Convolutional Neural Networks for Sign Language Detection and Interpretation to Assist Individuals with Hearing Impairment

Gustavo Antonio dos Santos Fontana[1], **Marlon Oliveira**[1], **Valter Blauth Junior**[1]
Merisandra C. Mattos[1], **Sérgio Coral**[1]

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

{gustavoasf77,marlon.oliveira,valterblauth,mem,sergiocoral}@unesc.net

Abstract. *This paper presents a web-based prototype for recognizing and classifying Brazilian Sign Language (LIBRAS) manual alphabet signs using convolutional neural networks trained with supervised learning and optimized via Stochastic Gradient Descent. The system integrates TensorFlow, Keras, and MediaPipe for real-time gesture detection with 64x64 RGB images. The model reached 98.67% accuracy on the validation set, with similar test results. Twelve out of 21 letters achieved 100% precision. Real-time webcam tests showed accuracy ranging from 50% to 100% depending on conditions. Results indicate strong potential for accessible browser-based applications.*

Resumo. *Este artigo apresenta um protótipo web para identificação e classificação de sinais do alfabeto manual da Língua Brasileira de Sinais (LIBRAS), utilizando redes neurais convolucionais treinadas com aprendizado supervisionado e otimizadas com Stochastic Gradient Descent. A solução emprega TensorFlow, Keras e MediaPipe para detecção em tempo real, com imagens RGB de 64x64 píxeis. O modelo alcançou 98,67% de acurácia na validação, com desempenho semelhante no teste. A análise por classe indicou 100% de precisão em 12 das 21 letras. Em testes com webcam, a acurácia variou entre 50% e 100% conforme o cenário. Os resultados demonstram o potencial do sistema em aplicações acessíveis no navegador.*

1. Introdução

A deficiência auditiva é um desafio no Brasil, afetando mais de 10 milhões de pessoas, sendo 2,7 milhões com surdez profunda [Junior et al. 2022]. A LIBRAS, principal meio de comunicação dessa população, enfrenta barreiras de aprendizagem, especialmente entre ouvintes, devido à escassez de profissionais, materiais didáticos e tecnologias acessíveis [Barral and Rumjanek 2018, Silva 2020, Rodrigues-Moura and Desidério 2023].

Com os avanços tecnológicos, a computação tem contribuído com soluções inclusivas. A Inteligência Artificial (IA) vem se destacando por oferecer recursos para reconhecimento de padrões e análise de imagens em tempo real [Li et al. 2020], viabilizando ferramentas assistivas mais eficazes [da Silva et al. 2020]. Considerando as projeções de crescimento da população com deficiência auditiva até 2050, há uma demanda urgente por tecnologias que favoreçam a acessibilidade e a inclusão digital [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.

O levantamento bibliográfico mostra que a Inteligência Artificial tem se consolidado como um pilar da inovação tecnológica, especialmente nas áreas de acessibilidade e inclusão social. No contexto da comunicação entre surdos e ouvintes, o uso de Redes Neurais Convolucionais (CNNs) e visão computacional tem se destacado por possibilitar a tradução em tempo real da LIBRAS, promovendo inclusão digital [Kuzenkov 2024]. Segundo [Szeliski 2022], essas tecnologias permitem que máquinas interpretem dados visuais por meio de segmentação e extração de padrões. Inspiradas no córtex visual humano, as CNNs aprendem bordas, formas e texturas, sendo eficazes na identificação de gestos [Goodfellow et al. 2016]. Combinadas à visão computacional, permitem sistemas automáticos de tradução. O algoritmo Stochastic Gradient Descent (SGD) também se destaca por sua eficiência na otimização de modelos supervisionados [Asritha and Soundari 2023].

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 envolveu 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.

Já a etapa de Aplicação WEB corresponde ao desenvolvimento de uma aplicação

Figura 1. Etapas da 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 de 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 et al. 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 et al. 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 et al. 2016].

