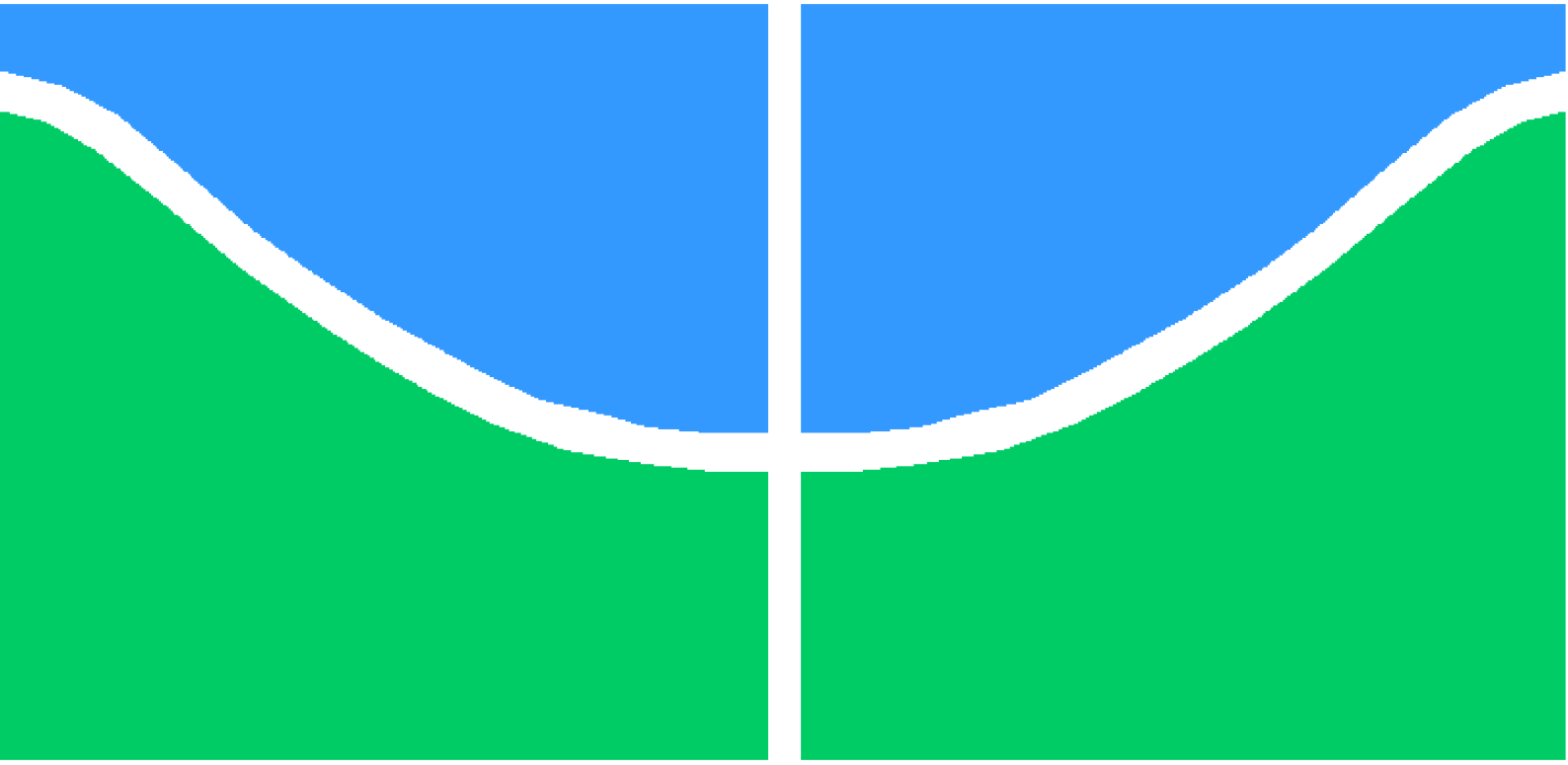


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Reconhecimento de Gestos Estáticos da Mão a partir de Rede Neural Artificial e Coordenadas do Esqueleto

Autor: Matheus Silva Pereira

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2022

Matheus Silva Pereira

Reconhecimento de Gestos Estáticos da Mão a partir de Rede Neural Artificial e Coordenadas do Esqueleto

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2022

Matheus Silva Pereira
Reconhecimento de Gestos Estáticos da Mão a partir de Rede Neural Artificial e Coordenadas do Esqueleto/ Matheus Silva Pereira. – Brasília, DF, 2022-
65 p. : il. (algumas color.) ; 30 cm.

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2022.

1. Reconhecimento de gestos. 2. Rede Neural Artificial. I. Profa. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Reconhecimento de Gestos Estáticos da Mão a partir de Rede Neural Artificial e Coordenadas do Esqueleto

CDU 02:141:005.6

Matheus Silva Pereira

Reconhecimento de Gestos Estáticos da Mão a partir de Rede Neural Artificial e Coordenadas do Esqueleto

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 05 de dezembro de 2022:

Profa. Dra. Carla Silva Rocha Aguiar
Orientador

Profa. Dra. Juliana Petrocchi Rodrigues
Convidado 1

Profa. Dra. Loana Nunes Velasco
Convidado 2

Brasília, DF
2022

Este trabalho é dedicado àqueles que,
assim como eu, não desistiram.

Agradecimentos

Quero expressar meus sinceros agradecimentos à minha família, meu pai Wilton, minha mãe Cleide e meu irmão João pelo apoio, paciência e amor incomensuráveis demonstrados para com a minha pessoa, permitindo-me seguir em frente na árdua tarefa, que para mim, foi a realização deste trabalho.

Agradeço aos genuínos amigos que fiz durante a minha jornada de graduação, em especial, à Vanessa, Ítalo e Emilie pelos momentos ímpares que passamos juntos, tanto de descontração e risadas, como também de estudos e aprendizados. Posso afirmar, com certeza, que conhecer vocês foi o que de melhor me aconteceu em todos esses anos de faculdade. Espero levá-los para a vida.

Por último, mas não menos importante, agradeço à Universidade de Brasília e seu corpo docente por todos os ensinamentos que contribuíram para o meu crescimento intelectual e profissional, em especial à minha orientadora Carla, pela paciência e compreensão a mim prestadas.

A todos, meu muito obrigado!

