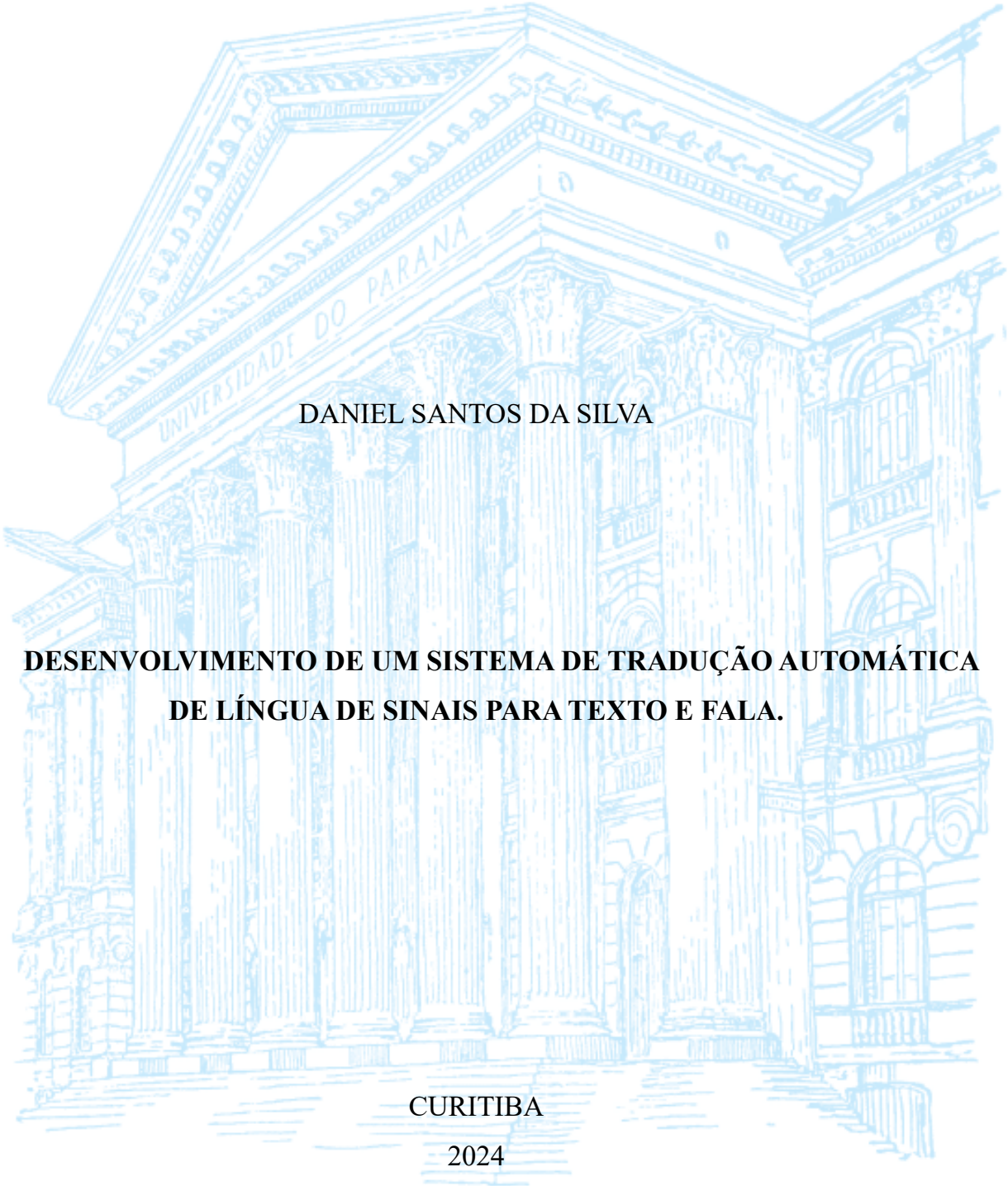
UNIVERSIDADE FEDERAL DO PARANÁ



DANIEL SANTOS DA SILVA

DESENVOLVIMENTO DE UM SISTEMA DE TRADUÇÃO AUTOMÁTICA DE LÍNGUA DE SINAIS PARA TEXTO E FALA.

CURITIBA

2024

DANIEL SANTOS DA SILVA - GRR20182083

DESENVOLVIMENTO DE UM SISTEMA DE TRADUÇÃO AUTOMÁTICA DE LÍNGUA DE SINAIS PARA TEXTO E FALA.

Relatório de Trabalho de Conclusão de Curso apresentado ao Curso de Graduação de Engenharia Elétrica, Ênfase em Eletrônica e Telecomunicações, Setor de Tecnologia, Universidade Federal do Paraná – Campus Curitiba, como requisito para à obtenção do título de Engenheiro Eletricista.

Orientador(a): Prof.ª Dra. Giselle Lopes Ferrari Ronque

Curitiba

2024

Resumo

Este trabalho apresenta o desenvolvimento de um sistema de tradução automática de gestos da Língua Brasileira de Sinais (LIBRAS) para texto e fala em tempo real, utilizando técnicas avançadas de visão computacional e aprendizado profundo. A solução foi projetada com base em bibliotecas como OpenCV e MediaPipe, responsáveis pela captura e interpretação de gestos a partir de *keypoints* das mãos, rosto e corpo. Os gestos foram segmentados em estáticos e dinâmicos, sendo os primeiros classificados por um modelo *Random Forest*, que atingiu 98,65% de precisão, e os dinâmicos, por uma rede LSTM, com acurácia de 96,5% no conjunto de teste.

O trabalho incluiu a aquisição de dados de gestos, o pré-processamento para assegurar consistência e qualidade, e a implementação de um sistema integrado capaz de traduzir os gestos reconhecidos em áudio e texto. Além disso, o sistema foi validado em diferentes condições de captura, como iluminação, ângulos e posições. Os resultados demonstraram que a solução é eficaz na promoção da inclusão social e da acessibilidade, permitindo que pessoas surdas e ouvintes se comuniquem de maneira eficiente e natural. Este projeto evidencia o potencial das tecnologias assistivas no Brasil e contribui para a democratização do acesso à comunicação.

Palavras-chave: LIBRAS, Visão Computacional, Redes LSTM, Tradução Automática, Inclusão Social.

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Acrônimos

ANN	*Artificial Neural Network* (Rede Neural Artificial)
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
DL	*Deep Learning*
FPS	*Frames Per Second* (Quadros por Segundo)
IA	Inteligência Artificial
LIBRAS	Língua Brasileira de Sinais
LSTM	*Long Short-Term Memory* (Memória de Longo e Curto Prazo)
ML	*Machine Learning* (Aprendizado de Máquina)
PDI	Processamento Digital de Imagens
RGB	*Red, Green, Blue* (Vermelho, Verde, Azul)
RNN	*Recurrent Neural Network* (Rede Neural Recorrente)
UFPR	Universidade Federal do Paraná

Sumário

1. Introdução

A comunicação é a essência da interação humana, uma ponte que conecta diferentes mundos, perspectivas e realidades. Entretanto, a barreira da linguagem muitas vezes se torna uma muralha intransponível, excluindo indivíduos surdos do pleno exercício desse direito fundamental. Neste contexto, surge a necessidade imperativa de explorar soluções tecnológicas inovadoras que não apenas superem essas barreiras, mas também promovam a inclusão e a igualdade de oportunidades para todos.

Neste Trabalho de Conclusão de Curso propõe-se a enfrentar esse desafio, adentrando no universo da língua de sinais e das tecnologias emergentes para desenvolver um Tradutor Automático de Língua de Sinais para Texto/Fala. A iniciativa visa criar uma ponte digital entre a comunidade surda e aquelas pessoas que não têm conhecimento da língua de sinais, ampliando as possibilidades de interação e compreensão mútua simultaneamente.

Neste projeto, busca-se não apenas apresentar uma solução tecnológica, mas também ressaltar a importância da inovação e do comprometimento ético na construção de um mundo mais acessível e inclusivo para todos. O desenvolvimento deste tradutor de língua de sinais representa um passo significativo em direção a uma sociedade que valoriza e respeita a diversidade linguística e cultural, permitindo que a comunicação seja verdadeiramente universal.

O desenvolvimento de sistemas automáticos de tradução de LIBRAS para texto ou fala é motivado pela necessidade de promover a inclusão social e comunicacional das pessoas surdas. Esses sistemas têm o potencial de transformar a vida de milhares de pessoas, facilitando a comunicação e garantindo acesso a informações e serviços essenciais, visto que a comunicação eficaz entre pessoas surdas e ouvintes é crucial em vários contextos. Em ambientes educacionais, por exemplo, sistemas de tradução podem auxiliar na compreensão de conteúdos didáticos, permitindo que estudantes surdos acompanhem aulas de forma mais eficiente e interajam com colegas e professores (QUADROS; KARNOPP, 2004). No Brasil, o Decreto nº 5.626/2005 regulamenta o uso da Língua Brasileira de Sinais (LIBRAS) em diferentes esferas, incluindo a educação, reconhecendo a importância de garantir acessibilidade para pessoas surdas em diversos contextos educacionais e profissionais. No ambiente profissional, tais sistemas podem facilitar a participação de trabalhadores surdos em reuniões, treinamentos e outras atividades, promovendo uma inclusão verdadeira e igualitária no local de trabalho. Socialmente, a tradução automática e o uso de *frameworks* como MediaPipe têm demonstrado resultados promissores em aplicações práticas, permitindo a interação de surdos em eventos, conferências e atividades culturais, ampliando suas oportunidades de participação (BRAGG, 2019).

Pessoas surdas enfrentam frequentemente barreiras ao acessar serviços públicos e privados. Em hospitais, por exemplo, a comunicação entre pacientes surdos e profissionais de saúde pode ser desafiadora sem a presença de intérpretes. Sistemas automáticos de tradução de LIBRAS para texto ou fala podem oferecer uma solução prática, permitindo uma comunicação direta e imediata. Da mesma forma, em repartições públicas, como delegacias de polícia, tribunais e serviços de assistência social, esses sistemas podem garantir que as pessoas surdas recebam atendimento adequado e compreendam plenamente seus direitos e deveres (QUADROS; KARNOPP, 2004).

A autonomia das pessoas surdas é significativamente ampliada quando elas têm acesso facilitado a informações e podem se comunicar de forma eficaz com ouvintes. Sistemas automáticos de tradução, como o VLibras, permitem que surdos acessem conteúdos na internet, utilizem aplicativos de smartphones e realizem tarefas cotidianas sem depender de intermediários. Isso não só aumenta sua independência, mas também promove a autoestima e a confiança, permitindo-lhes participar ativamente da sociedade em todas as suas dimensões (VLIBRAS, 2024).

A criação e implementação de sistemas de tradução automáticos também beneficiam a comunidade ouvinte, promovendo a disseminação do uso da LIBRAS. À medida que mais ouvintes interagem com esses sistemas, aumenta a sensibilização sobre a importância da LIBRAS e a inclusão social das pessoas surdas. Isso pode levar a um maior interesse e esforço para aprender LIBRAS, resultando em uma sociedade mais inclusiva e empática (BRAGG, 2019; QUADROS; KARNOPP, 2004).

O desenvolvimento de tais sistemas também impulsiona avanços tecnológicos em áreas como processamento de linguagem natural, reconhecimento de gestos, e inteligência artificial. A inovação nessas áreas não só beneficia a comunidade surda, mas também contribui para o progresso científico e tecnológico em geral, com aplicações potenciais em diversos outros campos. O trabalho de Capovilla e Raphael (2001) destaca a importância da integração e aceitação da LIBRAS na sociedade, apontando que a criação de sistemas automáticos de tradução é um passo significativo para a inclusão e valorização da cultura surda. Esses sistemas, ao facilitarem a comunicação, ajudam a romper barreiras históricas e estruturais que limitam a participação plena das pessoas surdas na sociedade.

Assim, o desenvolvimento de sistemas automáticos de tradução de LIBRAS para texto ou fala é uma iniciativa vital para promover a inclusão, autonomia e acessibilidade das pessoas surdas. Além de beneficiar diretamente essa comunidade, esses sistemas incentivam uma maior integração e compreensão entre surdos e ouvintes, contribuindo para uma sociedade mais justa e equitativa.

1.1 Objetivo Geral

O objetivo principal deste Trabalho de Conclusão de Curso é desenvolver um sistema inovador e eficiente de tradução automática de LIBRAS para texto e fala em tempo real, utilizando uma abordagem interdisciplinar que combina visão computacional, aprendizado de máquina e *deep learning*. Este sistema visa superar as barreiras comunicacionais enfrentadas pela comunidade surda, promovendo a inclusão social e a acessibilidade em diversos contextos, como educacional, profissional e social.

Por meio da criação de uma solução tecnológica robusta e acessível, o projeto busca facilitar a comunicação entre surdos e ouvintes, permitindo uma interação mais fluida e natural. Além disso, o desenvolvimento do sistema representa um avanço significativo no uso de tecnologias assistivas no Brasil, alinhando-se aos princípios de igualdade de oportunidades e respeito à diversidade linguística e cultural. O trabalho também se propõe a explorar o potencial transformador da tecnologia na promoção de um ambiente mais inclusivo, demonstrando como inovações no campo da inteligência artificial podem impactar positivamente a sociedade.

1.2 Objetivos Específicos

1.2.1) Capturar e interpretar gestos da língua de sinais

Desenvolver algoritmos de visão computacional para capturar e interpretar gestos de maneira precisa e eficiente.

