Universidade Estadual de Londrina

GABRIEL ÂNGELO PEREZ GASPARINI SABAUDO

RECONHECIMENTO DE SINAIS DE LIBRAS EM IMAGENS DIGITAIS

LONDRINA

2023

GABRIEL ÂNGELO PEREZ GASPARINI SABAUDO

RECONHECIMENTO DE SINAIS DE LIBRAS EM IMAGENS DIGITAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Vitor Valerio de Souza Campos

Coorientador: Prof. Dr. Guilherme Pina Cardim

LONDRINA

2023

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Sabaudo, Gabriel Ângelo Perez Gasparini.
RECONHECIMENTO DE SINAIS DE LIBRAS EM IMAGENS DIGITAIS / Gabriel Ângelo Perez Gasparini Sabaudo. - Londrina, 2023.
45 f. : il.

Orientador: Vitor Valerio de Souza Campos.
Coorientador: Guilherme Pina Cardim.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, Graduação em Ciência da Computação, 2023.
Inclui bibliografia.

1. Aprendizado de máquina - TCC. 2. Libras - TCC. 3. Visão Computacional - TCC. 4. Classificação - TCC. I. Souza Campos, Vitor Valerio de. II. Pina Cardim, Guilherme . III. Universidade Estadual de Londrina. Centro de Ciências Exatas. Graduação em Ciência da Computação. IV. Título.

CDU 519

GABRIEL ÂNGELO PEREZ GASPARINI SABAUDO

RECONHECIMENTO DE SINAIS DE LIBRAS EM IMAGENS DIGITAIS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Vitor Valerio de Souza Campos
Universidade Estadual de Londrina

Prof. Dr. Guilherme Pina Cardim
Universidade Estadual Paulista - UNESP

Prof. Dr. Evandro Bacarin
Universidade Estadual de Londrina - UEL

Londrina, 29 de Maio de 2023.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

AGRADECIMENTOS

Agradeço ao meu professor orientador Vitor e ao meu professor coorientador, Guilherme, por toda a ajuda e ensinamentos transmitidos a mim, e por sempre estarem disponíveis para sanar minhas dúvidas e esclarecê-las. Também agradeço ao Silas Luiz e ao Jauvane Oliveira pela disponibilização do conjunto de dados que me ajudou em minha pesquisa neste trabalho.

Agradeço também à todos os professores do Departamento de Computação que me auxiliaram até aqui, e que me forneceram a bagagem necessária para encerrar esta etapa na minha vida e começar um novo ciclo.

Agradeço aos meus amigos e colegas de sala, e especialmente meus companheiros para a vida Claudio Bento, Guilherme Silva, Vinícius Cesar e Felipe Barusso, por toda a experiência juntos, pelos momentos de risadas e por sempre estarem presentes nos momentos em que precisei.

Por fim, agradeço à minha família por terem sido a base da minha persistência, da minha formação e por todo o apoio em minhas decisões.

"Todas as verdades são fáceis de entender uma vez que são descobertas; o objetivo é descobri-las.
(Galileu Galilei)

SABAUDO, G. A.. **Reconhecimento de Sinais de Libras em Imagens Digitais**. 2023. 45f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2023.

RESUMO

A análise e detecção de gestos em imagens digitais se tornaram uma das grandes áreas de estudo e desenvolvimento no tocante ao uso de tecnologias que utilizam da imagem ou vídeo, sua principal forma de operação. A relevância da temática justifica-se pela importância da Libras no contexto brasileiro, como meio de comunicação e inclusão social para pessoas com deficiência auditiva. Partindo deste pressuposto, através de abordagens em conceitos de detecção em imagens digitais e um modelo de aprendizado de máquina (*machine learning*) utilizando uma rede neural, o objetivo deste trabalho é construir um modelo que possa realizar a classificação de sinais de Libras.

Palavras-chave: Imagem Digital. Detecção de Sinais de Libras. Rede Neural. Machine Learning. Reconhecimento de Gestos

SABAUDO, G. A.. **Libras Signal Recognizement in Digital Images**. 2023. 45p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2023.

ABSTRACT

The analysis and detection of gestures in digital images have become one of the major areas of study and development in the use of technologies that rely on image or video, their main mode of operation. The relevance of the subject is justified by the importance of Libras in the Brazilian context as a means of communication and social inclusion for people with hearing disabilities. Based on this assumption, through approaches in digital image detection concepts and a machine learning model using a neural network, the aim of this study is to build a model that can perform the classification of Libras signs.

Keywords: Digital Image. Libras Sign Detection. Neural Network. Machine Learning. Gesture Recognition

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RGB	Sistema de espaço de cores (*Red, Green, Blue*)
YIQ	Sistema de espaço de cores
HSV	Sistema de espaço de cores *(Hue, Saturation, Value*)
MLP	*Multi-Layer Perceptron*
KNN	*K-Nearest-Neighbors*
RNA	Rede Neural Artificial
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, o mercado procura atender necessidades para vários tipos de problemas, porém nem sempre todos possuem a atenção necessária. O mundo está interligado por soluções que visam praticidade, essencialmente pelo fato da maioria das pessoas possuírem um celular, por exemplo, o qual pode ser um facilitador para ter acesso a tais aplicações em meio ao cotidiano.

Estes fatores se firmam em relação à deficientes auditivos ao ensino da Libras. A dificuldade relacionada a comunicação com outras pessoas hoje pode ser superada, por exemplo, pelo uso de aplicativos que ajudam no ensino de sinais gestuais. A Lingua Brasileira de Sinais é reconhecida como meio legal de comunicação e expressão no país, e por este motivo engloba um núcleo muito grande de usuários [15]. A importância da Libras para essas pessoas é altíssima principalmente por fazer parte da educação deste grupo e por formar sua participação na sociedade.

Infelizmente, por muitas vezes as pessoas surdas não possuem condições mínimas de atendimento. Em repartições públicas, hospitais, lojas e locais adaptados que lidam com questões de acessibilidade, raramente há alguém preparado para atendê-las [16]. Partindo deste problema, as soluções relacionadas ao campo da visão computacional baseadas no reconhecimento de gestos se alicerçam muito bem para situações dessa natureza.

A partir da importância da Linguagem Brasileira de Sinais no contexto social e inclusivo, este trabalho descreve um método baseado na utilização de uma rede neural aplicada em imagens pré selecionadas, onde será realizado o reconhecimento de cada sinal de Libras detectado e a classificação dos mesmos. A vantagem do uso de uma rede neural é seu poder de aprendizado, o que pode garantir resultados finais com maior precisão e eficiência.

2 FUNDAMENTAÇÃO TEÓRICA

A visão computacional compreende um grande número de artigos relacionados ao reconhecimento de sinais de Libras ou detecção de gestos em geral, e por este motivo este trabalho revisou uma parcela de alguns destes trabalhos, os quais serão mencionados nesta seção.

Nas subseções abaixo será feita uma breve introdução aos conceitos que englobam imagens digitais, métodos de reconhecimento e *machine learning*.

2.1 Imagem Digital

Uma imagem pode ser definida por uma função bidimensional *f (x,y)*, onde x e y são coordenadas espaciais, e a amplitude de *f* em qualquer par *(x,y)* é chamada de nível de cinza naquele ponto da imagem. Quando x, y, e os valores de intensidade de f são finitos e discretos, dizemos que a imagem é uma imagem digital. De acordo com a Figura 1, os valores de x e y correspondem às linhas e colunas, que podem ser representadas através de uma matriz que envolve a imagem. Essas posições são chamadas de pixels de uma imagem [17].

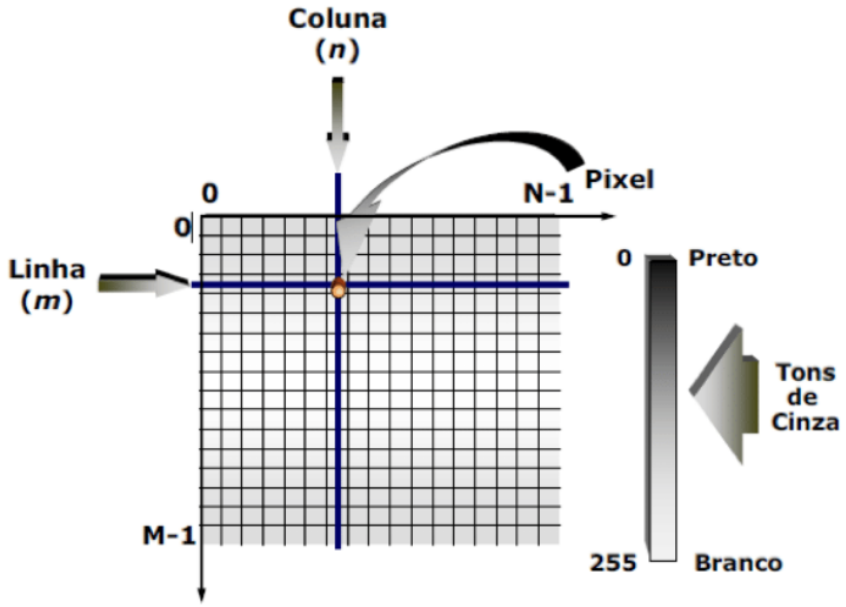

Figura 1 – Representação da matriz de Pixels, retirado de [1]

Analisando a Figura 2, as imagens digitais podem ser coloridas ou em níveis de cinza. No sistema RGB (muito utilizado em monitores, televisões e *smartphones*), por

exemplo, as imagens coloridas assumem três canais que são os canais R (*Red*), G (*Green*) e B (*Blue*), onde cada canal é representado por uma matriz [17].