"A inteligência artificial só será um problema se não reconhecermos o valor do ser humano."
(Raidalva de Castro)

Resumo

A visão computacional nasceu de uma necessidade do homem de fazer com que as máquinas pudessem resolver problemas complexos. Porém, conceder o sentindo da visão às máquinas não é tarefa simples, pois existem algumas dificuldades inerentes ao processo. A fim de contorná-las, vários métodos, abordagens e tecnologias foram desenvolvidas com intuito de distinguir características importantes dos objetos a serem identificados e/ou rastreados. Tendo a tarefa de reconhecimento de gestos estáticos da mão em foco, abordagens baseadas em esqueleto mostram-se especialmente úteis, já que fornecem informações altamente relevantes para a execução da tarefa. Diante disso, o objetivo deste trabalho foi reconhecer gestos estáticos da mão através de uma rede neural do tipo MLP tendo como entrada coordenadas 3D dos pontos do esqueleto. Para isso, um *dataset* da Linguagem de Sinais Americana (ASL) foi utilizado para o treinamento e avaliação da rede e, um outro produzido pelo autor, foi usado para teste do modelo. Os resultados obtidos atestam a importância dessa abordagem, uma vez que obteve-se uma performance com acurácias de 98% e 100% no reconhecimento dos gestos (sinais) no *dataset* da ASL utilizado e no do autor, respectivamente.

Palavras-chaves: Reconhecimento de gestos. Rede Neural Artificial. Visão Computacional.

Abstract

Computer vision was born out of a human need to make machines able to solve complex problems. However, granting the sense of vision to machines is not a simple task, as there are some difficulties inherent to the process. In order to circumvent them, several methods, approaches and technologies were developed in order to distinguish important characteristics of the objects to be identified and/or tracked. With the task of recognizing static hand gestures in focus, skeleton-based approaches prove to be especially useful, as they provide highly relevant information for performing the task. Therefore, the objective of this work was to recognize static hand gestures through a neural network MLP, having as input 3D coordinates of the skeletal joints. For this, an American Sign Language (ASL) dataset was used to train and evaluate the network, and another one produced by the author was used to test the model. The results obtained attest to the importance of this approach, since we obtained a performance with accuracies of 98% and 100% in the recognition of gestures (signs) in the used ASL dataset and in the author's, respectively.

Key-words: Gesture recognition. Artificial Neural Network. Computer Vision.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
ASL	*American Sign Language*
CSV	*Comma Separated Values*
GPU	*Graphics Processing Unit*
HDF5	*Hierarchical Data Format version 5*
IHC	Interação Humano-Computador
JSON	*JavaScript Object Notation*
ML	*Machine Learning*
MLP	*Multilayer Perceptron*
RNA	Rede Neural Artificial
TCU	*Tensor Processing Unit*
2D	Duas Dimensões
3D	Três Dimensões

Sumário

1 Introdução

Dentre os cinco sentidos humanos, a visão é, de acordo com Davies (2012), o sentido pelo qual o homem recebe a maior quantidade de informações do ambiente que o circunda. O número chega a ser, pelo menos, duas ordens de grandeza maior do que o cérebro obtém dos demais sentidos. Além disso, segundo National Research Council (2000) os seres humanos são aptos a reconhecer e interpretar facilmente o corpo e a linguagem de sinais, devido à uma combinação entre visão e interações sinápticas que foram formadas no decorrer do desenvolvimento do cérebro.

No entanto, o ato de ver não é um processo simples. Foram precisos milhões de anos de evolução para que a complexidade da visão fosse abstraída da percepção do homem. Com o intuito de que as máquinas possam realizar e resolver tarefas mais difíceis (como o reconhecimento de gestos) do que simples problemas de mecânica, é preciso fazer com que essas recebam o sentido da visão (DAVIES, 2012).

A fim de fazê-lo, ainda de acordo com Davies (2012), existem algumas dificuldades intrínsecas da implementação de visão de máquina que precisam ser contornadas, como distorções e ruídos nos dados dos padrões a serem identificados ou reconhecidos. Além disso, a depender do número de bits dos padrões, o problema de reconhecimento pode levar um tempo impraticável. Uma forma de lidar com essas dificuldades é distinguir as características mais importantes das demais informações dos padrões, selecionando desta forma, os bits de maior relevância. Tendo em vista a tarefa de reconhecimento de gestos da mão, existem diversas abordagens que buscam solucionar essa questão.

Oudah, Al-Naji e Chahl (2020) elencaram essas abordagens em sua revisão divididas em dois métodos. O primeiro é baseado no uso de sensores (luvas, eletrônicos, etc.) e o segundo apenas em visão computacional sem o auxílio de aparatos externos. Focando-se no segundo método, temos o reconhecimento por cor, aparência, movimento, profundidade, modelo 3D, aprendizagem profunda e esqueleto.

Em meio às diversas abordagens, o reconhecimento baseado em esqueleto se mostra especialmente vantajoso pois é possível extrair uma informação altamente semântica das juntas do esqueleto em um tamanho pequeno de dados (LI et al., 2019). Por fim, é possível, a partir dessas informações, utilizar uma arquitetura simples de Rede Neural Artificial (RNA) para classificar os gestos estáticos da mão com boa acurácia.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral do presente trabalho é reconhecer (classificar) gestos estáticos da mão através de uma rede neural *Multilayer Perceptron* (MLP) utilizando como entrada coordenadas 3D dos pontos do esqueleto, extraídas de imagens de uma base de dados da Linguagem de Sinais Americana (ASL), com o intuito de avaliar a performance desta abordagem.

1.1.2 Objetivos Específicos

1. Implementar um código para extrair as coordenadas de 21 pontos do esqueleto da mão das imagens de 24 sinais estáticos do alfabeto da ASL e gravar em um arquivo;
2. Implementar uma arquitetura de rede MLP para classificar as 24 classes de sinais;
3. Treinar a rede MLP com os dados do arquivo obtido no primeiro passo e criar dois arquivos com a estrutura e os pesos da rede neural treinada;
4. Avaliar o modelo;
5. Implementar um código para classificar uma única imagem de um dos sinais treinados usando os arquivos da rede neural criados no terceiro passo.

2 Referencial Teórico

Nesta seção serão apresentados os conceitos, métodos e trabalhos relevantes que fornecem o embasamento teórico ao tema desta monografia.

2.1 Gestos Estáticos e Dinâmicos

Um gesto pode ser definido como uma forma de comunicação onde o movimento do corpo é utilizado para transmitir uma mensagem de maneira não verbal ou não vocal. Os gestos se originam de várias partes do corpo, porém, os mais comuns e expressivos são os que surgem das mãos e da face. A maior classe de gestos é representada pelas mãos, pois sua forma possibilita um grande número de arranjos possíveis nitidamente distinguíveis. Portanto, os gestos da mão são de suma importância para a linguagem de sinais (AL-SAEDI; AL-ASADI, 2019).