1.2.2) Utilizar algoritmos de *Machine Learning*, *Deep Learning* e Redes Neurais para reconhecer e classificar os gestos

Implementar algoritmos de *machine learning* e subáreas deste segmento – *deep learning* e redes neurais – que possibilitem o reconhecimento e a classificação precisa dos gestos, garantindo a eficácia do sistema.

1.2.3) Desenvolver um sistema de tradução automática preciso e em tempo real

Criar um sistema que traduza os gestos reconhecidos para texto ou fala de forma precisa e em tempo real, permitindo uma comunicação fluida e instantânea.

1.2.4) Integrar tecnologias de saída para texto ou fala

Implementar saídas de texto ou fala para viabilizar a comunicação efetiva com pessoas que não conhecem a língua de sinais.

1.2.5) Promover a Inclusão e Acessibilidade

Assegurar que o sistema desenvolvido contribua significativamente para a inclusão e acessibilidade, possibilitando a comunicação eficaz entre pessoas surdas e aquelas que não têm conhecimento da língua de sinais.

Ao alcançar esses objetivos específicos, o projeto não apenas atenderá às demandas técnicas necessárias para a criação do tradutor de língua de sinais, mas também reforçará seu impacto social, proporcionando meios mais acessíveis e inclusivos para a comunicação, promovendo a igualdade de oportunidades e a interação harmoniosa entre diferentes comunidades linguísticas.

2. Fundamentação Teórica

Esse capítulo possui o objetivo de apresentar os conceitos relacionados ao desenvolvimento deste trabalho. Serão abordadas as noções sobre LIBRAS, explicando como essa forma de comunicação funciona e a importância de seu uso (Seção 2.1). Ainda neste capítulo, serão apresentados os conceitos de Visão Computacional (Seção 2.2), MediaPipe (Seção 2.3), Aprendizado de Máquina (Seção 2.4), *Deep learning* (Seção 2.5) e Redes Neurais (Seção 2.5) na construção do sistema deste projeto.

2.1 Língua Brasileira de Sinais (LIBRAS)

LIBRAS é a língua de sinais utilizada pela comunidade surda no Brasil. Reconhecida como uma língua natural, ela possui gramática e sintaxe próprias, conforme descrito por Quadros e Karnopp (2004). Caracteriza-se por ser visual-espacial e gestual, utilizando mãos, expressões faciais e movimentos corporais para a transmissão de significado. A comunicação por meio da LIBRAS desempenha um papel essencial na integração social e educacional de indivíduos surdos, possibilitando o acesso à informação e a interação social plena, frequentemente limitados pela ausência de meios acessíveis.

Dessa forma, temos que a LIBRAS é definida por gestos estáticos e dinâmicos, sendo uma língua essencialmente visual e espacial. A comunicação gestual permite que os surdos se expressem de maneira rica e complexa, englobando não apenas o movimento das mãos, mas também a utilização de expressões faciais e movimentos do corpo para transmitir nuances de significado. Essas informações e características da língua são essenciais para implementar um sistema que reconheça e detecte os gestos. Essa complexidade e riqueza tornam a LIBRAS uma ferramenta poderosa para a educação e socialização dos surdos, proporcionando-lhes um meio de comunicação completo e eficaz.

A Lei nº 10.436, promulgada em 24 de abril de 2002, e o Decreto nº 5.626, emitido em 22 de dezembro de 2005, são dois documentos legais que reconheceram a LIBRAS como meio de comunicação e expressão – sendo que este último delineia termos específicos relativos à propagação e adoção da LIBRAS no Brasil (BRASIL, 2002; 2005). Estas legislações enfatizam a importância da integração da população surda na sociedade.

Entretanto, mesmo com o reconhecimento legal e a crescente aceitação social da LIBRAS, ainda há uma carência significativa de ferramentas que promovam a acessibilidade para a comunidade surda. Essas ferramentas são cruciais para permitir que os surdos exponham seus pensamentos e opiniões de forma eficaz. A ausência de tais recursos pode limitar a plena integração e participação desses indivíduos em diversas esferas da sociedade.

Portanto, a promoção de uma maior acessibilidade e o desenvolvimento de mais ferramentas e recursos que suportem a utilização da LIBRAS são imperativos para garantir que a comunidade surda tenha igualdade de oportunidades na sociedade. Iniciativas que fomentem a inclusão e o reconhecimento das necessidades específicas dos surdos são essenciais para construir uma sociedade mais justa e equitativa.

2.2 Visão Computacional

2.2.1 Conceitos Gerais

A visão computacional é um campo da IA que permite aos computadores interpretarem e compreenderem o mundo visual de maneira similar aos humanos (SZELISKI, 2010). Uma pessoa, por exemplo, é capaz de observar determinado objeto e classificá-lo, reconhecer seu padrão e ser capaz de diferenciar diferentes tipos de figuras e formas. Na visão computacional o objetivo é replicar essa capacidade humana, permitindo com que a máquina consiga visualizar e entender o mundo visual ao seu redor. Um ser humano com visão funcional consegue, ainda, distinguir diferentes expressões faciais e rastrear movimentos.

Esta área de estudo é crucial para o desenvolvimento de sistemas de tradução de sinais, pois envolve a captura e análise de imagens e vídeos dos sinais em LIBRAS. A visão computacional possibilita que computadores reconheçam e interpretem os movimentos das mãos, expressões faciais e movimentos corporais, que são essenciais para a comunicação em LIBRAS.

A aquisição da imagem é crucial para que o modelo de inteligência artificial deste contexto possa iniciar a compreensão do ambiente ao seu redor. A captura da imagem, ou sequência de imagens que compõem um vídeo, pode ser feita através de câmeras e sensores. Uma vez que as imagens forem adquiridas, parte-se para o processamento destas.

A fase de processamento de imagem é onde, por meio de algoritmos adequados para cada contexto de aplicação, é possível transformar os dados presentes na captura, analisando e extraindo as informações relevantes presentes nesta. Características como coordenadas de um objeto presente no *frame* da imagem, coloração, formas etc. Todos esses dados são úteis para iniciar o processo de análise e interpretação.

Ao analisar e interpretar os dados processados na captura, é possível treinar o modelo de IA para reconhecer certos padrões, ou mesmo objetos e, ainda, detectar movimentos presentes em uma sequência de *frames*.

2.2.2 Aquisição e Processamento da Imagem

Para a aquisição e processamento das imagens, utilizou-se Python e a biblioteca OpenCV, amplamente utilizada no campo de visão computacional, oferecendo uma vasta gama de ferramentas para lidar com distintos cenários de aplicação.

Uma técnica amplamente utilizada para tratar dados de imagens com OpenCV é a escala de cinza, pois diminui o tamanho da complexidade a ser processada. A intensidade das cores passa a variar de 0 a 255, ao invés dos 3 canais do espectro RGB. Como resultado, tem-se uma imagem definida com tons de cinza, sendo definida por uma matriz contendo estes valores, conforme exemplificada na Figura 1 e Figura 2. Além de reduzir a complexidade computacional, a conversão para escala de cinza facilita a aplicação de diversos algoritmos de processamento de imagem, como detecção de bordas, análise de textura e segmentação. Este método é especialmente útil em contextos em que a informação de cor não é essencial, permitindo focar em características estruturais e de luminosidade da imagem. A simplicidade da escala de cinza também acelera o tempo de processamento e reduz a necessidade de recursos computacionais, tornando-a uma escolha eficiente para diversas aplicações em visão computacional.

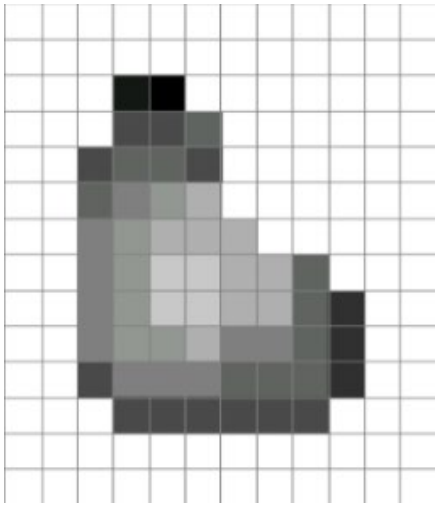


Figura 1 - Imagem em escala de cinza.

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Figura 2 - Imagem em escala numérica de cinza.

Para o processamento e análise, pode-se aplicar algoritmos de detecção a fim de captar gestos, objetos e expressões faciais. Uma das ferramentas que possibilita essa aplicação é o *framework* desenvolvido pela Google, chamado de Mediapipe, o qual será abordado em mais detalhes na Seção 2.3. Mediapipe oferece uma ampla variedade de soluções para reconhecimento de gestos, rastreamento de mãos, detecção de rostos e identificação de pontos-chave, utilizando técnicas avançadas de *machine learning* e visão computacional. Além disso, este *framework* permite uma implementação eficiente e em tempo real, sendo altamente configurável e adaptável a diferentes necessidades de aplicação. Com a integração de Mediapipe, é possível desenvolver sistemas robustos e precisos para interpretação de sinais visuais, facilitando a criação de interfaces interativas e inteligentes.

2.2.3 Análise e Interpretação da Imagem

Os principais desafios da visão computacional no contexto da tradução de LIBRAS incluem:

1. Variabilidade dos Sinais: Os sinais em LIBRAS podem variar amplamente em termos de velocidade, estilo e contexto. A mesma palavra ou expressão pode ser sinalizada de maneiras ligeiramente diferentes por diferentes pessoas ou até pela mesma pessoa em diferentes momentos. Isso exige que os sistemas de visão computacional sejam altamente adaptáveis e capazes de generalizar bem a partir de dados variados.

2. Captura de Expressões Faciais e Movimentos Corporais: Além dos movimentos das mãos, a LIBRAS utiliza expressões faciais e movimentos corporais para transmitir significados e nuances. Capturar e interpretar esses elementos não verbais é essencial para uma tradução

precisa, mas apresenta desafios técnicos significativos, pois requer uma análise detalhada e sincronizada de múltiplos pontos de dados.

3. Segmentação e Reconhecimento de Sinais Individuais: Em um fluxo contínuo de comunicação, identificar onde um sinal começa e termina pode ser complexo. Os sistemas devem ser capazes de segmentar sinais individuais e reconhecê-los com precisão, mesmo em sequências rápidas e intercaladas.

2.3 MediaPipe

Uma das ferramentas mais avançadas para segmentar os gestos que serão identificados no modelo de detecção de língua de sinais é o MediaPipe, um *framework* desenvolvido pela Google para a construção de pipelines multimodais de processamento de mídia, como imagens e vídeos (MEDIAPIPE SOLUTIONS GUIDE, 2024). O MediaPipe oferece uma série de funcionalidades que são particularmente úteis para o desenvolvimento de sistemas de tradução de LIBRAS:

1. Detecção de Mãos: O MediaPipe *Hands* é um módulo que pode detectar e rastrear mãos em imagens e vídeos em tempo real. Ele fornece uma representação precisa das posições das mãos, incluindo os pontos-chave dos dedos, o que é essencial para reconhecer os diferentes sinais em LIBRAS.

2. Detecção de Face e Expressões Faciais: O MediaPipe Face *Mesh* é um módulo que captura a estrutura da face e rastreia pontos-chave com alta precisão. Isso permite a análise das expressões faciais, que são fundamentais para a interpretação correta dos sinais.

3. Rastreamento de Corpo: O MediaPipe Pose é um módulo que rastreia a pose corporal completa, identificando a posição e orientação das partes do corpo. Isso é útil para captar os movimentos corporais que complementam os sinais manuais em LIBRAS.

4. Integração em Tempo Real: O MediaPipe é altamente eficiente e pode processar dados em tempo real, o que é crucial para a tradução simultânea de sinais. Ele pode ser integrado a aplicativos móveis e sistemas de *desktop*, oferecendo flexibilidade para diversas implementações.

O modelo holístico do MediaPipe é uma biblioteca avançada que combina a detecção de múltiplas partes do corpo humano simultaneamente. Esse modelo é projetado para realizar a detecção e rastreamento em tempo real de face, mãos e pose corporal de maneira integrada. A holística do MediaPipe é especialmente útil para aplicações que requerem uma compreensão abrangente do

movimento humano, como a tradução de língua de sinais. O modelo holístico captura os seguintes dados:

- *Face Landmarks*: Detecção de pontos-chave (*keypoints*) no rosto, permitindo a análise de expressões faciais e movimentos labiais.
- Hand Landmarks: Detecção de pontos-chave (*keypoints*) nas mãos, crucial para identificar gestos e sinais.
- *Pose Landmarks*: Detecção de pontos-chave (*keypoints*) no corpo, facilitando a análise de posturas e movimentos corporais.

Após a aplicação do modelo citado acima, é possível segmentar os *keypoints* presentes em cada *frame* de captura. Estes pontos são referências específicas identificadas pelo modelo em diferentes partes do corpo. Para o caso da mão, por exemplo, são 21 pontos por mão, identificando desde articulações até as pontas dos dedos. A partir da extração das coordenadas dos *keypoints*, é possível treinar modelos de *machine learning* que vão reconhecer os sinais de LIBRAS. A Figura 3 traz um exemplo de segmentação em uma captura.