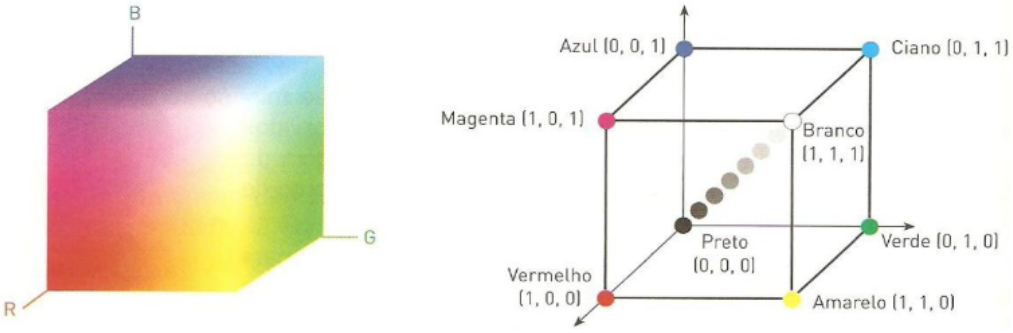

Figura 2 – Modelo RGB de cores, retirado de [2]

A manipulação de uma imagem digital (processo conhecido como processamento digital de imagens) consiste em manipular uma imagem (ou dado) por computador, de certa maneira que a entrada e a saída do processo são imagens. O grande objetivo do mesmo consiste em melhorar o aspecto visual de certas feições estruturais, de certa maneira que o analista consiga melhor interpretar, classificar e tomar decisões com base nos dados existentes na imagem [18].

2.2 Detecção das Mãos em Imagens Digitais

A detecção de mãos em imagens não é um processo trivial já que a análise corporal em uma imagem que inclui as mãos comumente possui elementos indesejados como: sombras, excesso de luz, outras partes do corpo, partes da roupa, objetos na cena, e até mesmo o cenário de fundo. Dessa forma, o rastreamento das mãos se torna muito mais viável após a realização de processamentos que possam minimizar o impacto destes elementos indesejados [19].

A maneira como a detecção é realizada é importante pois será determinante para a qualidade final dos dados que serão analisados, e pode, inclusive, impactar no modelo de previsão, gerado a partir dos dados [19].

Na Figura 3, a detecção das mãos é realizada com o auxílio de um processo de remoção de fundo da imagem feito através de uma técnica chamada *Background Subtraction* (subtração do fundo), que analisa a variação dos pixels do fundo de cena em comparação com os pixels das mãos, a fim de separá-los. Desta forma é possível obter uma imagem de saída com a mão isolada do fundo de cena.

Figura 3 – Exemplo de processo de remoção de fundo, retirado de [3]

A ideia de utilizar dispositivos vestíveis também é muito presente em pesquisas voltadas ao rastreamento de gestos das mãos, principalmente pela efetividade conferida ao usar sensores. A Figura 4 propõe a utilização de uma luva instrumentada constituída de sensores e dispositivos de captação de sinal de dados, que traduzem um sinal a partir da movimentação das mãos em tempo real e posteriormente são passados para um computador.

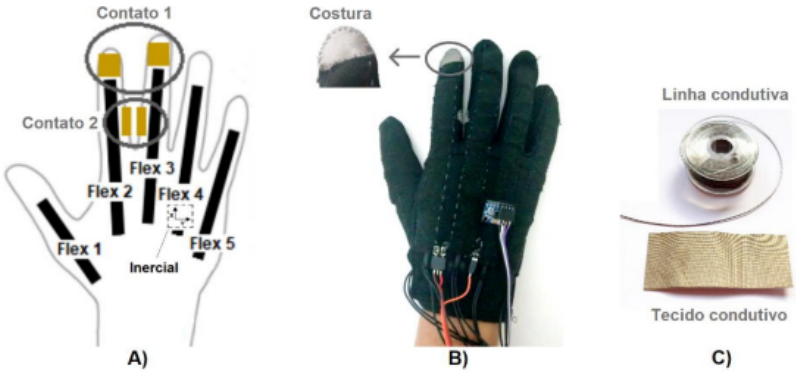

Figura 4 – Luva instrumentada para reconhecimento de padrões de gestos em Libras, criada por [4]

Apesar de eficiente, este tipo de medida para a detecção das mãos se torna mais restritiva por conta do orçamento necessário para a confecção da luva e aquisição dos materiais. Pensando em outras alternativas, diversos trabalhos apresentam propostas baseadas em processos de limiarização de imagens.

2.2.1 Limiarização de Imagens Digitais

A distribuição pela freqüência de ocorrência dos níveis de cinza de uma imagem digital é denominada de histograma desta imagem. Assim, em uma imagem que possui uma faixa de níveis de cinza no intervalo [0,...,L], seu histograma h(rk) = nk é uma função discreta, em que nk é o número total de ocorrência dos níveis de cinza rk dentro daquele intervalo [17]. Uma operação usual realizada sobre um histograma, com o objetivo de

realce, é a limiarização. Esta consiste num mapeamento de pixels de uma imagem original f(x,y) em outra imagem g(x,y), por exemplo, dado pela equação

$$g(x,y) = \begin{cases} a, & \text{if } f(x,y) \leq L \\ b, & \text{caso contrário,} \end{cases}$$

em que a, b e L são valores fixos, sendo que L é denominado de limiar, ou seja, é um valor limite que altera as propriedades dos pixels da imagem resultante g(x,y), a partir da imagem original [17].

Existem diferentes maneiras de se encontrar um limiar em um histograma devido aos diferentes cenários de distribuição das intensidades ou cores dos pixels de uma imagem, o que reflete em diferentes abordagens ao realizar uma segmentação. O histograma entrega resultados muito pertinentes à processamentos de imagens, principalmente pela quantidade de informações que podem ser extraídas de uma imagem digital para um eventual processamento da mesma.

Apesar do modelo de cores RGB não ser o modelo de cores mais intuitivo, os melhores resultados na área de segmentação de imagens coloridas são geralmente obtidos através deste formato de dados. A partir de um conjunto de amostra representativo das cores de interesse, pode-se obter uma estimativa da cor média que se deseja segmentar na imagem. O processo de segmentação consiste em classificar cada pixel da imagem dentro da faixa de interesse especificada para a cor. Para realizar esta classificação, utiliza-se a limiarização, um dos algoritmos de segmentação mais intuitivos e rápidos empregados em processamento de imagens [17].

A limiarização funciona através do estabelecimento de um limiar T que separa a imagem em duas regiões. Os pixels $P >= T$ são separados em uma classe (geralmente os pontos de interesse) enquanto os pontos $P < T$ são separados em outra classe (o que se pretende ignorar na imagem). Em sequência é realizada a binarização, onde ocorre a separação dos pixels em classes e cria-se uma imagem de saída onde os pixels são separados entre os valores 1 (branco) e 0 (preto) [17].

A proposta ilustrada na Figura 5 detecta a mão em imagens de profundidade e cria um modelo cinemático para a mesma. Um sensor Kinect foi utilizado para a captura das imagens, que em seguida passaram pelas fases de pré-processamento, extração e seleção das características, rastreamento da mão e modelo cinemático. Aqui, o limiar se baseia na altura aproximada da mão na imagem, onde os valores de profundidade dividem a cena entre objeto alvo (mão) e o restante a ser descartado.

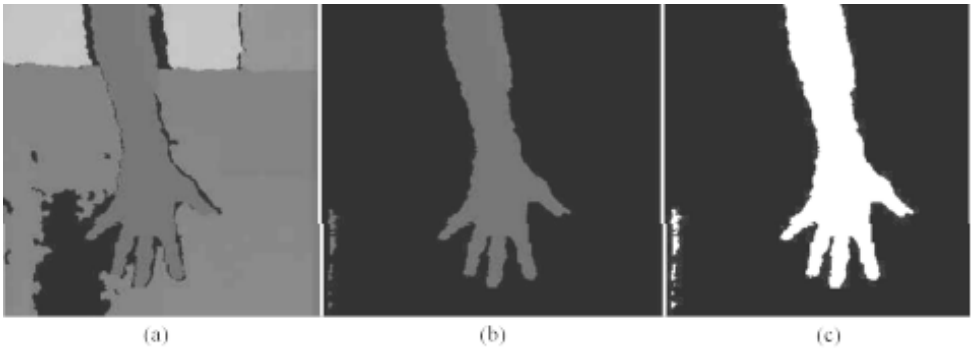

Figura 5 – Etapas da segmentação e binarização com a utilização de imagens de profundidade pelo método de [5]

Alternativamente, o rastreamento das mãos também pode ser realizado através de técnicas de identificação de cor de pele, como demonstra a abordagem de [6]. Utilizando espaço de cores YIQ, o valor do limiar foi determinado empiricamente pelos próprios pesquisadores para a definição de intervalos que classificam os pixels como pele ou não. Na Figura 6, é possível visualizar o resultado da técnica utilizada nesta abordagem, onde a detecção se deu de forma eficaz.

Figura 6 – Reconhecimento de pele pelo método de [6]

2.2.2 Reconhecimento Através do Mapeamento de Pontos-Chave da Mão

Uma abordagem que vem ganhando relevância consiste em separar o processo de reconhecimento em duas partes: a primeira utiliza um modelo que analisa uma imagem e identifica pontos de interesse na mão (chamado de esqueleto) e a segunda parte utiliza apenas os dados destes pontos para classificar os gestos. A partir desta perspectiva, é possível reduzir consideravelmente a complexidade dos modelos, uma vez que o modelo que analisa a imagem tem uma tarefa mais simples, e o modelo de reconhecimento de gestos pode passar a utilizar uma quantidade muito menor de dados.

Em [7], foi desenvolvido um modelo para abstrair o esqueleto de uma mão baseando-se em modelos de florestas aleatórias e mistura gaussiana. Desse modo, foi possível evitar a construção de um modelo extenso e complexo, e foi possível atingir uma inferência muito mais rápida, conseguindo operar em vídeos de até 50 fps. Na Figura 7 é possível visualizar o resultado obtido pelo modelo.

