Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Reconhecimento de sinais da Libras em vídeos no contexto educacional utilizando aprendizado de máquina

Gabriel Frasson Costa

JUIZ DE FORA
JANEIRO, 2026

Reconhecimento de sinais da Libras em vídeos no contexto educacional utilizando aprendizado de máquina

Gabriel Frasson Costa

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Luiz Maurílio da Silva Maciel

JUIZ DE FORA
JANEIRO, 2026

Reconhecimento de sinais da Libras em vídeos no contexto educacional utilizando aprendizado de máquina

Gabriel Frasson Costa

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Luiz Maurílio da Silva Maciel
Doutor em Engenharia de Sistemas e Computação

Marcelo Caniato Renhe
Doutor em Engenharia de Sistemas e Computação

Rodrigo Luis de Souza da Silva
Doutor em Engenharia Civil

JUIZ DE FORA

09 DE JANEIRO, 2026

Resumo

A inclusão da comunidade surda no ambiente educacional brasileiro enfrenta barreiras significativas devido à carência de intérpretes e à falta de fluência na Língua Brasileira de Sinais (Libras) por parte de professores e alunos ouvintes. Este trabalho apresenta o desenvolvimento de um modelo de reconhecimento de sinais dinâmicos da Libras focado no contexto de sala de aula, utilizando técnicas de aprendizado de máquina. Para isso, foi construído um conjunto de dados próprio contendo 900 vídeos de 20 sinais distintos, executados por nove sinalizadores. A metodologia consistiu na extração de pontos de referência (*keypoints*) por meio da ferramenta MediaPipe Holistic, seguida pelo cálculo de características geométricas (ângulos e distâncias) e de movimento. O modelo de classificação baseou-se em redes neurais recorrentes (RNN) do tipo Long Short-Term Memory (LSTM). Os resultados demonstram que a integração de características de movimento, juntamente com o emprego de modelos especialistas para distinguir classes ambíguas, proporcionaram uma acurácia geral do modelo de 88,89%. A abordagem contribui para o avanço dos estudos na tarefa de reconhecimento de sinais da Libras, oferecendo uma base para o desenvolvimento de futuras ferramentas de acessibilidade no contexto educacional.

Palavras-chave: Reconhecimento de sinais, aprendizado de máquina, sinais dinâmicos, contexto educacional, visão computacional.

Abstract

The inclusion of the deaf community in the Brazilian educational environment faces significant barriers due to the shortage of interpreters and the lack of fluency in Brazilian Sign Language (Libras) among hearing teachers and students. This work presents the development of a dynamic Libras sign recognition model focused on the classroom context, using machine learning techniques. For this purpose, a proprietary dataset containing 900 videos of 20 distinct signs, performed by nine signers, was constructed. The methodology consisted of extracting keypoints using the MediaPipe Holistic tool, followed by the calculation of geometric features (angles and distances) and movement features. The classification model was based on Long Short-Term Memory (LSTM) recurrent neural networks (RNN). The results demonstrate the integration of movement features, with the use of specialist models to distinguish ambiguous classes, provided an overall accuracy of the model of 88.89%. The approach contributes to the advancement of studies in the task of Libras sign recognition, providing a basis for the development of future accessibility tools in the educational context.

Keywords: Sign recognition, machine learning, dynamic signs, educational context, computer vision.

Agradecimentos

A Deus, à minha família, à minha namorada e aos meus amigos, agradeço pelo apoio incondicional, pela paciência e pelo incentivo constante em cada etapa desta jornada.

Ao professor Luiz Maurílio da Silva Maciel pela orientação constante e pela confiança desde a iniciação científica na área de visão computacional, até este presente trabalho. Sua competência e direcionamento foram fundamentais para minha formação acadêmica.

Aos alunos e professores da faculdade de Letras da UFJF que contribuíram imensamente na gravação dos sinais para compor o conjunto de dados: Aline Garcia Rodero Takahira, Hadassa Rodrigues Santos, Mariângela Simão Albani, Erick Enrique Rodrigues, Danielle, Lilian, Maria Vitória, Maria Eduarda e Mirella. Agradeço também ao Pedro Bittencourt pela grande ajuda com as gravações.

Por fim, aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ASL	*American Sign Language*
AdamW	*Adaptive Moment Estimation with Weight Decay*
BSL	*British Sign Language*
CNN	*Convolutional Neural Network*
CSL	*Chinese Sign Language*
CSLR	Continuous Sign Language Recognition
GIL	*Global Interpreter Lock*
HOG	*Histogram of Oriented Gradients*
ISL	*Indian Sign Language*
ISLR	*Isolated Sign Language Recognition*
Libras	Língua Brasileira de Sinais
LSF	Língua de Sinais Francesa
LSK	Língua de Sinais Kaapor
LSTM	*Long Short-Term Memory*
LTS	Língua Terena de Sinais
RNN	*Recurrent Neural Network*
SLTr	*Sign Language Transformer*
UFJF	Universidade Federal de Juiz de Fora
WSL	*Windows Subsystem for Linux*
YOLO	*You Only Look Once*

1 Introdução

A Língua Brasileira de Sinais (Libras) é a língua utilizada pela comunidade surda no Brasil, como meio principal de comunicação. Entretanto, apesar de a língua ser essencial na comunidade e reconhecida oficialmente por lei (BRASIL, 2002), apenas uma pequena parcela de ouvintes sabe se comunicar por meio dela, o que prejudica a inclusão dos surdos na sociedade.

A Libras utiliza recursos espaço-visuais na constituição dos sinais, cujos principais elementos são a configuração da mão, expressões não manuais, orientação da palma da mão, localização do sinal e movimento (ARAúJO; FERREIRA, 2016). Dado que a língua é essencialmente visual-motora, o movimento é uma característica muito presente e determinante na maioria dos sinais.

Considerando essa natureza essencialmente visual e dinâmica, a tarefa de reconhecimento de sinais é definida, no contexto da computação, como o processo de ensinar um sistema a identificar e classificar um sinal específico a partir de um vídeo. A ideia geral é que um algoritmo de inteligência artificial seja capaz de analisar um vídeo e determinar qual sinal foi executado, fornecendo a respectiva tradução em português.

Embora existam vários estudos na área de reconhecimento de sinais de outras línguas, como *American Sign Language* (ASL) (BUTTAR et al., 2023), *Chinese Sign Language* (CSL) (HE et al., 2023), *Indian Sign Language* (ISL) (DIKSHANT et al., 2024), os estudos para a Libras possuem espaço para muitos avanços e enfrentam diversos desafios.

Um dos principais desafios é a escassez de bases de dados de sinais públicas, uma vez que as bases existentes possuem uma gama razoável de sinais, porém com poucas amostras de sinalizadores para cada sinal. Essa característica dificulta consideravelmente o treinamento de modelos que realizam as tarefas de reconhecimento. Dessa forma, grande parte das pesquisas na área busca construir suas próprias bases de dados aplicadas em contextos específicos, combinando técnicas de aumento de dados (*data augmentation*) visando obter resultados melhores nos treinamentos.

Além disso, algumas pesquisas abordam a tarefa de reconhecimento de sinais com recursos auxiliares para aprimorar a detecção dos movimentos, como é o caso de sensores, luvas ou câmeras de profundidade. Apesar de haver um ganho no desempenho do modelo, essas técnicas restringem o contexto de utilização de uma ferramenta de reconhecimento de sinais, de modo a sempre depender desses recursos auxiliares. Em contrapartida, os estudos que realizam o reconhecimento utilizando apenas uma câmera também apresentam bons resultados e são mais flexíveis e acessíveis para uma implementação em um contexto prático na sociedade.

Diante do amplo espectro de desafios de inclusão da comunidade surda, o contexto educacional destaca-se como uma área consideravelmente crítica. A dificuldade de acesso a intérpretes em tempo integral nas salas de aula e a falta de fluência de professores e alunos na Libras conferem uma barreira à comunicação. Visto que o ambiente educacional de uma sala de aula é um espaço fundamental para o desenvolvimento e inclusão social, é necessária uma intervenção para que essas dificuldades sejam superadas. Logo, o foco deste trabalho consiste justamente no reconhecimento de sinais comumente utilizados em salas de aula.

Dessa forma, este estudo justifica-se pela necessidade de contribuir para o avanço na tarefa de reconhecimento de sinais da Libras com movimento, uma vez que a comunicação e a inclusão dos surdos na sociedade enfrentam inúmeras barreiras que precisam ser quebradas. Acredita-se que, com a contribuição proposta, mais um passo será dado para tornar a comunicação mais acessível.

1.1 Objetivos

O objetivo geral do trabalho é desenvolver e treinar um modelo capaz de reconhecer sinais da Libras com movimento no contexto educacional e no contexto da computação.

Os objetivos específicos consistem em:

- Construir uma base de dados de vídeos de sinais no contexto educacional e no contexto da computação;
- Desenvolver e treinar um modelo de aprendizado de máquina para a tarefa de reco-

nhecimento de sinais da Libras com movimento;

- Explorar a utilização da base de dados construída no treinamento do modelo e avaliar o desempenho.

2 Fundamentação Teórica

Este capítulo descreve os fundamentos teóricos necessários para a construção do trabalho. Ele é dividido em duas seções, sendo elas a Seção 2.1 que trata sobre a Língua Brasileira de Sinais (Libras) e a Seção 2.2 que discorre sobre o tipo de rede neural *Long Short-Term Memory* (LSTM).

2.1 Língua Brasileira de Sinais (Libras)

A Libras é a língua oficial da comunidade surda no Brasil, reconhecida pela lei nº 10.436/2002 (BRASIL, 2002). É o meio de comunicação pelo qual os surdos conseguem expressar-se livremente com recursos espaço-visuais. Entretanto, mesmo após mais de duas décadas do reconhecimento oficial, poucas pessoas ouvintes sabem se comunicar por meio dela, e os recursos de acessibilidade seguem insuficientes para incluir os surdos na sociedade de forma plena.

Embora seja a língua de sinais predominante no Brasil, a Libras não é a única existente. Existem inúmeras línguas de sinais por todo o mundo, com diferentes variações linguísticas, como regionalismos e sotaques. Cada língua possui sua própria estrutura léxica e gramatical, que pode diferir completamente de outras. Alguns exemplos de outras línguas conhecidas internacionalmente são: Língua de Sinais Americana (ASL), Língua de Sinais Francesa (LSF), Língua de Sinais Britânica (BSL), Língua de Sinais Chinesa (CSL) e Língua de Sinais Indiana (ISL). Dentro do Brasil, apenas a Libras é oficialmente reconhecida por lei como a língua da comunidade surda. No entanto, existem muitas outras línguas de sinais indígenas independentes da Libras, como a Língua de Sinais Kaapor (LSK) e a Língua Terena de Sinais (LTS).

Durante muitos anos, a língua de sinais sofreu grande resistência e preconceito na sociedade. A técnica do oralismo era a única forma permitida de comunicação entre os surdos, devido ao Congresso de Milão, que ocorreu em 1880. Somente a partir de 1960 as línguas de sinais voltaram a ser aceitas na sociedade, graças aos estudos de Stokoe (1960).

Ele concluiu que as línguas de sinais possuem estruturas léxicas e sintáticas completas, capazes de produzir infinitas frases e expressar ideias como qualquer outra língua. Dessa forma, entende-se que a Libras não é português gesticulado, uma vez que possui sintaxe, semântica e pragmática próprias, totalmente independentes da língua oral. Ela é completa em si mesma e carrega recursos suficientes para expressar qualquer ideia.