Via de regra, os gestos da mão podem ser divididos em duas categorias: estáticos e dinâmicos. Os gestos estáticos usualmente são representados pela postura ou forma das mãos e os gestos dinâmicos, por sua vez, são representados de acordo com o movimento das mãos (CHANG et al., 2006).

Os gestos estáticos ou posturas são constantes ao longo do tempo e podem ser compreendidos totalmente com apenas uma imagem, enquanto os gestos dinâmicos são uma sequência de posturas que só podem ser identificados quando analisado o contexto temporal da informação (AL-SAEDI; AL-ASADI, 2019).

Visto que os gestos estáticos conseguem por si só transmitir significado e, além disso, representar as transições específicas nos gestos dinâmicos, reconhecê-los é uma das partes mais importantes no reconhecimento de gestos em geral (CHANG et al., 2006).

2.2 Técnicas para Reconhecimento de Gestos da Mão

O reconhecimento de gestos é de extrema importância na criação de sistemas de interação humano-computador (IHC). As aplicações do reconhecimento de gestos são vastas e vão desde o reconhecimento de linguagem de sinais, passando por monitoramento do estado de alerta de motoristas até navegação e manipulação em ambientes virtuais (MITRA; ACHARYA, 2007).

Os gestos da mão, como visto na seção 2.1, constituem a maior classe de gestos e, por consequência, possuem um amplo campo de pesquisa. Existem diversos trabalhos na literatura que adotam muitas técnicas e abordagens diferentes para o reconhecimento de gestos da mão, sendo essas baseadas em tecnologia de sensores instrumentados e visão computacional (OUDAH; AL-NAJI; CHAHL, 2020).

A Figura 1 detalha as diferentes técnicas e abordagens para o reconhecimento de gestos da mão.

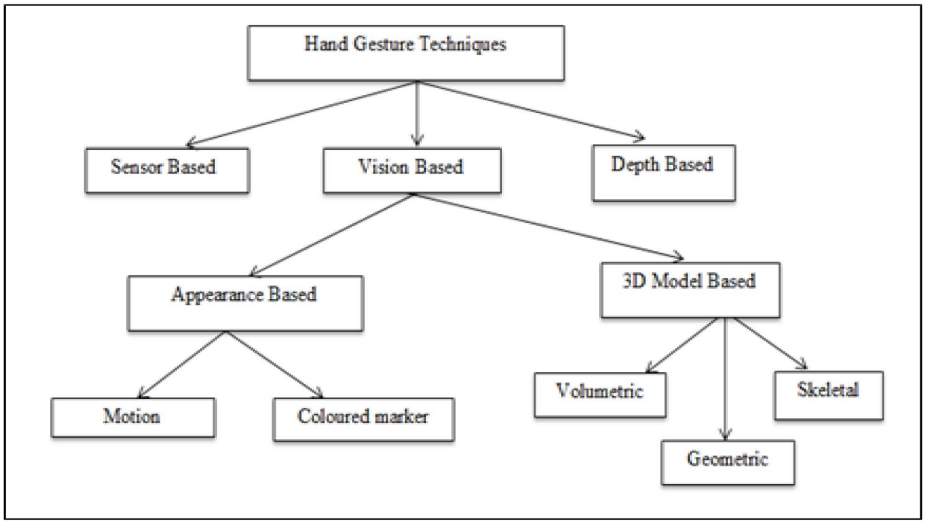


Figura 1 – Classificação de técnicas para o reconhecimento de gestos da mão. Fonte: (KAUR; RANI, 2016)

Mitra e Acharya (2007) defendem que técnicas que envolvem o uso de sensores e afins, embora apresentem bons resultados, dificulta a facilidade e naturalidade da interação do usuário com o computador. Também pode gerar desconforto e confusão devido a problemas de configuração e conexões de fios, além de que alguns sensores, a depender do tipo, podem ter um custo elevado.

Dessa forma, se deu o desenvolvimento de técnicas de baixo custo que substituíram o uso de luvas instrumentadas (sensores) por uma ou mais câmeras.

2.2.1 Técnica Baseada em Visão Computacional

Nesta técnica o uso de câmera(s) de vídeo é o único requisito necessário para a interação entre o usuário e o computador, o que a torna simples, natural e conveniente (GARG; AGGARWAL; SOFAT, 2009).

A Figura 2 ilustra o processo para a extração e reconhecimento de gestos da mão com base em visão computacional. O usuário realiza um gesto específico, com uma ou duas mãos, que é capturado pela(s) câmera(s) e processado por um *framework* na máquina utilizando uma das possíveis abordagens para extrair características e classificar o gesto em questão. A partir do resultado pode-se executar alguma ação em uma das aplicações possíveis.

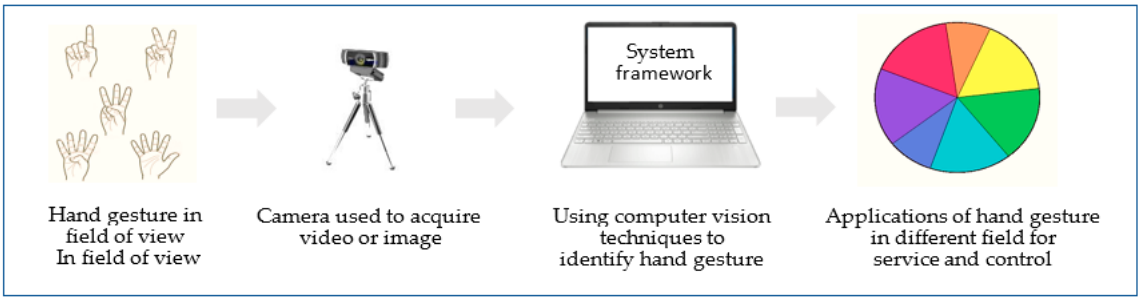


Figura 2 – Processo de reconhecimento de gestos baseado em visão computacional. Fonte: (OUDAH; AL-NAJI; CHAHL, 2020)

Entretanto, o uso desse tipo de técnica envolve alguns desafios a serem tratados, como por exemplo, mudanças na iluminação, oclusão parcial ou total da(s) mão(s) e/ou dedos, cenários de fundo dinâmicos e complexos e etc (OUDAH; AL-NAJI; CHAHL, 2020).

2.2.1.1 Representações de Gestos da Mão

As abordagens baseadas em visão são, comumente, caracterizadas por dois tipos de representações (modelagem) de gestos (Fig. 3) (PAVLOVIC; SHARMA; HUANG, 1997).