Figura 7 – Resultados do modelo retirado de [7]

Em alguns trabalhos onde o foco é o reconhecimento de gestos com os dados já processados, é muito comum utilizar conjuntos de dados previamente coletados, como é o caso do dataset SHREC'17. Nele, é realizada a identificação de 22 pontos para constituir o esqueleto da mão, podendo ser observados na Figura 8.

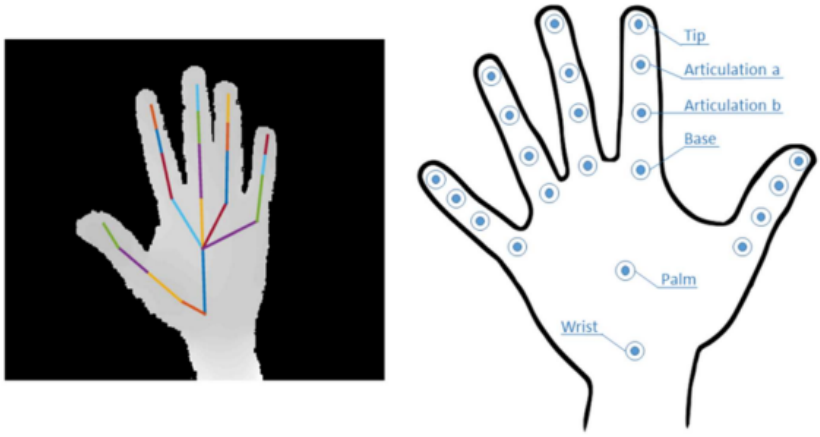

Figura 8 – Exemplo de modelo de dado do dataset SHREC'17, retirado de [8]

Em contrapartida, em vários trabalhos e aplicações, ao invés de utilizar um dataset pronto, tem se tornado comum utilizar modelos previamente treinados para capturar o esqueleto da mão numa imagem. Em particular, um modelo que tem se tornado popular neste aspecto é um modelo fornecido pelo *Google Reseach*, através de uma API chamada *MediaPipe*. O *MediaPipe* é um framework de livre distribuição que possui diversas funcionalidades e soluções para *Machine Learning*, incluindo detecção de objetos, detecção corporal, estimação de poses, detecção da íris dos olhos, entre muitas outras [20].

A Figura 9 demonstra o mapeamento das mãos com vários tipos de gestos, evidenciando como o *MediaPipe* pode ser uma ferramenta poderosa no quesito de reconhecimento de sinais.

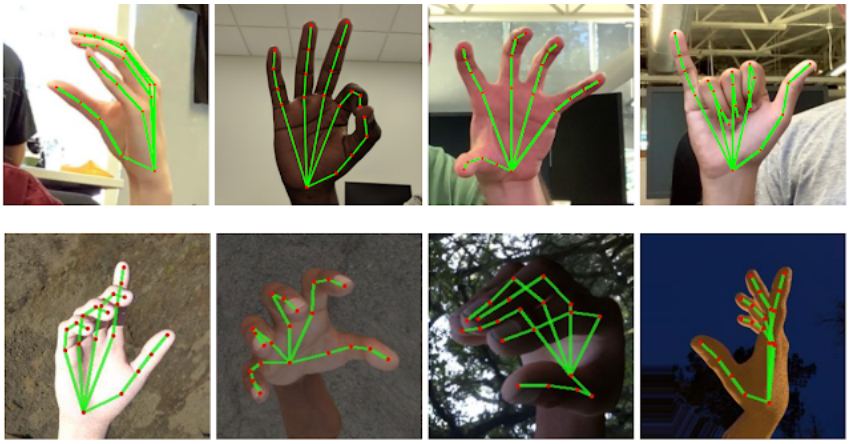

Figura 9 – Exemplo de reconhecimento das articulações das mãos, gerado pelo *framework MediaPipe*

Trabalhos como os de [21, 22, 23] demonstram como a utilização deste *framework* entrega resultados interessantes no tocante à detecção de gestos. Utilizando a solução de detecção de mãos, o *MediaPipe* realiza o mapeamento das articulações através de 21 pontos tanto para imagens ou para a utilização de vídeo (em tempo real), retornando suas localizações nos eixos x, y e z. A obtenção dos valores destas coordenadas viabilizam o processo de extração de características para trabalhos que não possuem *features* previamente definidas e que desejam fazer a sua própria aquisição de dados para treinar seus modelos.

2.3 Machine Learning

O aprendizado de máquina (ou *Machine Learning*) é um campo de pesquisa de inteligência artificial que busca desenvolver sistemas de computador que melhorem automaticamente seu desempenho por meio da experiência e treinamento. Embora programas de aprendizagem especializados existam hoje, o objetivo final é desenvolver sistemas mais amplamente aplicáveis e com capacidades de aprendizagem mais robustas [24]. O aprendizado de máquina divide os métodos de aprendizagem em três categorias: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço [25, 26, 27].

Um método de aprendizado supervisionado pode ser descrito como: um modelo que recebe um conjunto de pares de valores (entradas e saídas) rotulados que representam um determinado ambiente. Para que o modelo represente o ambiente, é necessário que ele passe por uma fase de treinamento. Nesta fase, o modelo recebe os valores que representam o ambiente (entradas) e para cada um deles é verificado se foi gerada a saída correspondente. Caso isto não ocorra, o modelo é reestruturado de forma a produzir o resultado correto [26]. Exemplos de métodos desta categoria são redes neurais artificiais do tipo *Multi-Layer Perceptron* (também chamado de MLP), SVM (*Support Vector Machines*), KNN

(*k-Nearest-Neighbour*) e árvores de decisão.

Já os métodos de aprendizado não supervisionados são técnicas de *machine learning* na qual o algoritmo é alimentado com dados não rotulados, ou seja, sem indicação prévia de qual é a saída correta para cada entrada. Dessa forma, o objetivo do algoritmo é encontrar padrões ou estruturas ocultas nos dados, sem a necessidade de um conjunto de dados de treinamento rotulado. Alguns exemplos de algoritmos de aprendizado não supervisionado incluem clustering, redução de dimensionalidade e detecção de anomalias.

Diferente do aprendizado supervisionado e não supervisionado, o aprendizado por reforço é baseado em um processo de tentativa e erro. Ele é caracterizado por interações de um agente com um ambiente. Nesse modelo, o agente interage com o ambiente através de ações, e recebe uma recompensa (ou punição) em resposta a cada ação tomada. O agente começa sem nenhum conhecimento prévio do ambiente e precisa explorar e experimentar diferentes ações para aprender qual é a melhor política a seguir.

À medida que o agente interage com o ambiente, ele vai aprendendo qual é a melhor ação a tomar em cada situação, ajustando sua política com base nas recompensas recebidas. Alguns exemplos de aplicações dos métodos de aprendizado por reforço incluem jogos de tabuleiro e até mesmo jogos eletrônicos.

2.3.1 *K-Nearest-Neighbors*

O algoritmo de classificação *k-Nearest-Neighbors* (KNN) é uma técnica de aprendizado de máquina que busca classificar uma amostra desconhecida a partir das classes de seus vizinhos mais próximos em um conjunto de dados. O KNN é um algoritmo simples e intuitivo, mas pode ser bastante eficaz em muitas tarefas de classificação, especialmente em problemas com dados não lineares ou com muitas dimensões. A Figura 10 ilustra um exemplo de aplicação do algoritmo.

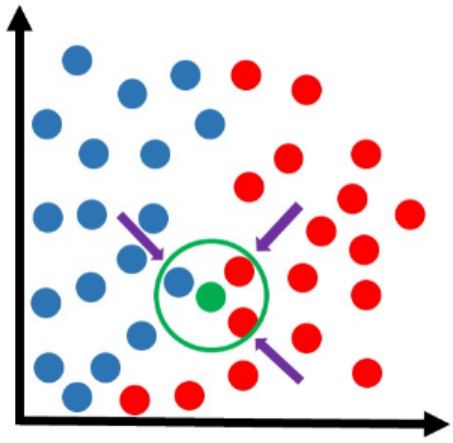

Figura 10 – Ilustração da utilização do KNN entre um conjunto de dados dividido em duas classes, retirado de [9]

Neste exemplo, a bolinha verde representa o elemento que estamos analisando, sendo k = 3 o número desejado de vizinhos a serem checados. Ao seu redor, o algoritmo detecta então os k vizinhos mais próximos e encontra a classe vermelha como dominante, atribuindo então a instância em questão como vermelha também. No geral, as instâncias podem ser consideradas como pontos dentro de um espaço de instância n-dimensional onde cada uma das n-dimensões corresponde a uma das n-características que são usadas para descrever uma instância [28].

Esta técnica se configura como um tipo de aprendizado supervisionado, onde o KNN baseia-se no princípio em que as instâncias dentro de um conjunto de dados geralmente existirão em proximidade com outras instâncias que possuem propriedades similares. Se as instâncias forem marcadas com um rótulo de classificação (*label*), o valor do rótulo de uma instância não classificada pode ser determinado observando a classe de seus vizinhos mais próximos [28].

2.3.2 *Support Vector Machines*

O SVM (*Support Vector Machine*) é uma técnica de aprendizado de máquina supervisionado usada principalmente para tarefas de classificação e regressão. O objetivo do SVM é encontrar o hiperplano que melhor separa os dados em duas classes distintas, maximizando a margem entre elas. O SVM é um método baseado em distância, no qual a escolha do hiperplano é baseada em encontrar a melhor separação entre os pontos de dados.

Como ilustra a Figura 11, é traçado um hiperplano que separa os pontos de dados das diferentes classes. O hiperplano é definido por uma linha (em 2D), um plano (em 3D) ou um espaço hiperplano (em mais de 3 dimensões) que maximiza a distância entre as classes. Essa distância é chamada de margem e é determinada pela distância entre os pontos mais próximos de cada classe, conhecidos como vetores de suporte [29].

