preparar a rede neural convolucional para o processo de treinamento. Nesse método, o SGD foi definido como o otimizador, ou seja, o componente responsável por ajustar os pesos das conexões entre os neurônios da rede, com o objetivo de minimizar os erros de previsão ao longo do tempo. Esse ajuste é feito com base em uma função matemática chamada função de perda, que mede a diferença entre a saída esperada e a saída real da rede.

A Figura 2 ilustra como o algoritmo Stochastic Gradient Descent atualiza os parâmetros da rede ao longo da curva da função de perda. O ponto inicial representa o estado inicial da rede, enquanto o ponto final indica o mínimo da função. A seta tracejada mostra o percurso em pequenos passos realizados a cada mini-batch, visando minimizar o erro (Goodfellow; Bengio; Courville, 2016).

Figura 2 - Trajetória de otimização na função de perda utilizando SGD.



Fonte: Elaborado pelo autor.

A Figura 3 ilustra a regra de atualização dos pesos conforme descrita por Bishop e Nasrabadi (2006), no contexto do algoritmo de descida do gradiente estocástico. Nessa formulação, os pesos do modelo são ajustados iterativamente com base no gradiente da função de erro, calculado para uma única amostra ou mini-lote. A cada iteração, subtrai-se do valor atual dos pesos um termo proporcional à inclinação da função de erro naquele ponto, escalado por um fator denominado taxa de aprendizado. Esse procedimento conduz os parâmetros em direção ao mínimo da função de custo, promovendo a redução progressiva do erro de previsão.

Figura 3 - Equação de atualização dos parâmetros na SGD.

$$\theta = \theta - \eta \nabla J\left(\theta; x^{(i)}; y^{(i)}\right)$$

Fonte: Adaptado de Bishop (2006).

Essa escolha mostrou-se especialmente adequada em contextos com conjuntos de dados de porte intermediário ou elevado, como no caso deste trabalho, que utilizou aproximadamente 1.650 imagens para cada uma das 21 classes de sinais, totalizando mais de 34 mil imagens. O SGD se destaca por atualizar os pesos da rede frequentemente, utilizando pequenos grupos de exemplos de treinamento por vez, técnica conhecida como mini-batches (Ruder, 2017). Esse comportamento torna o processo de aprendizado mais ágil e contribui para o modelo conseguir identificar padrões relevantes nos dados, o que é essencial para alcançar uma boa

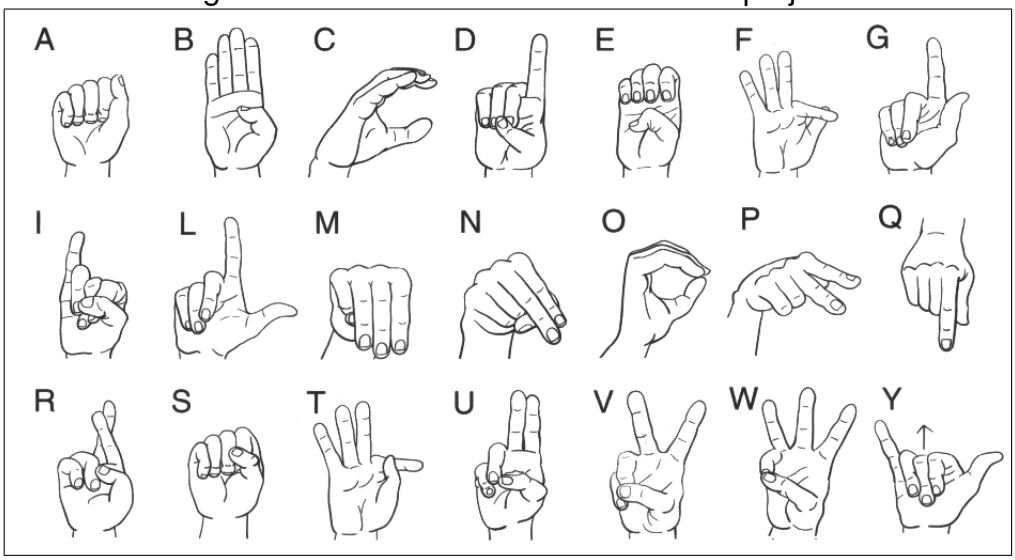
capacidade de generalização, ou seja, o desempenho da rede ao interpretar sinais que não estavam presentes durante o treinamento.