Segundo Araújo e Ferreira (2016), o sinal em Libras é formado por um conjunto de cinco parâmetros:

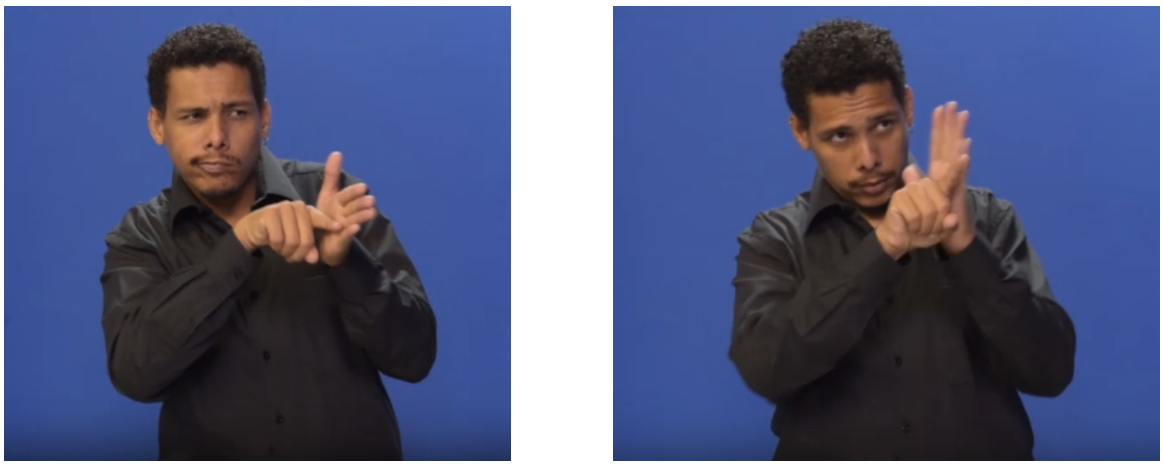
- **Configuração da mão:** Formas que as mãos assumem na realização do sinal, podendo ser em formas de letras, números ou outras;
- **Ponto de articulação:** Espaço em frente ao corpo onde os sinais são articulados;
- **Movimento:** Deslocamento da mão no espaço;
- **Orientação:** Orientação da palma da mão durante a realização do sinal, que pode ser para cima, para baixo, para dentro, para fora ou para o lado;
- **Expressões não manuais:** A expressão facial que é usada para definir ou intensificar os significados dos sinais.

A grande maioria dos sinais da Libras possui bastante movimento, porém nem todos se configuram dessa forma. Há também alguns sinais estáticos, como a datilologia, ou alfabeto manual, que é a representação das letras do alfabeto da língua portuguesa por meio de configurações de mão na Libras.

As Figuras 2.1a e 2.1b exemplificam dois sinais da Libras “Pesquisar” e “Perguntar” que se distinguem majoritariamente pelo movimento.

2.2 Long Short-Term Memory (LSTM)

As Redes Neurais Recorrentes (RNNs) são uma classe de arquiteturas de aprendizado profundo projetadas especificamente para o processamento de dados sequenciais, como séries temporais, áudio ou vídeo. Diferente das redes neurais convencionais, as RNNs possuem conexões cíclicas que permitem que a informação persista, funcionando como uma



(a) Sinal “Pesquisar” (b) Sinal “Perguntar”

Figura 2.1: Sinais de “Pesquisar” e “Perguntar” (QUADROS, R., 2023).

memória interna que retém conhecimentos de entradas anteriores para influenciar a saída. Entretanto, as RNNs tradicionais apresentam limitações ao lidar com sequências longas, por conta do problema do desaparecimento ou explosão do gradiente, o que as impede de aprender dependências de longo prazo de maneira eficaz (LIPTON; BERKOWITZ; ELKAN, 2015).

Para mitigar esses problemas, Hochreiter e Schmidhuber (1997) introduziram a arquitetura de Memória de Longo e Curto Prazo (*Long Short-Term Memory* - LSTM). O diferencial da LSTM consiste em uma estrutura de célula de memória, que regula o fluxo de informações por meio de portas (*gates*). Essas portas utilizam pesos para decidir quais dados devem ser armazenados, descartados ou carregados para o próximo estado. A unidade LSTM é composta por:

- **Porta de Esquecimento (*Forget Gate*):** Responsável por identificar quais informações do estado anterior da célula não são mais relevantes e devem ser descartadas.
- **Porta de Entrada (*Input Gate*):** Decide quais novas informações da entrada atual serão atualizadas e armazenadas no estado da célula.
- **Porta de Saída (*Output Gate*):** Determina qual parte do estado atual da célula será enviada como saída para a próxima etapa da sequência.

No domínio do reconhecimento de sinais, a rede LSTM destaca-se pela capacidade

de processamento temporal, sendo útil para capturar a dinâmica do movimento, que é determinante para o significado da maioria dos sinais na Libras. Como a execução de um sinal envolve uma trajetória de movimentos das mãos e do corpo, a LSTM consegue aprender as correlações entre os quadros, permitindo uma rastreabilidade do movimento ao longo do tempo.

3 Trabalhos Relacionados

Existem inúmeras formas de abordar a tarefa de reconhecimento de sinais em visão computacional, e diferentes métodos são utilizados de acordo com as características do problema. A categorização é definida em relação à continuidade dos sinais e à presença ou não de movimento. A tarefa de reconhecer sinais individuais, muitas vezes em nível de letras ou palavras, é chamada de *Isolated Sign Language Recognition* (ISLR). Já a tarefa de interpretar um fluxo contínuo de vídeo em uma sequência de sinais, sem necessidade de pausas entre os sinais, é chamada de *Continuous Sign Language Recognition* (CSLR). Por outro lado, os sinais também são categorizados pela presença de movimento, de modo que os sinais sem movimento são chamados de sinais estáticos, enquanto os demais, com movimento, são comumente chamados de sinais dinâmicos.

Alguns estudos utilizam algoritmos clássicos para classificar os sinais, como o Histograma de Gradientes (*Histogram of Oriented Gradients* - HOG) (BASTOS, 2015). Porém esses métodos geralmente ficam limitados a sinais estáticos, pois não desempenham bem sozinhos quando se adiciona o movimento. Para reconhecimento de sinais estáticos, o modelo *You Only Look Once* (YOLO) também é muito utilizado em diferentes versões e apresenta bons resultados (BUTTAR et al., 2023).

Já para a tarefa de reconhecimento de sinais com movimento, os algoritmos clássicos raramente são utilizados isoladamente na literatura. As estratégias mais utilizadas são baseadas em algoritmos de aprendizado profundo e existem inúmeras maneiras de abordar o problema. Assim, frequentemente, encontram-se trabalhos que exploram diferentes combinações de Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN) e Redes Neurais Recorrentes (*Recurrent Neural Network* - RNN) que alcançam um bom desempenho no reconhecimento. Rajapakshe et al. (2025) aborda a tarefa combinando uma CNN treinada para extrair características dos vídeos, com uma rede LSTM responsável pelo reconhecimento dos sinais a partir dos dados extraídos.

As CNNs são redes especializadas em analisar dados com topologia de grade, como imagens, sendo ótimas para processar os quadros dos vídeos de sinais. Enquanto

isso, as RNNs são projetadas para processar dados sequenciais, como séries temporais, utilizando uma memória para reter informações de entradas anteriores, o que contribui na análise da continuidade dos sinais e rastreabilidade do movimento. Já a rede LSTM é um tipo de RNN bastante presente na literatura em tarefas de reconhecimento de sinais.

É possível processar os dados de entrada das redes de maneiras diferentes. A abordagem mais comum é extrair pontos relevantes dos quadros (*keypoints*) e utilizá-los diretamente como entrada da rede. He et al. (2023) faz a extração dos *keypoints* do esqueleto e da mão e os utiliza como entrada em uma rede LSTM para realizar o reconhecimento da língua de sinais chinesa. Além disso, uma outra estratégia interessante, utilizada por Alves, Boldt e Paixão (2024), é codificar os *keypoints* extraídos dos quadros em imagens bidimensionais, contendo informações espaciais e temporais. Em seguida, uma rede CNN construída sobre a rede residual profunda de 18 camadas (ResNet18) é alimentada com os quadros codificados para classificar os sinais da Libras.

Ao invés de alimentar uma rede neural com *keypoints*, também é possível fornecer uma sequência de quadros diretamente a uma rede. É o caso do trabalho de Singh (2021), que organiza os quadros sequencialmente e os envia como entrada para uma rede neural 3D-CNN. Então, ela processa os quadros e já devolve a classificação como saída, sem precisar de uma extração prévia de *keypoints*. Uma vantagem dessa abordagem é que isso pode contribuir para a detecção de outros elementos importantes no sinal, além do movimento, como, por exemplo, a expressão facial.

Estudos mais recentes também fazem uso de modelos mais complexos na tarefa de reconhecimento de sinais. Damdoo e Kumar (2025) constroem um modelo chamado *SignEdgeLVM*, baseado na arquitetura *Sign Language Transformer* (SLTr). Ele processa as representações de quadros espaço-temporais usando camadas de auto-atenção (*self-attention*) e *feed forward*. Por conta do mecanismo de auto-atenção, que ajuda o modelo a entender o contexto temporal, foi possível realizar o reconhecimento de sinais contínuos (CSLR). Essa tarefa é consideravelmente mais complexa e exige tanto um modelo mais profundo quanto um volume bem maior de dados para treinamento, além de um poder computacional elevado.

Algumas técnicas para aprimorar a performance dos modelos também estão pre-

sentes na literatura, como o *fine-tuning* em modelos pré-treinados. Dentre os benefícios de se utilizar modelos pré-treinados, um dos mais perceptíveis e relevantes é obter um melhor desempenho quando há poucos dados no conjunto do problema a ser resolvido. Uma vez que os modelos são pré-treinados em conjuntos de dados consideravelmente maiores, as redes aprendem características de alto nível que servem como fundamentos para extrair novos aprendizados a partir de conhecimentos já consolidados. Além disso, o tempo de treinamento e o sobreajuste (*overfitting*) podem ser reduzidos com a aplicação dessa técnica.

A arquitetura TCNet do estudo proposto por Lu, Salah e Poppe (2024) utiliza uma rede SpyNet pré-treinada para estimar movimento e alcançou resultados do estado da arte na tarefa de reconhecimento de sinais. Outro exemplo é o estudo de Liu et al. (2021), que utiliza a arquitetura VGG16 pré-treinada como base. Os autores utilizaram as primeiras quatro camadas da VGG16 e adicionaram mais cinco camadas de convolução dilatada no modelo, atingindo uma taxa de reconhecimento de 93,5% no conjunto de dados público *Sahand*.

Modelos pré-treinados também são utilizados para realizar tarefas de extração de características, como no estudo de Bansal e Jain (2023), em que a ResNet50 pré-treinada é aplicada nos vídeos do conjunto de dados para servir de entrada para uma rede RNN posteriormente. Uma ideia semelhante se aplica ao estudo de Zuo e Mak (2022), que utiliza a arquitetura VGG11 para extrair características visuais dos quadros dos vídeos de entrada.

Além disso, outra técnica muito presente é o aumento de dados. Essa abordagem é comumente utilizada quando não há muitos dados no conjunto, de forma a tentar reduzir o *overfitting* e alcançar uma melhor generalização do modelo durante o treinamento. Alguns aumentos podem ser aplicados, como transformações em imagens (rotação, inversão, translação, brilho, contraste, ruído, *zoom*), transformações em vídeos (deformação temporal elástica, recorte) e outras que contribuam para a generalização dos dados.

Os autores Oropesa, Felicen e Guzman (2024) realizaram um aumento de 10 vezes dos dados, partindo de um conjunto de 97 vídeos por sinal, para um novo conjunto de 970 vídeos por sinal. Com o melhor modelo e o conjunto de dados aumentado, foram obtidas

acurácias de 98% no treinamento e 86% nos testes em ambiente real, o que sugere um bom aproveitamento dessa abordagem.

Existe também a possibilidade de extrair características personalizadas, resultantes da combinação de outras características. Sarmento (2023) treinou um modelo utilizando um vetor de 90 características aglomeradas, provenientes da extração dos ângulos entre as conexões dos *keypoints* de cada mão e da extração da distância entre os *keypoints* da pose. Para os ângulos ou distâncias que não estavam presentes, foi atribuído o valor 0 como forma de preenchimento (*padding*). Comparada às outras abordagens testadas pela autora, esta obteve os melhores resultados de acurácia, além de facilitar o treinamento do modelo.

Este presente trabalho se propõe em utilizar uma abordagem muito semelhante à de Sarmento (2023), com a mesma arquitetura do modelo e extração de características personalizadas. Entretanto, como contribuição deste trabalho, são implementados modelos especialistas para sinais ambíguos e elaboradas outras características geométricas e de movimento. Além disso, são realizados experimentos variando o tamanho da rede e quantidade de quadros selecionados.

4 Conjunto de dados

Quando se trata de treinar um modelo de aprendizado profundo para uma tarefa de reconhecimento, é necessário ter um conjunto de dados representativo. Atualmente, existem alguns conjuntos conhecidos na literatura que apresentam uma gama razoável de sinais. Porém o maior desafio de utilizar esses conjuntos no modelo é a pequena quantidade de vídeos por sinal. Como mostra a Tabela 4.1, existem, no máximo, 7 vídeos por classe em um único conjunto de dados. Além disso, o tamanho dos vídeos, a resolução e o plano de fundo não são padronizados, exigindo um trabalho de pré-processamento para fornecer os dados ao modelo. Existem também os conjuntos de dados MINDS - Libras[1] e Libras - UFOP[2] que são mais abrangentes, porém também não focam em sinais do contexto educacional.