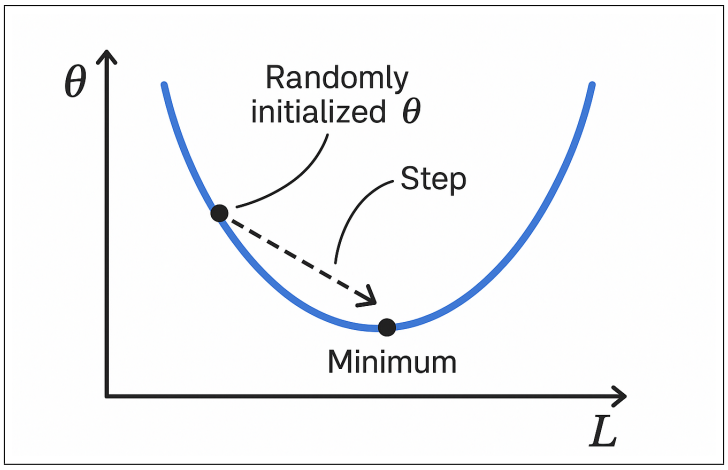

Figura 2. Trajetória de otimização na função de perda utilizando SGD.

A Figura 3 ilustra a regra de atualização dos pesos conforme descrita por [Bishop and 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.

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

Figura 3. Equação de atualização dos parâmetros na SGD.

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 and 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 et al. 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 imagens foram obtidas de um repositório público no GitHub e representam sinais estáticos do alfabeto manual. A base não inclui sinais dinâmicos nem expressões faciais, o que limita a aplicabilidade prática em comunicação real[1]. A base contém imagens de mãos representando sinais estáticos correspondentes ao alfabeto manual de Libras, contendo 21 classes distintas.

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

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

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 Aplicação Web

Foi desenvolvida uma aplicação web com HTML5 e JavaScript para reconhecer sinais da LIBRAS de forma acessível em computadores e dispositivos móveis. Executada diretamente no navegador, a aplicação utiliza a webcam para capturar imagens em tempo real, detectar a presença de uma mão, delimitar uma região central da imagem e classificar o gesto com um modelo de rede neural convolucional previamente treinado.

O modelo, originalmente em formato .h5, foi convertido para TensorFlow.js usando o utilitário tensorflowjs_converter, gerando arquivos model.json e .bin com os pesos, organizados na pasta model_tfjs e carregados diretamente no navegador via TensorFlow.js, permitindo inferência local sem necessidade de servidor.

A interface utiliza elementos como video e canvas para capturar e processar a imagem, redimensionando a região de interesse para 64×64 píxeis. As previsões são exibidas em tempo real em um campo h2, com atualização a cada segundo, informando o sinal detectado e o nível de confiança. A aplicação, hospedada no GitHub Pages, inclui recursos como alternância de tema e ativação dos *landmarks* da mão via MediaPipe.

A Figura 4 ilustra a interface da aplicação em funcionamento, demonstrando o reconhecimento dos sinais em tempo real em diferentes dispositivos, com destaque para a região de interesse, elementos da interface e exibição da predição, com ou sem os *landmarks* ativados.

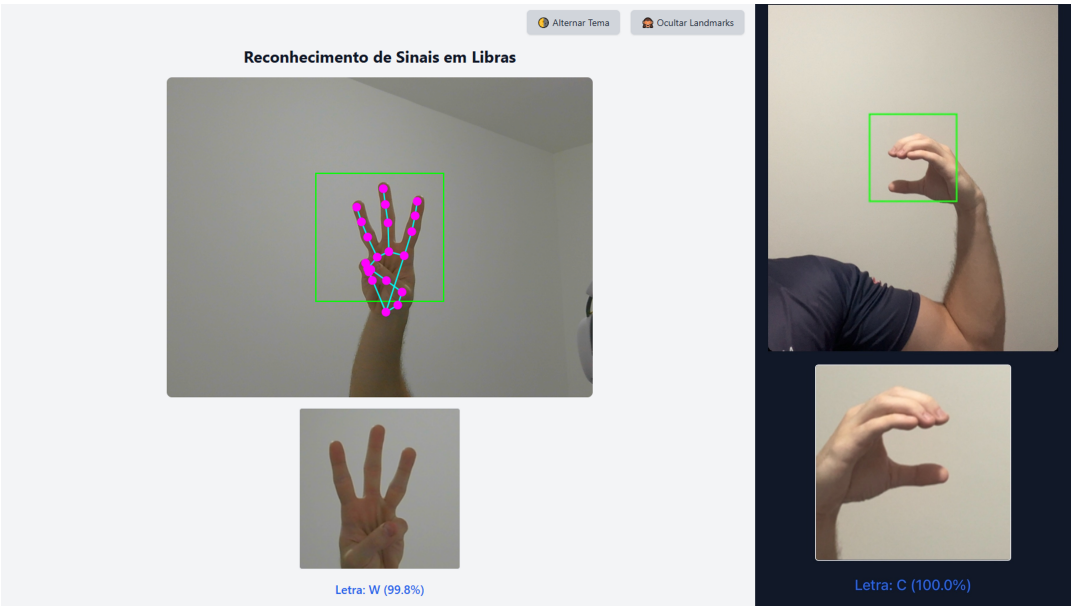

Figura 4. Aplicação em Execução.