Além do próprio algoritmo de otimização, o uso eficaz do método SGD depende de outras ferramentas auxiliares. Um exemplo é o ImageDataGenerator da biblioteca Keras, que serve para preparar as imagens antes de treinar a rede e também aumentar a quantidade de dados disponíveis. Isso é feito com pequenas variações nas imagens, como rotações ou mudanças de brilho, ajudando a melhorar o aprendizado do modelo (Shorten; Khoshgoftaar, 2019). Outra ferramenta importante é o Early Stopping, que interrompe automaticamente o treinamento quando o desempenho do modelo deixa de melhorar nos dados de validação. Isso ajuda a evitar que o modelo se ajuste demais aos dados de treino, aprendendo até mesmo ruídos ou padrões irrelevantes, o que comprometeria sua capacidade de funcionar bem com novos dados, um problema conhecido como sobreajuste (overfitting) (Goodfellow; Bengio; Courville, 2016).

2.1.2 Preparação do Conjunto de Dados

Para o treinamento e avaliação do modelo de reconhecimento de sinais da Língua Brasileira de Sinais, foi utilizado um conjunto de dados obtido de um repositório público na internet[3]. A base contém imagens de mãos representando sinais estáticos correspondentes ao alfabeto manual de Libras, contendo 21 classes distintas, conforme ilustrado na Figura 4.

Figura 4 - Letras do alfabeto usadas no projeto.



Fonte: Elaborado pelo autor.

[3]https://github.com/lucaaslb/cnn-libras

O conjunto de dados foi previamente dividido em dois subconjuntos: um para treinamento e outro para teste. A porção de treinamento contém aproximadamente 1.650 imagens por classe (cerca de 34.650 amostras), enquanto o conjunto de teste possui cerca de 550 imagens por classe (totalizando aproximadamente 11.550 amostras). Essa divisão corresponde a uma proporção de 75% para o treinamento e 25% para o teste.

A escolha dessa proporção foi baseada no equilíbrio entre fornecer volume suficiente de dados para o modelo aprender padrões relevantes e garantir um conjunto de teste representativo para avaliar seu desempenho. De acordo com UÇar et al. (2020), divisões como 70/30, 75/25 e 80/20 são comuns e devem ser escolhidas conforme a natureza do conjunto de dados. No presente estudo, a divisão 75/25 se mostrou adequada, ao evitar overfitting e ainda oferecer uma validação confiável.

Todas as imagens foram organizadas em diretórios separados por classe, conforme a estrutura exigida pelo recurso ImageDataGenerator da biblioteca Keras.

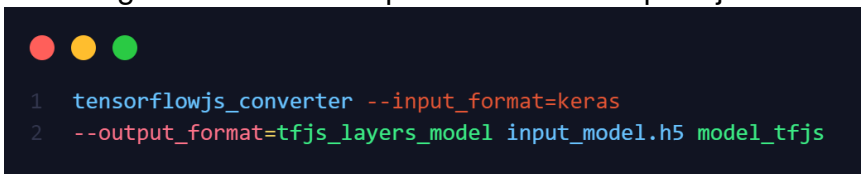
Durante a preparação, as imagens foram redimensionadas para 64×64 píxeis e mantidas em RGB, sendo em seguida normalizadas para o intervalo [0,1] para garantir estabilidade numérica durante o treinamento. Em seguida, a classe ImageDataGenerator foi configurada não somente para repetir essa normalização em ambos os conjuntos, mas também, no treino, para aplicar *data augmentation* por meio de inversão horizontal, zoom de até 20% e cisalhamento de até 20%. Embora não tenha sido necessário balancear as classes, já que cada uma contava com aproximadamente 1.650 imagens no treino e 550 no teste.

2.2 DESENVOLVIMENTO DA ETAPA APLICAÇÃO WEB

Nesta etapa, foi desenvolvida uma aplicação web utilizando HTML5 e JavaScript para realizar o reconhecimento de sinais da LIBRAS de forma acessível tanto em computadores quanto em dispositivos móveis. A aplicação executa diretamente no navegador, realizando a captura de imagens da webcam em tempo real, detectando a presença de uma mão, delimitando dinamicamente uma região central da imagem como área de interesse e utilizando o modelo de rede neural convolucional previamente treinado para classificar o sinal apresentado.

O primeiro passo foi a conversão do modelo no formato .h5 para o formato compatível com TensorFlow.js. Para isso, foi utilizado o utilitário tensorflowjs_converter, com o comando apresentado na Figura 5.

Figura 5 - Comando para converter h5 para json.



Fonte: Elaborado pelo autor.