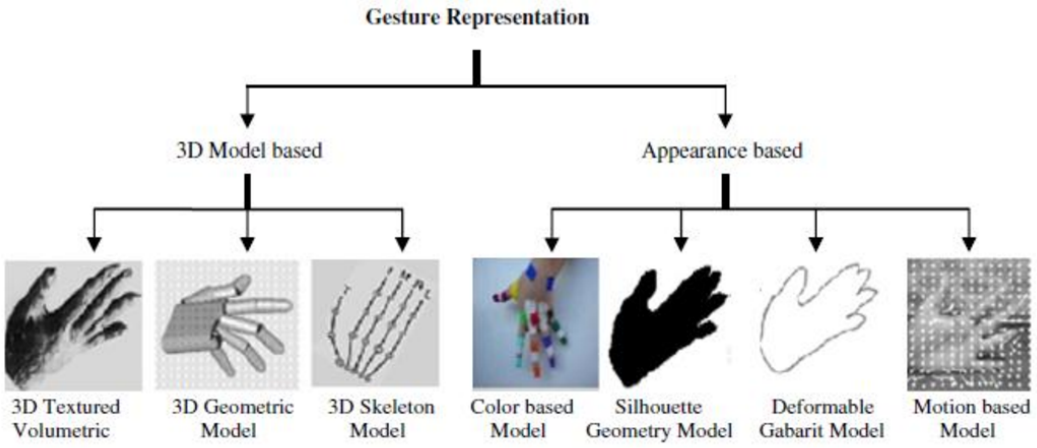


Figura 3 – Modelos de gestos da mão baseados em visão. Fonte: (BOURKE; O'BRIEN; ÓLAIGHIN, 2007)

Na modelagem baseada em aparência busca-se extrair características a partir de atributos visuais (cor, silhueta, posição, movimento e etc.), inferidos diretamente de imagens 2D, para criar uma representação (modelo) da aparência da mão (gesto). Esses parâmetros são então comparados às características extraídas das imagens de entrada para reconhecer gestos específicos. Essa modelagem é considerada mais simples, em relação ao modelo 3D, pois é mais fácil trabalhar com propriedades de imagens bidimensionais. (GARG; AGGARWAL; SOFAT, 2009) Apesar disso, é mais suscetível a ruídos como variações de iluminação e objetos em segundo plano (AL-SAEDI; AL-ASADI, 2019).

Já na modelagem baseada em modelos 3D utiliza-se uma representação visual cinemática tridimensional da mão para analisar os gestos. Os parâmetros de movimento (posição da palma, ângulos das articulações e etc.) da mão são estimados das imagens de entrada por comparação com a possível aparência bidimensional do gesto projetada a partir do modelo 3D. Os gestos são reconhecidos através de tais movimentos (GARG; AGGARWAL; SOFAT, 2009).

Os modelos 3D podem ser classificados em volumétricos e esqueléticos. O modelo volumétrico visa descrever a aparência visual 3D elaborada da mão. Porém este tipo de modelagem pode ser um problema, visto que lida com vários parâmetros da mão que possuem uma grande dimensionalidade, sendo complexa e muito custosa computacionalmente falando. O modelo esquelético é uma alternativa para lidar com esse tipo de problema, pois limita o número de parâmetros para gerar a estrutura tridimensional da forma do esqueleto da mão (PAVLOVIC; SHARMA; HUANG, 1997).

Em geral, abordagens com base em aparência têm desempenho melhor na detecção de gestos em tempo real em comparação com abordagens baseadas em modelos 3D. Todavia, os modelos 3D podem representar uma grande variedade de gestos (AL-SAEDI; AL-ASADI, 2019).

2.2.1.2 Abordagem Baseada em Esqueleto

Conforme Kaur e Rani (2016), o modelo esquelético contém parâmetros que otimizam a detecção de características complexas. Há várias formas de representação de dados do esqueleto do modelo que são usadas para a classificação de gestos, onde as características mais comuns utilizadas são a orientação, localização e espaço entre as articulações esqueléticas (pontos do esqueleto). E ainda, ângulos, trajetória e curvatura das articulações (OUDAH; AL-NAJI; CHAHL, 2020).

2.2.1.2.1 *Google's MediaPipe Hands*

Zhang et al. (2020) propuseram uma solução de alta fidelidade para o rastreamento da mão em tempo real, implementado via *Google Mediapipe*, que estima o esqueleto a

partir de um único *frame*. A solução localiza vinte e um pontos 3D do esqueleto da mão (Fig. 4) utilizando um *Machine Learning (ML) pipeline* que consistem em dois modelos trabalhando em conjunto:

1. Um detector de palma que atua na imagem de entrada inteira e localiza a palma da(s) mão(s) delimitando-as em caixas (*bounding box*);

2. Um modelo chamado *hand landmark model* que opera na imagem cortada, fornecida pelo detector de palma, estimando o esqueleto. Este modelo possui três saídas:

   a) 21 (vinte e um) pontos do esqueleto que consistem em x, y e profundidade relativa;

   b) Um indicador da probabilidade de presença de mão(s) na imagem de entrada;

   c) Um classificador de lateralidade, indicando se é a mão esquerda ou direita.

O modelo se mostra robusto mesmo em cenários de mãos parcialmente visíveis e auto-oclusões, o que permite estimar gestos estáticos básicos com qualidade razoável, podendo ser usado em aplicações como o reconhecimento de gestos.

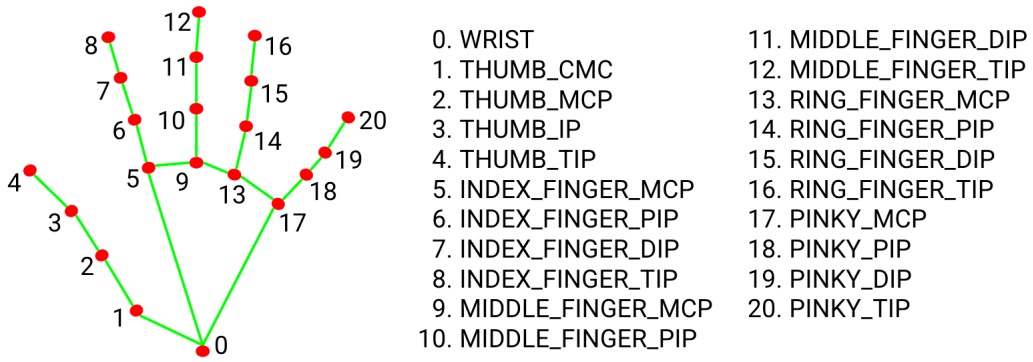


Figura 4 – Pontos do esqueleto da mão. Fonte: Site do MediaPipe[1]

.