Tabela 4.1: Quantidade de classes, sinalizadores e vídeos por classe nas diferentes fontes de dados.

Fonte	**Nº classes distintas**	**Nº sinalizadores**	**Nº vídeos por classe**	**Nº total de vídeos**
UFPE[3]	1396	4	De 1 a 7	4221
UFV[4]	1004	3	De 1 a 4	1029
INES[5]	237	1	De 1 a 2	282
SignBank[6]	485	1	1	485

Fonte: Sarmento (2023).

Dessa forma, a maioria dos autores na literatura escolhe construir um conjunto de dados próprio. Dado que o objetivo deste trabalho é reconhecer sinais do contexto educacional e, considerando os fatores citados, também optou-se por construir um conjunto de dados específico para essa tarefa. Ao todo, foram escolhidos 20 sinais diferentes para constituir o conjunto de dados, de forma que todos apresentem movimento e se enquadrem no contexto de sinais utilizados em uma sala de aula. Mais especificamente,

[1] https://www.kaggle.com/datasets/j0aopsantos/minds-libras
[2] https://www.kaggle.com/datasets/andersonls/libras-ufop-dataset
[3] https://libras.cin.ufpe.br/
[4] https://sistemas.cead.ufv.br/capes/dicionario/
[5] https://www.ines.gov.br/dicionario-de-libras/
[6] https://signbank.libras.ufsc.br/pt

oito sinais estão ligados à área da computação e podem ser utilizados em uma sala de aula, enquanto doze sinais são mais genéricos no contexto educacional. Os sinais escolhidos foram: "Professor", "Estudar", "Aluno", "Dúvida"[7], "Perguntar", "Responder", "Entender", "Pesquisar"[8], "Aprender"[9], "Quadro", "Prova", "Trabalho escolar", "Computador"[10], "Internet"[11], "Tecnologia", "Algoritmo", "Inteligência artificial", "Software", "Código fonte" e "Compilador". Os doze primeiros são os sinais genéricos de sala de aula, enquanto os oito últimos são os sinais específicos da computação.

4.1 Captura dos vídeos

Os vídeos dos sinais foram gravados por alunos e professores da faculdade de Letras da Universidade Federal de Juiz de Fora (UFJF) em dois dias diferentes. A primeira gravação foi feita no dia 28 de julho de 2025, com quatro pessoas sinalizando. Já a segunda gravação foi realizada no dia 30 de julho de 2025, com cinco pessoas sinalizando. Dos sinalizadores presentes, oito deles são ouvintes e um é surdo. Cada pessoa gravou 5 repetições de todos os 20 sinais, totalizando 45 repetições por sinal. Além disso, é importante ressaltar que todos os participantes possuem conhecimento em Libras.

Todos os vídeos foram gravados no mesmo local, em uma sala preparada para a captura de vídeos de sinais na faculdade de Letras, mantendo a mesma configuração de iluminação, plano de fundo e câmera em toda a coleta. Pode haver pequenas divergências quanto à distância dos elementos dos vídeos gravados no primeiro dia, comparados aos vídeos do segundo dia, porém não houve impacto significativo no resultado obtido.

O plano de fundo foi um tecido *chromakey* verde, que cobre toda a área de gravação dos vídeos. O uso do *chromakey* é útil para possibilitar edições do plano de fundo dos vídeos, porém para este presente trabalho, nenhuma edição do plano de fundo foi realizada. Para a iluminação, contou-se com quatro fontes de luz: uma lâmpada de teto localizada no centro da sala e três holofotes posicionados sobre um tripé, localizados à esquerda, à direita e à frente do sinalizador e direcionados a ele durante todas as

[7] https://libras.cin.ufpe.br/sign/460
[8] https://libras.cin.ufpe.br/sign/1080
[9] https://libras.cin.ufpe.br/sign/848
[10] https://libras.cin.ufpe.br/sign/354
[11] https://libras.cin.ufpe.br/sign/378

capturas. O sinalizador se posicionava a aproximadamente 50 centímetros à frente do *chromakey* e mantinha-se no mesmo local em todos os vídeos. A área gravada dos vídeos sempre capturava o sinalizador a partir de alguns centímetros acima da cabeça até alguns centímetros abaixo da região da cintura e sempre posicionado ao centro horizontalmente.

A câmera utilizada foi Canon SL3 e os vídeos foram gravados na resolução 1920 × 1080 (*Full HD*), a 60 quadros por segundo. Foi acordado que, para cada sinal, uma pessoa informaria ao sinalizador qual seria o próximo sinal e o sinalizaria para padronizar a execução. Em seguida, a gravação era iniciada e o sinalizador repetia o mesmo sinal 5 vezes, sempre iniciando e terminando o sinal com a posição de repouso das mãos. Cada vídeo ficou com uma duração média de 15 segundos antes de ser processado.

4.2 Preparação dos vídeos

Após todos os 180 vídeos serem capturados (20 sinais × 9 sinalizadores = 180 vídeos), ainda era necessário cortá-los para isolar uma repetição do sinal por vídeo, uma vez que cada vídeo contemplava 5 repetições de um sinal. Os vídeos foram cortados manualmente, por conta da dificuldade de estimar com precisão em quais instantes cada vídeo precisaria ser cortado de forma automática, uma vez que a duração da execução do sinal varia para cada sinalizador. Dessa forma, utilizou-se o software "Fotos Microsoft"[12] para cortar cada vídeo em cinco novos vídeos. O motivo de utilizar esse software foi que os vídeos cortados resultantes mantinham a resolução e a taxa de quadros iguais às dos vídeos originais. Testou-se também a utilização do software "Microsoft Clipchamp"[13], porém este reduzia a resolução dos vídeos resultantes na versão gratuita.

A posição de repouso das mãos foi preservada no início e no final de cada vídeo resultante para manter a padronização. Além disso, nenhum quadro de um vídeo se repete em qualquer outro vídeo, pois eles foram cortados um a um, mantendo o quadro inicial de um vídeo sempre posterior ao quadro final do vídeo anterior. Ao final do processamento, consolidou-se um conjunto de 900 vídeos (20 sinais × 9 sinalizadores × 5 repetições por sinal = 900 vídeos), com tempo médio de duração de 2 segundos por vídeo.

[12]https://www.microsoft.com/pt-br/windows/tips/photos-app
[13]https://clipchamp.com/pt-br/

Cada vídeo foi salvo com o nome “XX-sinalizador-YY-execucao” (Exemplo: 00-sinalizador-00-execucao), de modo que o primeiro número representa o identificador do sinalizador, enquanto o segundo representa o identificador da execução do sinal em questão, variando de 00 a 04. Agruparam-se os vídeos por sinais em pastas, logo cada pasta contém todas as execuções do sinal, de todos os sinalizadores. As pastas foram padronizadas com o nome “XX-nomedosinal” (Exemplo: 00-professor), de forma que o número representa o identificador do sinal, variando de 00 a 19. Posteriormente, o identificador do sinal no nome das pastas é utilizado como rótulo das classes dos sinais fornecidos ao modelo.

5 Metodologia

Para a execução da tarefa de reconhecimento de sinais, adotou-se a estratégia de extrair os *keypoints* dos quadros dos vídeos, calcular características personalizadas a partir deles e, em seguida, utilizá-las para treinar uma rede neural com blocos LSTM. Utilizou-se como ponto de partida a metodologia de melhor resultado proposta por Sarmento (2023). Todo o processo foi reproduzido e, em seguida, adaptado para melhorar ainda mais o desempenho do modelo no reconhecimento dos sinais escolhidos. A Seção 5.1 aborda como o pré-processamento dos dados foi realizado para prepará-los para serem usados como entrada do modelo. A Seção 5.2 apresenta informações a respeito da arquitetura do modelo e lista todas as configurações iniciais dos parâmetros utilizados. Já na Seção 5.3, discute-se o treinamento do modelo.

5.1 Pré-processamento dos dados

Antes de treinar o modelo, é necessário preparar os dados da maneira como a rede os espera. A proposta de Sarmento (2023) que obteve a melhor acurácia consiste em extrair características personalizadas a partir dos *keypoints* obtidos. Uma amostragem de quadros também é realizada para selecionar aqueles com maior potencial de fornecer informações relevantes ao modelo, visando melhorar o reconhecimento. Além disso, os *keypoints* são normalizados, e o conjunto de dados é aumentado de forma a reduzir o *overfitting*. Todas essas técnicas foram aplicadas nesta etapa inicial de pré-processamento dos dados dos vídeos. O código foi inteiramente desenvolvido na linguagem Python, com o auxílio de algumas bibliotecas e ferramentas. Um fluxo de execução é apresentado na Figura 5.1.

Inicialmente, todos os arquivos com a extensão ".mp4" são buscados nas pastas de vídeos, seguindo a ordem dos sinais da classe "00" até a classe "19". Os vídeos são então processados na mesma ordem em que estão listados nas pastas. A partir do nome da pasta e do nome do arquivo do vídeo, obtém-se o identificador da classe e do sinalizador, respectivamente.

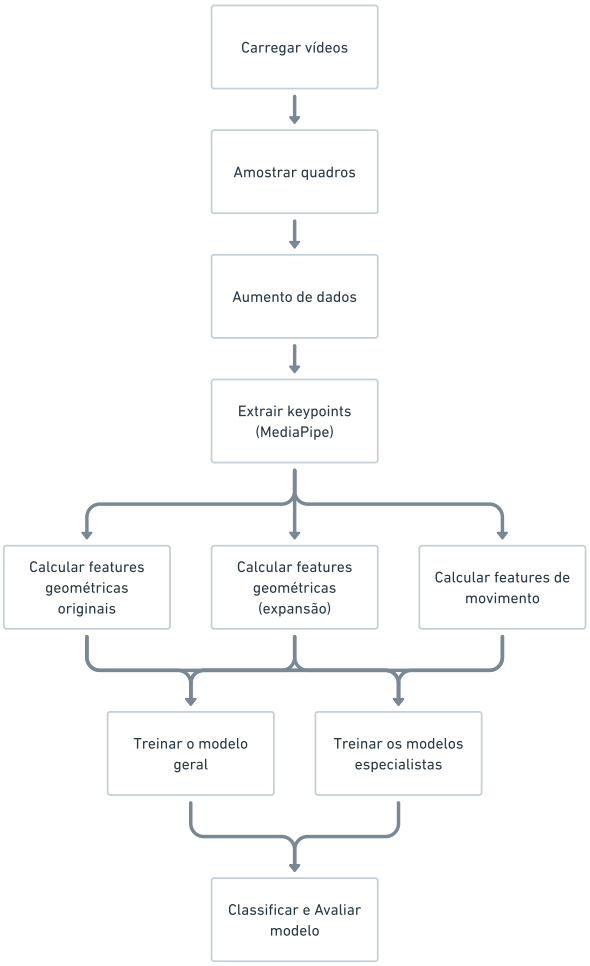


Figura 5.1: Fluxo de execução.

A primeira etapa do processamento dos vídeos é a amostragem de quadros. Utilizou-se a biblioteca OpenCV[14] para fazer a leitura e o processamento de cada vídeo. É necessário que o vídeo possua pelo menos a quantidade de quadros a ser amostrada para que ele possa ser utilizado no modelo. Uma técnica de *padding* poderia ser aplicada nos casos de vídeos com menos quadros. Porém, dado que nenhum vídeo do conjunto de dados construído apresenta essa característica, optou-se por não aplicá-la.

Seguindo a metodologia base de Sarmento (2023), optou-se por amostrar 15 quadros de cada vídeo. Existem diferentes formas de escolher os quadros de um vídeo. Uma possível abordagem seria amostrar segundo uma distribuição uniforme, porém isso não garante que os quadros mais relevantes serão selecionados. Utilizar a distribuição uniforme

[14]https://opencv.org/

para amostrar os quadros produziria a mesma chance de selecionar qualquer quadro do vídeo. No entanto, os quadros do início e do final do vídeo contêm trechos da posição de repouso das mãos, que não correspondem especificamente ao sinal em questão. Dessa forma, é interessante aplicar uma distribuição que privilegie os quadros mais relevantes, que se encontram no meio do vídeo. Tendo isso em vista, a distribuição escolhida para a amostragem foi a distribuição normal, que tende a selecionar mais quadros do meio do vídeo e menos quadros iniciais e finais.