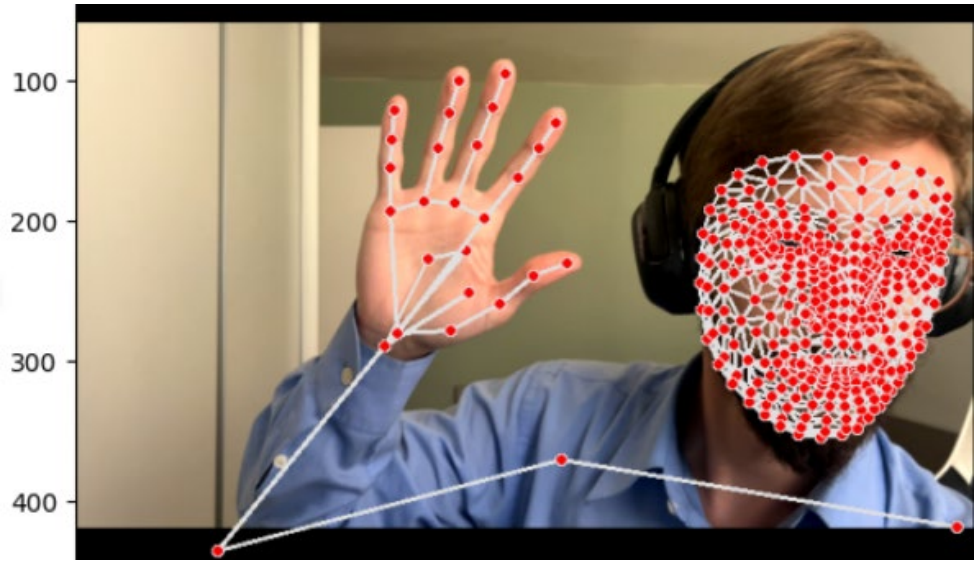


Figura 3 - Segmentação da captura MediaPipe.

As *Landmarks*, por sua vez, são os pontos específicos detectados na imagem. O Mediapipe desenha conexões entre esses pontos para formar uma representação visual compreensível das partes do corpo. No projeto, unifica-se ombros, braço e antebraço, como podemos visualizar na Figura 4.

Figura 4 - Landmarks: mão, face, corpo.

A visão computacional, combinada com a ferramenta MediaPipe, oferece uma base sólida para o desenvolvimento de sistemas automáticos de tradução de LIBRAS para texto e fala. Esses sistemas não apenas facilitam a comunicação e a inclusão social, mas também promovem a autonomia das pessoas surdas e sensibilizam a sociedade sobre a importância da LIBRAS. O avanço contínuo nesse campo promete tornar o mundo mais acessível e equitativo para todos.

Uma vez que as imagens podem ser segmentadas, é possível extrair os pontos chaves (*keypoints*) de cada segmentação, gerando coordenadas que irão definir cada gesto e sinal da linguagem. Com um grande volume de informações compondo os dados do projeto, é necessário explorar técnicas de *Machine learning* para começar a treinar o modelo deste trabalho, a fim de possibilitar a detecção, identificação e predição em tempo real dos gestos da língua de sinais.

2.4 *Machine learning*

Machine Learning, ou aprendizado de máquina, é um campo da inteligência artificial que envolve o desenvolvimento de algoritmos que permitem aos computadores aprenderem a partir de dados e fazerem previsões ou tomarem decisões sem serem explicitamente programados para isso. Técnicas de ML podem ser aplicadas em diversas áreas, incluindo visão computacional, processamento de linguagem natural, e análise preditiva.

A visão computacional é um campo da computação que estuda maneiras de extrair informações dos objetos de uma imagem. Juntamente com o processamento digital de imagens (PDI), a visão computacional analisa imagens para obter um resultado próximo do conhecimento humano. Em termos simples, a visão computacional é o processo de modelagem e replicação da visão humana utilizando recursos de *hardware* e *software*, compreendendo uma cena tridimensional (3D) a partir de imagens bidimensionais (2D) em termos das propriedades das estruturas presentes na cena.

O reconhecimento de objetos é uma das principais funções da visão computacional e está intimamente ligado ao reconhecimento de padrões. Um objeto pode ser definido por seus padrões de textura, cor, forma, dimensão, coordenadas no espaço, entre outros. O reconhecimento individual destes padrões pode caracterizar um objeto como um todo, ou seja, defini-lo como uma classe. Uma classe de padrões é uma família de padrões que compartilham entre si propriedades em comum.

Já para máquinas, reconhecer padrões envolve técnicas de atribuição de padrões às suas respectivas classes de forma automática e com a menor intervenção humana possível (GONZALEZ E WOODS, 2010). Um exemplo disso é a utilização de algoritmos de aprendizagem de máquina para a classificação de objetos. Ao extrair informações de uma imagem, é necessário analisar tais propriedades com o auxílio dos algoritmos de aprendizado de máquina, estimando um padrão em um novo exemplo a partir do aprendizado previamente adquirido.

Existem dois tipos principais de aprendizado de máquina: o supervisionado e o não supervisionado.

- Aprendizado Supervisionado: Utiliza dados com a classe previamente especificada. O algoritmo é treinado em um conjunto de dados de entrada onde as saídas desejadas são conhecidas. Este método é amplamente utilizado em tarefas de classificação e regressão.
- Aprendizado Não Supervisionado: Utiliza dados sem o atributo classe. Este tipo de aprendizado utiliza técnicas de agrupamento ou regras de associação para descobrir padrões ou estruturas ocultas nos dados.

A Figura 5 indica o fluxo de funcionamento de um modelo supervisionado (dados rotulados).

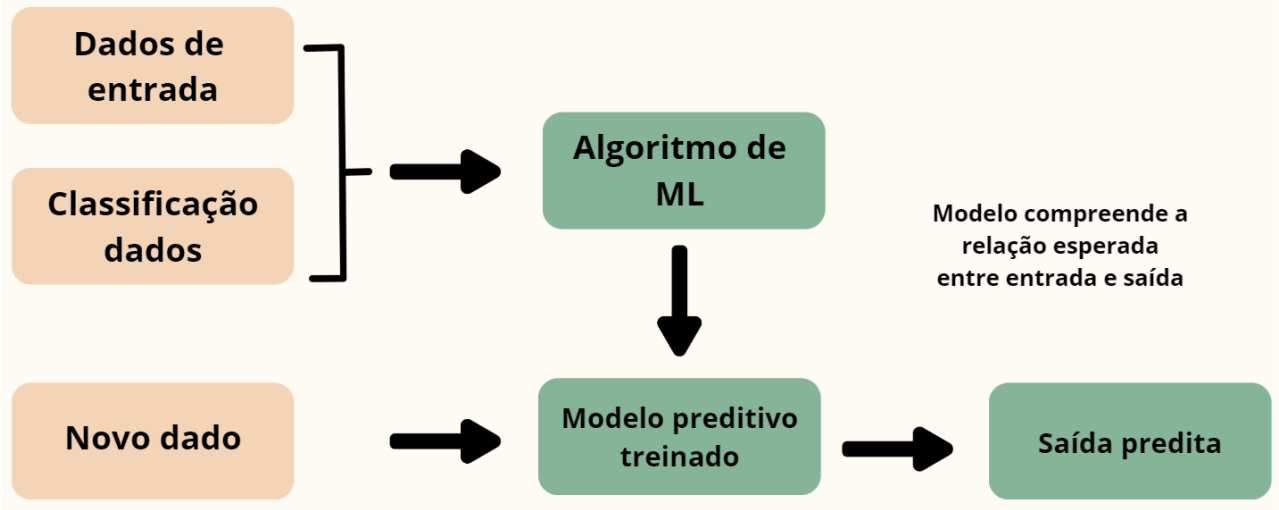


Figura 5- Fluxo Modelo Supervisionado de ML.

O sistema de ML é alimentado e treinado com dados de entrada devidamente classificados. Desenvolve-se um modelo preditivo já treinado, de modo que ele seja capaz de retornar uma saída prevista para cada novo dado inserido no sistema. O objetivo principal é que o sistema compreenda a relação esperada entre a entrada e a saída correspondente.

No contexto de problemas de classificação em aprendizado de máquina, utiliza-se conjuntos distintos de dados, a saber:

- Conjunto de Treinamento: Consiste em registros cujos rótulos são conhecidos e é utilizado para construir modelos de classificação.
- Conjunto de Testes: Consiste em registros cujos rótulos das classes são conhecidos, mas não participam do treinamento. Este conjunto é usado para avaliar a performance do modelo.

A avaliação do modelo se dá pela sua capacidade de classificar corretamente as instâncias do conjunto de testes, o que é medido pela precisão. A exatidão representa a proporção de predições corretas em relação ao total de predições feitas pelo modelo.

A classificação da base de treinamento é feita dividindo os dados entre dois grupos: base de treino e base de teste. Uma vez que possuímos a base de treino, utiliza-se ela para treinar o algoritmo, de modo que ele consiga classificar os dados adequadamente. A base de teste é processada removendo seus “rótulos verdadeiros”, de modo que esses dados sem rótulos vão ser enviados para o modelo classificar. Assim, obtém-se as “classes preditas nos testes do modelo”. Estas, por sua vez, são comparadas com os “rótulos verdadeiros”, onde é possível analisar a porcentagem de acertos que o modelo teve ao predizer as classes da base de teste sem rótulos. Esse valor percentual define a mencionada exatidão do modelo de ML. O fluxograma deste processo está na Figura 6.

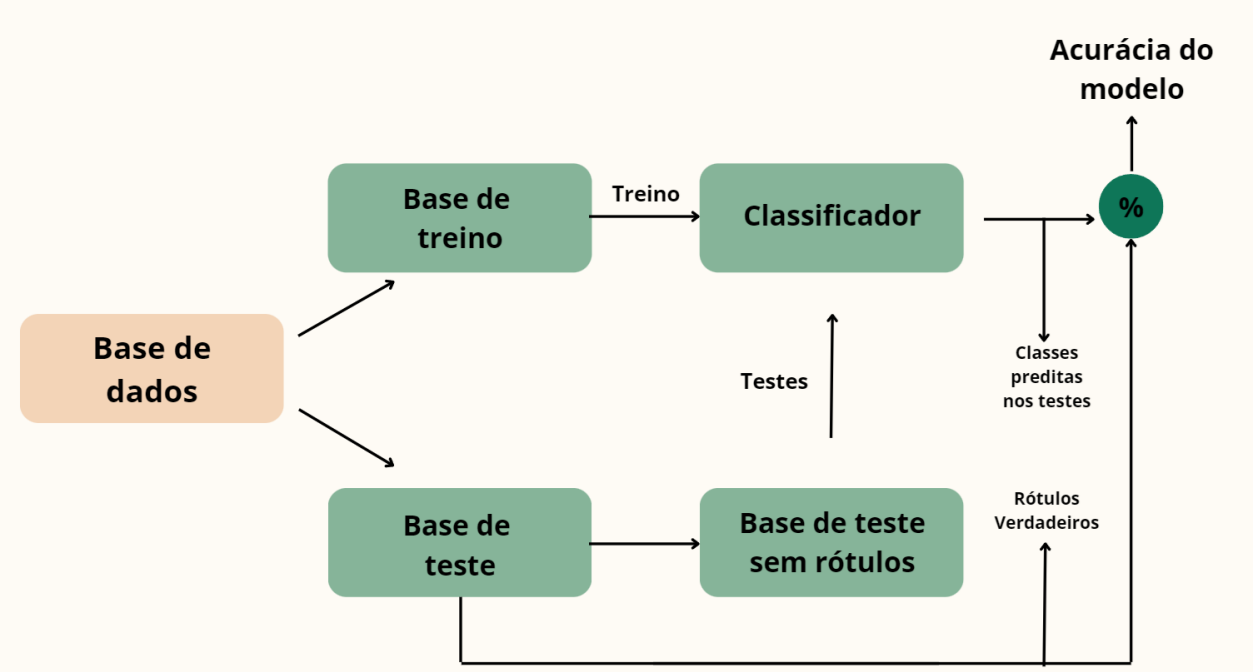


Figura 6 - Fluxo divisão base de dados.

Uma vez que que o modelo está treinado e realizando predições, é construída uma matriz de confusão, que permite analisar a distribuição de acertos do classificador (TAN ET AL., 2009).

Após treinar o modelo, implementa-se uma matriz de confusão, conforme indicado no Quadro 1. Esta é uma ferramenta que mostra a performance de um modelo de classificação, sendo composta por:

- Verdadeiros Positivos (VP): Casos em que as instâncias foram classificadas corretamente como pertencentes à classe positiva.
- Verdadeiros Negativos (VN): Casos em que as instâncias foram classificadas corretamente como pertencentes à classe negativa.
- Falsos Positivos (FP): Casos em que as instâncias foram classificadas incorretamente como pertencentes à classe positiva.
- Falsos Negativos (FN): Casos em que as instâncias foram classificadas incorretamente como pertencentes à classe negativa.

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Quadro 1 - Matriz de Confusão. Fonte: O autor (2024)

Por fim, calcula-se a precisão geral seguindo a fórmula abaixo:

$$Precisão = \frac{Número\ de\ previsões\ corretas}{Número\ total\ de\ amostras}$$

A tarefa de um algoritmo classificador é organizar os objetos entre diversas categorias pré-definidas. Em geral, os classificadores constroem modelos a partir dos dados de entrada (TAN ET AL., 2009). Alguns exemplos de métodos de classificação incluem:

- Árvores de Decisão: Estruturas de decisão que utilizam uma série de regras baseadas nas características dos dados.
- Redes Neurais Artificiais: Modelos inspirados no funcionamento do cérebro humano, compostos por camadas de neurônios artificiais.