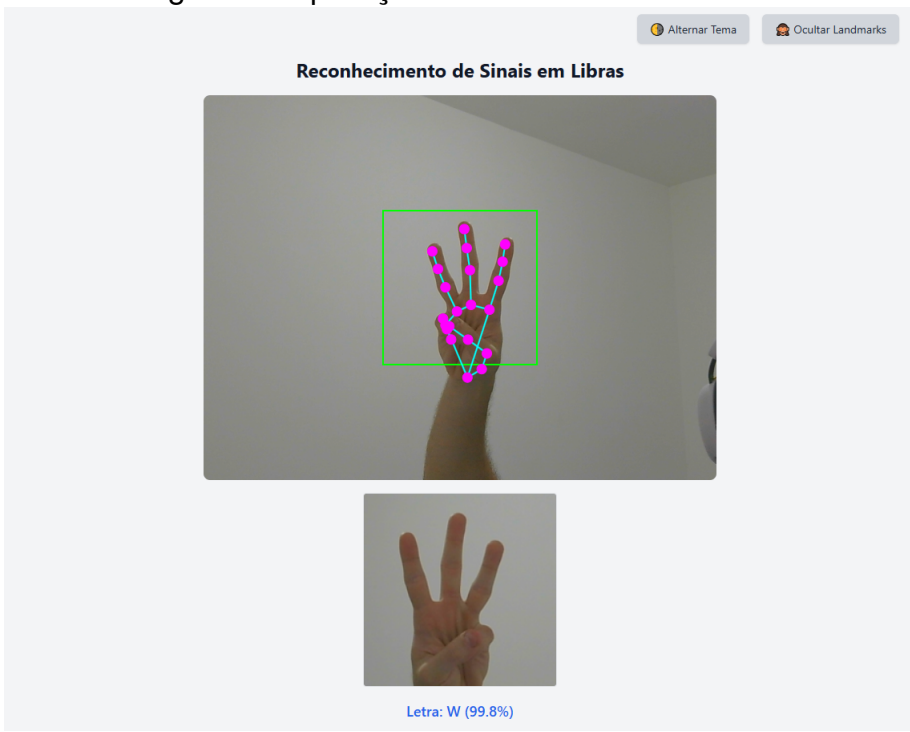
Esse processo gera um arquivo model.json e arquivos binários (.bin) contendo os pesos da rede neural, os quais são organizados em uma pasta chamada model_tfjs e utilizados diretamente no front-end da aplicação. O carregamento do modelo no navegador é realizado por meio da biblioteca TensorFlow.js, permitindo que as inferências ocorram localmente, sem depender de servidores.

A interface web da aplicação foi projetada com múltiplos elementos coordenados para viabilizar a classificação dos sinais em tempo real no navegador. Um elemento <video> capta continuamente imagens da webcam, enquanto um <canvas> sobreposto destaca dinamicamente a região de interesse. Outro <canvas> redimensiona essa área para 64x64 píxeis, compatibilizando-a com o formato esperado pelo modelo. Um terceiro <canvas> auxilia o posicionamento da mão ao exibir visualmente a região destacada. O resultado da previsão é exibido em tempo real em um campo <h2> com a letra identificada.

Durante a execução, a aplicação realiza predições a cada segundo. Quando uma mão é detectada, a área central (196x196 píxeis) é extraída, redimensionada, normalizada para o intervalo [0,1] e utilizada como entrada do modelo. O resultado é exibido junto à confiança percentual e, na ausência de detecção, uma mensagem é mostrada. Toda a aplicação é hospedada no GitHub Pages, podendo ser executada diretamente no navegador, sem instalação. Há ainda botões para alternar entre temas claro/escuro e ativar ou desativar a visualização dos *landmarks* da mão fornecidos pela biblioteca MediaPipe.

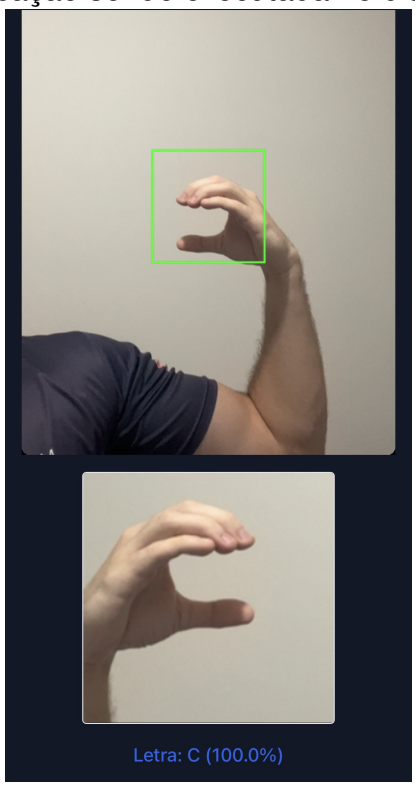
As Figuras 6 e 7 ilustram a interface da aplicação em funcionamento, demonstrando dois exemplos distintos de predição. A Figura 6 mostra a detecção da letra "W", com os pontos de referência (*landmarks*) visíveis sobre a mão, enquanto a Figura 7 apresenta a letra "C", com os marcadores ocultos.