De modo a melhorar a generalização do modelo e reduzir o *overfitting*, aplicou-se a técnica de aumento de dados. Ela consiste em criar cópias modificadas dos dados do conjunto, visando aumentar a diversidade sem a necessidade de coletar novos dados reais. Para isso, algumas transformações foram combinadas aleatoriamente, de forma que os dados aumentados fossem diferentes. O conjunto de dados foi aumentado em 20 vezes, variando aleatoriamente os parâmetros das transformações em cada vídeo. É importante ressaltar que as transformações são aplicadas diretamente nos quadros selecionados e, a cada aumento de um vídeo, novos quadros são amostrados pela distribuição normal. Com isso, aumenta-se ainda mais a diversidade, pois diferentes quadros podem estar presentes no conjunto de dados aumentado. Além disso, o mesmo conjunto de transformações aplicado a um quadro deve ser aplicado a todos os outros da amostra de aumento em questão, logo os mesmos parâmetros são mantidos para cada sequência de quadros.

As seguintes transformações foram aplicadas:

- **Espelhamento horizontal**: Escolhe-se aleatoriamente se o vídeo será espelhado horizontalmente ou não. Essa transformação é especialmente útil para simular a execução dos sinais utilizando a mão dominante inversa;
- **Rotação**: Rotaciona-se de -5 a 5 graus, simulando diferentes orientações;
- **Translação**: Translada-se horizontalmente e/ou verticalmente de -20 a 20 *pixels* (até 20 *pixels* para a esquerda ou para a direita e/ou até 20 *pixels* para cima ou para baixo);
- **Corte centralizado**: Corta-se até 10% das bordas dos quadros, mantendo-os sempre centralizados;

- **Contraste**: Alteração no contraste de 70% até 130% do contraste original (100%);
- **Brilho**: Alteração do brilho de -20 até 20 níveis de intensidade de *pixel*.

Cada quadro foi redimensionado para 640 *pixels* de largura por 480 *pixels* de altura, de forma a prepará-lo para extrair as características. Em seguida, os pontos de referência (*keypoints*) são estimados utilizando a ferramenta "MediaPipe Holistic"[15] do *Google*. Ela é capaz de extrair *keypoints* não só das mãos, mas também da pose e da face simultaneamente.

Os parâmetros utilizados no MediaPipe foram:

- **Modo de imagem estática**: Determina a natureza da entrada de dados. Utilizou-se a opção desativada, na qual o sistema trata as imagens como um fluxo de vídeo contínuo: ele detecta a pessoa nos primeiros quadros e, em seguida, passa a rastrear apenas os *keypoints* nos quadros subsequentes;
- **Complexidade do modelo**: Define o nível de complexidade do modelo de *keypoints*. Utilizou-se o valor 2, que determina a complexidade mais alta, resultando em maior precisão, porém com aumento na latência de inferência;
- **Suavização dos *keypoints***: Utilizou-se o modo ativo, que aplica um filtro aos *keypoints* ao longo das imagens para reduzir tremores;
- **Habilitação de segmentação**: Desativada, para não gerar uma máscara de segmentação da pessoa;
- **Suavização da segmentação**: Desativada, uma vez que a segmentação também foi desabilitada;
- **Refinamento de marcos faciais**: Desativado, optando por não realizar refinamentos adicionais nos pontos ao redor dos olhos e lábios;
- **Confiança mínima de detecção**: O valor mínimo de confiança exigido do modelo para que a detecção seja considerada bem-sucedida. Utilizou-se uma confiança de 90%;

[15] https://pypi.org/project/mediapipe/

- **Confiança mínima de rastreamento**: O valor mínimo de confiança exigido do modelo de rastreamento para que a pose seja considerada rastreada com sucesso. Utilizou-se uma confiança de 90%.

Em um único quadro, são extraídos pontos na forma de coordenadas tridimensionais da pose, da face e das mãos. Ao todo, obtém-se 543 coordenadas por quadro (33 da pose + 468 da face + 21 da mão esquerda + 21 da mão direita = 543 coordenadas). Para acelerar a extração, até 4 processos são criados para extrair os pontos, de modo que cada processo fica responsável por processar um quadro independente. Utilizou-se o paralelismo em nível de processo, visando evitar o Bloqueio Global do Interpretador (*Global Interpreter Lock* - GIL) do Python quando o paralelismo ocorre em nível de *thread*. Os pontos das mãos detectados pelo MediaPipe são exibidos na Figura 5.2.

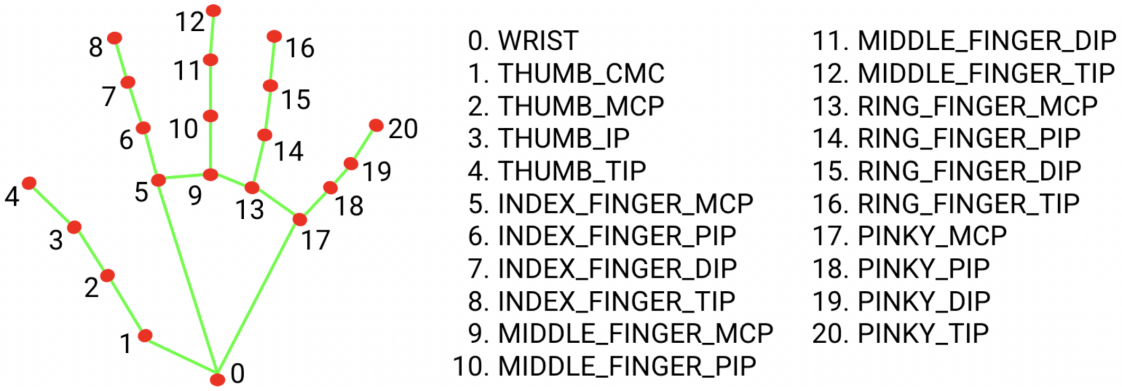


Figura 5.2: *Keypoints* da mão, obtidos pelo MediaPipe.

5.1.1 Características Geométricas

Após todos os pontos terem sido extraídos dos quadros selecionados de um vídeo, inicia-se a etapa de extração de características personalizadas a partir desses dados. Foram calculados os ângulos internos de diferentes pontos de articulação de cada mão e diferentes distâncias entre pontos estratégicos da pose. Os ângulos calculados são exibidos na Figura 5.3.

A partir de um ponto de articulação, obtém-se dois vetores $\mathbf{v_1}$ e $\mathbf{v_2}$ com origem na articulação e extremidades em dois pontos vizinhos na mão. Com isso, é possível calcular o ângulo entre os vetores através da normalização, gerando os vetores unitários $\hat{\mathbf{v}}_\mathbf{1} = \frac{\mathbf{v_1}}{||\mathbf{v_1}||}$

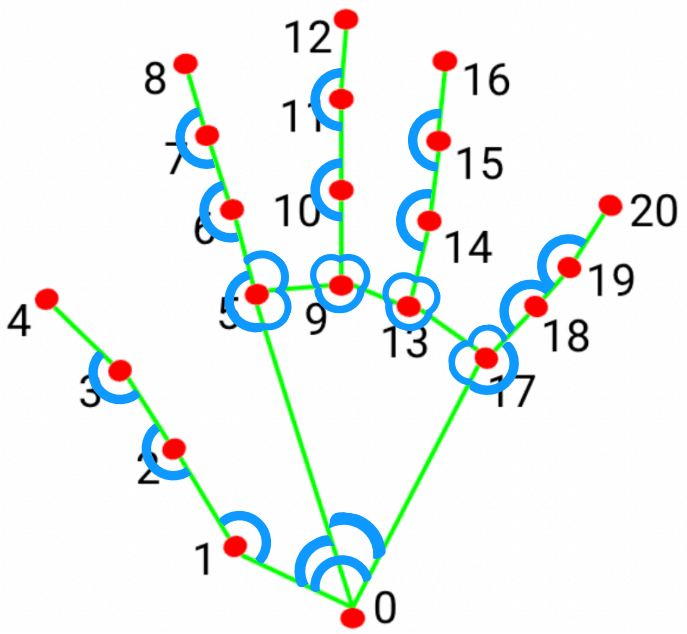


Figura 5.3: Ângulos da mão calculados a partir dos pontos.

e $\hat{\mathbf{v}}_2 = \frac{\mathbf{v_2}}{||\mathbf{v_2}||}$. Em seguida, calcula-se o produto escalar entre esses vetores normalizados. O ângulo θ é então obtido através da relação $\theta = \arccos(\hat{\mathbf{v}}_1 \cdot \hat{\mathbf{v}}_2)$.

A operação completa de cálculo do ângulo entre dois vetores pode ser expressa pela equação:

$$\theta = \arccos\left(\frac{\mathbf{v_1}}{||\mathbf{v_1}||} \cdot \frac{\mathbf{v_2}}{||\mathbf{v_2}||}\right) \tag{5.1}$$

.

Antes de calcular as distâncias entre os *keypoints* da pose, é necessário normalizá-los, visando padronizar a posição e o tamanho do corpo na imagem. Dessa forma, todos os pontos da pose são ajustados para que o centro dos ombros se torne a origem do sistema de coordenadas. Além disso, todos os pontos também são divididos pelo tamanho do tronco. Isso garante que o modelo entenda os sinais da mesma forma, independentemente da distância do sinalizador em relação à câmera. Todos os pontos detectados pelo MediaPipe são exibidos na Figura 5.4.

Seja P_i o vetor de coordenadas (x, y, z) do i-ésimo *keypoint*. O processo de normalização para obter o ponto normalizado $\hat{P}_i$ é definido pelas seguintes etapas:

1. **Cálculo do centro de referência:** O ponto central (C) é definido como o ponto médio entre os ombros (pontos 11 e 12):

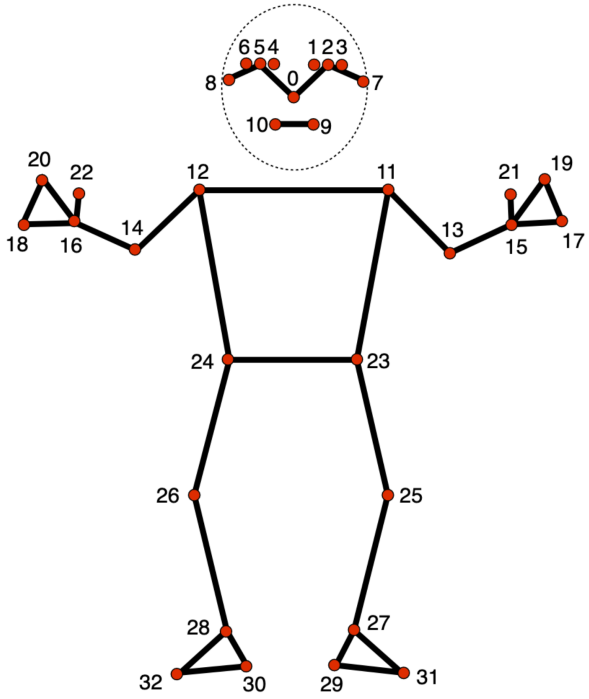


Figura 5.4: *Keypoints* da pose, obtidos pelo MediaPipe.

$$C = \frac{P_{11} + P_{12}}{2} \tag{5.2}$$

.

2. **Cálculo do Fator de Escala (Tamanho do Tronco):** O fator de escala (s) é a distância entre o centro dos ombros e o centro dos quadris (pontos 23 e 24):

$$s = \left\| C - \frac{P_{23} + P_{24}}{2} \right\| \tag{5.3}$$

.

3. **Normalização:** Cada ponto P_i é transladado e reescalado:

$$\hat{P}_i = \frac{P_i - C}{s} \tag{5.4}$$

.

Após a normalização ter sido aplicada, calcula-se o segundo grupo de características personalizadas. Trata-se das distâncias entre pares de pontos específicos da

pose. Foram determinados 36 pares de pontos para calcular distâncias e compor os dados de entrada do modelo, juntamente com os ângulos das mãos. Os 36 pares são listados na Tabela 5.1.

Tabela 5.1: Pares de pontos da pose para calcular distâncias.