Neste projeto de reconhecimento de gestos de língua de sinais, será necessário reconhecer imagens. Para este caso, classificadores lineares não são os mais indicados, uma vez que eles preveem uma classe por meio de uma função que delimita uma separação através de um hiperplano entre as classes. Para captura de *frames*, há mudanças de posição, orientação e iluminação dos objetos, garantindo maior complexidade para os dados. As imagens, ainda, fazem com que o sistema desenvolvido interprete os dados de alta dimensionalidade. Neste contexto, técnicas de extração de características automáticas, como o *Deep Learning* e Redes Neurais, são introduzidas para produzir características invariantes importantes para a discriminação de um objeto na cena (LECUN ET AL., 2015).

2.5 *Deep Learning* & Redes Neurais

2.5.1 Introdução ao *Deep Learning*

Deep Learning é um subcampo dentro da área de *Machine Learning* que se concentra no uso de redes neurais artificiais profundas para aprender e modelar padrões complexos nos dados. A característica distintiva do *Deep Learning* é a capacidade de aprender representações hierárquicas de dados através de múltiplas camadas de processamento não linear. Vamos explorar mais detalhadamente as características dessa ferramenta.

2.5.2 Características do *Deep Learning* e Redes Neurais

Redes neurais são sistemas computacionais inspirados pela estrutura e funcionamento do cérebro humano. Elas consistem em unidades interconectadas chamadas neurônios, que são organizadas em diferentes camadas. Cada camada é composta por vários neurônios que processam dados de entrada e passam informações para a próxima camada.

Estrutura das Redes Neurais

A estrutura básica de uma rede neural inclui três tipos de camadas:

- **Camada de Entrada**: Recebe os dados brutos que serão processados.
- **Camadas Ocultas**: Realizam a maior parte do processamento. Uma rede neural profunda possui múltiplas camadas ocultas, permitindo que a rede aprenda representações cada vez mais abstratas dos dados.
- **Camada de Saída**: Produz o resultado, como uma classificação ou predição.

O termo "profundo" (*deep*) em *Deep Learning* refere-se à profundidade da rede, ou seja, o número de camadas ocultas (LE CUN; BENGIO; HINTON, 2015). Cada camada aprende uma representação dos dados que é mais complexa que a camada anterior. Por exemplo, em uma rede neural para reconhecimento de imagens, as primeiras camadas podem aprender a detectar bordas e texturas, enquanto camadas mais profundas podem reconhecer formas e objetos completos.

Devido à sua habilidade de aprender representações hierárquicas e abstratas, os modelos de *Deep Learning* geralmente generalizam bem para novos dados, desde que o conjunto de treinamento seja representativo. Isso significa que o modelo consegue lidar melhor com novos exemplos ou novas informações, sendo uma característica essencial no campo de visão computacional, visto que, em cada captura, pode-se variar a iluminação do local, o ângulo da captura, a posição e até mesmo a visibilidade. Assim, com uma alta capacidade de abstração e generalização, um modelo de DL consegue fornecer predições mais precisas.

Existem vários tipos de redes neurais utilizadas em *Deep Learning*, cada uma adequada para diferentes tipos de problemas:

- **Redes Neurais Artificiais (ANNs):** As ANNs são a forma mais básica de redes neurais, compostas por camadas de neurônios totalmente conectadas. Adequadas a tarefas de classificação e regressão, são frequentemente usadas em problemas como reconhecimento de padrões, previsão de séries temporais e classificação de dados.
- **Redes Neurais Convolucionais (CNNs)**: Especialmente eficazes para tarefas de visão computacional, como reconhecimento de imagens e detecção de objetos.
- **Redes Neurais Recorrentes (RNNs)**: Adequadas para dados sequenciais, como séries temporais e processamento de linguagem natural.

A Figura 7 traz um exemplo de arquitetura de uma RNN. Nela, percebemos a camada de entrada e a camada de saída. As camadas ocultas exibem os neurônios (Σ), onde são realizados os cálculos dos pesos do modelo. Cada saída é representada pelo bloco "g", o qual irá conter as saídas dos aprendizados do modelo, informação esta que será enviada para outros neurônios de outras camadas e, ainda, retornará para os mesmos neurônios anteriores – por isso o termo "recorrente".

Essa estrutura permite que as RNNs processem dados sequenciais, mantendo uma memória das entradas anteriores para influenciar as saídas futuras. O estado interno das RNNs, mantido através de *loops* recorrentes, é essencial para capturar dependências temporais nos dados, como séries temporais ou texto.

Em uma RNN básica, a saída de cada neurônio não é apenas uma função da entrada atual, mas também do estado anterior, criando um ciclo de retroalimentação. No contexto do reconhecimento de gestos dinâmicos de LIBRAS, essa capacidade de considerar o contexto anterior é crucial para interpretar corretamente os movimentos ao longo do tempo.

Além disso, a habilidade de transmitir informações ao longo da sequência de dados permite que as RNNs sejam particularmente eficazes em tarefas onde o padrão temporal ou a ordem dos eventos são importantes. No entanto, para capturar dependências de longo prazo de forma mais eficiente e mitigar problemas como o desvanecimento do gradiente, utilizamos uma variante avançada das RNNs, chamada *Long Short-Term Memory* (LSTM).

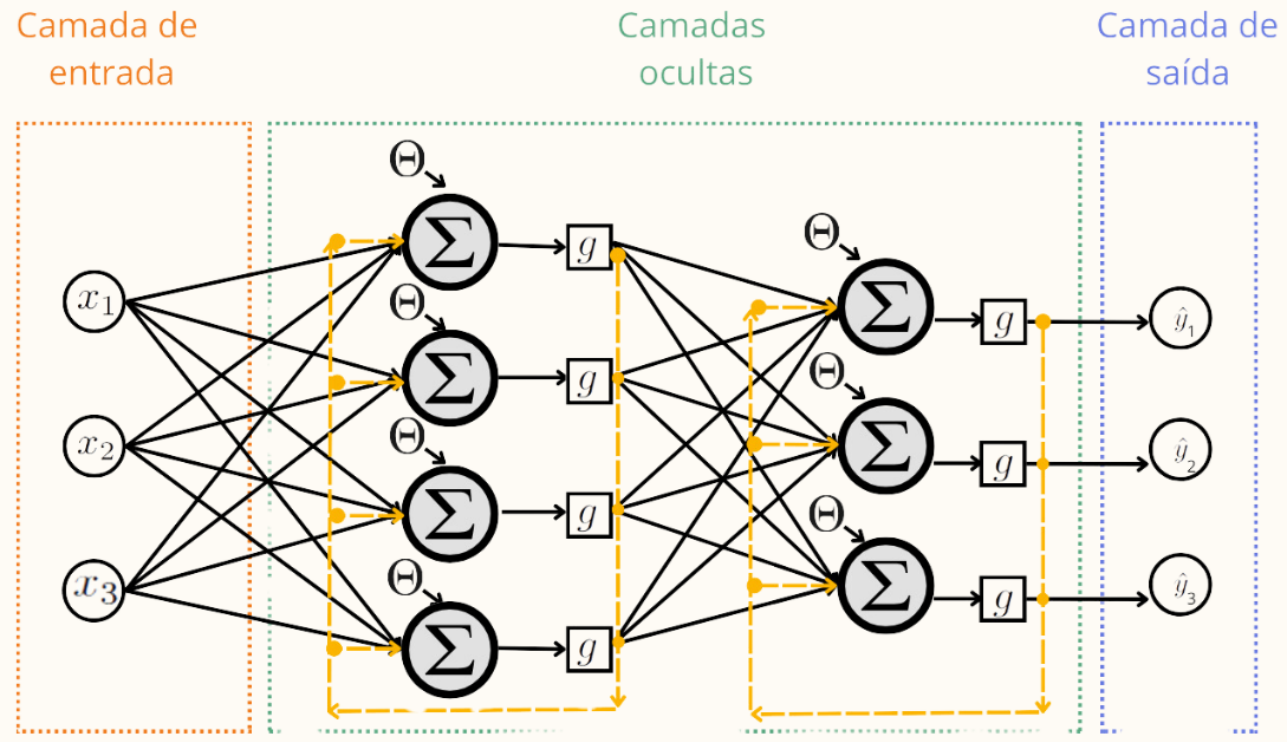


Figura 7 - Arquitetura Rede Neural Recorrente (RNN).

Dentro do vasto campo das redes neurais e seus tipos, existem diversas arquiteturas de modelos, cada uma projetada para lidar com diferentes tipos de dados e tarefas específicas. Para uma RNN, por exemplo, existe o modelo LSTM. Para este trabalho, optou-se por utilizar justamente o modelo de LSTM, devido às suas capacidades únicas de lidar com dados sequenciais e que possuem dependências temporais (HOCHREITER; SCHMIDHUBER, 1997). Os gestos dinâmicos da língua de sinais são compostos por uma sequência de *frames*, cada um representando uma parte do movimento. Capturar a evolução temporal desses gestos é crucial para um reconhecimento preciso e confiável.

2.5.3 Justificativas da Escolha do Modelo *Long Short-Term Memory*

1. **Dependências Temporais**:
    - **Definição**: Os gestos dinâmicos não são eventos isolados. A compreensão de um gesto depende do contexto fornecido pelos *frames* anteriores e seguintes. Isso caracteriza uma dependência temporal.
    - **LSTM**: As LSTMs são projetadas para lembrar informações por longos períodos e esquecê-las quando não são mais relevantes, o que é fundamental para capturar o contexto temporal dos gestos.

2. **Padrões Sequenciais Complexos**:

    - **Definição**: A língua de sinais envolve movimentos complexos e sutis que devem ser compreendidos como um todo. Não é suficiente reconhecer um *frame* individualmente; é necessário entender a sequência completa.
    - **LSTM**: As LSTMs possuem células de memória que podem capturar e reter informações sequenciais complexas, permitindo uma compreensão mais holística dos gestos.

3. **Robustez e Flexibilidade**:

    - **Aplicabilidade**: A robustez das LSTMs na captura de dependências temporais e padrões sequenciais complexos as torna ideais para uma ampla gama de aplicações, desde reconhecimento de gestos até tradução de língua de sinais e outros dados sequenciais.
    - **LSTM**: Essa flexibilidade é essencial para nosso projeto, que requer a detecção e classificação precisa de diferentes gestos de língua de sinais.

4. **Desempenho Empírico**:

    - **Evidências**: Estudos anteriores e experimentos em várias tarefas sequenciais demonstram que as LSTMs geralmente superam outras arquiteturas de redes neurais, como RNNs simples e CNNs, em tarefas que envolvem dados temporais.
    - **LSTM**: Baseando-se nessas evidências, a escolha da LSTM para o modelo proposto foi guiada pela expectativa de alcançar uma alta precisão e generalização no reconhecimento dos gestos dinâmicos.

5. **Comparações com Outra Arquiteturas:**

    - **ANN (*Artificial Neural Network*)**:
        - **Limitação**: ANNs tradicionais não possuem a capacidade de lidar com dependências temporais, pois tratam cada entrada de forma independente.
        - **LSTM**: Em contraste, as LSTMs são projetadas para processar e aprender a partir de sequências temporais, tornando-as mais adequadas para nosso projeto.

- **CNN (*Convolutional Neural Network*)**:
    - **Limitação**: Embora CNNs sejam eficazes na captura de padrões espaciais, elas não são projetadas para capturar dependências temporais ao longo de sequências de *frames*.
    - **LSTM**: As LSTMs, por outro lado, são especializadas em capturar padrões temporais e contextuais, essenciais para o reconhecimento de gestos de língua de sinais.

Assim, a escolha da LSTM foi baseada em sua capacidade de lidar com as complexidades temporais e sequenciais inerentes aos gestos dinâmicos da língua de sinais. Sua arquitetura permite capturar e reter informações temporais cruciais, garantindo uma alta precisão e generalização no reconhecimento dos gestos, fazendo dela a escolha ideal para este projeto.

3. Metodologias de Desenvolvimento

Para alcançar o objetivo de criar um modelo capaz de traduzir gestos de língua de sinais para texto e fala, foi elaborado um fluxograma para facilitar a compreensão passo a passo do que foi desenvolvido para alcançar os resultados desejados. Na Figura 8 é possível visualizar o fluxo do raciocínio empregado ao longo das etapas de desenvolvimento deste trabalho.