Figura 6 - Aplicação sendo executada no PC.



Fonte: Elaborado pelo autor.

Figura 7 - Aplicação sendo executada no dispositivo móvel.



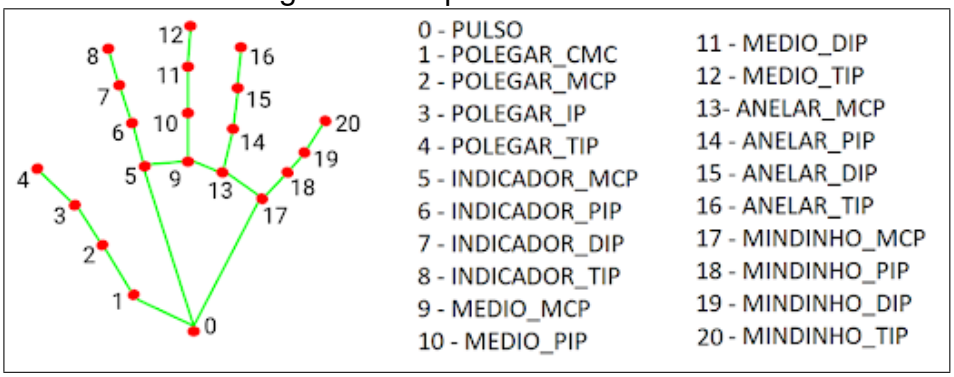
Fonte: Elaborado pelo autor.

2.2.1 Mediapipe

Para realizar a detecção da mão em tempo real, utilizou-se a biblioteca MediaPipe Hands, desenvolvida pelo Google. Essa ferramenta oferece uma solução eficiente e leve para rastreamento de mãos, utilizando visão computacional baseada em aprendizado de máquina. O modelo consegue detectar até 21 pontos de referência por mão (Oliveira et al., 2023).

No contexto deste projeto, o MediaPipe foi utilizado diretamente no navegador, por meio do pacote JavaScript disponibilizado via CDN. A integração com o elemento <video> da câmera permitiu capturar quadros em tempo real, os quais foram processados para identificar a presença de uma mão. Quando detectada, o sistema delimitava automaticamente uma região de interesse centralizada na imagem, de onde a mão era recortada para ser redimensionada e classificada pelo modelo de rede neural previamente treinado. A Figura 8 ilustra o mapeamento dos pontos de referência da mão realizado pelo MediaPipe.

Figura 8 - Mapeamento da mão.



Fonte: Albino (2022, p. 9).

2.3 PROTOCOLO DE TESTES

Para validar a eficácia e a confiabilidade do modelo de reconhecimento dos sinais do alfabeto manual da LIBRAS, foi definido um protocolo de testes com avaliações automáticas, análise por classe e testes práticos em condições reais.

Após o treinamento da rede neural convolucional, são gerados automaticamente gráficos de desempenho que relacionam a perda e a acurácia de treino e validação ao longo das épocas. Essas curvas permitem identificar se o modelo apresentou comportamento de convergência estável

e ajudam a verificar a ocorrência de *overfitting*[4] ou *underfitting*[5]. Um modelo com bom desempenho é esperado apresentar queda consistente na perda e estabilização ou aumento da acurácia de validação, com mínima diferença em relação ao treino (Chollet, 2021).

Além disso, uma matriz de confusão é gerada com o conjunto de teste, evidenciando os acertos e erros por classe. Essa análise ajuda a identificar confusões entre sinais parecidos e a avaliar a precisão individual de cada classe (Geron, 2019).

Além das avaliações automáticas, o modelo também foi testado em tempo real por meio da webcam, considerando diferentes condições de uso. Os testes contemplaram variações de iluminação, presença de objetos e cores no plano de fundo em contraste com o fundo branco utilizado durante o treinamento, além de mudanças na distância e no ângulo da mão em relação à câmera. Para cada cenário, os sinais C, E e L foram executados repetidamente, totalizando 30 tentativas por combinação de sinal e condição. A acurácia foi calculada conforme a Equação 1. A escolha desses sinais se deu com base em suas características visuais distintas entre si.

$$acuracia = \frac{numero\ de\ acertos}{numero\ de\ tentativas} \tag{1}$$

Por fim, a acurácia geral obtida com o modelo desenvolvido é comparada com a apresentada no trabalho de Albino (2022), permitindo avaliar os avanços alcançados e identificar possíveis limitações a serem aprimoradas futuramente. A escolha por essa comparação se justifica pelo fato de o presente estudo ser uma continuação do trabalho de Albino, utilizando como base a mesma proposta inicial e avançando em aspectos como a arquitetura do modelo e o uso de técnicas de pré-processamento.