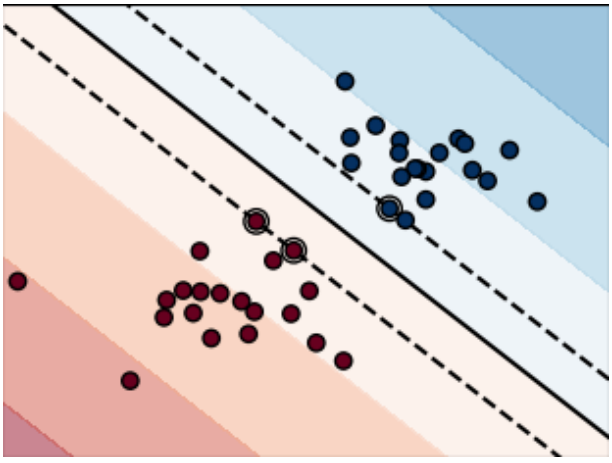

Figura 11 – Hiperplano linear segregando duas classes, retirado de [10]

O classificador SVM possui alguns parâmetros importantes que podem ser configurados para ajustar o modelo. Alguns desses parâmetros incluem o controle do *trade-off* entre a maximização da margem e a minimização do erro de classificação e também o tipo de kernel a ser utilizado para transformar os dados em um espaço de maior dimensionalidade.

Alguns exemplos de kernels usados no SVM são o linear, polinomial e gaussiano. Cada tipo de kernel tem suas próprias características e pode ser mais adequado para diferentes tipos de dados e problemas de classificação [30].

2.3.3 *Naive Bayes*

O classificador *Naive Bayes* é um método de classificação probabilístico baseado no teorema de Bayes. Ele é frequentemente utilizado em problemas de classificação de textos, mas pode ser aplicado a qualquer conjunto de dados com variáveis categóricas ou discretas [31]. O método é chamado de *naive* (ingênuo em português) porque assume que as variáveis (*features*) são independentes entre si, o que nem sempre é verdade, mas que simplifica bastante o cálculo das probabilidades.

Esta técnica de classificação utiliza o teorema de Bayes, que é uma fórmula de probabilidade que calcula a possibilidade de um evento acontecer com base em um conhecimento que pode estar relacionado ao evento. A fórmula é dada pela Equação 2.1:

$$P(y|x_1, ..., x_n) = \frac{(P(y) * P(x_1|y) * ... * P(x_n|y))}{P(x_1, ..., x_n)} \tag{2.1}$$

onde:

- P(y) é a probabilidade da classe y;
- P(x1, ..., xn) é a probabilidade das características x1, x2, ..., xn;
- P(y|x1, ..., xn) é a probabilidade da classe y dado as características x1, x2, ..., xn;
- P(x1, ..., xn|y) é a probabilidade das características x1, x2, ..., xn dado a classe y.

Existem diversas variações do classificador *Naive Bayes*, dependendo da distribuição de probabilidade assumida para as características de entrada (por exemplo, Gaussiana, Bernoulli, multinomial), onde cada variação tem sua própria equação.

2.3.4 Rede Neural

Uma rede neural, ou também chamada de rede neural artificial (RNA), é um método de inteligência artificial que ensina computadores a processar dados de uma forma

inspirada pelo cérebro humano. Chamada também de aprendizagem profunda (*Deep Learning*), tem a característica de usar nós ou neurônios interconectados em uma estrutura em camadas, semelhante ao cérebro humano [32]. Um neurônio é um objeto de uma rede neural que recebe dados de neurônios e os pesos de cada ligação, e retorna um valor calculado utilizando essas entradas.

Uma rede neural básica tem neurônios artificiais interconectados em três camadas:

- Camada de Entrada, responsável por receber informações externas. Os nós de entrada processam os dados, analisam ou categorizam esses dados e os encaminham para a próxima camada;
- Camada Oculta, responsável por ocultar as entradas da camada de entrada ou de outras camadas ocultas. As RNAs podem ter várias camadas ocultas. Cada camada oculta analisa o resultado da camada anterior, processa-o mais um pouco e o encaminha para a próxima camada;
- Camada de Saída, responsável por fornecer o resultado final de todos os dados processados pela rede neural artificial. Ela pode ter um ou vários nós. Por exemplo, se tivermos um problema de classificação binária (sim/não), a camada de saída terá um nó de saída, o que fornecerá o resultado como 1 ou 0. Porém, se tivermos um problema de classificação de várias classes, a camada de saída poderá ter mais de um nó de saída.

A Figura 12 mostra uma representação visual de uma arquitetura básica de uma RNA. É possível observar que cada neurônio é conectado a todos os neurônios da próxima camada.

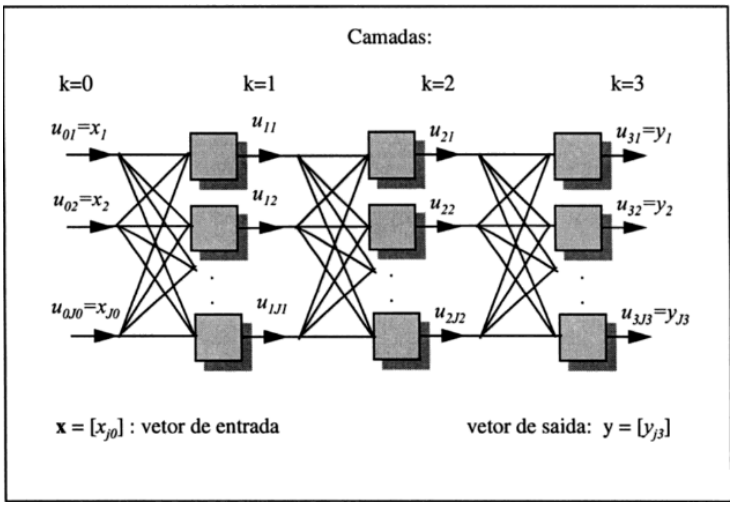

Figura 12 – Uma RNA multicamada com uma camada de entrada (k=0), duas camadas ocultas (k=1, k=2) e uma camada de saída (k=3), retirado de [11]

Nota-se pela ilustração que cada conexão entre neurônios possui um peso relacionado. Os pesos são os valores que determinam a importância relativa de cada entrada para a saída do neurônio. Eles são ajustados durante o treinamento da rede neural, de modo que a rede possa aprender a reconhecer padrões nos dados de entrada e produzir saídas corretas, além de estarem ligados diretamente às funções que determinam se um neurônio deve ser ativado ou não. Estas funções são chamadas de função de ativação.

2.3.4.1 Função de Ativação

A função de ativação é aplicada a cada neurônio da rede e pode ser pensada como uma espécie de limiar (*bias*). Quando o resultado da soma ponderada das entradas de um neurônio ultrapassa esse limiar, ele é ativado e envia um sinal para a próxima camada da rede. Caso contrário, o neurônio permanece inativo [33].

A Figura 13 ilustra um esquema básico de um neurônio artificial (também conhecido como *perceptron*), onde é possível observar a função de ativação sendo aplicada com base na soma ponderada das entradas.

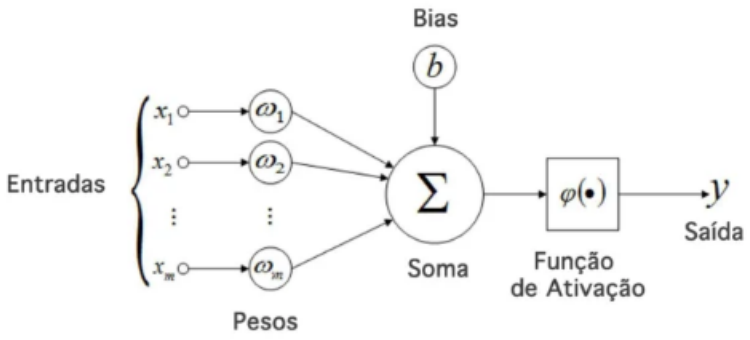

Figura 13 – Ilustração de um modelo simples de neurônio artificial, retirado de [12]

Existem vários tipos de funções de ativação, onde a escolha da mesma dependerá do tipo de problema que se deseja resolver e das características da arquitetura da rede neural a ser utilizada. Algumas das funções de ativação mais comuns são a função sigmoidal, a função ReLU (*Rectified Linear Unit*), a função tanh e a função *softmax* [33].

2.3.4.2 Otimização de Peso

Outra característica muito importante na arquitetura das redes neurais é o ajuste dos valores dos pesos. A otimização de peso em uma rede neural é a etapa onde os pesos das conexões são atualizados de forma a minimizar a função de perda da rede para que ela possa aprender a mapear corretamente as entradas para as saídas desejadas. Esse processo é essencial para que a rede possa aprender com os dados de treinamento e generalizar para novos dados.

Existem vários tipos de ajustes de peso em redes neurais artificiais, sendo o método de Descida de Gradiente um dos mais utilizados neste aspecto [34].

Descida do Gradiente é um algoritmo iterativo que ajusta os pesos de uma RNA por meio do cálculo do gradiente da função de perda em relação aos pesos e subtrai uma fração desse gradiente do valor atual do peso [34]. Esse processo é repetido várias vezes até que um mínimo global ou local da função de perda seja alcançado. O gradiente é calculado por meio da retropropagação (também conhecido como *backpropagation*) do erro da rede, que é um algoritmo que propaga o erro de saída da rede de volta às camadas anteriores. O gradiente é então usado para atualizar os pesos, diminuindo a função de perda da rede.

2.3.4.3 Topologia

A forma como os neurônios estão conectados, também chamada topologia, influencia diretamente no aprendizado da RNA. Tradicionalmente, as RNAs possuem uma topologia onde seus neurônios ficam dispostos em camadas. Essas topologias são classificadas em três grupos principais: *perceptron*, *Multi-Layer Perceptron* (MLP) e recorrente. O *Perceptron*, ou também conhecido como *single-layer feedforward*, consiste na configuração mais básica de RNA. Composta de apenas uma camada de entrada e outra de saída e sem ciclos entre conexões dos neurônios, esta RNA fica limitada à resolução de problemas lineares [35].