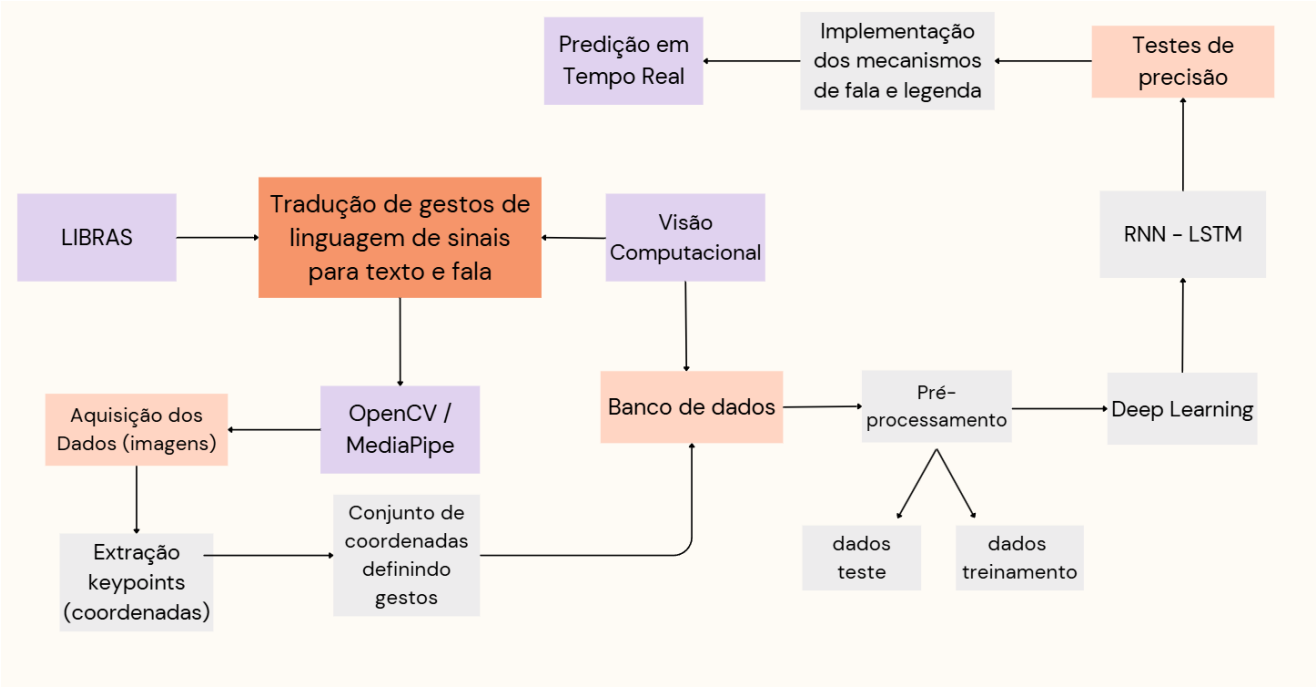


Figura 8 - Fluxo da metodologia de desenvolvimento.

Num primeiro momento, busca-se entender sobre LIBRAS, a fim de saber como funcionam os gestos e o mecanismo da comunicação como um todo. A fim de trabalhar com visão computacional interpretando gestos de LIBRAS, necessita-se de uma base de dados contendo os gestos que serão reconhecidos. A aquisição das imagens se deu utilizando a biblioteca OpenCV do Python juntamente com o *framework* MediaPipe responsável por fazer toda a segmentação. Através das imagens já segmentadas, extraíram-se as coordenadas que delimitam cada gesto presente nos *frames* de captura. Cada gesto da língua de sinais terá seu conjunto de coordenadas definindo o mesmo. Esse compilado de pontos e informações é o que compõem o nosso banco de dados.

Após a aquisição de todos esses conjuntos de informações, parte-se para o pré-processamento, onde os dados são tratados, normalizados e separados em dados de teste e dados de treinamento do modelo.

Utilizou-se *Deep Learning* para desenvolver o modelo em questão, devido à alta complexidade e dimensionalidade dos dados. Optou-se por uma rede neural recorrente na arquitetura LSTM, pois é a única que permite a previsão de séries temporais e interpretar padrões de dados sequenciais.

Por fim, realizam-se testes de precisão no modelo e, uma vez que estejam satisfatórios, parte-se implementar o mecanismo de fala responsável por traduzir para áudio os gestos reconhecidos e, por fim, realizam-se as predições em tempo real.

3.1 Aquisição dos Dados (Imagens)

3.1.1 Utilização da Biblioteca OpenCV

Para a aquisição dos dados visuais, utilizou-se a biblioteca OpenCV (*Open Source Computer Vision Library*), que é amplamente utilizada no contexto de visão computacional. A OpenCV fornece uma infraestrutura eficiente para a captura, processamento e análise de imagens e vídeos, o que a torna ideal para projetos que envolvem detecção e rastreamento de movimentos em tempo real (OPENCV, 2024).

A primeira etapa do processo de aquisição de dados envolve a conexão com a *webcam* do computador. A OpenCV facilita essa conexão através da classe *VideoCapture*, que permite acessar a transmissão em tempo real da câmera e capturar cada *frame* exibido.

Uma vez tendo sido estabelecida a conexão com a *webcam*, cada *frame* da transmissão de vídeo é capturado em tempo real. Esses *frames* são então processados individualmente para extrair informações relevantes. Isso é feito por meio da segmentação dos *Keypoints* presentes em cada imagem, utilizando-se MediaPipe Holistic.

Uma vez que os *frames* são capturados, o *framework* do MediaPipe é utilizado para segmentar e identificar os *keypoints* (pontos de referência) nas imagens. Essa é uma ferramenta bastante robusta pois permite detectar simultaneamente *keypoints* das mãos, face e corpo, proporcionando uma representação completa e integrada dos movimentos.

Utilizou-se uma função chamada "*mediapipe_detection*", responsável por capturar os pontos na imagem. O MediaPipe Holistic não apenas detecta e desenha os *keypoints*, mas também fornece as coordenadas exatas de cada ponto de referência. A função "*draw_landmarks*" é a que vai desenhar os pontos na imagem. Essas coordenadas são fundamentais para a análise subsequente e para o treinamento dos modelos de *Deep Learning* que irão reconhecer e interpretar os sinais em LIBRAS. O resultado dessa captura pode ser visto na Figura 9.

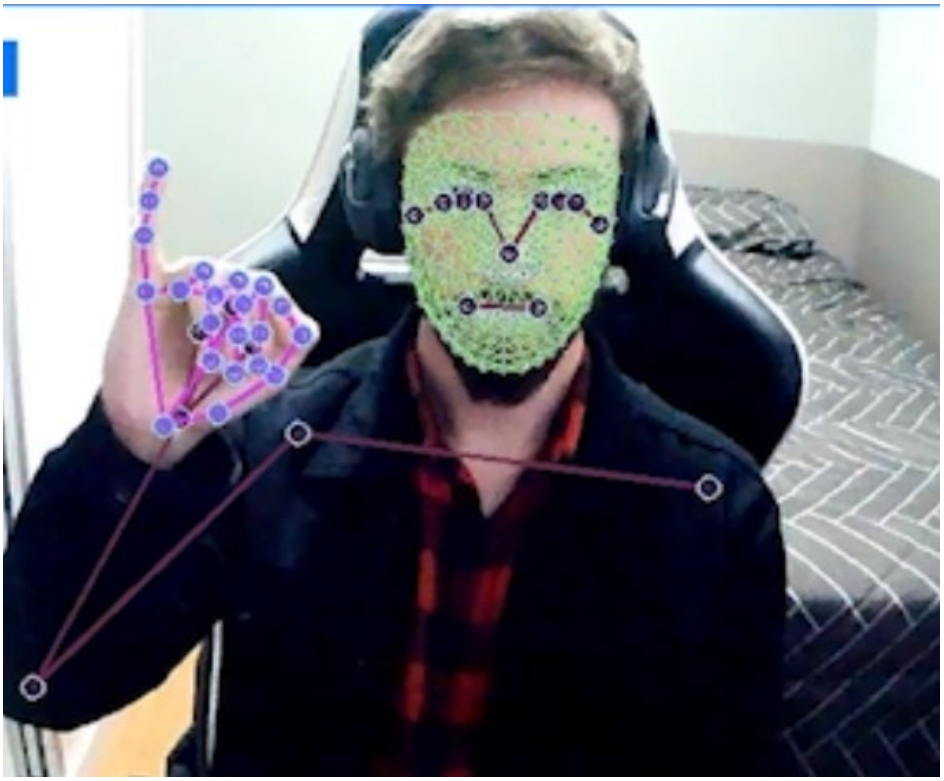


Figura 9 - Captura do gesto.

Os pontos específicos detectados na imagem são chamados de "*landmarks*". O MediaPipe Holistic é capaz de identificar e fornecer as coordenadas de *keypoints* em diferentes partes do corpo, incluindo as mãos, o rosto e o corpo. Para cada tipo de *keypoint*, cria-se um *array* para armazenar as coordenadas, permitindo uma organização e análise estruturada dos dados. Os principais grupos de *keypoints* detectados são:

- ***Landmarks* das Mãos**: Incluem 21 pontos para cada mão, que cobrem desde as articulações até as pontas dos dedos.

- ***Landmarks* do Rosto**: Contêm um conjunto de pontos que mapeiam a estrutura facial, essenciais para capturar expressões e movimentos.
- ***Landmarks* do Corpo**: Incluem pontos como ombros, braços e antebraços, que são críticos para capturar a pose e movimentos corporais.

Cada *array* criado possui um conjunto de coordenadas, sendo 3 coordenadas para os *arrays* de "*landmarks* mãos" e "*landmarks* da face", e 4 coordenadas para "*landmarks* do corpo". A coordenada extra neste caso representa a visibilidade do corpo em si (quantos porcento do corpo está visível na captura). Essa abordagem de captura de dados usando OpenCV e MediaPipe oferece várias vantagens:

1. **Precisão e Confiabilidade**: A detecção precisa dos *keypoints* permite uma análise detalhada dos movimentos.
2. **Eficiência em Tempo Real**: O processamento em tempo real possibilita a tradução simultânea dos sinais, essencial para aplicações práticas.
3. **Facilidade de Integração**: As bibliotecas OpenCV e MediaPipe são altamente integráveis, permitindo uma implementação fluida e eficiente.

Desse modo, a aquisição de dados visuais através da combinação dessas duas ferramentas estabelece uma base sólida para o desenvolvimento do sistema de tradução de LIBRAS. A captura precisa e eficiente dos *keypoints* de mãos, face e corpo possibilita a análise detalhada e a interpretação dos gestos, avançando significativamente na criação de uma ferramenta robusta e prática para a comunidade surda.

3.1.2 Aquisição dos Gestos

Para treinar e avaliar o sistema de reconhecimento de sinais, realizou-se a captura de vídeos para cada gesto. Foram definidos os seguintes gestos dinâmicos: "Oi", "Como", "Amigo", "Obrigado", "Você", "Está", "Bom Dia", "Eu", "Eu Te Amo", "Meu", "Nome", "Estudo", Engenharia", "Elétrica", "UFPR", além do alfabeto de A-Z (26 gestos estáticos). A Figura 10 traz as letras do alfabeto de LIBRAS. Esse processo é fundamental para construir o conjunto de dados que permite ao modelo aprender e generalizar os diferentes gestos realizados por cada indivíduo.

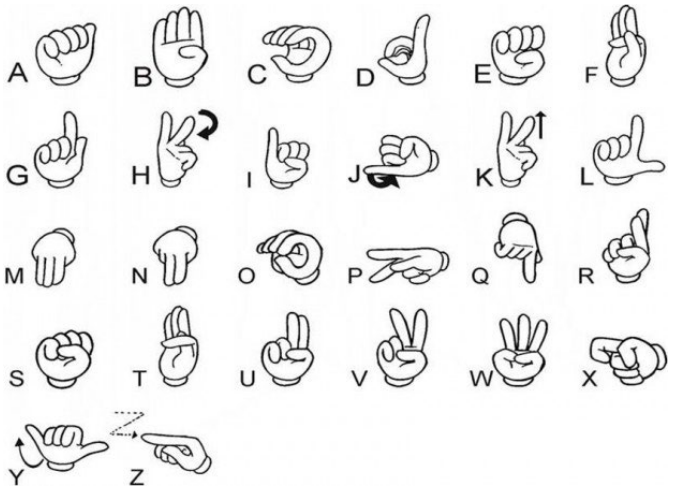


Figura 10 – Alfabeto de Libras (A-Z).

A implementação das letras do alfabeto em LIBRAS foi realizada utilizando um dataset disponibilizado na plataforma Kaggle (SAU, 2023). O mencionado material continha 3000 imagens capturadas para cada letra do alfabeto, contemplando uma ampla variabilidade de condições, incluindo diferentes níveis de iluminação, ângulos de captura e distâncias da mão em relação à câmera. Essa diversidade foi fundamental para garantir que o modelo pudesse generalizar bem para situações reais e não dependesse de condições específicas para realizar as classificações.

O pipeline de processamento incluiu a extração das *landmarks* da mão utilizando o MediaPipe *Hands*, uma biblioteca poderosa para detecção e rastreamento de mãos em tempo real (ROUNDTREE, 2021). Para cada imagem, foram extraídas as coordenadas (x, y) dos 21 pontos anatômicos principais das mãos, formando um vetor de características que serviu como entrada para o modelo de *machine learning*.

Esses dados foram então armazenados, juntamente com seus respectivos rótulos (a letra correspondente), em um arquivo no formato *.pickle* para facilitar a manipulação e o treinamento subsequente. Para o treinamento do modelo, foi escolhido o algoritmo *Random Forest*, um método baseado em múltiplas árvores de decisão que combina os resultados de diferentes árvores para produzir uma classificação robusta. A escolha desse método deve-se à sua eficiência em lidar com dados tabulares e à capacidade de capturar padrões complexos, mesmo sem a necessidade de pré-processamentos extensivos ou ajustes muito detalhados no modelo. Como o problema envolve apenas imagens estáticas e não uma sequência temporal de *frames*, a *Random Forest* mostrou-se uma escolha simples, porém eficaz.