2.2.1.3 Processo de Sistemas de Reconhecimento de Gestos

Um sistema típico de reconhecimento de gestos da mão baseado em visão computacional, segundo Choudhury, Talukdar e Sarma (2015) abrange quatro importantes fases internas (Fig. 5): segmentação, rastreamento, extração de características e classificação.

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

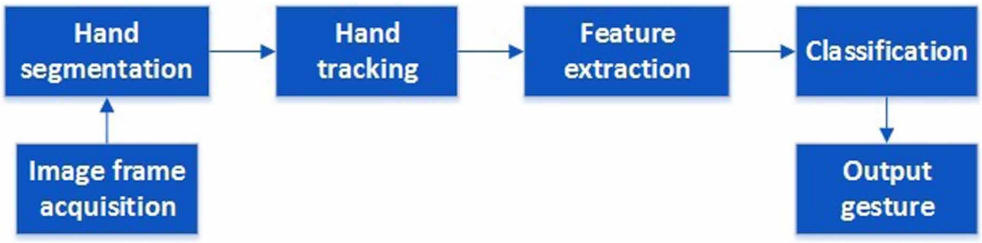


Figura 5 – Processo geral de sistemas de reconhecimento de gestos baseados em visão. Fonte: (CHOUDHURY; TALUKDAR; SARMA, 2015)

Após a captura dos *frames*, na fase de segmentação, as mãos são detectadas e então as partes correspondentes à elas são segmentadas (separadas) do plano de fundo da imagem. No rastreamento, utiliza-se as informações espaciais e temporais de *frames* consecutivos para calcular as trajetórias dos movimentos das mãos durante a execução de um gesto. Na fase de extração de características as informações úteis da trajetória ou postura da mão são extraídas para que sirvam de entrada em algoritmos de classificação (CHOUDHURY; TALUKDAR; SARMA, 2015). A forma com que o sistema de reconhecimento realiza as etapas retratadas até o momento, muda de uma aplicação para outra. As diferentes abordagens apresentadas e, algumas, discutidas nas seções anteriores são utilizadas para tal.

Por fim, na etapa de classificação utiliza-se as características extraídas na fase anterior para identificar os gestos específicos. Classificação, conforme Theodoridis e Koutroumbas (2009), é um método estatístico que recebe um conjunto de características (dados) como entrada e fornece uma saída rotulada de classe. Um dos classificadores que é amplamente utilizado no reconhecimento de gestos é a Rede Neural Artificial (*feedfoward*). Esse tipo de classificador é preferencialmente usado para representar e reconhecer gestos estáticos, pois gestos dinâmicos mudam ao longo do tempo e uma única RNA (*feedforward*) treinada não é adequada para interpretar esse tipo de gesto (CHOUDHURY; TALUKDAR; SARMA, 2015).

2.3 Rede Neural Artificial (RNA)

De acordo com Braga, Ludermir e Carvalho (2000), RNAs são sistemas paralelos distribuídos constituídos por um grupo de unidades de processamento (neurônios ou nodos) que realizam o calculo de funções matemáticas não-lineares (em sua grande maioria). Essas unidades compõem um rede de camadas interconectadas por um grande número de conexões, geralmente unidirecionais, que são associadas a pesos. O funcionamento desta rede é inspirado na estrutura física do cérebro humano.

A solução de problemas por meio de RNAs dá-se a partir de um procedimento de aprendizagem, onde um conjunto de dados é inserido na rede, que por sua vez, extrai

de forma automática as características necessárias para representar os dados fornecidos. Estas características são usadas, posteriormente, para gerar respostas para o problema em contexto.

A capacidade de aprendizagem e generalização de informações das RNAs, as tornam atrativa para a solução de problemas complexos. Por outro lado, o tipo e complexidade do problema a ser resolvido pela rede se restringe a sua arquitetura. Por exemplo, as redes de uma só camada (redes *Perceptron*) lidam apenas com problemas linearmente separáveis, enquanto que redes com múltiplas camadas (MLP) possuem a capacidade de resolver problemas mais difíceis (não linearmente separáveis) (BRAGA; LUDERMIR; CARVALHO, 2000).

2.3.1 Redes *Multilayer Perceptron* (MLP)

Uma rede MLP, ou ***perceptrons* de múltiplas camadas**, é composta por uma camada de entrada (*Input Layer*), uma ou mais camadas escondidas (*Hidden Layer(s)*) e uma camada de saída (*Output Layer*) (Fig. 6) (DUDA; HART; STORK, 2000).

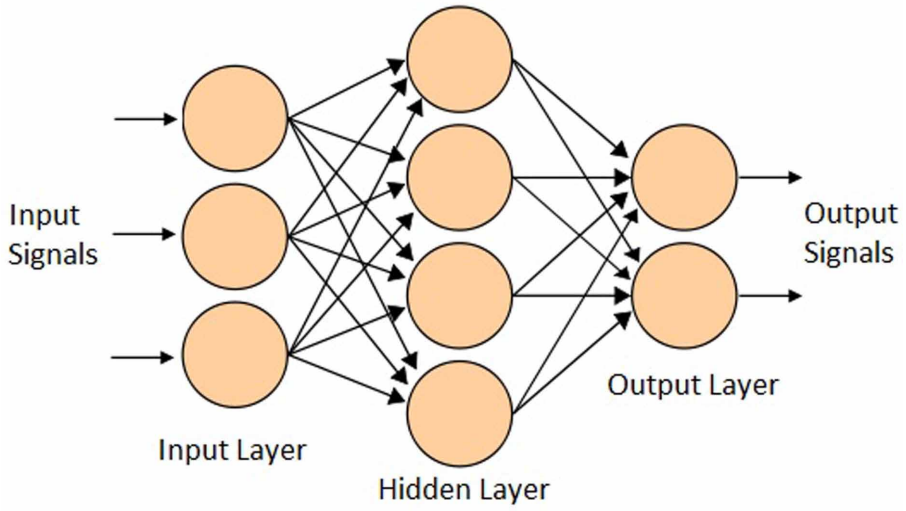


Figura 6 – Representação Gráfica de uma rede MLP. Fonte: (CHOUDHURY; TALUKDAR; SARMA, 2015)

Redes MLP são *feedforward* ou acíclicas, onde o sinal de entrada se propaga para frente, sempre em uma única direção, camada por camada. Ou seja, não há realimentação (*feedback*). Os *perceptrons* de múltiplas camadas possuem treinamento supervisionado que utiliza um algorítimo conhecido como *back-propagation* baseado na regra de aprendizagem por correção de erro. O processo de aprendizagem baseado nesta regra consiste, basicamente, na realização de dois passos: propagação e retro-propagação (HAYKIN, 2007).