3 DISCUSSÃO E RESULTADOS

Para validar a eficácia e a confiabilidade do modelo de reconhecimento dos sinais do alfabeto manual da Libras, foi seguido um protocolo de testes que incluiu avaliações quantitativas automatizadas com métricas de desempenho, especificamente acurácia e função de perda, análise de-

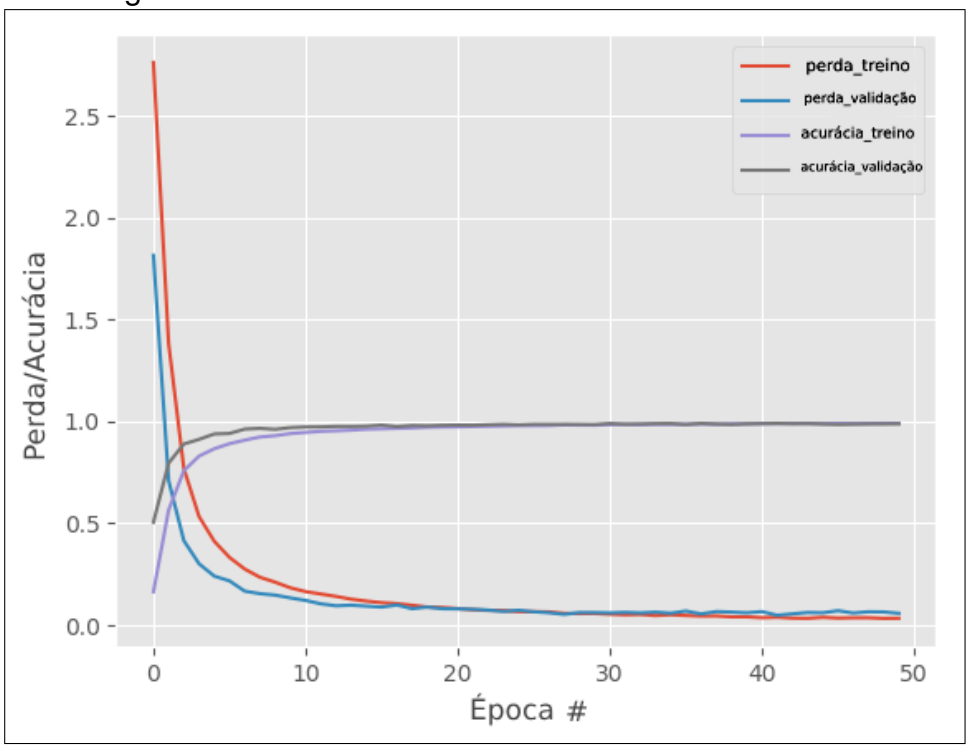
[4]Ocorre quando o modelo é excessivamente ajustado aos dados de treino, perdendo capacidade de generalização. Fonte: IBM. What is underfitting? Disponível em: https://www.ibm.com/topics/underfitting. Acesso em: 22 jun. 2025.

[5]Situação em que o modelo é simples demais para capturar os padrões dos dados, resultando em baixo desempenho. Fonte: IBM. What is underfitting? Disponível em: https://www.ibm.com/topics/underfitting. Acesso em: 22 jun. 2025.

talhada por classe a partir da matriz de confusão, testes práticos com webcam em condições reais de uso e comparação de acurácia entre trabalhos anteriores e o presente estudo.

Após o treinamento da rede neural convolucional por 50 épocas, foi gerado automaticamente um gráfico que ilustra a evolução da acurácia e da função de perda para os conjuntos de treino e validação. Como mostrado na Figura 9, observa-se uma rápida convergência durante as primeiras 10 épocas, com a perda do treinamento diminuindo progressivamente até se estabilizar em valores próximos de zero. A perda da validação segue tendência similar, evidenciando a ausência de overfitting. A acurácia, por sua vez, aumenta consistentemente durante o treinamento, atingindo 98,67% no conjunto de validação ao final da última época, valor semelhante ao obtido no conjunto de teste.

Figura 9 - Gráfico de Precisão e Perda do Treinamento.