Durante o treinamento, foi necessário ajustar o comprimento dos vetores de características, garantindo que todos os exemplos tivessem um comprimento consistente. Isso foi feito adicionando zeros aos vetores menores, um passo crucial para evitar erros ao alimentar os dados no modelo.

O modelo foi avaliado utilizando uma divisão do conjunto de dados, separando 20% das amostras para teste. Após o treinamento, o modelo atingiu uma precisão de 98,65%, demonstrando sua capacidade de classificar corretamente a maioria das amostras no conjunto de teste. A Figura 11 apresenta o reconhecimento da letra “A” do alfabeto. Enquanto a Figura 12 indica como o sistema consegue detectar os *keypoints* das duas mãos simultaneamente e identificar os gestos sendo executados, classificando corretamente as letras “A” e “B”.

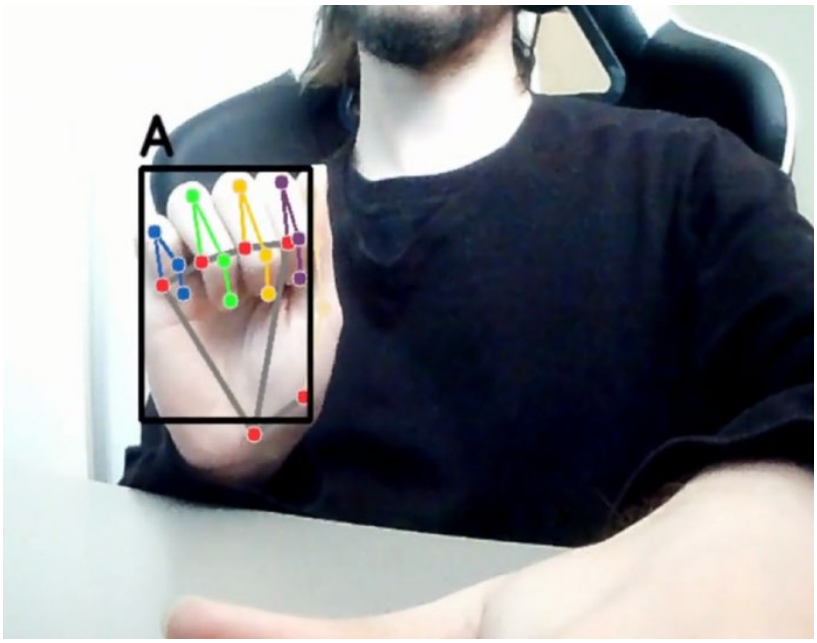


Figura 11 - Modelo identificando a letra A.

Para a inferência em tempo real, foi desenvolvido um script que utiliza a *webcam* para capturar *frames* e processar as *landmarks* da mão em tempo real, novamente utilizando o MediaPipe. As coordenadas extraídas são transformadas no formato necessário para o modelo, e a classificação é exibida diretamente no *feed* de vídeo. Além disso, o sistema desenha as *landmarks* da mão e um retângulo ao redor da região detectada, permitindo que o usuário visualize claramente a área sendo analisada. A interface suporta uma interação intuitiva, onde a letra reconhecida aparece sobre a mão em tempo real.

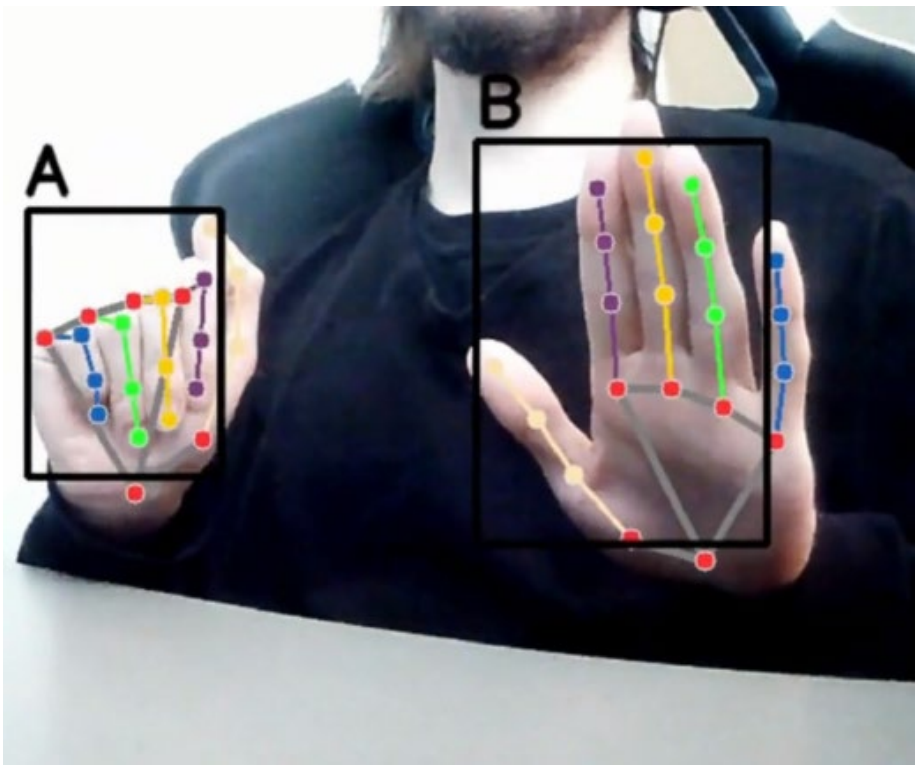


Figura 12 - Capaz de identificar os gestos das duas mãos simultaneamente.

Esse projeto demonstra o potencial de soluções simples e acessíveis para criar sistemas de reconhecimento de sinais em LIBRAS, sendo um primeiro passo para aplicações mais complexas que possam integrar sequências de gestos ou expandir o vocabulário reconhecido.

Já para a metodologia para a aquisição dos gestos dinâmicos é descrita abaixo:

- **Número de Vídeos por Gesto:** Para cada gesto, realizaram-se 100 registros. Este número é suficiente para proporcionar uma diversidade adequada de exemplos, o que ajuda o modelo a reconhecer variações naturais nos movimentos.
- ***Frames* por Vídeo:** Cada vídeo é composto por 60 *frames*. Isso resulta em uma sequência suficiente de imagens para capturar a dinâmica do gesto.
- ***Array* de *Keypoints*:** Cada *frame* é processado para extrair *keypoints*, resultando em 60 *arrays* de *keypoints* por vídeo.
- **FPS Médio:** A taxa de quadros por segundo média durante a captura é de aproximadamente 16 FPS.
- **Duração dos Vídeos:** Cada vídeo de captura tem uma duração de 3,75 segundos, o que é suficiente para capturar a execução completa de cada gesto.

Cada *array* irá conter um conjunto de 1662 *keypoints*, sendo 21 *keypoints* de cada mão multiplicado por 3 coordenadas. O mesmo para os pontos da face, sendo 461 *keypoints* multiplicado

por 3 coordenadas. E, por fim, os 33 pontos do corpo, multiplicados por 4 coordenadas conforme mencionado anteriormente.

O processo de aquisição dos gestos começa com a captura dos *frames* da *webcam* utilizando "*cap.read()*". Cada *frame* capturado é então processado pelo modelo MediaPipe Holistic para detectar *keypoints* das mãos, rosto e corpo. Esses *keypoints* são desenhados no *frame* utilizando a função "*draw_styled_landmarks*". Durante a coleta, são exibidas mensagens informativas na tela para indicar o início da coleta e especificar qual gesto e vídeo estão sendo capturados, proporcionando um *feedback* visual ao operador.

Os *keypoints* extraídos de cada *frame* são salvos em *arrays*, que são armazenados em arquivos ".npy". Isso facilita o posterior treinamento do modelo de *Deep Learning*, já que esses arquivos podem ser facilmente acessados e manipulados. O feed da *webcam*, com os *keypoints* desenhados, é exibido em uma janela chamada *"OpenCV Feed",* permitindo monitorar visualmente o processo de captura. A fim de melhorar a precisão do modelo e torná-lo mais robusto para lidar com uma série de variações dos gestos, realizou-se parte das capturas em horários diferentes do dia, explorando variações dos ângulos de captura e, também, das distâncias de cada gesto em relação a câmera, conforme observado na Figura 13. Dessa forma, o banco de dados para cada gesto apresenta diferentes cenários, reduzindo o risco de enviesamento no modelo e aumentando sua capacidade de generalizações para situações reais.

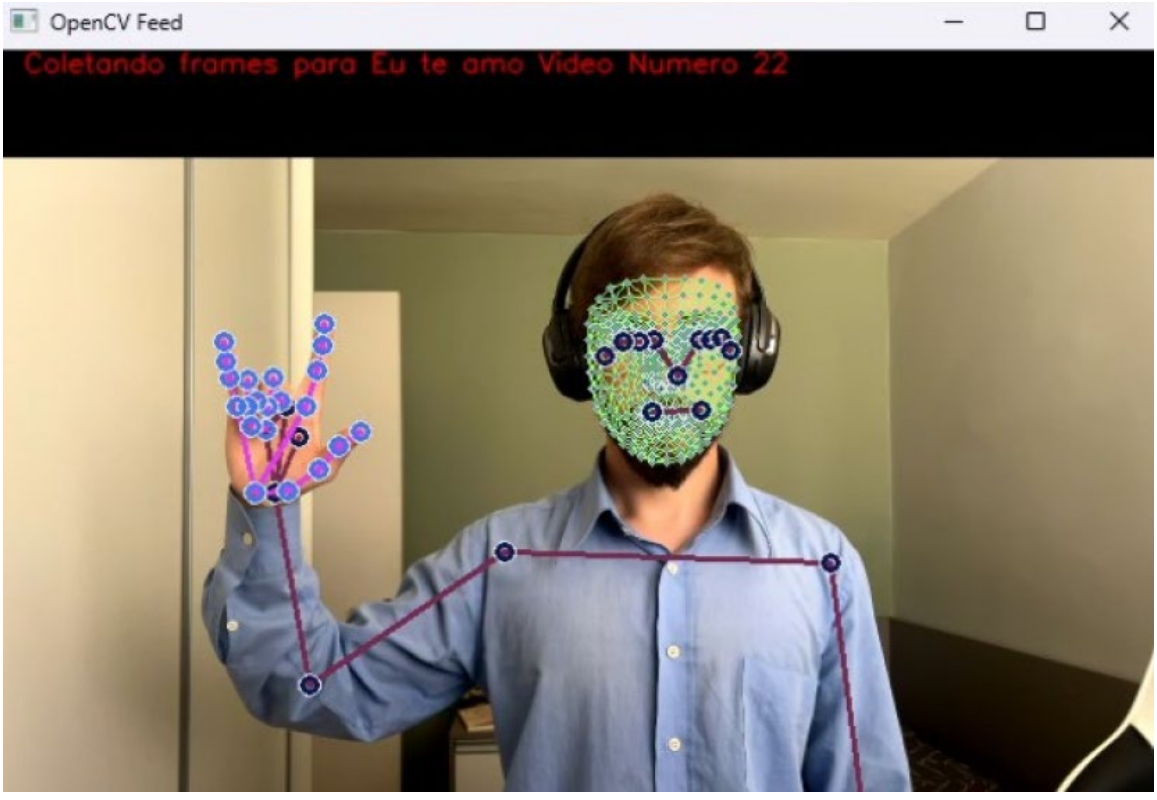


Figura 13 - Coletando frames para gesto "Eu te amo".

3.1.3 Pré-processamento dos dados

Nesta etapa, garante-se que os dados estão no formato adequado para o treinamento do modelo. Primeiramente, cria-se um mapeamento das ações (“Oi”, “Como”, “Amigo”, “Obrigado”, “Você”, “Está”, “Bom Dia”, “Eu”, “Eu Te Amo”, “Meu”, “Nome”, “Estudo”, Engenharia”, “Elétrica”, “UFPR”).

A ideia é mapear para números, utilizando um dicionário onde cada ação é associada a um número inteiro “0” ou “1”, facilitando a categorização durante o treinamento do modelo. Por exemplo, a lista [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0] indica que o gesto reconhecido é “Oi”, enquanto a lista [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0] indicaria “Amigo”.

Cada um dos 60 *frames* dos 100 vídeos é adicionado a uma lista chamada *sequences*, resultando em um total de 1500 vídeos, cada um com 60 *frames* e 1662 *keypoints*. Esses *frames* são lidos a partir dos arquivos .npy gerados na etapa anterior e adicionados à lista *sequences*, enquanto as *labels* correspondentes são adicionadas à lista *labels*. O formato da lista *sequences* será (1500, 60, 1662), indicando 1500 vídeos, cada um com 60 *frames* e 1662 *keypoints* por *frame*.

Com o objetivo de aumentar a precisão do modelo, reduzir a complexidade do treinamento e minimizar os riscos de confusão durante o processo de aprendizado, os *keypoints* da face foram removidos. Essa decisão baseou-se no fato de que, para os gestos presentes no banco de dados, a expressão facial não possui relevância para a correta identificação destes. Os movimentos das mãos e do corpo são os únicos aspectos cruciais para a classificação dos gestos. Portanto, foram mantidos exclusivamente os *keypoints* das mãos e do corpo, o que resultou em uma significativa redução no número de características processadas, sem comprometer a qualidade da análise e garantindo maior eficiência no treinamento do modelo, como será apresentado ao longo deste trabalho.

Converteu-se, assim, as listas *sequences* e *labels* em *arrays* NumPy, o que facilita o manuseio dos dados e é necessário para a maioria das operações de aprendizado de máquina.