Pontos	**Posições na pose**
P_0, P_{15}	Nariz e pulso esquerdo
P_0, P_{16}	Nariz e pulso direito
P_{12}, P_{16}	Ombro direito e pulso direito
P_{12}, P_{15}	Ombro direito e pulso esquerdo
P_{11}, P_{16}	Ombro esquerdo e pulso direito
P_{11}, P_{15}	Ombro esquerdo e pulso esquerdo
P_{12}, P_{14}	Ombro direito e cotovelo direito
P_{12}, P_{13}	Ombro direito e cotovelo esquerdo
P_{11}, P_{14}	Ombro esquerdo e cotovelo direito
P_{11}, P_{13}	Ombro esquerdo e cotovelo esquerdo
P_{16}, P_{18}	Pulso direito e dedo mínimo direito
P_{16}, P_{17}	Pulso direito e dedo mínimo esquerdo
P_{15}, P_{17}	Pulso esquerdo e dedo mínimo esquerdo
P_{15}, P_{18}	Pulso esquerdo e dedo mínimo direito
P_{16}, P_{20}	Pulso direito e dedo indicador direito
P_{16}, P_{19}	Pulso direito e dedo indicador esquerdo
P_{15}, P_{20}	Pulso esquerdo e dedo indicador direito
P_{15}, P_{19}	Pulso esquerdo e dedo indicador esquerdo
P_{18}, P_{20}	Dedo mínimo direito e dedo indicador direito
P_{18}, P_{19}	Dedo mínimo direito e dedo indicador esquerdo
P_{17}, P_{20}	Dedo mínimo esquerdo e dedo indicador direito
P_{17}, P_{19}	Dedo mínimo esquerdo e dedo indicador esquerdo
P_{18}, P_{22}	Dedo mínimo direito e dedo polegar direito
P_{18}, P_{21}	Dedo mínimo direito e dedo polegar esquerdo
P_{17}, P_{21}	Dedo mínimo esquerdo e dedo polegar esquerdo
P_{17}, P_{22}	Dedo mínimo esquerdo e dedo polegar direito
P_{20}, P_{22}	Dedo indicador direito e dedo polegar direito
P_{20}, P_{21}	Dedo indicador direito e dedo polegar esquerdo
P_{19}, P_{21}	Dedo indicador esquerdo e dedo polegar esquerdo
P_{19}, P_{22}	Dedo indicador esquerdo e dedo polegar direito
P_{21}, P_{22}	Dedo polegar esquerdo e dedo polegar direito
P_{19}, P_{20}	Dedo indicador esquerdo e dedo indicador direito
P_{17}, P_{18}	Dedo mínimo esquerdo e dedo mínimo direito
P_{15}, P_{16}	Pulso esquerdo e pulso direito
P_{13}, P_{14}	Cotovelo esquerdo e cotovelo direito
$\frac{P_{11}+P_{12}}{2}$, $\frac{P_{23}+P_{24}}{2}$	Centro dos ombros e centro do quadril

Fonte: Elaborada pelo autor (2026).

Assim como nos ângulos das mãos, as distâncias também são calculadas para cada quadro selecionado do vídeo. Ao todo, são calculadas 88 características (26 ângulos da mão esquerda + 26 ângulos da mão direita + 36 distâncias da pose = 88 características). Elas são concatenadas em um único vetor que é utilizado como entrada do modelo.

As características extraídas dos vídeos são serializadas e salvas em diferentes arquivos, para que um vídeo que foi pré-processado uma vez não precise passar pelo mesmo procedimento. No início da etapa de pré-processamento, verifica-se se existem características extraídas para o vídeo em questão, com o objetivo de pular essa etapa e evitar uma computação desnecessária.

5.1.2 Expansão do Conjunto de Características Geométricas

De modo a tentar extrair ainda mais desempenho do modelo, algumas variações de características foram planejadas e calculadas. A primeira variação corresponde à adição de mais distâncias, além das descritas na Tabela 5.1, adição de vetor normal das palmas das mãos e à remoção de algumas distâncias entre dedos retornados pelo modelo da pose, substituindo-os pelos *keypoints* do modelo das mãos que possuem uma precisão melhor. As variações de características geométricas foram:

- **Distâncias internas das mãos (18 características)**: Foram calculadas 9 distâncias para cada mão, normalizadas pela escala da própria mão. Dentre elas, 4 distâncias são entre a ponta do polegar e a ponta dos outros dedos, enquanto 5 distâncias são do pulso até a ponta de todos os dedos;

- **Normais das Palmas (6 características)**: Foi calculado o vetor normal (x, y, z) da palma de cada mão através do produto vetorial entre os vetores do pulso ao dedo indicador e do pulso ao dedo mínimo;

- **Distâncias entre as mãos e a face (24 características)**: Foram calculadas distâncias de 2 *keypoints* de cada mão até 6 *keypoints* da face. Essas distâncias foram normalizadas pelo tamanho do tronco;

- **Remoção de distâncias das mãos pelo modelo da pose (10 características)**: Os *keypoints* das mãos apresentam precisão superior quando obtidos pelo modelo das mãos, comparado ao modelo da pose. Logo, optou-se por priorizar a utilização do modelo das mãos para calcular as distâncias internas. Assim, 10 distâncias anteriormente calculadas entre *keypoints* da pose foram removidas e substituídas por distâncias entre *keypoints* do modelo das mãos. Estes são os pares de pontos cujas distâncias foram removidas: (P_{16}, P_{18}), (P_{15}, P_{17}), (P_{16}, P_{20}), (P_{15}, P_{19}), (P_{18}, P_{20}), (P_{17}, P_{19}), (P_{18}, P_{22}), (P_{17}, P_{21}), (P_{20}, P_{22}) e (P_{19}, P_{21}).

Assim, totalizam-se 38 novas características (18 + 6 + 24 - 10 = 38), visto que 10 características foram apenas substituídas pelas mesmas distâncias, utilizando os *keypoints* das mãos em vez da pose.

5.1.3 Características de Movimento

Para capturar a dinâmica temporal dos sinais, um conjunto de características baseadas na velocidade e aceleração dos *keypoints* foi implementado. Para isso, considere $P_{t,i} \in \mathbb{R}^3$ o vetor de posição (x, y, z) do i-ésimo *keypoint* no quadro t, onde $t \in [1, T]$ e T é o número total de quadros.

Cálculo de Velocidade e Aceleração

A velocidade instantânea $V_{t,i}$ é aproximada pela diferença entre *keypoints* em quadros consecutivos:

$$V_{t,i} = P_{t,i} - P_{t-1,i}, \quad \text{para } t > 1. \tag{5.5}$$

A aceleração instantânea $A_{t,i}$ é calculada como a taxa de variação da velocidade:

$$A_{t,i} = V_{t,i} - V_{t-1,i}, \quad \text{para } t > 1. \tag{5.6}$$

Para $t = 1$, assume-se $V_{1,i} = 0$ e $A_{1,i} = 0$.

Características por *keypoints*

Para cada *keypoint* selecionado, quatro características são extraídas a cada quadro:

1. **Magnitude da Velocidade Normalizada**:

$$\hat{v}_{t,i} = \frac{\|V_{t,i}\|}{\max_t(\|V_{t,i}\|) + \epsilon} \tag{5.7}$$

2. **Magnitude da Aceleração Normalizada**:

$$\hat{a}_{t,i} = \frac{\|A_{t,i}\|}{\max_t(\|A_{t,i}\|) + \epsilon} \tag{5.8}$$

3. **Direção do Movimento (X e Y)**:

   Componentes normalizados do vetor de velocidade no $\mathbb{R}^2$:

$$d_{x,t,i} = \frac{V_{x,t,i}}{\|V_{xy,t,i}\| + \epsilon}, \quad d_{y,t,i} = \frac{V_{y,t,i}}{\|V_{xy,t,i}\| + \epsilon}, \tag{5.9}$$

onde:

- $\|V_{xy}\|$ é a norma Euclidiana apenas nos componentes x e y.
- $V_{x,t,i}$ e $V_{y,t,i}$ são os componentes da velocidade instantânea do i-ésimo *keypoint* no quadro t ao longo dos eixos x e y, respectivamente.
- $\epsilon = 10^{-6}$ é utilizado nos denominadores para evitar divisão por 0.

***Keypoints* Selecionados**

As características acima são calculadas para um subconjunto de 15 *keypoints* selecionados:

- **Mãos (6 por mão)**: Pulso, e as pontas dos 5 dedos.
- **Pose (3 pontos)**: Nariz, Pulso Esquerdo e Pulso Direito.

Isso resulta em $15 \times 4 = 60$ características locais por quadro. A inclusão duplicada dos pulsos, pelo modelo de mãos e modelo de pose foi feita de maneira redundante intencionalmente para cobrir casos em que os pontos são detectados por somente um dos modelos.

Características de Movimento Globais

Além das características locais, 5 descritores globais são computados para resumir o comportamento do movimento ao longo de toda a sequência de vídeo:

1. **Quadro de Pico de Velocidade (ρ)**: Esta característica identifica o momento do sinal em que ocorre a maior intensidade de movimento:

$$\rho = \frac{\arg\max(v_{comb}(t))}{T}, \tag{5.10}$$

onde:

- $v_{comb}(t) = \|V_{t,Esq}\| + \|V_{t,Dir}\|$ é a velocidade combinada das duas mãos no instante t.

- T é o número total de quadros do vídeo.
- $\arg\max$ retorna o índice t onde o valor é máximo.

2. **Variância da Velocidade Normalizada (σ^2)**: Esta característica mede a consistência ou variabilidade da velocidade ao longo da execução. Sinais com ritmo constante possuem variância baixa, enquanto sinais com pausas possuem variância alta:

$$\sigma^2 = \frac{\frac{1}{T}\sum_{t=1}^{T}(v_{comb}(t) - \bar{v}_{comb})^2}{\max(v_{comb}) + \epsilon}, \tag{5.11}$$

onde:

- $\bar{v}_{comb}$ é a velocidade média do sinal.
- O denominador $\max(v_{comb})$ normaliza o valor.

3. **Coordenação do Movimento**: Esta característica avalia o grau de coordenação entre as mãos esquerda e direita. É útil para distinguir sinais simétricos de sinais em que as mãos se movem de forma independente ou alternada. Se ambas as mãos estão em movimento, calcula-se o coeficiente de correlação de Pearson entre a magnitude das velocidades da mão esquerda e da mão direita para determinar a simetria. O resultado pode variar de -1 a 1, de modo que:

- Coeficiente = 1: As mãos possuem um movimento coordenado;
- Coeficiente = 0: As mãos possuem movimentos independentes;
- Coeficiente = -1: As mãos possuem movimentos alternados.

4. **Picos de Aceleração e Suavidade (K)**: Esta característica mede a complexidade do movimento contando quantas vezes a aceleração muda de sentido (sinal). Movimentos fluidos possuem poucas mudanças, enquanto movimentos complexos ou trêmulos possuem muitas.

$$K = \frac{1}{T}\sum_{t=2}^{T} \mathbb{1}[\text{sgn}(a_t) \neq \text{sgn}(a_{t-1})], \tag{5.12}$$

onde:

- a_t é a aceleração da velocidade combinada no instante t.
- $\text{sgn}(x)$ é a função sinal (retorna $+1$ ou -1).
- $\mathbb{1}$ retorna 1 se houver mudança de sinal, ou 0 caso contrário.

5. **Indicador de Mão Dominante (D)**: Determina qual mão é a dominante no sinal, comparando a quantidade total de movimento realizada por cada uma:

$$D = \frac{\sum_{t=1}^{T}(\|V_{t,Dir}\| - \|V_{t,Esq}\|)}{\sum_{t=1}^{T}(\|V_{t,Dir}\| + \|V_{t,Esq}\|)}, \tag{5.13}$$

onde valores próximos a $+1$ indicam dominância da mão direita, próximos a -1 indicam dominância da mão esquerda, e valores próximos a 0 indicam uso balanceado das duas mãos (ou nenhum movimento).

6. **Razão de Duração do Movimento (R)**: Calcula a porcentagem do vídeo em que houve movimento significativo, filtrando os momentos estáticos, ou com pouco movimento:

$$R = \frac{1}{T}\sum_{t=1}^{T}\mathbb{1}[v_{comb}(t) > 0,1 \cdot \max(v_{comb})], \tag{5.14}$$

onde o limiar de $0,1$ (10%) é aplicado para filtrar movimentos com baixa velocidade em relação ao pico de velocidade do vídeo.

Ao todo, 66 características de movimento foram calculadas (60 locais + 6 globais).

5.2 Arquitetura do Modelo

O modelo responsável pela tarefa de reconhecer os sinais da Libras, baseado na arquitetura de Sarmento (2023), consiste em uma rede neural com uma camada LSTM. Além dessa camada, outras também são adicionadas para cumprir tarefas importantes na composição geral do modelo.

A camada de entrada define o formato dos dados que entram na rede, preparando-a para receber uma sequência de quadros. Já a camada de normalização padroniza os

dados de entrada para que tenham média zero e variância unitária. Isso acelera a convergência e evita problemas com escalas de valores diferentes. A camada LSTM é a principal do modelo e é eficaz no processamento temporal. Para a tarefa de reconhecimento de sinais, a LSTM se destaca por ser capaz de aprender dependências de longo prazo, o que é essencial, pois o movimento dos sinais contribui profundamente para o significado. Uma função de regularização L1 também é aplicada ao *kernel*. Em seguida, uma camada de *dropout* desativa aleatoriamente uma porcentagem de neurônios durante o treinamento, o que contribui para reduzir o *overfitting*. Por fim, uma camada de saída com ativação *softmax* gera as probabilidades finais de cada classe a partir das características aprendidas.