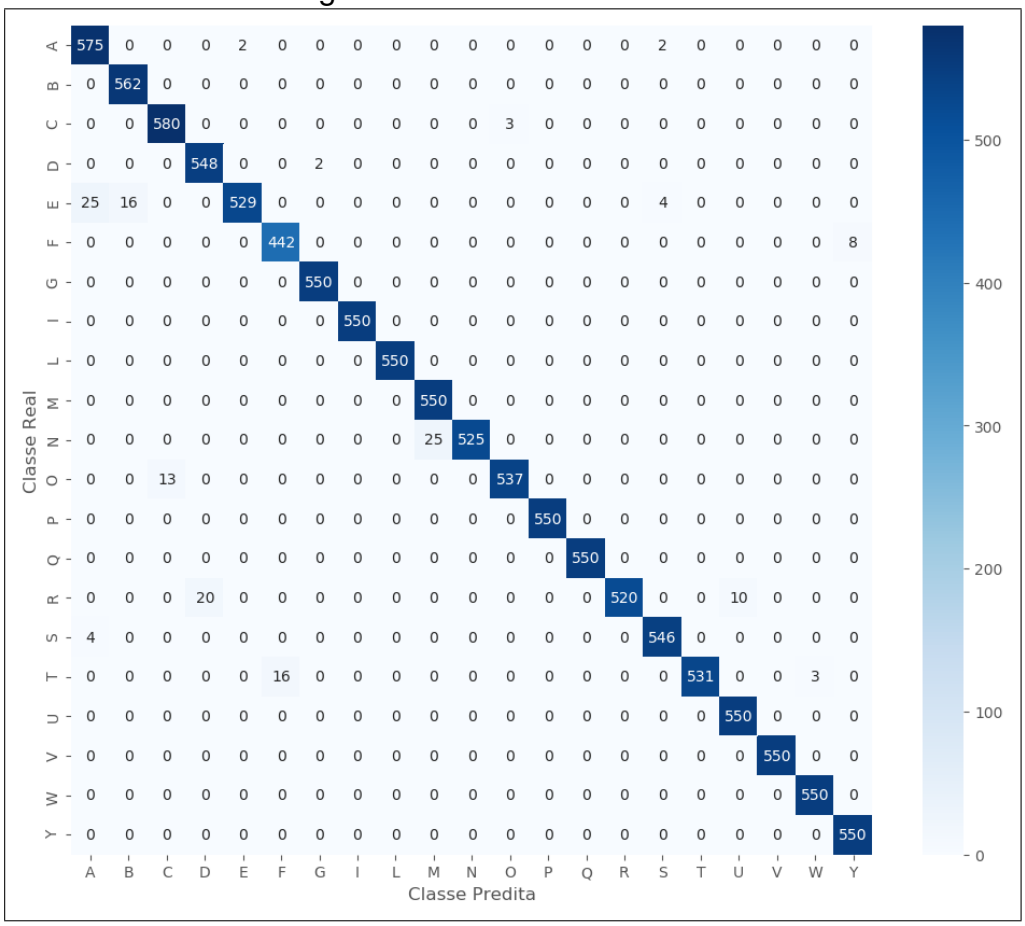


Fonte: Elaborado pelo autor.

Esse comportamento indica que o modelo aprendeu eficientemente os padrões presentes nos dados de entrada, mantendo uma boa capacidade de generalização para dados não vistos. A diferença entre as curvas de treino e validação é pequena e estável, o que confirma que estratégias como *data augmentation*, Early Stopping e o uso do otimizador Stochastic Gradient Descent foram eficazes para evitar o sobreajuste.

Além do bom desempenho geral, foi feita uma análise detalhada por classe, por meio da matriz de confusão apresentada na Figura 10. Essa avaliação permite identificar quais letras do alfabeto foram melhor reconhecidas pelo modelo e onde ocorreram as principais falhas.

Figura 10 - Matriz de Confusão.



Fonte: Elaborado pelo autor.

Observa-se que as letras B, G, I, L, M, P, Q, U, V, W, X, Y foram classificadas corretamente em 100% dos casos, sem qualquer erro, o que sugere que seus gestos possuem características visuais suficientemente distintas e consistentes para o modelo realizar a distinção com alta confiança.

Por outro lado, algumas letras apresentaram confusões notáveis. A letra A foi confundida com E e S, e a letra C com O, devido à semelhança nas configurações manuais fechadas e circulares, respectivamente. A letra D foi confundida com G, enquanto a letra E teve o maior número de erros, sendo incorretamente classificada como A, B e S, indicando dificuldade em distinguir dedos agrupados.

A letra F foi confundida com Y, possivelmente por falhas na visu-

alização do polegar e dedo mínimo. A letra N foi confundida com M, ambas com gestos similares. A letra O foi confundida com C, e a letra R com D e U, devido à proximidade dos dedos envolvidos. Também ocorreram confusões entre S e A, e entre T, F e W.