Resumidamente, na propagação os sinais de entrada são aplicados aos nodos da rede e se propagam por todas as camadas até que um conjunto de saídas é obtido como

resposta real da rede. Tal resposta é então comparada à resposta desejada (alvo) para produzir um sinal de erro. Durante esta etapa os pesos da rede permanecem fixos. Então, na retro-propagação, este sinal de erro é propagado para trás através da rede e os pesos são ajustados para fazer com que a resposta real da rede caminhe para mais perto do alvo, em uma perspectiva estatística. Este tipo de aprendizagem com o algorítimo *back-propagation* é denominada aprendizagem por retro-propagação (HAYKIN, 2007).

Portanto, esse tipo de rede é utilizado no reconhecimento de padrões, onde os sinais na camada de entrada correspondem aos componentes de um vetor de características (a serem aprendidas ou classificadas) e os sinais emitidos nas unidades de saída são produtos de funções discriminantes usadas para classificação (DUDA; HART; STORK, 2000).

3 Metodologia

Nesta seção serão detalhados procedimento metodológico, ferramentas e base de dados adotados para o reconhecimento de gestos estáticos da mão proposto como objetivo geral e específicos desta monografia.

3.1 Procedimento Metodológico

O fluxograma do procedimento metodológico pode ser visualizado na Fig. (7). A abordagem para o reconhecimento de gestos consiste em duas fases: Fase de Treinamento e Avaliação e Fase de Teste. Na primeira fase, um subconjunto dos sinais estáticos do alfabeto de uma base de dados da Linguagem de Sinais Americana (ASL) foi utilizado para a criação de um *dataset* com as coordenadas dos pontos do esqueleto da mão dos diferentes sinais. E então, este foi usado para o treinamento e avaliação de uma rede MLP. Já na fase de teste, utilizou-se a rede MLP já treinada para o reconhecimento dos gestos (sinais) a partir de imagens feitas pelo autor.

As fases, processos, *datasets* e arquivos são detalhados a seguir.

3.1.1 Fase de Treinamento e Avaliação

O processo iniciou-se a partir da definição e tratamento da base de dados (*dataset*) ASL.

3.1.1.1 *Dataset ASL*

O *Dataset* utilizado é de autoria de Barczak et al. (2011). O mesmo é composto por 2425 imagens de 36 sinais da ASL feitos por cinco indivíduos diferentes, incluindo os sinais dos números (0-9) e letras do alfabeto (A-Z). As imagens foram capturadas em um ambiente controlado e processadas com auxílio de técnicas de processamento de imagem para a segmentação e redução de ruídos. Variações nas condições de iluminação, ângulos e postura da mão foram intencionalmente executadas com intuito de fornecer o maior número de exemplos e cenários possíveis para que técnicas e algoritmos de *machine learning* possam ser eficientemente treinados.

Na Figura 8 pode-se ver exemplos do sinal da letra 'A' do alfabeto da ASL realizados por dois indivíduos diferentes (cima para baixo). É possível notar a variação de iluminação e postura. A iluminação varia (esquerda para direita) na figura em: inferior, difusa, esquerda, direita e topo.

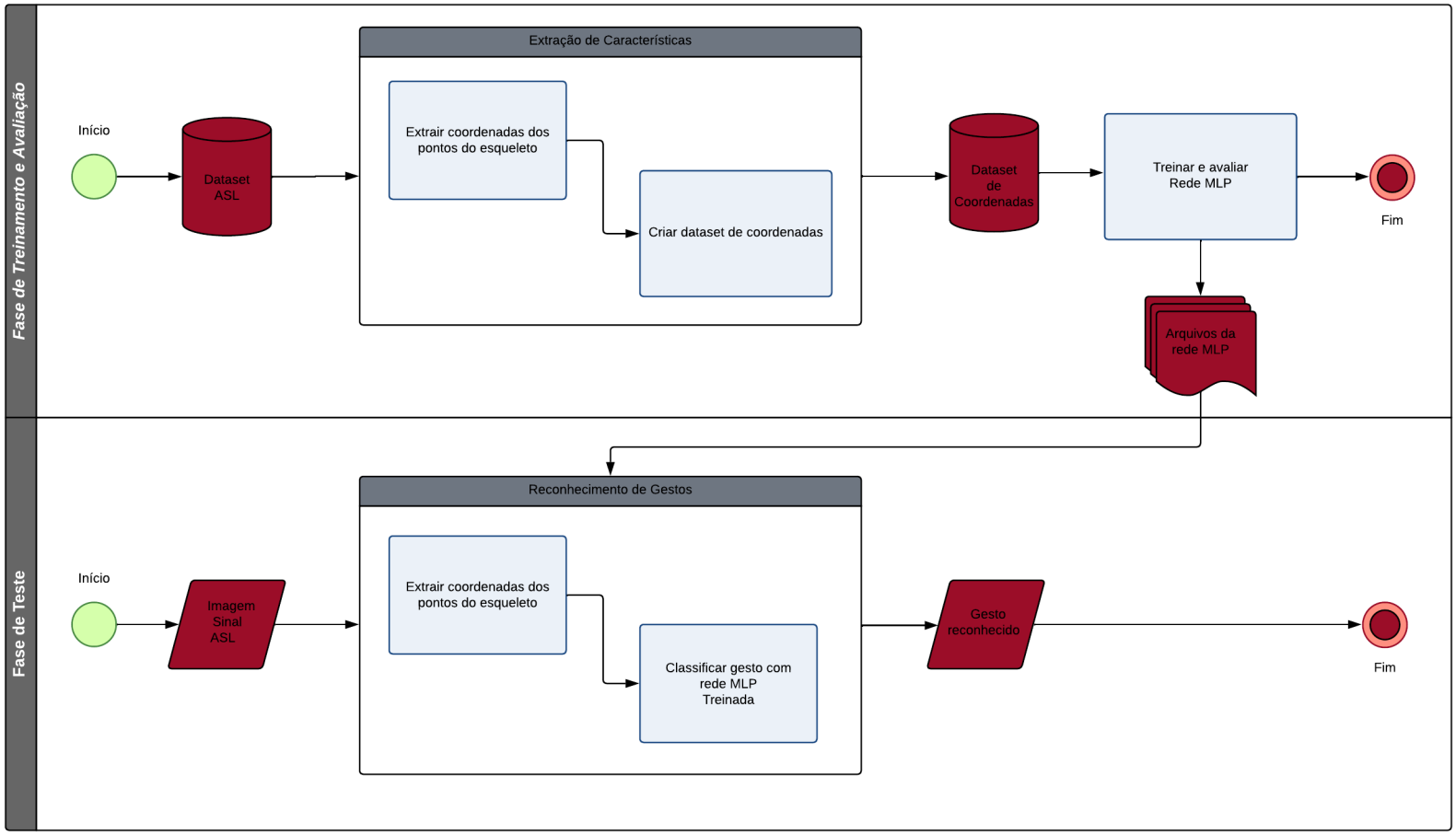


Figura 7 – Fluxograma do processo metodológico adotado.