Além disso, foi utilizado o otimizador Adam com decaimento de peso (*Adaptive Moment Estimation with Weight Decay* - AdamW) (LOSHCHILOV; HUTTER, 2017) para proporcionar uma convergência mais rápida e estável do modelo. O AdamW decide como ajustar os pesos da rede neural para que ela cometa menos erros durante o aprendizado. Já a técnica de decaimento de peso aplica uma penalização que tende a manter os pesos do modelo com valores pequenos. A utilização do otimizador resulta em uma melhor generalização e convergência do modelo no treinamento. A arquitetura completa do modelo é descrita na Tabela 5.2.

Tabela 5.2: Arquitetura do modelo para reconhecimento de sinais.

Camada	**Dimensão de saída**
Entrada	(Tamanho do lote, Nº de quadros, Nº de características)
Normalização	(Tamanho do lote, Nº de quadros, Nº de características)
LSTM	(Tamanho do lote, Nº de neurônios)
Dropout	(Tamanho do lote, Nº de neurônios)
Classificação (Densa)	(Tamanho do lote, Nº de classes)

Fonte: Elaborada pelo autor (2026).

Os parâmetros utilizados foram:

- **Número de características:** 88;
- **Número de classes:** 20;
- **Fator de regularização L1:** 0,001;

- **Taxa de *Dropout*:** 0,4;
- **Tamanho do vídeo de entrada:** 640 × 480;
- **Taxa de aprendizado:** 0,0001;
- **Taxa de decaimento de peso:** 0,005;
- **Função de perda:** Entropia cruzada categórica esparsa.

5.3 Treinamento do Modelo

Como as etapas de pré-processamento foram realizadas previamente e as características extraídas salvas, a etapa de treinamento foi consideravelmente mais leve, uma vez que os dados não precisaram ser re-processados a cada novo treinamento.

A base foi dividida de modo que cada sinalizador estivesse presente em apenas um grupo (treinamento, validação ou teste), para não enviesar o modelo. A estratégia utilizada foi a validação cruzada *9-fold*, sempre utilizando os vídeos de 7 sinalizadores para treinamento, 1 sinalizador para validação e 1 sinalizador para teste. A cada mudança de *fold*, um novo sinalizador era escolhido para compor o conjunto de validação e outro sinalizador era escolhido para o conjunto de teste.

Visando evitar que o modelo decorasse os dados de treinamento, foi utilizada a técnica de parada antecipada (*early stopping*), com paciência de 20 épocas. Essa técnica de regularização visa evitar o *overfitting* monitorando o desempenho do modelo no conjunto de validação e parando o treinamento quando a performance para de melhorar. Ao final do treinamento, os pesos da época de melhor acurácia de validação foram selecionados. Escolheu-se realizar o treinamento por 200 épocas.

O treinamento foi realizado em um computador com processador Intel(R) Core(TM) i5-12450H (2.00 GHz), placa de vídeo NVIDIA Geforce RTX 2050 4GB e 16GB de memória RAM de 4800 MT/s.

Realizar o treinamento sem a utilização da placa de vídeo é possível, porém prejudica consideravelmente a performance e aumenta o tempo gasto em cada teste. A execução desta etapa utilizando somente a CPU, no sistema operacional *Windows*, levava

em torno de 6 horas para ser finalizada. Dessa forma, preparou-se o ambiente para executar o treinamento com a GPU, garantindo um tempo de execução viável de testes. O meio mais acessível encontrado para isso foi utilizar o *Windows Subsystem for Linux* (WSL) e instalar o conjunto de ferramentas CUDA da NVIDIA. Tendo preparado o ambiente de execução, cada treinamento realizado com a GPU levava em torno de 45 minutos, cerca de 8 vezes mais rápido que na CPU.

Após o treinamento de cada *fold*, o modelo com os melhores pesos previu os dados de teste e foi avaliado quanto à acurácia. Ao final do treinamento de todos os *folds*, os resultados foram combinados de modo a obter a acurácia geral da base de dados completa. Além disso, a matriz de confusão foi gerada para uma melhor visualização do comportamento do modelo para cada classe, juntamente com gráficos de evolução da acurácia e da função de perda.

6 Experimentos e Resultados

Neste capítulo são apresentados e discutidos os resultados dos experimentos realizados para o reconhecimento de sinais da Libras. O objetivo é avaliar o impacto de diferentes configurações de hiperparâmetros, conjuntos de características e modificações na arquitetura do modelo, visando superar o desempenho do modelo base utilizado como referência. A métrica utilizada para a comparação dos modelos foi a acurácia geral do conjunto de dados completo.

6.1 Modelo Base

Para garantir a comparabilidade dos resultados, todos os experimentos foram realizados utilizando a mesma base de dados de sinais dinâmicos. Como ponto de partida, definiu-se como modelo base o apresentado no trabalho de Sarmento (2023), que foi descrito na Seção 5.2. Ele utiliza apenas os atributos geométricos mencionados na Seção 5.1.1, uma seleção normal de 15 quadros, uma camada LSTM com 512 unidades e hiperparâmetros descritos na Seção 5.2. Utilizando as mesmas configurações, atingiu-se uma acurácia de 84,00% para o conjunto dos 20 sinais estudados neste trabalho.

A Figura 6.1 mostra como o modelo base se desempenhou em cada classe, considerando o treinamento com todo o conjunto de dados. A matriz de confusão foi normalizada pelas linhas (rótulos verdadeiros). Isso significa que cada valor na diagonal principal representa a proporção de previsões corretas para aquela classe específica em relação ao total de amostras reais daquela classe, também conhecido como *recall* da classe.

Observa-se que algumas classes tiveram um alto *recall*, como é o caso de: “Professor”, “Prova”, “Computador”, “Internet”, “Tecnologia”, “Algoritmo”, “Código” e “Compilador”. Entretanto, as classes “Perguntar” e “Pesquisar” apresentaram alta confusão entre si, assim como as classes “Inteligência Artificial” e “Software”. Isso se deve ao fato de que esses sinais são ambíguos e possuem elementos muito semelhantes na sua construção. Já as demais classes tiveram *recall* razoável, com falsos negativos mais distribuídos.

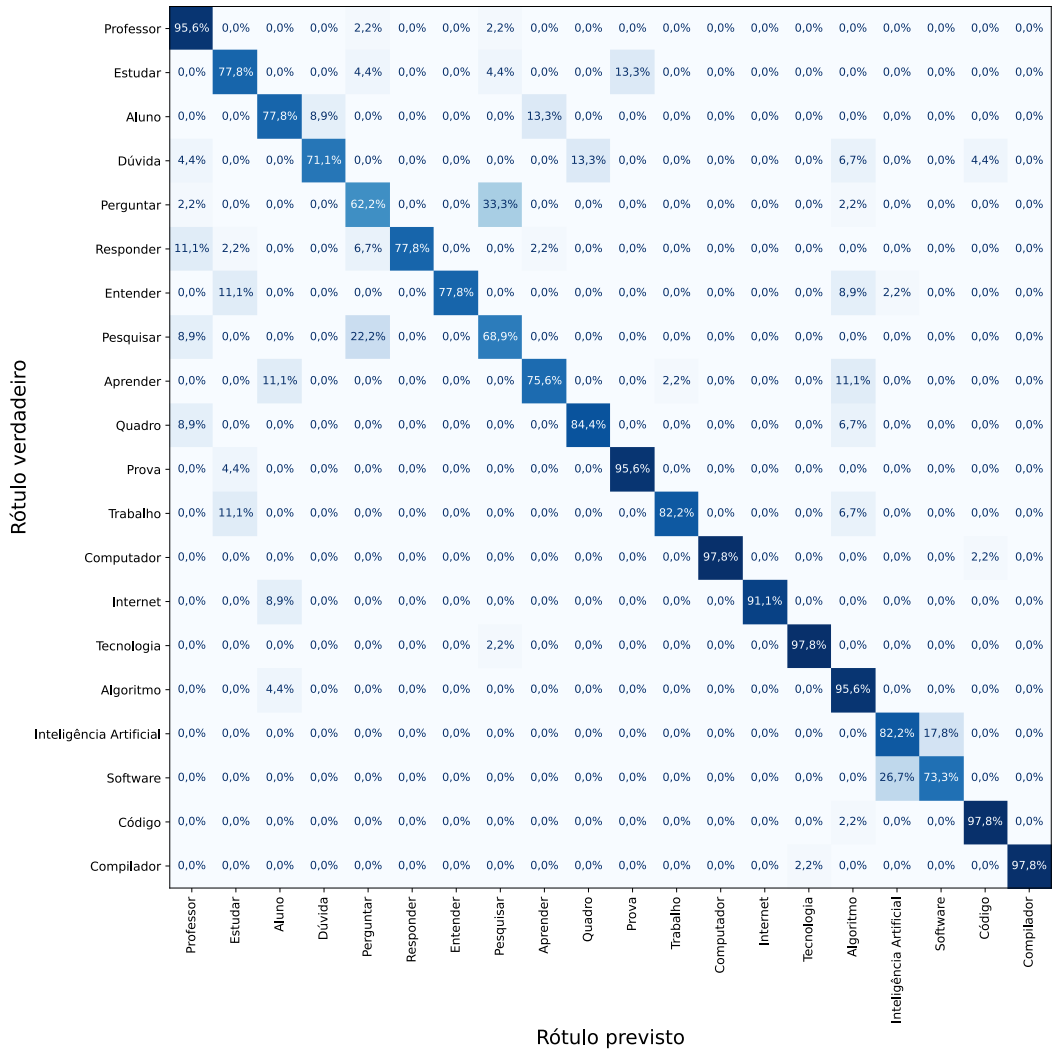


Figura 6.1: Matriz de confusão do modelo base.

6.2 Análise do Número de Quadros por Vídeo

O primeiro conjunto de experimentos buscou avaliar o impacto do número de quadros selecionados no desempenho do modelo. A hipótese é que um maior número de quadros permite capturar nuances mais finas do movimento dos sinais que podem ajudar o modelo a distingui-los. A Tabela 6.1 apresenta o comparativo entre modelos utilizando 15 e 30 quadros.

Tabela 6.1: Impacto da alteração do número de quadros no desempenho do modelo.

Modelo	Características	Nº Quadros	Unidades LSTM	Acurácia
M_1	**Originais**	**30**	**512**	**84,22%**
Base	Originais	15	512	84,00%

Fonte: Elaborada pelo autor (2026).

Os resultados indicam que a ampliação para 30 quadros por vídeo proporcionou um ganho de aproximadamente 0,22% em relação ao modelo base.

6.3 Análise da Arquitetura do Modelo

Alguns experimentos foram realizados com foco na arquitetura do modelo, variando o tamanho do lote e a profundidade da rede. A Tabela 6.2 apresenta o comparativo entre modelos utilizando 256, 512 e 1024 unidades LSTM. O tamanho do lote foi fixado em 1024, com exceção do modelo M_1 que precisou ser reduzido para 512 por limitação de recursos da máquina. Além disso, os mesmos testes foram realizados com 15 e 30 quadros para realizar um comparativo mais amplo.

Tabela 6.2: Impacto da alteração de hiperparâmetros e profundidade do modelo.

Modelo	Características	Nº Quadros	Unidades LSTM	Tamanho do Lote	Acurácia
M_2	**Originais**	**30**	**1024**	**512**	**87,22%**
M_1	Originais	30	512	1024	84,22%
M_3	Originais	30	256	1024	84,00%
M_4	Originais	15	1024	1024	84,33%
Base	Originais	15	512	1024	84,00%
M_5	Originais	15	256	1024	83,67%

Fonte: Elaborada pelo autor (2026).

Os resultados indicam que o aumento das unidades LSTM contribuiu para um melhor desempenho do modelo. Para os testes realizados com 15 quadros por vídeo, o aumento das unidades LSTM de 512 (configuração do modelo base) para 1024 produziu um ganho de 0,33% de acurácia. Já nos testes com 30 quadros por vídeo, o aumento das unidades LSTM de 512 para 1024 gerou um ganho de 3,00%. Além disso, esse cenário evidencia ainda mais o impacto positivo do aumento do número de quadros, ao comparar os modelos com as duas melhores acurácias dentre os testes.