Esses erros ocorreram majoritariamente entre gestos de aparência semelhante, evidenciando os limites da distinção visual entre alguns sinais. Ainda assim, o modelo manteve alta precisão geral, com os erros concentrados em um grupo reduzido de classes.

Para avaliar o desempenho do sistema em cenários mais próximos da aplicação real, foram realizados testes em tempo real com o uso de uma webcam. As condições avaliadas incluíram iluminação adequada, presença de objetos no plano de fundo e variações no ângulo de exibição da mão em relação à câmera. Os sinais C, E e L foram selecionados por apresentarem estruturas gestuais visivelmente distintas, permitindo uma análise mais robusta da capacidade de generalização do sistema. Cada sinal foi executado 30 vezes em três diferentes condições, totalizando 270 amostras. A Tabela 1 apresenta um resumo dos resultados obtidos.

Tabela 1- Acurácia dos sinais C, E e L em diferentes cenários.

Sinal	Cenário de Teste	Acertos	Tentativas	Acurácia (%)
C	Iluminação adequada	30	30	100
C	Fundo com objetos	21	30	70
C	Ângulo da mão	29	30	97
E	Iluminação adequada	25	30	83
E	Fundo com objetos	20	30	66
E	Ângulo da mão	15	30	50
L	Iluminação adequada	29	30	97
L	Fundo com objetos	22	30	73
L	Ângulo da mão	21	30	70

Fonte: Elaborado pelo autor.

Os resultados demonstram que o modelo mantém alta precisão em condições semelhantes às do conjunto de treinamento, como fundo branco e iluminação adequada, mas apresenta queda significativa de desempenho em ambientes com ruídos visuais ou com alterações na posição da mão. A presença de objetos no fundo, interfere na segmentação da mão e impacta diretamente o desempenho da rede, principalmente devido a variações de cor que dificultam o destaque da mão em relação ao plano de fundo. Além disso, mudanças no ângulo de exibição da mão em relação à câmera prejudicam a detecção consistente, dificultando o reconhecimento.

O cenário mais crítico foi o de ângulo da mão desfavorável, no qual a acurácia caiu para 50% no caso do sinal E, evidenciando a dificuldade do modelo em interpretar corretamente os gestos quando partes dos dedos ficam ocultas ou distorcidas. Em testes adicionais, foi observado que, em ambientes com alto nível de interferência visual no plano de fundo, a acurácia pode cair drasticamente, chegando a 0% em alguns casos.

A comparação com o trabalho de Albino (2022) mostra um avanço significativo na acurácia do reconhecimento dos sinais da Libras. Enquanto Albino alcançou 92,5% sem utilizar técnicas como aumento de dados, normalização e otimizador SGD, o modelo desenvolvido neste estudo atingiu 98,67% ao aplicar essas estratégias. Apesar de os dois estudos não utilizarem a mesma base de imagens, as melhorias adotadas nesta pesquisa contribuíram para uma melhor generalização e maior precisão do sistema, evidenciando o progresso alcançado.

Apesar dos resultados promissores, o modelo apresentou algumas limitações relevantes. Observou-se uma redução significativa no desempenho em ambientes não controlados, especialmente em cenários com ruído visual no plano de fundo ou com ângulos desfavoráveis da mão em relação à câmera, comprometendo a segmentação da mão e afetando diretamente a consistência das predições. Embora tenham ocorrido algumas confusões entre gestos com configurações manuais semelhantes, como ilustrado na Figura 10 esses casos representaram uma parcela menor dos erros. A maioria das falhas foi atribuída a condições externas adversas, como interferências no plano de fundo. Esses fatores evidenciam limitações na capacidade de generalização do modelo, que se mostrou mais confiável em ambientes controlados.

4 CONCLUSÃO

O objetivo deste trabalho foi desenvolver um protótipo de aplicação web para identificação e classificação dos sinais do alfabeto manual da língua brasileira de sinais, utilizando redes neurais convolucionais treinadas com aprendizado supervisionado e o otimizador Stochastic Gradient Descent, integrando as bibliotecas TensorFlow, Keras e MediaPipe.

Os resultados obtidos demonstraram que o modelo desenvolvido apresenta alta precisão, atingindo 98,67% de acurácia no conjunto de validação, evidenciando a eficácia da arquitetura proposta e das técnicas de pré-processamento adotadas, como aumento de dados e normalização. A análise detalhada por classe e os testes práticos em tempo real confirma-

ram que o sistema consegue reconhecer sinais em condições controladas com grande confiabilidade, embora apresente redução de desempenho em ambientes com ruído visual ou ângulos desfavoráveis da mão.