Figura 8 – Exemplos do sinal 'A' do *dataset* da ASL.

Para o escopo deste trabalho, utilizou-se um subconjunto do *dataset* contendo apenas os sinais estáticos do alfabeto da ASL. Dessa forma, os sinais correspondentes aos números (0-9) e às letras 'J' e 'Z' (gestos dinâmicos) não foram utilizados. Portanto, o *dataset* final utilizado possui 1675 imagens de 24 sinais, sendo 70 imagens para cada sinal do alfabeto.

3.1.1.2 Extração de Características

Nesta etapa as imagens dos sinais do *dataset* ASL final (já tratado) foram processadas utilizando a solução do *Google MediaPipe Hands* para a extração das coordenadas de 21 pontos do esqueleto da mão. Após a extração um *dataset* foi criado contendo as informações das coordenadas extraídas. Para isso, foi implementado um código em *Python* (versão 3.7.13) em uma máquina virtual do *Google Colaboratory*, que já possui um ambiente configurado com as principais bibliotecas de *machine learning*, inteligência artificial e análise de dados. E, além disso, é possível a utilização de GPUs e TCUs para agilizar o processamento de imagens e treinamento dos algoritmos.

3.1.1.2.1 Extrair Coordenadas dos pontos do esqueleto

Para a extração das coordenadas, utilizou-se a *Python Solution API*[1] (versão 0.8.9.1) do *MediaPipe*. A solução fornece três saídas após o processamento da imagem:

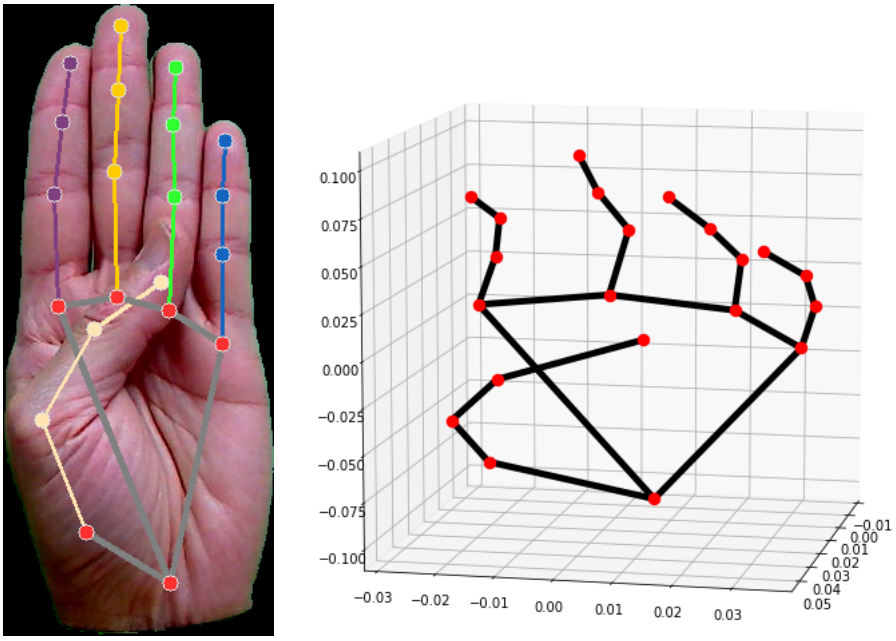
- *MULTI_HAND_LANDMARKS*: Uma coleção das mãos detectadas/rastreadas, onde cada mão é representada por uma lista de 21 pontos de referência (pontos do es-

[1] Documentação disponível em: https://google.github.io/mediapipe/solutions/hands.html#solution-apis

queleto) e cada um possui as coordenadas x, y e z (Fig. 9 (a)). Os valores de x e y correspondem a largura e altura da imagem onde se encontra o ponto em questão, e z representa a profundidade do ponto em relação à câmera. Quanto menor o valor de z, mais próximo o ponto está da câmera;

- *MULTI_HAND_WORLD_LANDMARKS*: Uma coleção de mãos detectadas/rastreadas, onde cada mão é representada por uma lista de 21 pontos de referência (pontos do esqueleto) em coordenadas reais (Fig. 9 (b)). Cada ponto é formado por x, y e z que são coordenadas 3D do mundo real em metros com a origem no centro geométrico aproximado da mão;

- *MULTI_HANDEDNESS*: Uma coleção de lateralidade das mãos detectadas/rastreadas, onde é atribuído para cada mão um *label* e uma pontuação. O *label* é uma *string* de valor *Left* ou *Right* e a pontuação é a probabilidade estimada da lateralidade.



(a) (b)

Figura 9 – Saídas da solução do *MediaPipe* para o sinal 'B' da ASL.

Para fins de treinamento da rede MLP, usou-se as coordenadas 3D do mundo real em metros, uma vez que estas são inferidas a partir do centro geométrico aproximado da mão e não possuem relação com a posição em que a mão se encontram na imagem, fornecendo, dessa forma, dados confiáveis para o treinamento do algoritmo.

Por fim, as 1675 imagens de sinais foram processadas para a extração das coordenadas 3D dos 21 pontos do esqueleto da mão. O código para a extração de coordenadas encontra-se no apêndice A.

3.1.1.2.2 Criar *dataset* de coordenadas

Nesta etapa criou-se um arquivo CSV contendo as coordenadas dos pontos do esqueleto da mão extraídas das imagens dos sinais processadas no passo anterior.

- ***Dataset* de coordenadas**

O *dataset* de coordenadas (Fig. 10) é formado por 64 colunas rotuladas, sendo 63 colunas correspondentes às coordenadas 3D de cada um dos 21 pontos do esqueleto (onde cada ponto possui coordenadas x, y e z) e uma coluna que indica a classe do sinal. A classe dos sinais é um número de 0 à 23, onde o 0 representa o sinal 'A', 1 o sinal 'B' e segue assim (em ordem alfabética) até o 23 que corresponde ao sinal 'Y'. Lembrando que os sinais 'J' e 'Z' não estão no escopo por se tratarem de sinais dinâmicos da ASL.