6.4 Análise de Características

Um dos focos centrais deste trabalho foi a exploração de novos conjuntos de características para enriquecer a representação dos sinais. Foram testadas combinações de características geométricas (ângulos e distâncias das mãos e da pose) mencionadas na Seção 5.1.1, um segundo grupo expandido de distâncias entre pontos específicos da face e das mãos descritas

na Seção 5.1.2, e características de movimento, calculadas pela diferença entre quadros consecutivos, apresentadas na Seção 5.1.3. Os testes foram realizados com 512 unidades LSTM e tamanho de lote 1024. A Tabela 6.3 resume os testes realizados variando as características.

Tabela 6.3: Comparativo do impacto de diferentes conjuntos de atributos. Orig.: Geométricas originais, Exp.: Geométricas expansão, Mov.: Movimento.

Modelo	**Características**	**Nº Caracte.**	**Nº Quadros**	**Acurácia**
$\boldsymbol{M_6}$	**Orig. + Exp. + Mov.**	**192**	**30**	**86,78%**
M_7	Orig. + Mov.	154	30	86,33%
M_8	Orig. + Exp.	126	30	85,11%
M_9	Exp. + Mov.	114	30	83,67%
M_1	Orig.	88	30	84,22%
M_{10}	Exp.	48	30	80,56%
M_{11}	Mov.	66	30	73,78%
Base	Orig.	88	15	84,00%

Fonte: Elaborada pelo autor (2026).

A análise dos dados revela que a combinação completa de características (Geométricas, Novas Distâncias e Movimento) atingiu o melhor desempenho geral. Observa-se que o uso isolado de atributos de movimento não é suficiente para uma classificação precisa (73,78% de acurácia), mas sua integração como informação complementar aos atributos geométricos é fundamental para distinguir sinais que possuem configurações de mão semelhantes, mas dinâmicas de movimento distintas.

6.5 Arquitetura com Modelos Especialistas

Para tratar as classes que apresentavam maior confusão no modelo base, foi implementada uma arquitetura com um modelo especialista. Esse modelo consiste em uma rede treinada especificamente para distinguir entre classes semelhantes, como “Pesquisar” e “Perguntar”, ou “Software” e “Inteligência artificial”.

De modo a avaliar o desempenho do modelo especialista, cada um foi treinado individualmente, buscando a melhor configuração para ser utilizada juntamente com o modelo completo. Quanto à configuração, a rede especialista manteve a arquitetura do modelo geral, porém configurada com 1024 unidades LSTM e tamanho de lote de 32. A Tabela 6.4 apresenta o desempenho variando as características utilizadas no treinamento

do modelo para as classes “Pesquisar” e “Perguntar”.

Tabela 6.4: Variação das características no modelo especialista das classes “Perguntar” e “Pesquisar”. Orig.: Geométricas originais, Exp.: Geométricas expansão, Mov.: Movimento.

Modelo	**Características**	**Nº Características**	**Nº Quadros**	**Acurácia**
E_1	Orig.	88	30	71,11%
$\boldsymbol{E_2}$	**Mov.**	**66**	**30**	**77,78%**
E_3	Orig. + Exp.	126	30	67,78%
E_4	Orig. + Exp. + Mov.	192	30	63,33%

Fonte: Elaborada pelo autor (2026).

Destaca-se pela análise dos resultados que a melhor configuração para o modelo especialista das classes “Perguntar” e “Pesquisar” é a utilização unicamente das características de movimento, que alcançou uma acurácia de 77,78%. Em contrapartida, o modelo base obteve uma acurácia inferior de 65,50% entre as duas classes. A matriz de confusão do melhor modelo pode ser observada na Figura 6.2.

Ao contrário do modelo geral, o modelo especialista desempenha melhor utilizando apenas características de movimento provavelmente por conta das outras características possuírem valores muito similares. Logo, utilizar um conjunto menor de características que é capaz de distinguir os sinais é mais vantajoso.

Para o modelo especialista entre as classes “Software” e “Inteligência artificial”, os mesmos testes foram realizados. A Tabela 6.5 apresenta o resultado do treinamento variando as características do modelo especialista para as duas classes.

Tabela 6.5: Variação das características no modelo especialista das classes “Software” e “Inteligência artificial”. Orig.: Geométricas originais, Exp.: Geométricas expansão, Mov.: Movimento.

Modelo	**Características**	**Nº Caracte.**	**Nº Quadros**	**Acurácia**
E_5	Orig.	88	30	88,89%
$\boldsymbol{E_6}$	**Mov.**	**66**	**30**	**98,52%**
E_7	Orig. + Exp.	126	30	87,78%
E_8	Orig. + Exp. + Mov.	192	30	85,56%

Fonte: Elaborada pelo autor (2026).

Da mesma forma que o modelo anterior, o modelo das classes “Software” e “Inteligência artificial” obteve o melhor resultado utilizando apenas as características de movimento, alcançando uma acurácia de 98,52%. Em contrapartida, o modelo base ob-

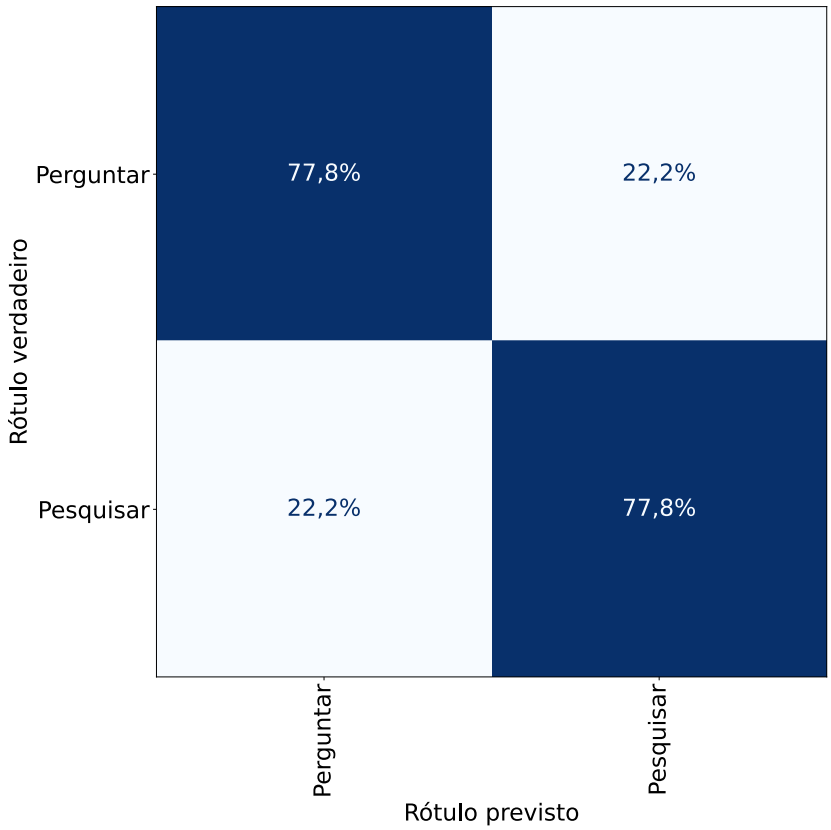


Figura 6.2: Matriz de confusão do modelo especialista para classes “Pesquisar” e “Perguntar” utilizando características de movimento.

teve uma acurácia inferior, de 77,50%, entre as duas classes. A matriz de confusão do melhor modelo pode ser observada na Figura 6.3.

Em seguida, os modelos especialistas foram testados em funcionamento junto com o modelo geral. Os testes contemplam a presença de apenas o primeiro modelo especialista operando no reconhecimento, apenas o segundo modelo especialista e os dois modelos especialistas juntos. Quando um modelo especialista é utilizado, o modelo geral é treinado com 19 classes, ao invés de 20, pois duas classes são agregadas em uma. Da mesma forma, quando dois modelos especialistas são utilizados, o modelo geral é treinado com 18 classes. Os modelos são treinados de forma independente. Após a classificação inicial pelo modelo geral, as instâncias identificadas como classe agregada são encaminhadas ao modelo especialista, que realiza a distinção final entre os dois sinais específicos. Os testes utilizaram 1024 unidades LSTM tanto para o modelo geral, quanto para os modelos especialistas. Quanto ao tamanho de lote foi utilizado 512 no modelo geral e 32 nos modelos especialistas. A Tabela 6.6 apresenta os resultados dos testes do modelo geral

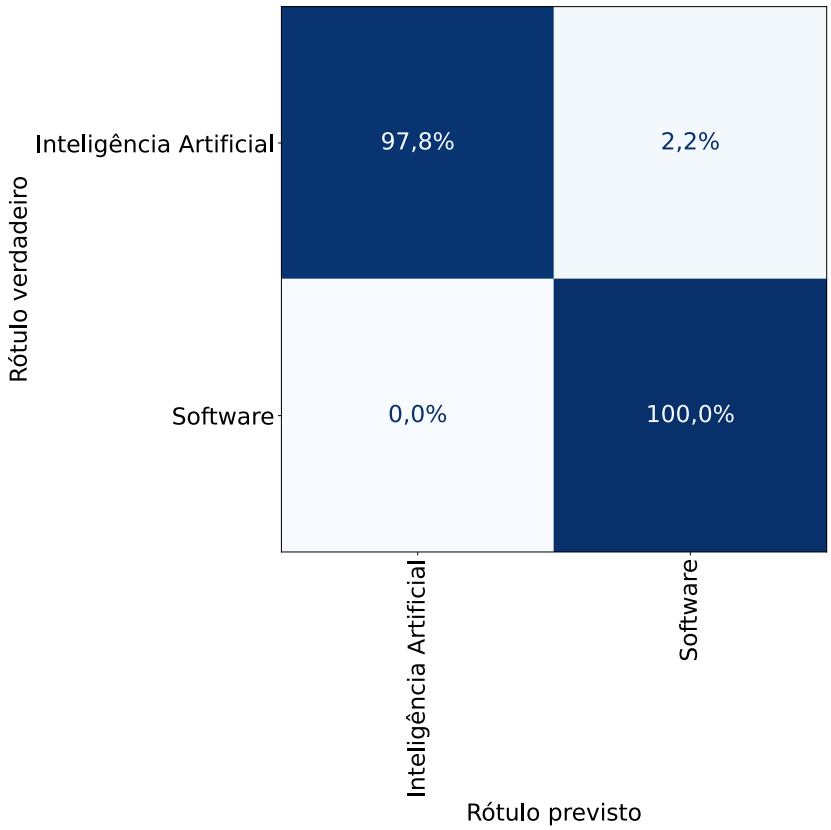


Figura 6.3: Matriz de confusão do modelo especialista para classes "Software" e "Inteligência Artificial" utilizando características de movimento.

combinado com os especialistas.

Tabela 6.6: Modelo geral combinado com modelos especialistas. Orig.: Geométricas originais, Exp.: Geométricas expandidas, Mov.: Movimento.

Modelo	Características modelo geral	Características especialistas	Especialista Perg. - Pesq.	Especialista Soft. - I.A.	Acurácia
$M_{14} + E_2$	Orig. + Exp. + Mov.	Mov.	Sim	Não	87,89%
$M_{15} + E_6$	Orig. + Exp. + Mov.	Mov.	Não	Sim	87,00%
$\boldsymbol{M_{16} + E_2 + E_6}$	**Orig. + Exp. + Mov.**	**Mov.**	**Sim**	**Sim**	**88,89%**

Fonte: Elaborada pelo autor (2026).

Verifica-se que a combinação de todas as características no modelo geral, juntamente com os dois modelos especialistas (utilizando apenas características de movimento), cada um com 1024 unidades LSTM, constitui o modelo de maior acurácia. Pela matriz de confusão da Figura 6.4 observa-se que os sinais "Perguntar/Pesquisar" foram menos confundidos entre si, comparado ao modelo base. Em ambos os sinais, o *recall* das classes foi de 82,2%, ao passo que no modelo base os *recalls* foram de 62,2% e 68,9% respectivamente. Da mesma forma, os sinais "Inteligência artificial/Software" alcançaram *recall* superior de 97,8% nas classes, enquanto no modelo base os *recalls* foram de 82,2% e 73,3%.

Isso mostra que os modelos especialistas contribuíram para melhorar o reconhecimento dos sinais ambíguos, principalmente quando utilizam as características de movimento.