O *Multi-Layer Perceptron* ou *multi-layer Feedforward* adiciona as camadas ocultas ao modelo *perceptron* tradicional elevando a capacidade de resolução para problemas não lineares, bem como melhoria da precisão dos resultados da RNA [12]. Na topologia Recorrente há a presença de um laço que conecta a saída do neurônio para a entrada de outro, de forma a realimentar a RNA.

2.3.5 *Cross-Entropy*

A entropia cruzada (*cross-entropy*, em inglês) é uma medida comumente usada para avaliar a diferença entre duas distribuições de probabilidade. No contexto de aprendizado de máquina, a entropia cruzada é frequentemente usada como uma função de custo (*loss function*) em tarefas de classificação, onde o objetivo é minimizar a diferença entre a distribuição de probabilidade prevista pelo modelo e a distribuição de probabilidade real dos rótulos das amostras [34]. A fórmula geral para a entropia cruzada está descrita na Equação 2.2:

$$H(p, q) = -\sum_{i=1}^{n} p_i \log q_i \tag{2.2}$$

O valor p é a distribuição de probabilidade real dos rótulos e q é a distribuição de probabilidade prevista pelo modelo.

Durante o treinamento do modelo, a entropia cruzada é usada como uma função de custo, que é minimizada usando algoritmos de otimização, como o gradiente descendente. O objetivo é encontrar os pesos do modelo que minimizam a entropia cruzada em todo o conjunto de treinamento, o que significa que o modelo está prevendo com mais precisão a distribuição de probabilidade real dos rótulos. Em suma, quanto menor o valor da entropia, por se tratar de um valor de perda, maior a qualidade do modelo.

2.4 Cálculo de desempenho de métricas

Para avaliar o desempenho de uma rede neural, é importante definir métricas adequadas que possam indicar como a rede está performando e qual é o erro em relação aos valores esperados. A análise correta das métricas de modelo aplicado em um conjunto de dados pode revelar problemas na rede treinada. Na literatura, algumas das métricas comuns usadas para avaliar o desempenho de redes neurais são matrizes de confusão, acurácia, precisão, revocação e *F1 score*. Portanto, é fundamental escolher as métricas apropriadas para monitorar e melhorar o desempenho da rede neural.

2.4.1 Matriz de confusão

Uma matriz de confusão é uma tabela usada para avaliar o desempenho de um modelo de classificação. É uma ferramenta útil para avaliar a precisão do modelo, especialmente em problemas de classificação binária ou multiclasse. Com base na matriz de confusão, é possível calcular outras métricas de desempenho, como acurácia, precisão, revocação e *F1 score*.

A Tabela 1 mostra onde as saídas, reais e preditas, devem ser colocados para classificar como verdadeiro positivo (VP), verdadeiro negativo (VN), falso positivo (FP) e falso negativo (FN). Verdadeiro ou falso indica se a predição da rede neural está correta. Positivo ou negativo indica qual foi o valor predito pela rede.

Real	**Previsto**	
	Positivo	**Negativo**
Positivo	Verdadeiro positivo (VP)	Falso negativo (FN)
Negativo	Falso positivo (FP)	Verdadeiro negativo (VN)

Tabela 1 – Matriz de confusão para um classificador binário.

2.4.2 Acurácia

A acurácia é uma medida geral que indica a quantidade de predições corretas da rede, sejam elas positivas ou negativas. O cálculo da acurácia é realizado conforme apresentado na Equação 2.3

$$Acc = \frac{VP + VN}{VP + FP + FN + VN} \tag{2.3}$$

2.4.3 Precisão

A precisão é uma medida que expressa a proporção de predições corretas em relação a tudo o que foi classificado como positivo. O cálculo da precisão é realizado por meio da Equação 2.4

$$Prec = \frac{VP}{VP + FP} \tag{2.4}$$

2.4.4 Revocação

A revocação, também conhecida como recall, é uma medida que expressa a proporção de casos positivos que foram corretamente identificados pelo modelo em relação ao total de casos positivos existentes. A Equação 2.5 demonstra como é realizado seu cálculo.

$$Rev = \frac{VP}{VP + FN} \tag{2.5}$$

2.4.5 *F1 score*

O *F1 Score* é uma métrica que combina as medidas de precisão e revocação em uma única medida, representando a média harmônica entre elas. Ele é útil para avaliar o desempenho geral de um modelo em relação a uma classe específica, levando em conta tanto a taxa de verdadeiros positivos quanto a taxa de falsos positivos. Sua fórmula é apresentada conforme a Equação 2.6.

$$F1Score = 2 * \frac{Prec * Rev}{Prec + Rev} \tag{2.6}$$

3 DESENVOLVIMENTO

O fluxo geral da metodologia do projeto pode ser visualizado na Figura 14. O projeto propôs utilizar imagens digitais contendo sinais de Libras retiradas de um *dataset*, extrair suas características utilizando o *framework MediaPipe* e aplicá-las em um modelo de *machine learning* baseado em rede neural que, através da análise dos padrões encontrados, classificará o sinal de Libras em questão.

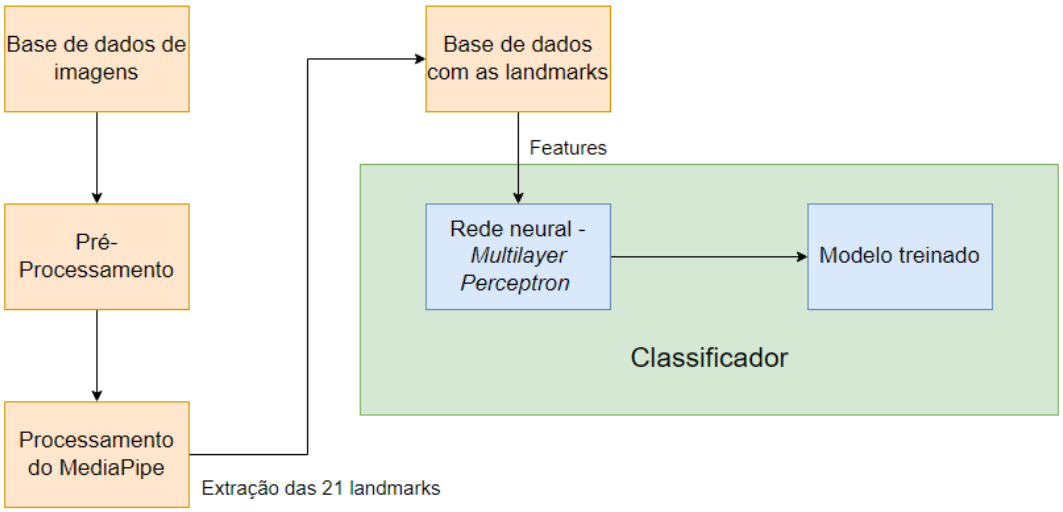

Figura 14 – Fluxo da visão geral do projeto

3.1 Seleção de Imagens

As imagens selecionadas para a realização do trabalho foram retiradas do *dataset* desenvolvido no IME-RJ (Instituto Militar de Engenharia - SE/8 do Rio de Janeiro) [13]. Os fatores implicantes para a seleção deste banco de imagens se caracterizaram principalmente pelo número significativo de imagens disponíveis, além da compatibilidade das mesmas para o processamento realizado no *MediaPipe*, o qual oferece suporte somente para imagens em formato RGB. O *dataset* criado no trabalho de [13] foi construído a partir de quadros extraídos em vídeos. Para a realização do presente trabalho, foram fornecidas 78.000 imagens, onde as classes estão separadas em 26 diretórios havendo 3.000 imagens cada um. A Figura 15 ilustra alguns dos exemplos de imagens do banco em questão.

Figura 15 – Sinais de Libras representando as letras A, B e C respectivamente. Retirado de [13]

3.1.1 Pré-Processamento

O único processamento aplicado diretamente nas imagens foi o aumento de brilho. No momento da utilização do *MediaPipe*, para extrair os pontos de marcação da mão, foi notado que algumas das imagens não estavam sendo detectadas pelo algoritmo por conta da iluminação irregular na região da mão. Este tipo de problema pode afetar a qualidade final do modelo, afinal a quantidade de amostras detectadas para cada tipo de classe ficaria muito desbalanceada.

Este problema foi resolvido criando um algoritmo que aumenta o brilho em níveis de pixel. Ele utiliza a biblioteca *OpenCV* para converter a imagem em um espaço de cor HSV, manipulando o canal de brilho e posteriormente convertendo a imagem de volta para o espaço de cor RGB. O aumento de brilho é especificado como um valor inteiro, podendo ser ajustado conforme necessário. A Figura 16 ilustra a diferença da imagem original, com a imagem que sofreu alteração no brilho.

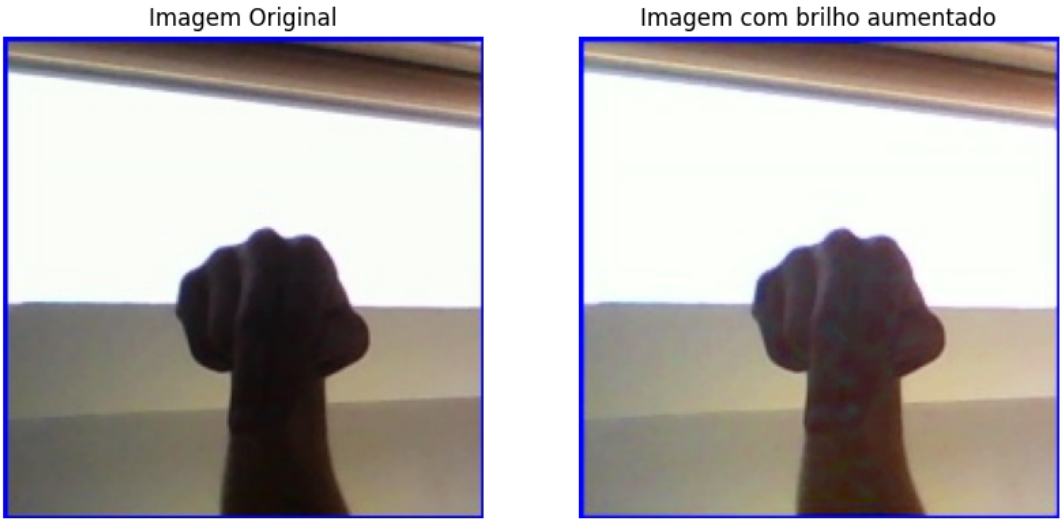

Figura 16 – Diferença da imagem antes e após alterar o brilho

3.2 Identificação das Mãos

O fato das imagens estarem em formato RGB possibilitou o uso do *MediaPipe* para o mapeamento do esqueleto das mãos e consequentemente sua identificação. A solução de detecção de mãos do *MediaPipe* é baseada em uma rede neural profunda treinada previamente pelos desenvolvedores da Google para detectar as mãos em imagens e vídeos em uma variedade de condições, incluindo mudanças de iluminação, fundos complexos e oclusões parciais [14].