Esses resultados indicam que o modelo é promissor para aplicações que visem facilitar a comunicação entre surdos e ouvintes por meio da Libras, especialmente em ambientes controlados. Contudo, para a aplicação em cenários reais, faz-se necessário aprimorar a robustez do sistema frente a variações ambientais e explorar estratégias que aumentem a generalização do modelo.

Como sugestões para trabalhos futuros, destacam-se: a incorporação de técnicas avançadas de segmentação e filtragem para melhorar a detecção da mão em fundos variados, o uso de arquiteturas de deep learning mais complexas, que possam lidar melhor com variações temporais e espaciais, a realização de testes com usuários reais, a fim de avaliar a usabilidade e a efetividade do sistema em contextos do cotidiano, e a ampliação do conjunto de classes, incluindo não somente o alfabeto manual, mas também sinais dinâmicos e expressões faciais.

REFERÊNCIAS

ALBINO, M. d. S. **Reconhecimento de linguagem de sinais utilizando o algoritmo Dynamic Time Warping para auxiliar surdos**. [S.l.: s.n.], 2022.

ARAÚJO, A. d. O. **2DCNN: Reconhecimento de sinais de libras utilizando uma rede neural convolucional 2D**. [S.l.: s.n.], 2023.

ASRITHA, K. R.; SOUNDARI, A. G. **Animal Collision Detection in Real Time to Prevent Accident using KNN Classifier in comparison with Stochastic Gradient Descent**. [S.l.]: IEEE, 2023. v. 6. 1478-1482 p.

BARRAL, J.; RUMJANEK, V. M. **Empréstimos linguísticos para sinais científicos na área de Biociências**. [S.l.: s.n.], 2018.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4.

CHOLLET, F. **Deep learning with Python**. [S.l.]: simon and schuster, 2021.

FAN, Y. **The Improvements for the Hands Gesture Recognition Based on the Mediapipe**. [S.l.]: IEEE, 2023. 748-753 p.

GERON, A. **Hands-on machine learning with Scikit-Learn, Keras & TensorFlow**. [S.l.]: OReilly, Sebastopol, 2019.

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

JAIN, E.; KAPOOR, S.; SINGH, M. **Advanced Chicken Disease Detection with Keras and TensorFlow Deep Learning Models**. [S.l.]: IEEE, 2024. 577-581 p.

JUNIOR, A. K.; SANTOS, N. dos; PENTEADO, F. C. **As pessoas surdas, as organizações e o compliance**. [S.l.: s.n.], 2022. v. 15.

KUZENKOV, S. **The Possibilities of Using Artificial Intelligence Technologies in Education and Science**. [S.l.]: IEEE, 2024. 88-91 p.

LEACINA, G. S. **Luva para interpretação de libras com comunicação a um aplicativo educacional**. 2021.

LI, D. et al. **Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison**. [S.l.]: IEEE, 2020. 1459–1469 p.

OLIVEIRA, E. F. et al. **Identificação do uso de cinto de segurança em crianças em veículos de passeio**. [S.l.]: Universidade Presbiteriana Mackenzie, 2023.

RODRIGUES-MOURA, S.; DESIDéRIO, R. **Ensino de física e educação para surdos no Brasil: uma revisão sistemática de literatura**. [S.l.: s.n.], 2023. v. 16. e023003 p.

RUDER, S. **An overview of gradient descent optimization algorithms**. 2017.

SHORTEN, C.; KHOSHGOFTAAR, T. M. **A survey on image data augmentation for deep learning**. Journal of big data, 2019, Springer, v. 6, n. 1, p. 1–48.

SILVA, E. P. da et al. **Silfa: Sign language facial action database for the development of assistive technologies for the deaf**. [S.l.]: IEEE, 2020. 688–692 p.

SILVA, L. d. **Aquisição de segunda língua: o estado da arte da Libras**. [S.l.]: SciELO Brasil, 2020. v. 64. e11861 p.

SOUZA, R. F. d. et al. **Camlibras: ferramenta complementar e acessível ao ensino de LIBRAS**. [S.l.]: Instituto Federal Goiano, 2023.

SZELISKI, R. **Computer vision: algorithms and applications**. [S.l.]: Springer Nature, 2022.

UÇAR, M. K. et al. **The effect of training and testing process on machine learning in biomedical datasets**. Mathematical Problems in Engineering, 2020, v. 2020, n. 1, p. 2836236.

YEBOAH, D.; SANOUSSI, M. S. A.; AGORDZO, G. K. **Image Classification Using TensorFlow GPU**. [S.l.]: IEEE, 2021. 1-5 p.