Aqui, X representa as features (os *keypoints*) e y as *labels*, que são convertidas para uma forma categórica (*one-hot encoding*) utilizando a função *to_categorical*. Finalmente, divide-se os dados em conjuntos de treinamento e teste. Utilizamos a função *train_test_split* da biblioteca scikit-learn para realizar essa divisão, destinando 30% dos dados para teste e 70% para treinamento. Isso resulta, para cada gesto, em 70 vídeos para treinamento e 30 vídeos para teste. Essa divisão é essencial para avaliar o desempenho do modelo em dados que ele não viu durante o treinamento, permitindo medir sua capacidade de generalização.

3.1.4 Criando e Treinando o Modelo de Rede Neutral LSTM

Nesta etapa, aborda-se a criação e o treinamento de um modelo de rede neural LSTM, uma arquitetura de rede neural recorrente adequada para tarefas de aprendizado sequencial, como a detecção de gestos a partir de vídeos.

Primeiramente, define-se a arquitetura do modelo LSTM. A estrutura do modelo inclui múltiplas camadas LSTM seguidas de camadas densas (*fully connected*). A escolha das camadas e o número de unidades em cada camada foram definidos de maneira empírica, testando os resultados de precisão do modelo com diferentes números de épocas. Cada época indica quantas vezes o modelo recalculou os pesos. O modelo é compilado com a função de perda *categorical_crossentropy*, apropriada para problemas de classificação multiclasse, e o otimizador Adam, que é eficaz para esse tipo de problema.

Épocas	Acurácia
100	0,6
200	0,8
300	0,8
400	0,8
500	0,4
600	0,8
700	0,6
800	0,4
900	0,4
1000	0,4
1100	0,8
1200	0,6
1300	1,0
1400	1,0
1500	0,6

Tabela 1 - Tabela com épocas e respectivas precisões. Fonte: O autor (2024)

O valor escolhido para o treinamento do modelo foi de 1100 épocas, devido à alta precisão obtida, correspondente a 0,8. Observamos que essa precisão também foi atingida em 200, 300, 400 e 600 épocas. No entanto, optamos por 1100 épocas porque um número maior de épocas permite mais treinamentos e recálculos de pesos, sugerindo uma melhor generalização dos dados. Esse processo torna o modelo mais robusto para lidar com novos exemplos que surgirão durante os testes em tempo real.

Embora tenhamos alcançado uma exatidão de 1 em 1300 e 1400 épocas, isso indicou que o

modelo sofreu de *overfitting*. Isso ocorre quando o modelo se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar para novos exemplos. Isso é evidenciado pelo fato de que, ao alcançar 100% de precisão, o modelo deixou de aprender e generalizar os resultados, comprometendo seu desempenho em situações reais. Portanto, a escolha de 1100 épocas oferece um equilíbrio adequado entre treinamento e generalização, evitando os problemas associados ao *overfitting*.

O modelo é treinado usando os dados de treinamento (*X_train* e *y_train*). Utilizou-se a função *model.fit* para treinar o modelo, especificando o número de épocas (1100) e *callbacks*, que permitem monitorar o treinamento e ajustar automaticamente a taxa de aprendizado ou interromper o treinamento com base em certas condições. Durante o treinamento, a precisão categórica e a perda (*loss*) são exibidas para cada época, permitindo acompanhar o desempenho do modelo em cada estágio do treinamento. Observamos a evolução desses valores ao longo das épocas (Figura 14), o que ajuda a identificar possíveis problemas como *overfitting*.

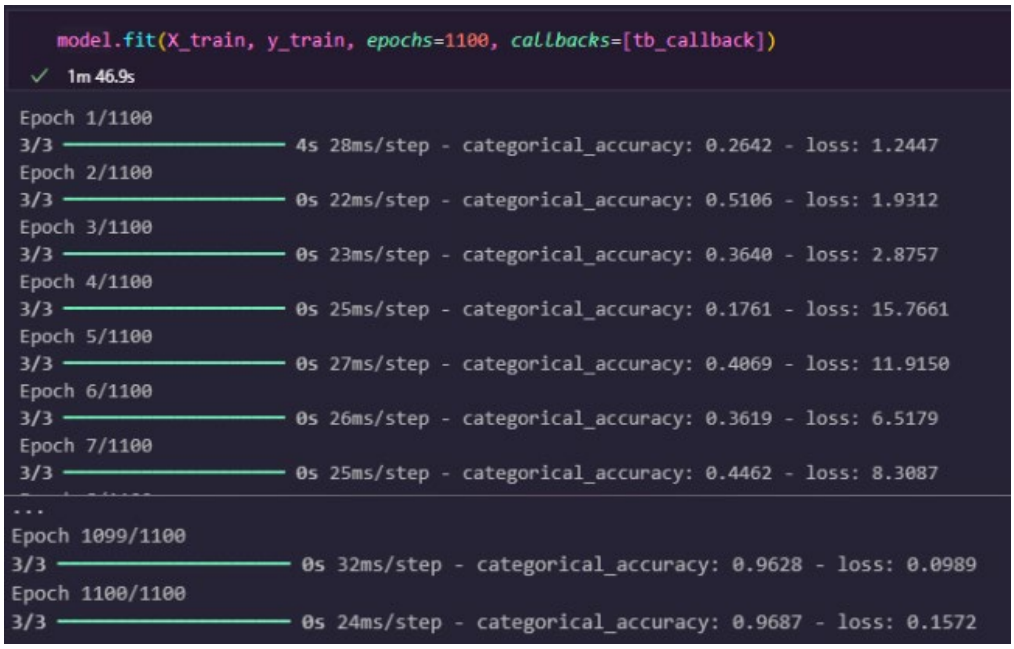


Figura 14 - Precisão do modelo de acordo com cada época.

Por fim, foi criada uma matriz de confusão para avaliar o desempenho do modelo. A matriz de confusão permite visualizar a performance do algoritmo de classificação, mostrando o número de previsões corretas e incorretas para cada classe (SULLIVAN, 2017). A precisão do modelo foi calculada utilizando a equação:

$$Precisão = \frac{Número\ de\ previsões\ corretas}{Número\ total\ de\ amostras}$$

Foram realizados 10 testes independentes do modelo para obter uma estimativa mais robusta do seu desempenho. A média de precisão obtida foi de 0,965 (96,5%), com um desvio padrão de 0,11.

Essa média indica que o modelo tem um desempenho consistente, enquanto o desvio padrão fornece uma medida da variabilidade dos resultados entre diferentes execuções do teste. Através dessa análise, verificamos que o modelo é capaz de generalizar bem para novos dados, mantendo uma boa performance nos testes realizados.

É importante destacar que a precisão de 0.8 (80%) apresentada pelo modelo na época 1100 indicada na Tabela 1, refere-se a precisão calculada no conjunto de treinamento em cada época. Durante esse processo, o modelo está aprendendo a ajustar os pesos da rede para minimizar a função de perda. Desse modo, esse número reflete o desempenho no conjunto de treinamento usado para monitorar o progresso do modelo. Já a precisão de 96,5% é calculada após o treinamento completo, utilizando um conjunto de teste independente, o qual contém dados nunca vistos pelo modelo durante seu processo de treino. Ou seja, reflete a capacidade do algoritmo de generalizar novos dados. Esse aumento na precisão do modelo sugere que ele conseguiu generalizar melhor ao lidar com dados reais, não apenas com as informações usadas para ajustar os pesos na rede neural.

3.1.5 Mecanismo de voz

O sistema desenvolvido para detecção de gestos em tempo real conta com um mecanismo de voz integrado, implementado utilizando a biblioteca *pyttsx3*, que permite a síntese de fala diretamente no dispositivo sem a necessidade de conexão à internet. Esse recurso foi projetado para oferecer *feedback* auditivo ao usuário, tornando o sistema mais acessível e interativo. A configuração inicial do mecanismo de voz incluiu ajustes na velocidade da fala, definida em 150 palavras por minuto, e no volume, configurado para 90% da capacidade máxima. Esses valores foram escolhidos para garantir uma comunicação clara, compreensível e adequada a diferentes ambientes de uso.

Para garantir a fluidez no processamento do sistema, o mecanismo de voz foi implementado de forma assíncrona. Isso foi alcançado por meio da biblioteca *threading*, que permite executar a síntese de fala em uma *thread* separada do fluxo principal. Assim, o sistema consegue continuar capturando *frames* e processando gestos em tempo real, mesmo enquanto a reprodução da fala ocorre. Essa abordagem foi fundamental para evitar atrasos ou interrupções que comprometessem a experiência do usuário.

A integração do mecanismo de voz ao *pipeline* de detecção ocorre após a validação dos gestos detectados. Quando um gesto é confirmado como consistente, ou seja, detectado consecutivamente por cinco *frames*, ele é adicionado à sentença dinâmica exibida na tela. Antes de ser reproduzido em voz, o gesto passa por uma etapa de ajuste textual, implementada por meio da função *adjust_caption*. Essa função realiza correções específicas para garantir que as palavras estejam contextualizadas e adequadas ao significado do gesto, como, por exemplo, transformar “UFPR” em “na UFPR” ou

ajustar “Esta” para “Está?”. Essa personalização contribui para que o *feedback* auditivo seja mais natural e alinhado às necessidades de comunicação em LIBRAS.

Os benefícios do mecanismo de voz incluem maior acessibilidade, interatividade e versatilidade. O *feedback* auditivo torna o sistema mais inclusivo, especialmente para usuários que preferem ou necessitam de comunicação por áudio em vez de texto. Além disso, a combinação de *feedback* visual e auditivo aumenta o engajamento e a clareza durante o uso do sistema. Essa funcionalidade é especialmente útil em aplicações educacionais, como o ensino de LIBRAS, ou em contextos de comunicação assistida, onde a rapidez e a clareza na transmissão de informações são essenciais.

Durante os testes do sistema, o mecanismo de voz mostrou-se eficiente e responsivo. A reprodução em voz ocorreu de forma clara, com um tempo médio de resposta insignificante entre a detecção do gesto e a reprodução auditiva. A implementação assíncrona garantiu que o *feedback* por voz não interferisse no processamento de vídeo ou no desempenho geral do sistema, mesmo em condições de uso intensivo. Em resumo, a implementação do mecanismo de voz foi bem-sucedida, demonstrando seu potencial para melhorar a acessibilidade e a usabilidade do sistema de detecção de gestos. Sua integração ao fluxo principal e a personalização das legendas ajustadas para LIBRAS foram pontos-chave para o sucesso dessa funcionalidade, que se destaca como uma solução robusta e prática para aplicações em diversos contextos.

3.1.6 Mecanismo de texto (legendas)

No sistema desenvolvido para a detecção de gestos em tempo real, a exibição das legendas e a visualização das probabilidades associadas aos gestos detectados desempenham o principal papel na interação com o usuário. Essas funcionalidades foram implementadas de forma a garantir clareza e fluidez na apresentação das informações, permitindo que o usuário acompanhe o processamento do sistema de maneira intuitiva e visualmente agradável.

As legendas dos gestos detectados são exibidas em tempo real no topo da tela, integradas ao *feed* de vídeo capturado pela *webcam*. A cada gesto validado, ele é adicionado a uma sentença dinâmica, que exibe os últimos cinco gestos detectados. Essa abordagem tem como objetivo fornecer um contexto contínuo da comunicação, facilitando o entendimento do usuário sobre os gestos reconhecidos pelo sistema. Para implementar essa funcionalidade, foi utilizada a biblioteca OpenCV, que desenha uma faixa no topo do vídeo para exibir o texto da legenda. Essa faixa é destacada por uma cor de fundo fixa, garantindo contraste com o conteúdo do *feed* de vídeo. O texto é renderizado utilizando uma fonte legível e dimensionada adequadamente para que seja compreensível mesmo em telas menores. Além disso, a legenda exibida passa por uma função de ajuste textual, chamada *adjust_caption*, que corrige palavras e expressões para torná-las mais apropriadas ao contexto da

LIBRAS. Essa função inclui, por exemplo, ajustes como transformar "UFPR" em "na UFPR" ou "Esta" em "Esta?", tornando a legenda mais clara e contextualizada para o usuário.

Além das legendas, o sistema inclui uma funcionalidade de visualização de probabilidades associadas a cada gesto detectado. A função *prob_viz* foi implementada para apresentar graficamente a confiança do modelo na classificação dos gestos. Essa visualização é feita por meio de barras horizontais desenhadas no *feed* de vídeo, onde a largura de cada barra é proporcional à probabilidade calculada pelo modelo para cada classe de gesto.

Cada barra é desenhada em uma cor destacada, e o texto indicando o gesto e sua probabilidade percentual é exibido ao lado da barra correspondente, como pode ser observado na Figura 15. A função organiza as barras verticalmente, começando no topo da tela, e atualiza os valores em tempo real à medida que novos *frames* são processados. Isso permite que o usuário visualize não apenas o gesto mais provável, mas também as probabilidades de outras classes, oferecendo uma percepção mais detalhada do comportamento do modelo.