Este sistema de reconhecimento de mãos do *MediaPipe* é dividido em 2 etapas, sendo elas o reconhecimento da palma e o reconhecimento de pontos de marcação da mão. Primeiro, foi treinado um detector de palma em vez de um detector de mão, já que estimar caixas delimitadoras (*bounding box*) de objetos rígidos como palmas e punhos é significativamente mais simples do que detectar mãos com dedos articulados. O treinamento em questão foi realizado utilizando diferentes técnicas com o propósito de abranger situações variadas e garantir o reconhecimento das mesmas. De acordo com [14], a equipe de desenvolvimento conseguiu uma taxa de precisão média de 95,7% no reconhecimento da palma da mão.

Após a detecção da palma em toda a imagem, o modelo subsequente de pontos de marcação da mão (também chamado de *landmarks*) executa a localização precisa de pontos-chave de 21 coordenadas 3D de juntas da mão dentro das regiões da mão detectadas por meio de regressão, como mostra a Figura 17.

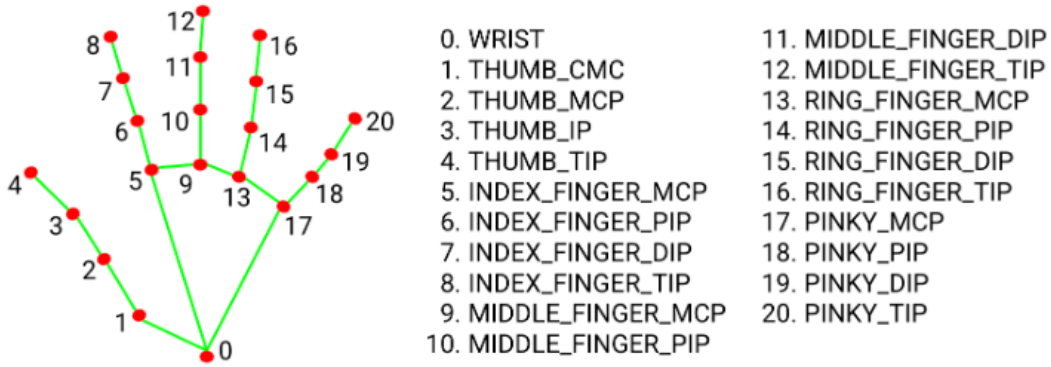

Figura 17 – Pontos-chave de marcação da mão criados pelo *MediaPipe*, retirado de [14]

Utilizando a solução de detecção de mãos do *MediaPipe*, a primeira etapa de processamento das imagens foi realizar a aquisição dos 21 pontos-chave de marcação da mão de todas as imagens presentes do conjunto de dados, dada a Figura 18 como um exemplo deste processamento.

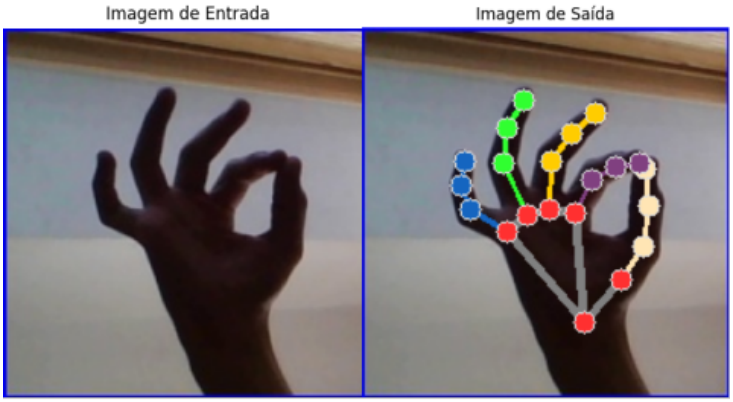

Figura 18 – À esquerda uma imagem representando a letra F, à direita a imagem processada pelo *MediaPipe*

3.3 Extração das *Features*

De modo inicial, houve dificuldade em encontrar maneiras de extrair características das mãos, já que o *dataset* disponibilizado somente continha imagens, sem nenhuma informação de *features*. A etapa de seleção de características é crucial para a qualidade final do modelo, sendo importante ter em mente de que selecionar as características certas pode afetar significativamente o desempenho do modelo de reconhecimento de mãos. Existem várias técnicas e abordagens para selecionar *features* de mãos para modelos de reconhecimento, incluindo técnicas de extração de características baseadas em imagem, como detecção de bordas ou descritores de textura.

O problema destas abordagens para este trabalho se mostrou na complexidade e no custo computacional, já que os cálculos destes métodos de extração seriam realizados em cada uma das milhares de imagens do banco. A partir desse problema, o uso do *MediaPipe* se mostrou uma ferramenta muito útil, sendo possível garantir o reconhecimento da mão e de seus pontos de marcação sem afetar significamente o desempenho do modelo, garantindo também um modelo de reconhecimento preciso e confiável.

Conforme a Figura 17, cada uma das 21 marcações retorna uma coordenada tridimensional, incluindo os valores de x, y e z. Somente foram levados em conta os valores de x e y, já que z se trata da distância de fundo entre a mão e a câmera. Deste modo, o valor de z foi ignorado por ser uma informação irrelevante neste processo.

A extração de *features* é realizada a cada iteração nas imagens. Conforme ela ocorre, o algoritmo de detecção de *landmarks* do *MediaPipe* é aplicado e as informações de coordenadas dessas *landmarks* são adquiridas em vetores no formato [x,y].

Esses dados foram transferidos para um *dataframe* criado pela biblioteca *Pandas*, pertencente ao *Python*. Cada *landmark* (do 0 ao 20) foi inserido como uma coluna no *dataframe*, totalizando originalmente 22 colunas, incluindo a coluna de rótulo. No entanto,

como essas informações não estavam em um formato compatível com a camada de entrada da rede neural, foi necessário realizar tratamentos nos dados.

Em particular, os dados foram normalizados e as coordenadas [x,y] foram transformadas em colunas separadas para cada ponto, com uma coluna indicando a coordenada x e outra indicando a coordenada y. Além disso, os valores foram formatados para remover caracteres indesejados, como ',' e ']'. Com essas transformações, os dados foram preparados para serem passados para a rede neural.

3.4 Arquitetura da Rede Neural

A rede neural desenvolvida se trata de um *Perceptron* Multicamadas (MLP). Sua escolha se deu principalmente por conta de sua adaptabilidade aos dados fornecidos, já que isso o torna flexível para lidar com uma ampla variedade de problemas de aprendizado de máquina. A rede foi construída utilizando um framework para *machine learning* chamado *Scikit-learn* (também conhecido como sklearn). O *Scikit-learn* é uma biblioteca de aprendizado de máquina em *Python* que oferece uma ampla gama de algoritmos para classificação, regressão, agrupamento e outras tarefas de mineração de dados e análise estatística. A biblioteca é construída em cima de outras bibliotecas de computação científica do *Python*, como *NumPy*, *SciPy* e *Matplotlib*.

A rede é composta por 27 camadas no total, incluindo 25 camadas ocultas, uma camada de entrada e uma de saída. Os dados para treino e teste passados para a rede foram respectivamente separados em 70% e 30% do conjunto total. A camada inicial recebe estes dados de entrada como uma matriz *numpy* contendo todas as *features* presentes no conjunto, ou seja, todas as informações de *landmarks*. Os dados de coordenadas são passados como um vetor x enquanto as *labels* são passadas em um vetor y. Os parâmetros para o desenvolvimento da rede ficaram da seguinte forma:

- *Solver* = sgd - Parâmetro responsável pela escolha do algoritmo que a rede utilizará para a otimização de peso. O valor SGD (*Stochastic Gradient Descent*) é um algoritmo que ajusta os pesos do modelo iterativamente, atualizando-os a cada passo com base na direção do gradiente negativo da função de custo. Isso é feito em pequenos lotes de dados (por isso é chamado de gradiente estocástico), tornando a técnica mais rápida e escalável para conjuntos de dados grandes;
- *activation* = logistic - Parâmetro responsável pela função de ativação a ser utilizada nas camadas ocultas. O valor *logistic* se refere à função sigmoide como função de ativação. Ela é muito comum para modelos de classificação, já que a função sigmoide é diferenciável em todos os pontos. Em outras palavras, a inclinação da curva sigmoide pode ser calculada para qualquer valor de entrada;

- *hidden_layer_sizes* = 25 - Parâmetro responsável por sinalizar quantas camadas ocultas a rede terá. Após experimentos e análises, foi verificado que uma rede com 25 camadas ocultas apresentou um desempenho superior em relação a outras configurações testadas, como mostra a Figura 19;

- *learning_rate_init* = 0.01 - Parâmetro responsável por controlar a taxa de aprendizado da rede neural;

- *max_iter* = [100,500,800,1000,1180] - Parâmetro responsável pelo número máximo de iterações realizadas na rede. Como mostra a Figura 19, foram testados diferentes valores máximos de iterações com o objetivo de encontrar a melhor acurácia e o ponto em que a função SGD converge;

- *random_state* = 1 - Parâmetro responsável por controlar a aleatoriedade dos valores de inicialização para os pesos e o *bias* na função de ativação.