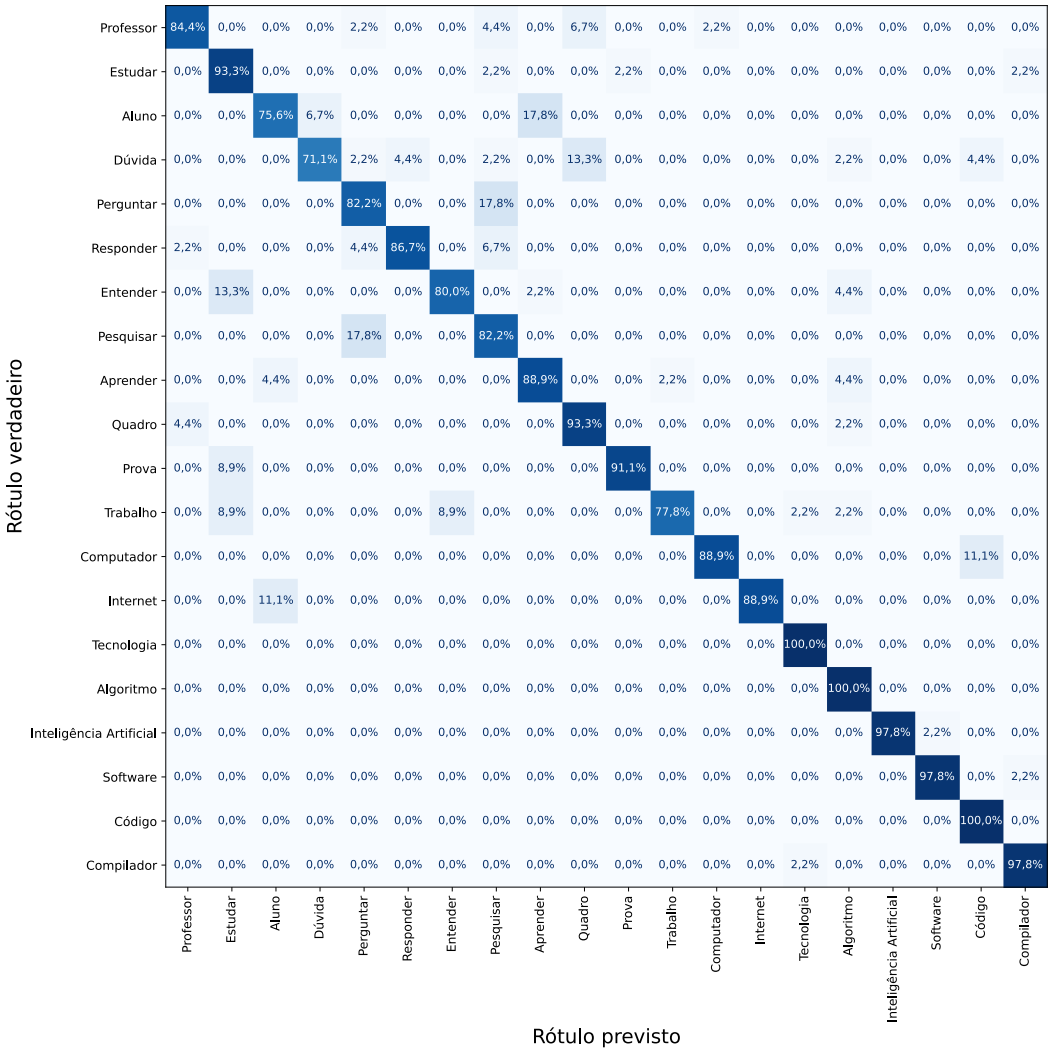


Figura 6.4: Matriz de confusão do melhor modelo combinando o modelo geral com os especialistas.

7 Conclusão

Este trabalho propôs o desenvolvimento de um sistema de reconhecimento de sinais dinâmicos da Língua Brasileira de Sinais (Libras) focado no contexto educacional, visando auxiliar nos estudos para a mitigação das barreiras de comunicação enfrentadas pela comunidade surda em salas de aula. O objetivo geral foi alcançado por meio da criação de um conjunto de dados específico e da implementação de modelos de aprendizado de máquina capazes de classificar sinais com movimento.

A metodologia envolveu a construção de uma base de dados própria contendo 900 vídeos, contemplando 20 sinais distintos realizados por 9 sinalizadores. Para o processamento desses dados, utilizou-se a ferramenta MediaPipe Holistic para a extração de *keypoints* e a técnica de aumento de dados em 20 vezes para buscar maior generalização do modelo no treinamento.

A arquitetura do modelo baseou-se em redes neurais recorrentes do tipo *Long Short-Term Memory* (LSTM), explorando diferentes conjuntos de características, como ângulos e distâncias, além de atributos de movimento.

Os experimentos demonstraram que a evolução incremental do modelo base, permitiu ganhos de desempenho na tarefa de reconhecimento de sinais. Verificou-se que:

- A ampliação do número de quadros amostrados (de 15 para 30) e do número de unidades LSTM (de 512 para 1024) foi determinante para capturar melhor a dinâmica dos sinais.

- A integração de características de movimento foi essencial para distinguir sinais com configurações de mão semelhantes, mas trajetórias distintas.

- A implementação de uma arquitetura com modelos especialistas mostrou-se eficaz para tratar ambiguidades, como as observadas entre os pares “Pesquisar/Perguntar” e “Software/Inteligência Artificial”.

- O modelo geral combinado com os especialistas atingiu uma acurácia final de 88,89%,

superando a referência inicial de 84,00% do modelo base.

Apesar dos resultados promissores, este trabalho ainda apresenta lacunas para avanços. Como propostas para trabalhos futuros, sugerem-se os seguintes pontos:

- **Expansão do Vocabulário:** Ampliar a base de dados para incluir uma gama maior de sinais do contexto educacional e de outras áreas.
- **Reconhecimento Contínuo:** Evoluir da classificação de sinais isolados (ISLR) para o reconhecimento de sinais contínuos (CSLR), permitindo a tradução de frases completas sem a necessidade de pausas.
- **Refinamento da Arquitetura:** Explorar novas arquiteturas para os modelos geral e especialistas, como a otimização de hiperparâmetros específicos e adição de novas camadas relevantes.
- **Modelos Especialistas:** Explorar a utilização de outros modelos especialistas para melhorar o desempenho do modelo em classes que ainda apresentam confusão, como “Aluno”, “Dúvida”, “Entender”, “Trabalho”, “Computador” e “Internet”.
- **Implementação em Tempo Real:** Testar a eficiência computacional do modelo em dispositivos móveis para uso prático em ambientes de ensino.

As contribuições deste estudo reforçam a viabilidade do uso de aprendizado de máquina e visão computacional para promover a acessibilidade, servindo como base para futuras ferramentas de apoio à inclusão social e educacional de surdos no Brasil.

Bibliografia

ALVES, C. E. G. R.; BOLDT, F. d. A.; PAIXãO, T. M. Enhancing brazilian sign language recognition through skeleton image representation. *arXiv e-prints*, Apr 2024.

ARAúJO, C. S. S. B. d.; FERREIRA, A. C. d. A. Análise do parâmetro movimento e sua importância para a comunicação em libras. In: *Anais do II Congresso Internacional de Educação e Inclusão (CINTEDI)*. Campina Grande: Realize Editora, 2016. Disponível em: ⟨https://editorarealize.com.br/editora/anais/cintedi/2016/TRABALHO_EV060_MD1_SA7_ID2936_21092016170248.pdf⟩.

BANSAL, N.; JAIN, A. Dynamic isl word recognition system using resnet50 and rnn deep learning models. In: . [s.n.], 2023. p. 497 – 500. Cited by: 0. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85195238024&doi=10.1109%2fICIIP61524.2023.10537746&partnerID=40&md5=ac3a2529f86174ccd9f71bd9e3aff58e⟩.

BASTOS, I. L. O. *Reconhecimento de sinais da Libras utilizando descritores de forma e redes neurais artificiais*. Dissertação (Dissertação de Mestrado) — Universidade Federal da Bahia, Salvador, May 2015. Disponível em: ⟨https://repositorio.ufba.br/handle/ri/19374⟩.

BRASIL. *Lei nº 10.436, de 24 de abril de 2002*: Dispõe sobre a língua brasileira de sinais - libras e dá outras providências. Brasília, DF, 2002. Disponível em: ⟨https://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm⟩.

BUTTAR, A. M. et al. Deep learning in sign language recognition: A hybrid approach for the recognition of static and dynamic signs. *Mathematics*, v. 11, n. 17, 2023. Cited by: 39; All Open Access, Gold Open Access. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85176402392&doi=10.3390%2fmath11173729&partnerID=40&md5=d5614da79048ed3ff97c4a7b99b3532a⟩.

DAMDOO, R.; KUMAR, P. Signedgelvm transformer model for enhanced sign language translation on edge devices. *Discover Computing*, v. 28, n. 1, 2025. Cited by: 0; All Open Access, Hybrid Gold Open Access. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-86000790010&doi=10.1007%2fs10791-025-09509-1&partnerID=40&md5=0faa21546c1bbd3517b598d87402be5a⟩.

DIKSHANT et al. Handwave: A convolutional model for indian sign language recognition: A deep learning approach. In: . [s.n.], 2024. p. 502 – 508. Cited by: 0. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-105004904743&doi=10.1109%2fICPIDS65698.2024.00084&partnerID=40&md5=7ae2b902a709ff982d6b42ae1ded16a0⟩.

HE, X. et al. Ai chinese sign language recognition interactive system based on audio-visual integration. In: . [s.n.], 2023. p. 962 – 968. Cited by: 6. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85187317466&doi=10.1109%2fICEACE60673.2023.10442295&partnerID=40&md5=06eae712b461225fb0a88ce74d4a6d8f⟩.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, p. 1735–1780, 11 1997.

LIPTON, Z. C.; BERKOWITZ, J.; ELKAN, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015. Disponível em: ⟨https://arxiv.org/abs/1506.00019⟩.

LIU, J. et al. Dynamic gesture recognition based on cnn-lstm-attention. In: . [s.n.], 2021. Cited by: 1. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85118447551&doi=10.1109%2fICSPCC52875.2021.9565034&partnerID=40&md5=0e31e981a4f02defd4c53539a773ea03⟩.

LOSHCHILOV, I.; HUTTER, F. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. Disponível em: ⟨http://arxiv.org/abs/1711.05101⟩.

LU, H.; SALAH, A. A.; POPPE, R. Tcnet: Continuous sign language recognition from trajectories and correlated regions. In: . [s.n.], 2024. v. 38, n. 4, p. 3891 – 3899. Cited by: 4; All Open Access, Gold Open Access, Green Open Access. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85189537431&doi=10.1609%2faaai.v38i4.28181&partnerID=40&md5=f53d11a25f26cb7414954f3ace6c3eef⟩.

OROPESA, A. R. M.; FELICEN, G. L. R.; GUZMAN, J. A. D. Senyas: A filipino sign language recognition system using mediapipe and cnn-lstm. In: . [s.n.], 2024. p. 956 – 960. Cited by: 0. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-105000430003&doi=10.1109%2fTENCON61640.2024.10902785&partnerID=40&md5=fa78485e1ff9e19aa6c6d88f8b89ebe0⟩.

QUADROS, R. *Libras SignBank*. 2023. Acesso em: 09 jan. 2026. Disponível em: ⟨https://signbank.libras.ufsc.br/pt⟩.

RAJAPAKSHE, A. et al. A robust vision-based dynamic sign language recognition using a hybrid cnn-lstm model. In: . [s.n.], 2025. Cited by: 0. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-105004558554&doi=10.1109%2fICARC64760.2025.10963007&partnerID=40&md5=250d6b0cc6540c62720ff12fdcb42cd4⟩.

SARMENTO, A. H. d. A. *Integração de Datasets de Vídeo para Tradução Automática da LIBRAS com Aprendizado Profundo*. Dissertação (Dissertação de Mestrado) — Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo (USP), São Carlos, 2023. Disponível em: ⟨https://www.teses.usp.br/teses/disponiveis/55/55137/tde-10012024-093541/pt-br.php⟩.

SINGH, D. K. 3d-cnn based dynamic gesture recognition for indian sign language modeling. *Procedia Computer Science*, v. 189, p. 76–83, 2021. ISSN 1877-0509. AI in Computational Linguistics. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1877050921011650⟩.

STOKOE, W. C. *Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf*. Buffalo: Department of Anthropology and Linguistics, University of Buffalo, 1960.

ZUO, R.; MAK, B. Local context-aware self-attention for continuous sign language recognition. In: . [s.n.], 2022. v. 2022-September, p. 4810 – 4814. Cited by: 12. Disponível em: ⟨https://www.scopus.com/inward/record.uri?eid=2-s2.0-85140053608&doi=10.21437%2fInterspeech.2022-164&partnerID=40&md5=7a035ecb1a3dc3a69dc34d8f6ae977d2⟩.