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 5 ilustra o mapeamento dos pontos de referência da mão realizado pelo MediaPipe.

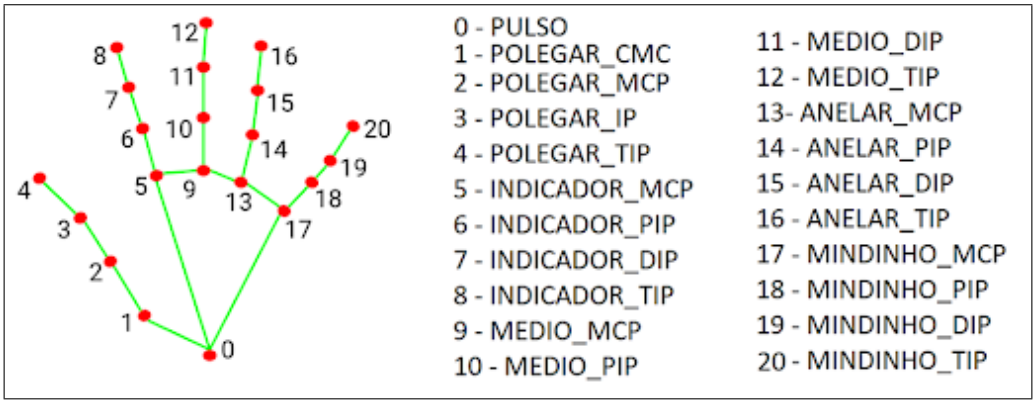

Figura 5. Mapeamento da mão.

2.3. Protocolo de Teste

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*[2] ou *underfitting*[3]. 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].

[2]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.

[3]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.

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.

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

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.

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, análise detalhada 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 6, 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.

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, a análise por classe com base na matriz de confusão (Figura 7) revelou que as letras B, G, I, L, M, P, Q, U, V, W, X e Y foram classificadas com 100% de acerto, indicando gestos visualmente distintos e de fácil identificação pelo modelo.

Em contrapartida, ocorreram confusões entre letras com gestos semelhantes. A foi confundida com E e S; C com O; D com G; e E a mais problemática com A, B e S, devido a agrupamentos de dedos similares. F apresentou confusão com Y, N com M, e R com D e U. Também houve trocas entre S e A, e entre T, F e W.

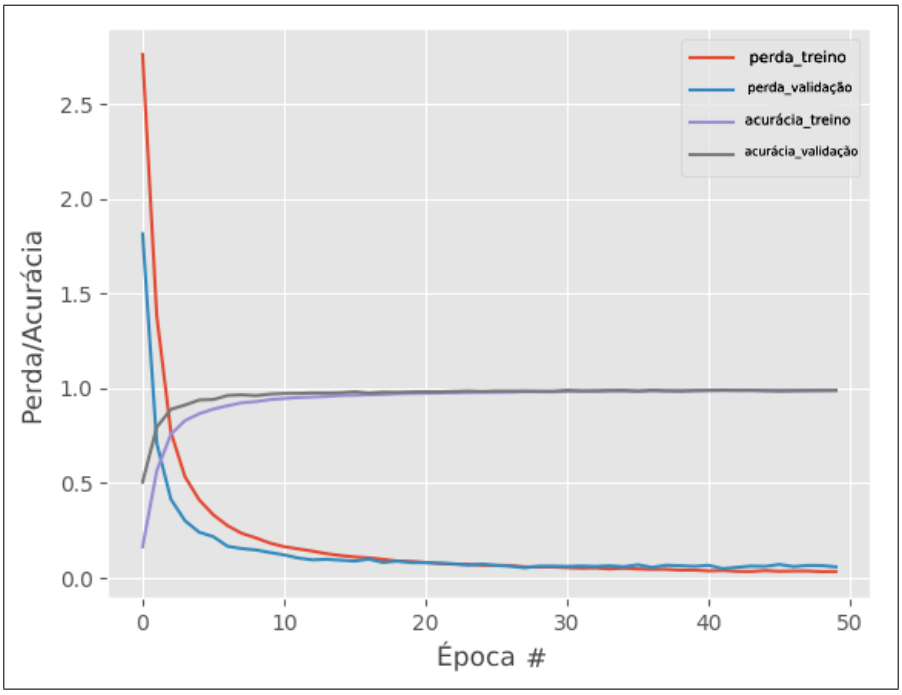

Figura 6. Gráfico de Precisão e Perda do Treinamento.