Desta forma, o processo de iteração na rede MLP desenvolvida consiste em propagar os dados de entrada para frente através das camadas ocultas e de saída, calcular o erro comparando a saída com as etiquetas, retropropagar o erro de volta para ajustar os pesos da MLP, atualizar os pesos e repetir o processo até que a rede atinja um desempenho satisfatório ou o número máximo de iterações seja alcançado. Após o treinamento, a MLP pode ser usada para fazer previsões em dados não vistos (dando margem para a criação de aplicações).

4 RESULTADOS

Para a realização dos testes, foram selecionadas de forma aleatoria 70% da amostras para treino e os outros 30% para testes. Durante esta fase, foram testados diferentes número de iterações na rede com o intuito de alcançar a melhor acurácia e o ponto em que a função de otimização de peso SGD convergisse. Foram observados os valores de acurácia e *cross-entropy* para cada um destes testes. Após atingir o número de 1180 iterações, foi possível visualizar os seguintes resultados nos testes, como mostra a Figura 19.

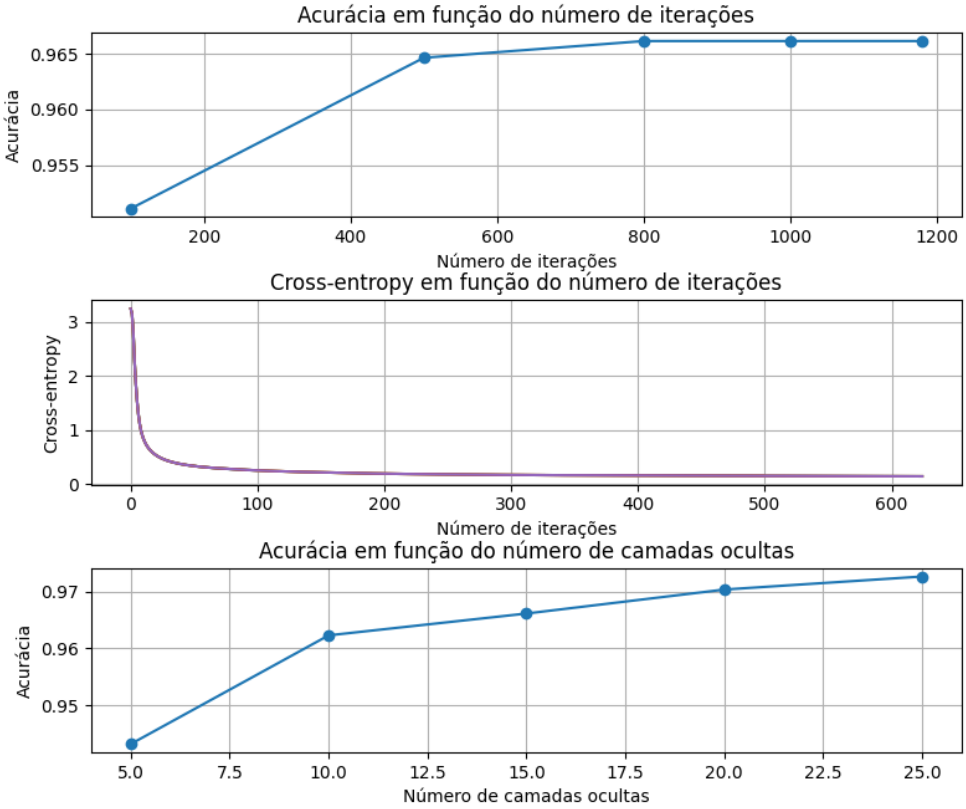

Figura 19 – Gráfico de acurácia por número de iterações, camadas ocultas e *cross-entropy* no conjunto de validação

Para definir o melhor resultado deve-se observar os valores de acurácia e *cross-entropy* em conjunto. Como pode-se notar, o melhor resultado ocorreu em 1180 iterações, como mostra a Tabela 2.

Foi observado que, ao testar valores de iterações máximas maiores de 1180, a rede neural atinge seu limite e apresenta pouca ou nenhuma alteração nos resultados, tornando-se inviável aumentar esse valor. É interessante notar que a partir de 500 iterações o valor da acurácia muda de forma sutil, porém o valor de perda (entropia) diminui de forma significativa. Isso significa que a rede começa a chegar em seu limite em acurácia enquanto

Iterações	Acc./treino	Cross-ent./treino
100	0.917	0.468
500	0.954	0.226
800	0.958	0.187
1000	0.960	0.173
1180	0.960	0.164

Tabela 2 – Comparação entre valores de acurácia e *cross-entropy*

continua ajustando seus pesos para os dados de treinamento até atingir o valor máximo de iterações.

A matriz de confusão obtida com 30% dos dados de teste pode ser visualizada na Figura 20. Neste exemplo, os valores da diagonal principal se mostram diferentes do restante da matriz. Os valores coloridos representam um número alto de predições corretas realizadas pelo modelo, enquanto os valores em roxo representam um número baixo de predições erradas, indicando que o modelo está performando bem. No entanto, quando estamos lidando com muitas classes, pode ser inviável verificar apenas a matriz de confusão devido à dificuldade de visualização pois há muitos números, o que causa um pouco de confusão na identificação dos mesmos. Por esse motivo, podemos verificar outras métricas para avaliar o modelo treinado. É possível usar métricas como acurácia, precisão, revocação, *F1-score*, entre outras, que podem apresentar uma ideia mais clara do desempenho do modelo para cada classe individualmente e para o conjunto de classes como um todo.

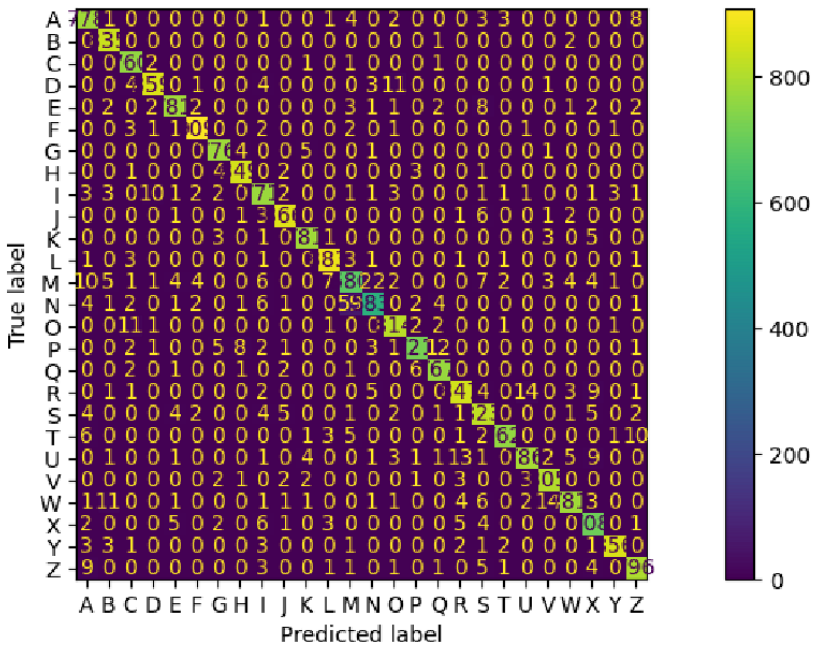

Figura 20 – Matriz de confusão (modelo com 1180 iterações máximas)

As métricas de precisão, *recall*, *F1-score* e acurácia para cada classe estão ilustradas na Tabela 3, que mostra resultados satisfatórios quanto a predição das classes.

	Precision	**Recall**	**F1-score**
A	0.95	0.97	0.96
B	0.97	1.00	0.98
C	0.96	0.99	0.98
D	0.98	0.97	0.98
E	0.98	0.97	0.97
F	0.99	0.99	0.99
G	0.98	0.99	0.98
H	0.98	0.99	0.98
I	0.94	0.96	0.95
J	0.98	0.98	0.98
K	0.98	0.98	0.98
L	0.98	0.99	0.98
M	0.89	0.89	0.89
N	0.94	0.87	0.90
O	0.97	0.98	0.97
P	0.98	0.95	0.97
Q	0.97	0.98	0.98
R	0.96	0.95	0.96
S	0.94	0.96	0.95
T	0.99	0.96	0.97
U	0.97	0.95	0.96
V	0.97	0.98	0.98
W	0.98	0.94	0.96
X	0.94	0.96	0.95
Y	0.99	0.98	0.99
Z	0.97	0.97	0.97
Accuracy: 97,15%			

Tabela 3 – Relatório de classificação

4.1 Comparação de resultados com outros modelos

Foram realizadas algumas comparações entre diferentes modelos de classificação durante o desenvolvimento deste estudo. A comparação de modelos é uma técnica importante para podermos compreender quais métodos de classificação são mais adequados para o problema que estamos lidando considerando fatores como precisão, velocidade, eficiência e outros critérios relevantes. Os classificadores escolhidos para esta comparação são frequentemente utilizados em outros estudos, sendo assim, os resultados obtidos podem ser comparados com outras pesquisas, e assim, proporcionar uma visão mais ampla do desempenho de cada método em diferentes contextos.

Modelo	Recall	F1-score	Acurácia	Precisão
KNN (k=1)	0.98	0.98	0.98	0.98
KNN (k=3)	0.97	0.97	0.97	0.97
KNN (k=5)	0.96	0.96	0.96	0.96
MLP	0.97	0.97	0.97	0.97
SVM	0.95	0.95	0.96	0.97
Naive Bayes	0.44	0.44	0.44	0.51

Tabela 4 – Comparação de resultados entre 4 modelos de acordo com as métricas de *recall*, *F1-score*, acurácia e precisão.