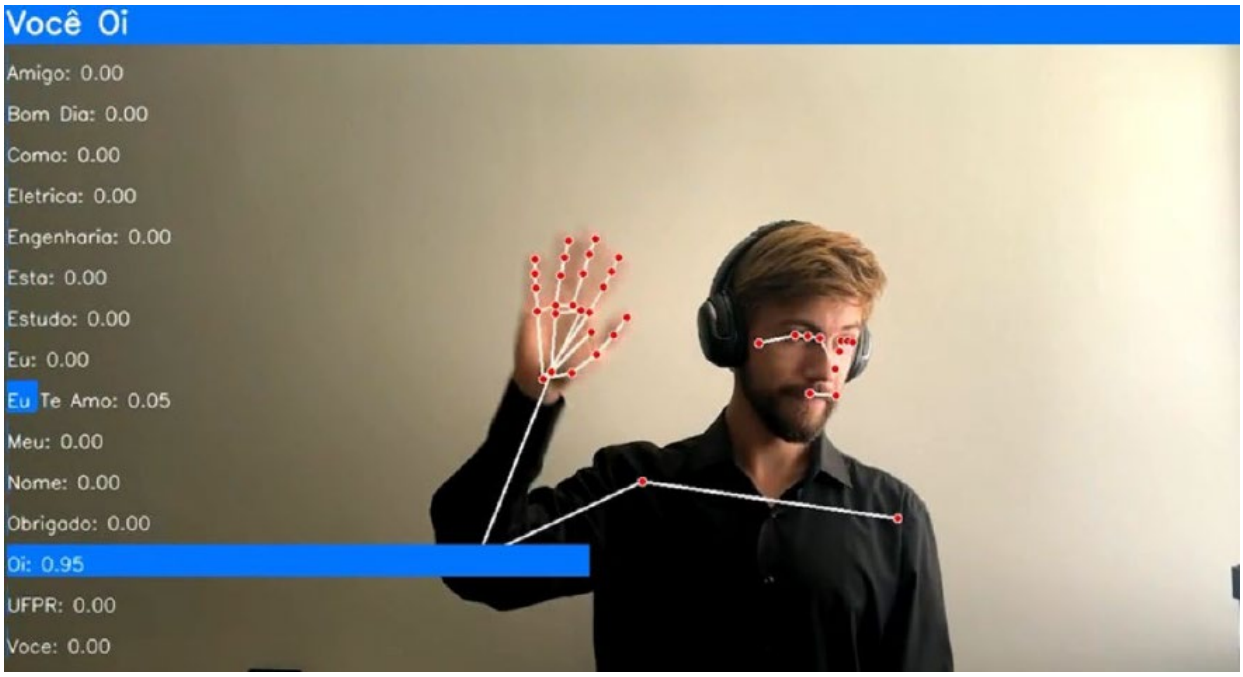


Figura 15 - Reconhecimento dos gestos indicando a legenda, lista de sinais e a função de probabilidade para cada.

4 Resultados

Os resultados obtidos neste trabalho não apenas atingiram os objetivos propostos, mas também destacaram o potencial transformador da aplicação de tecnologias avançadas no campo da inclusão social. A seguir, são detalhados o alcance de cada objetivo e os impactos observados ao longo do desenvolvimento.

4.1. Captura e Interpretação de Gestos com LIBRAS

O sistema desenvolvido demonstrou ótima capacidade para capturar e interpretar gestos de LIBRAS, atingindo alta precisão e eficiência, conforme indicado pelos resultados de teste de exatidão de 96,5% para gestos dinâmicos e 98,65% para gestos estáticos. Utilizando a biblioteca OpenCV e o *framework* MediaPipe Holistic, foi possível identificar e mapear os *keypoints* de mãos, rosto e corpo, criando uma base sólida de dados estruturados para análise. Cada gesto, dinâmico ou estático, foi representado por *arrays* contendo coordenadas tridimensionais (x, y, z), permitindo a análise detalhada dos movimentos.

Esse processo foi testado com um vocabulário de 15 gestos dinâmicos, como "Eu te amo" e "Como", além das 26 letras do alfabeto em LIBRAS. A detecção em tempo real demonstrou excelente capacidade de adaptação a variações nos ângulos de captura, iluminação e distância, como pode ser observado na Figura 16, que ilustra diferentes *frames* de captura durante a criação do banco de dados para o gesto “Oi”.

Figura 16 - Frames de capturas em diferentes condições de ângulo, iluminação e posição.

4.2. Classificação de Gestos Usando *Machine learning* e Redes Neurais

A classificação dos gestos foi outro marco importante deste trabalho, consolidando o uso de algoritmos de *Machine Learning* e *Deep Learning*. Para os gestos estáticos, como as letras do alfabeto, foi empregado o modelo *Random Forest,* que atingiu uma precisão de 98,65% no conjunto de teste. Sua escolha foi baseada na eficiência em lidar com dados tabulares e na capacidade de capturar padrões complexos.

Já para os gestos dinâmicos, a arquitetura LSTM provou ser a abordagem ideal. Por lidar com sequências temporais, a LSTM conseguiu interpretar com eficácia padrões dinâmicos dos movimentos, atingindo uma precisão de 96,5% nos testes.

4.3. Tradução Automática em Tempo Real

A tradução automática em tempo real foi um dos aspectos mais importantes deste trabalho. O sistema não apenas detecta e classifica os gestos em tempo real, mas também os converte imediatamente em texto, exibindo as palavras reconhecidas na tela de forma dinâmica. Isso possibilita a formação de frases completas, como “Oi como você está?” mostrado na Figura 17, promovendo uma comunicação fluida entre os interlocutores.

Figura 17 - Sequência de Capturas gerando a frase "Oi Como Você Está?".

Durante os testes, o sistema mostrou-se capaz de acompanhar gestos contínuos sem perda significativa de desempenho, mesmo em situações de maior complexidade, como quando os gestos eram realizados em rápida sucessão. Essa fluidez na tradução reforça a aplicabilidade prática do sistema em contextos reais, como salas de aula, ambientes corporativos e atendimentos ao público.

4.4. Integração de Saída para Texto e Fala

A integração com a biblioteca *pyttsx3* para síntese de fala amplia significativamente a acessibilidade do sistema. Ao converter automaticamente o texto gerado em áudio. Esse recurso foi projetado de forma assíncrona, permitindo que a conversão em voz ocorresse sem prejudicar o desempenho do sistema na detecção dos gestos.

A escolha por uma solução local, sem dependência de conexão à internet, reforçou a robustez do sistema e garantiu sua usabilidade em ambientes com baixa conectividade, ampliando o alcance e a aplicabilidade do projeto.

4.5. Promoção de Inclusão e Acessibilidade

Por fim, o principal objetivo deste trabalho, promover a inclusão social e a acessibilidade, foi plenamente alcançado, conforme demonstrado pelos resultados obtidos, que indicaram precisão de 96,5% para gestos dinâmicos e 98,65% para sinais estáticos de maneira satisfatória. O sistema desenvolvido permite que pessoas surdas e ouvintes se comuniquem de maneira eficiente e natural, superando barreiras históricas impostas pela ausência de soluções práticas e acessíveis para tradução de LIBRAS.

Além disso, a interface intuitiva, que combina *feedback* visual (legendas dinâmicas) e auditivo (mecanismo de fala), torna o sistema uma ferramenta inclusiva para diferentes perfis de usuários. Assim, a combinação de texto em tempo real e síntese de voz proporciona um ambiente de comunicação adaptável e eficaz, contribuindo diretamente para a democratização do acesso à comunicação.

5 Conclusão

O desenvolvimento deste trabalho de conclusão de curso proporcionou uma oportunidade significativa para explorar e aplicar tecnologias avançadas no contexto da inclusão social e da acessibilidade. Por meio da criação de um sistema capaz de traduzir gestos da Língua Brasileira de Sinais para texto e fala em tempo real, foram alcançados resultados que demonstram a relevância técnica, social e acadêmica da pesquisa.

A partir de uma abordagem interdisciplinar que combinou visão computacional, *Machine Learning* e processamento de linguagem natural, foi possível integrar bibliotecas e *frameworks* como

OpenCV, MediaPipe, pyttsx3 e TensorFlow, resultando em um sistema funcional e de alta precisão. A utilização de modelos como *Random Forest* para gestos estáticos e redes neurais recorrentes LSTM para gestos dinâmicos comprovou a capacidade dessas tecnologias de lidar com os desafios inerentes ao reconhecimento de sinais em LIBRAS, como a variabilidade nos movimentos, diferentes condições de captura e a necessidade de interpretar padrões sequenciais complexos.

Os resultados obtidos validaram o objetivo inicial de criar um sistema acessível e prático. Com precisão de 98,65% para gestos estáticos e 96,5% para gestos dinâmicos, o sistema se mostrou robusto e capaz de generalizar para novos dados, mantendo desempenho consistente em testes em tempo real. Além disso, a integração com mecanismos de saída, como texto dinâmico e síntese de voz, garantiu uma experiência de comunicação fluida entre surdos e ouvintes, destacando a aplicabilidade do sistema em diferentes contextos, como ambientes educacionais, corporativos e de atendimento ao público.

A contribuição deste trabalho ultrapassa o âmbito técnico, pois reforça o papel da tecnologia como ferramenta de transformação social. O projeto atende a uma demanda crescente por soluções inclusivas no Brasil, onde a LIBRAS é reconhecida como meio legal de comunicação, mas ainda enfrenta barreiras práticas para sua disseminação. Ao criar uma ponte entre a comunidade surda e os ouvintes, este sistema promove o diálogo, a igualdade de oportunidades e a inclusão.

O trabalho também oferece valor acadêmico ao explorar métodos híbridos para classificação de gestos, avaliando suas vantagens e limitações, e ao apresentar um *pipeline* completo que pode ser replicado e expandido em pesquisas futuras. Este modelo, que inclui captura, pré-processamento, treinamento, avaliação de modelos e integração de saída, é um guia prático para outros estudos na área de visão computacional e aprendizado de máquina aplicados à língua de sinais.

Contudo, reconhece-se que este é apenas o início de um extenso caminho repleto de desafios. Entre as limitações do projeto estão a necessidade de ampliar o vocabulário de gestos reconhecidos, incluir variações regionais da LIBRAS e adaptar o sistema para dispositivos móveis, o que aumentaria sua acessibilidade e aplicabilidade prática. Além disso, futuras iterações poderiam explorar arquiteturas mais avançadas de *deep learning* para melhorar ainda mais a precisão e a capacidade de generalização do modelo.

Em suma, este trabalho representa uma contribuição significativa para o avanço da tecnologia assistiva no Brasil, alinhando-se aos princípios de acessibilidade universal e inclusão social. Através dos resultados obtidos e da metodologia desenvolvida, espera-se não apenas beneficiar a comunidade surda, mas também inspirar outros pesquisadores e profissionais a continuar desenvolvendo soluções que promovam a igualdade e a integração. Este projeto reafirma que a inovação tecnológica, quando aliada a objetivos sociais, pode transformar vidas e construir um futuro mais inclusivo e conectado.

6 Referências Bibliográficas

1. QUADROS, R. M. de; KARNOPP, L. *Língua de sinais brasileira: estrutura e processos*. São Paulo: Editora XYZ, 2004.
2. BRASIL. Decreto nº 5.626, de 22 de dezembro de 2005. Regulamenta a Lei nº 10.436, de 24 de abril de 2002, que reconhece a Língua Brasileira de Sinais – LIBRAS – como meio legal de comunicação e expressão. *Diário Oficial da União*, Brasília, DF, 23 dez. 2005.
3. BRAGG, D.; KOLLER, O.; BELLARD, M.; BERKE, L.; BOWDEN, R. Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective**.** *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19)*. Disponível em: https://dl.acm.org/doi/10.1145/3308561.3353774. Acesso em: 3 jun. 2024.
4. VLIBRAS. Tecnologias que ajudam surdos no dia a dia. Disponível em: https://www.vlibras.com.br/tecnologiass-que-ajudam-surdos-no-dia-a-dia/. Acesso em: 03 dez. 2024.
5. SAU, Debashish**.** American Sign Language Alphabet Dataset. Kaggle, 2023. Disponível em: https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset. Acesso em: 18 out. 2024.
6. MEDIAPIPE SOLUTIONS GUIDE**.** Real-time Media Processing *Framework*. Disponível em: https://ai.google.dev/edge/mediapipe/solutions/guide. Acesso em: 3 dez. 2024.
7. ROUNDTREE, D. *Hands-on MediaPipe: Build Real-Time Applications with MediaPipe*. Packt Publishing, 2021.
8. SULLIVAN, D. J. *Confusion matrix: understanding its meaning and purpose*. Wiley, 2017.
9. LE CUN, Y.; BENGIO, Y.; HINTON, G. *Deep learning*. *Nature*, v. 521, n. 7553, p. 436-444, 2015.
10. HOCHREITER, S.; SCHMIDHUBER, J. *Long short-term memory*. *Neural Computation*, v. 9, n. 8, p. 1735-1780, 1997.
11. CORNELL UNIVERSITY**.** Lecture 1: Filtering and Edge Detection. Disponível em: https://www.cs.cornell.edu/courses/cs5670/2017sp/lectures/lec01_filter.pdf. Acesso em: 12 dez. 2024.

12. KUBRUSLY, J. Curso de Machine Learning - Redes Neurais. Disponível em: https://bookdown.org/jessicakubrusly/curso_de_machine_learning/_book/07-redesneurais.html. Acesso em: 12 dez. 2024.
13. PATERNO, U. Passo 1: Aprender o alfabeto manual e os números. Disponível em: https://professoruesleipaterno.wordpress.com/2021/10/27/passo-1-aprender-o-alfabeto-manual-e-os-numeros/. Acesso em: 12 dez. 2024.