Esses erros refletem limitações na distinção de gestos visualmente próximos, mas a alta precisão em diversas classes reforça a robustez do modelo em cenários com maior contraste visual.

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.

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

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

Os resultados indicam que o modelo mantém alta precisão em condições similares

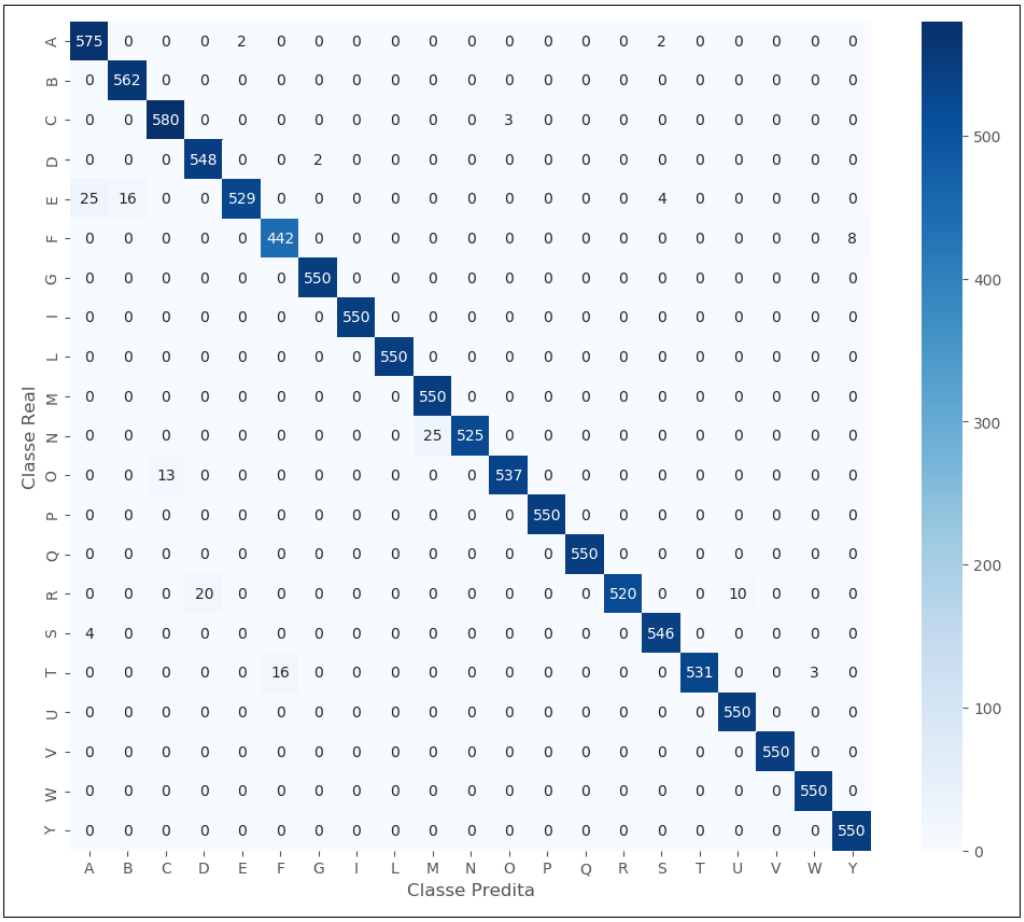

Figura 7. Matriz de Confusão.

ao conjunto de treinamento, como fundo branco e iluminação adequada, mas apresenta queda significativa em ambientes com ruídos visuais ou mudanças na posição da mão. Objetos no fundo interferem na segmentação da mão devido a variações de cor, afetando o desempenho da rede. Alterações no ângulo da mão em relação à câmera também dificultam a detecção consistente, como no caso do sinal E, cuja acurácia caiu para 50% quando partes dos dedos ficam ocultas ou distorcidas. Em ambientes com alto nível de interferência visual, a acurácia pode reduzir drasticamente, chegando a 0% em alguns casos.

A comparação com [Albino 2022] revela um avanço relevante: enquanto Albino obteve 92,5% sem técnicas como aumento de dados, normalização e uso do otimizador SGD, o modelo deste estudo alcançou 98,67% aplicando essas estratégias, o que contribuiu para melhor generalização e maior precisão.