Neste estudo em particular, foram avaliados quatro modelos de classificação: MLP, SVM, KNN e *Naive Bayes*. A análise dos resultados ilustrados na Tabela 4 mostra que o KNN com k = 1 obteve a maior acurácia, com um valor inclusive superior ao MLP, enquanto o *Naive Bayes* apresentou uma acurácia consideravelmente inferior, em torno de 44%. No geral, os modelos KNN, SVM e MLP obtiveram ótimos resultados com acurácia a partir de 95%.

O que pode explicar o resultado superior do KNN em relação à rede MLP é a sua sensibilidade a *outliers*. *Outliers* no contexto de inteligência artificial é um dado que se distancia radicalmente dos demais que compõem a amostra analisada, ou seja, um valor atípico. Neste contexto, O MLP é um modelo mais complexo que o KNN, e pode ser mais sensível a *outliers* e ruídos nos dados. Nestas situações, o KNN pode ser mais robusto pois se baseia diretamente na proximidade entre as instâncias.

Essa comparação permitiu identificar as vantagens e desvantagens de cada modelo em relação à classificação de sinais de libras, permitindo a escolha do modelo mais adequado para o problema em questão. Além disso, os resultados obtidos podem servir como referência para estudos futuros na área de processamento de sinais e classificação de imagens.

4.2 Demonstração prática

Com o intuito de testar a eficácia do modelo com a entrada de novos dados, uma aplicação prática foi desenvolvida para demonstração. A aplicação consiste em abrir uma janela de vídeo através de uma *webcam*, onde a detecção do sinal de Libras é realizada em tempo real. A Figura 21 ilustra alguns exemplos de sinais identificados com sucesso.

Em contrapartida, a captura dos sinais de algumas letras funcionou somente em alguns pontos específicos no campo de visão da câmera. Como as *features* da rede são

(a) Letra C do alfabeto de Libras

(b) Letra F do alfabeto de Libras

(c) Letra I do alfabeto de Libras

(d) Letra P do alfabeto de Libras

Figura 21 – Exemplos de sinais capturados com sucesso na demonstração

coordenadas, este problema pode ter sido ocasionado pelo fato dos sinais no conjunto de dados de treinamento terem sido disponibilizados em poucas variedades de posições. Dessa forma, alguns sinais estão sendo capturados somente em regiões específicas da *webcam*. A Figura 22 ilustra um exemplo deste problema, mostrando a letra D na posição em que foi treinada sendo detectada de maneira correta, Figura 22 (a), em relação ao mesmo sinal sendo detectado de forma incorreta em outra posição na câmera, Figura 22 (b).

(a) Letra D do alfabeto de Libras na posição em que foi treinada - previsão correta

(b) Letra D do alfabeto de Libras em outra posição - previsão incorreta

Figura 22 – Exemplo do problema detectado na demonstração

5 CONCLUSÃO

Tendo por objetivo a classificação de sinais de Libras em imagens digitais, este trabalho avaliou o uso de uma rede neural artificial do tipo *Multi-Layer Perceptron* para construir um modelo de reconhecimento. Ao alcançar uma acurácia de 97% com a utilização do *MediaPipe*, pôde-se concluir que é possível utilizar uma RNA MLP para uma tarefa de classificação de sinais em imagens. Analisando outros métodos de aprendizado de máquina como o KNN ou SVM, também foi possível encontrar bons resultados no tocante à classificações desta natureza.

Além disso, a abordagem adotada neste trabalho pode ser aplicada em diferentes contextos e cenários, como por exemplo, na educação inclusiva, na comunicação entre surdos e ouvintes e na acessibilidade digital. Desta forma, este trabalho se alinha aos princípios do uso da tecnologia social para promover o bem-estar e a inclusão de pessoas com deficiência auditiva.

Sugere-se como trabalhos futuros a realização de mais experimentos com o *MediaPipe*, utilizando diferentes arquiteturas de RNA MLP, bem como outras categorias de RNA, tais como as redes convolucionais e as redes recorrentes. É interessante concluir que a partir destes testes, os problemas encontrados na demonstração prática podem ser corrigidos em trabalhos futuros a partir do uso (ou da criação) de um *dataset* que possua os sinais de Libras em diferentes posições no campo da imagem, a fim de garantir uma maior cobertura e eficácia no reconhecimento dos padrões dos sinais.

REFERÊNCIAS

[1] OLIVEIRA, G. H. *Representação da Matriz de Pixels*. 2019. <https://www.researchgate.net/figure/Figura-3-Representacao-da-Matriz-de-Pixels_fig1_348443999>.

[2] VALENTE, J. P. F. *Modelo RGB*. <https://sites.google.com/site/aimcjv/modelo-rgb>.

[3] SIMOES, W. Remoção do fundo da cena para detecção da silhueta da mão humana e detecção de movimentos. *SIGES 2011 - Simpósio de Informática e Geotecnologia de Santarém*, Oct 2011.

[4] DIAS, T. S. Luva instrumentada para reconhecimento de padrÕes de gestos em libras. *CT - Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial*, Mar 2020.

[5] SANTOS, T. N. Detecção e rastreamento da mão utilizando dados de profundidade. *Mestrado em Mecatrônica*, 2017.

[6] BHUIYAN, M. et al. Face detection and facial feature localization for human-machine interface. *NII Journal*, p. 25–39, 03 2003.

[7] SRIDHAR, S. et al. Fast and robust hand tracking using detection-guided optimization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015.

[8] SMEDT, Q. D. et al. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In: PRATIKAKIS, I.; DUPONT, F.; OVSJANIKOV, M. (Ed.). *Eurographics Workshop on 3D Object Retrieval*. [S.l.]: The Eurographics Association, 2017. ISBN 978-3-03868-030-7. ISSN 1997-0471.

[9] O que é e como funciona o algoritmo KNN? Disponível em: <https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-knn/>.

[10] SVM Margins Example. Disponível em: <https://scikit-learn.org/stable/auto_examples/svm/plot_svm_margin.html#sphx-glr-auto-examples-svm-plot-svm-margin-py>.

[11] KOVACS, Z. L. *Redes Neurais Artificiais, Fundamentos e Aplicações*. 4th. ed. [S.l.]: Livraria da Física, 2006.

[12] S., H. *Neural networks: A comprehensive foundation*. 2nd. ed. [S.l.]: Prentice Hall, 1998.

[13] FURTADO, S.; OLIVEIRA, J. Computer vision and neural networks for libras recognition. In: *Anais do XVII Workshop de Visão Computacional*. Porto Alegre, RS, Brasil: SBC, 2021. p. 137–142. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/wvc/article/view/18903>.

[14] HANDS. Disponível em: <https://google.github.io/mediapipe/solutions/hands.html>.

[15] LEI Que Oficializa a lingua brasileira de Sinais Completa Seis Anos. 2008. Disponível em: <https://www.interlegis.leg.br/comunicacao/noticias/2008/04/ha-6-anos-a-lingua-brasileira-de-sinais-libras-era-oficializada-como-meio-legal-de-expressao>.

[16] PEREIRA, G. K. *Curso de Libras*. 2010. <https://ufsj.edu.br/portal2-repositorio/File/incluir/libras/curso_de_libras_-_graciele.pdf>.

[17] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 3rd. ed. [S.l.]: Pearson, 2007.

[18] GIOVANINI, A. *Processamento digital de imagens: o que é?* 2016. <https://adenilsongiovanini.com.br/blog/processamento-digital-de-imagens/>.

[19] GOMES, P. C. T. *Pré-Processamento de Dados: Conheça as técnicas E as etapas!* 2022. Disponível em: <https://www.datageeks.com.br/pre-processamento-de-dados/>.

[20] HOME. Disponível em: <https://google.github.io/mediapipe/>.

[21] ZHANG, F. et al. *MediaPipe Hands: On-device Real-time Hand Tracking*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2006.10214>.

[22] SUNG, G. et al. *On-device Real-time Hand Gesture Recognition*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2111.00038>.

[23] ALVES, M. et al. Desenvolvimento de aplicativo para promover comunicação e interação social para surdos. *VI Encontro de Iniciação Científica e Tecnológica VI EnICT*, 2021.

[24] MITCHELL, T. et al. Machine learning. *Annual Review of Computer Science*, 1990.

[25] BARANAUSKAS, J. A.; MONARD, M. C. Combining symbolic classifiers from multiple inducers. *Knowledge-Based Systems*, v. 16, n. 3, p. 129–136, 2003.

[26] MITCHELL, T. M. *Machine Learning: A multistrategy approach*. 1st. ed. [S.l.]: McGraw-Hill Education (ISE Editions), 1997.

[27] CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. An overview of machine learning. in: Michalski, r.s., carbonell, j.g., mitchell, t.m. *Machine Learning: An Artificial Intelligence Approach, Berlin, Heidelberg: Springer*, p. 3–23, 1984.

[28] KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. In: *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*. NLD: IOS Press, 2007. p. 3–24. ISBN 9781586037802.

[29] BURGES, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, v. 2, n. 2, p. 121–167, June 1998.

[30] SUPPORT Vector Machines. Disponível em: <https://scikit-learn.org/stable/modules/svm.html>.

[31] MÜLLER, A.; GUIDO, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016. ISBN 9781449369897. Disponível em: <https://books.google.com.br/books?id=vbQlDQAAQBAJ>.

[32] SERVICES, A. W. *O que é uma rede neural?* 2020. <https://aws.amazon.com/pt/what-is/neural-network/>.

[33] NIELSEN, M. A. misc, *Neural Networks and Deep Learning*. Determination Press, 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/>.

[34] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Book in preparation for MIT Press. Disponível em: <http://www.deeplearningbook.org>.

[35] BRAGA, A.; LUDERMIR, T.; CARVALHO, A. *Redes neurais artificiais: Teoria e aplicações*. 2nd. ed. [S.l.]: Rio de Janeiro: LTC, 2007.