Apesar dos bons resultados, o modelo mostrou limitações em ambientes não controlados, especialmente com ruídos visuais e ângulos desfavoráveis, comprometendo a segmentação da mão e a consistência das predições. Algumas confusões ocorreram entre gestos com configurações semelhantes, mas representaram uma parcela menor dos erros. A maioria das falhas deve-se a condições externas adversas, evidenciando que o modelo é 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 Libras, utilizando redes neurais convolucionais treinadas com aprendizado supervisionado e o otimizador Stochastic Gradient Descent, integrando TensorFlow, Keras e MediaPipe.

Os resultados mostraram que o modelo atingiu 98,67% de acurácia no conjunto de validação, evidenciando a eficácia da arquitetura e das técnicas de pré-processamento, como aumento de dados e normalização. A análise por classe e os testes em tempo real confirmaram a confiabilidade do sistema em condições controladas, embora haja queda de desempenho em ambientes com ruído visual ou ângulos desfavoráveis da mão.

O modelo é promissor para facilitar a comunicação entre surdos e ouvintes por meio de LIBRAS, especialmente em ambientes controlados. Para uso em cenários reais, é necessário aprimorar a robustez frente a variações ambientais e explorar estratégias para aumentar a generalização.

Como trabalhos futuros, sugerem-se técnicas avançadas de segmentação para melhorar a detecção da mão em fundos variados, uso de arquiteturas deep learning mais complexas para lidar com variações temporais e espaciais, testes com usuários reais para avaliar usabilidade, e ampliação do conjunto de classes, incluindo sinais dinâmicos e expressões faciais.

Referências

Albino, M. d. S. (2022). *Reconhecimento de linguagem de sinais utilizando o algoritmo Dynamic Time Warping para auxiliar surdos*.

Araújo, A. d. O. (2023). *2DCNN: Reconhecimento de sinais de libras utilizando uma rede neural convolucional 2D*.

Asritha, K. R. and Soundari, A. G. (2023). *Animal Collision Detection in Real Time to Prevent Accident using KNN Classifier in comparison with Stochastic Gradient Descent*, volume 6. IEEE.

Barral, J. and Rumjanek, V. M. (2018). *Empréstimos linguísticos para sinais científicos na área de Biociências*. Number 49.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Chollet, F. (2021). *Deep learning with Python*. simon and schuster.

da Silva, E. P., Costa, P. D. P., Kumada, K. M. O., and De Martino, J. M. (2020). *Silfa: Sign language facial action database for the development of assistive technologies for the deaf*. IEEE.

Fan, Y. (2023). *The Improvements for the Hands Gesture Recognition Based on the Mediapipe*. IEEE.

Geron, A. (2019). Hands-on machine learning with scikit-learn, keras & tensorflow.

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

Jain, E., Kapoor, S., and Singh, M. (2024). *Advanced Chicken Disease Detection with Keras and TensorFlow Deep Learning Models*. IEEE.

Junior, A. K., dos Santos, N., and Penteado, F. C. (2022). *As pessoas surdas, as organizações e o compliance*, volume 15.

Kuzenkov, S. (2024). *The Possibilities of Using Artificial Intelligence Technologies in Education and Science*. IEEE.

Li, D., Rodriguez, C., Yu, X., and Li, H. (2020). *Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison*. IEEE.

Oliveira, E. F., Takada, K. Y. P., Santos, O. A. A. K., and Brocco, R. M. (2023). *Identificação do uso de cinto de segurança em crianças em veículos de passeio*. Universidade Presbiteriana Mackenzie.

Rodrigues-Moura, S. and Desidério, R. (2023). *Ensino de física e educação para surdos no Brasil: uma revisão sistemática de literatura*, volume 16.

Ruder, S. (2017). An overview of gradient descent optimization algorithms.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

Silva, L. d. (2020). *Aquisição de segunda língua: o estado da arte da Libras*, volume 64. SciELO Brasil.

Souza, R. F. d. et al. (2023). *Camlibras: ferramenta complementar e acessível ao ensino de LIBRAS*. Instituto Federal Goiano.

Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.

UÇar, M. K., Nour, M., Sindi, H., and Polat, K. (2020). The effect of training and testing process on machine learning in biomedical datasets. *Mathematical Problems in Engineering*, 2020(1):2836236.

Yeboah, D., Sanoussi, M. S. A., and Agordzo, G. K. (2021). *Image Classification Using TensorFlow GPU*. IEEE.