THUMB_MCPx	THUMB_MCPy	THUMB_MCPz	THUMB_IPx	...	PINKY_PIPx	PINKY_PIPy	PINKY_PIPz	PINKY_DIPx	PINKY_DIPy	PINKY_DIPz	PINKY_TIPx	PINKY_TIPy	PINKY_TIPz	CLASS
0.050233	0.015462	-0.001144	0.044805	...	-0.038351	0.016146	-0.026220	-0.024097	0.029614	-0.036051	-0.017673	0.037162	-0.020410	0
0.045784	0.024090	-0.021246	0.046297	...	-0.035816	0.002813	-0.027232	-0.022762	0.013940	-0.042476	-0.021741	0.027300	-0.030225	0
0.048201	0.022854	-0.026555	0.045784	...	-0.035744	0.003307	-0.029363	-0.024594	0.015536	-0.044417	-0.023017	0.026570	-0.031134	0
0.045306	0.023747	-0.021973	0.046496	...	-0.036339	0.002605	-0.027781	-0.022496	0.014111	-0.042588	-0.021477	0.028029	-0.030237	0
0.046323	0.022925	-0.022097	0.049161	...	-0.032162	0.006512	-0.032452	-0.020096	0.018824	-0.043651	-0.023923	0.028630	-0.027866	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.050416	0.025647	-0.014709	0.068518	...	-0.041572	-0.009614	-0.004348	-0.042906	-0.032528	-0.021425	-0.045056	-0.042653	-0.034453	23
0.054671	0.015373	-0.016243	0.063098	...	-0.039664	-0.003736	-0.021106	-0.036239	-0.025313	-0.034195	-0.031498	-0.039940	-0.043952	23
0.054419	0.015753	-0.027470	0.057944	...	-0.037890	-0.011814	-0.023195	-0.034776	-0.033580	-0.035455	-0.028827	-0.050963	-0.045614	23
0.044022	0.035231	-0.021250	0.064465	...	-0.039685	-0.018694	-0.009412	-0.033450	-0.038870	-0.025313	-0.030642	-0.052258	-0.032281	23
0.048494	0.028947	-0.019870	0.065085	...	-0.041434	-0.007061	-0.000956	-0.046954	-0.025329	-0.014980	-0.049965	-0.037364	-0.023569	23

Figura 10 – *dataset* de coordenadas.

3.1.1.3 Treinar e avaliar Rede MLP

Para o treinamento e avaliação da rede MLP, primeiramente, carregou-se os dados do *dataset* de coordenadas com a biblioteca **pandas** (versão 1.3.5). Logo após, a biblioteca **sklearn** (versão 1.0.2) foi usada para a criação das bases de treino e teste. Aplicou-se um embaralhamento (*shuffle*) nos dados antes da divisão, para evitar que o algoritmo da RNA aprendesse alguma informação relacionada a ordem dos dados.

Posteriormente, uma rede MLP foi implementada com a API **keras** da biblioteca tensorflow (versão 2.8.0). A rede é composta por quatro camadas: Uma camada de entrada contendo 63 neurônios (nodos) correspondentes ao número de coordenadas do vetor de características, duas camadas escondidas (*hidden layers*) contendo 66 neurônios cada e uma camada de saída com 24 neurônios que correspondem às classes de sinais do alfabeto da ASL. A função de ativação utilizada nas camadas foi a *ReLu*, exceto na camada de saída, onde se utilizou a função *Softmax* que fornece um vetor de probabilidades onde o maior valor corresponde à classe prevista para o sinal de entrada. A estrutura e parâmetros da rede podem ser vistos na Fig. (11).

O número de camadas escondidas foi definido de acordo com o trabalho de Cybenko (1988) que diz que, teoricamente, duas camadas intermediárias (escondidas) são suficientes para aproximar qualquer função matemática. Já a quantidade de neurônios presentes nessas camadas foram definidos conforme método apresentado por Heaton (2008): O número de neurônios ocultos deve ser 2/3 do tamanho da camada de entrada, mais o tamanho da camada de saída. Assim sendo, as camadas escondidas devem ter (63 * 2/3) + 24 = 66 neurônios.

Ao final, a rede MLP foi compilada com o método otimizador do gradiente descendente *Adam*, a função de erro *SparseCategoricalCrossentropy* e as métricas *Accuracy*. E então, executou-se o treinamento da rede e a posterior avaliação. O código pode ser consultado no apêndice B.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense (Dense)                (None, 66)                4224

 dense_1 (Dense)              (None, 66)                4422

 dense_2 (Dense)              (None, 24)                1608

=================================================================
Total params: 10,254
Trainable params: 10,254
Non-trainable params: 0
_________________________________________________________________
```

Figura 11 – Estrutura da rede MLP.

- **Arquivos da rede MLP**

Após o treinamento da rede MLP, gerou-se dois arquivos. Um arquivo no formato JSON contendo a estrutura da rede neural e outro arquivo no formato HDF5 contendo os pesos da rede MLP já treinada.

3.1.2 Fase de Teste

Nesta fase utilizou-se a rede MLP treinada para o reconhecimento dos sinais estáticos da ASL. Para isso, o autor criou um banco de imagens (Fig. 12) contendo os 24 sinais treinados na fase anterior (uma imagem para cada sinal). Então, Implementou-se um código (Apêndice C) em *Python* (versão 3.7.13) no *Google Colaboratory* para ler uma imagem e classifica-lá.

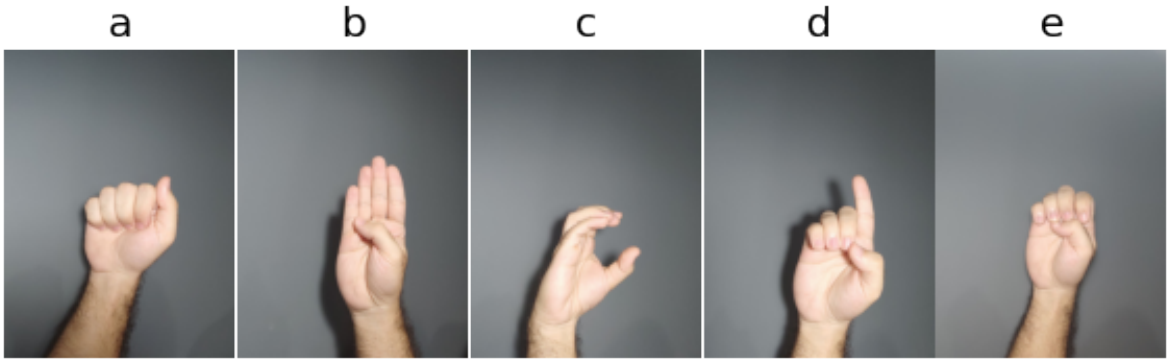


Figura 12 – Exemplos do banco de dados de imagens feitas pelo autor.

3.1.2.1 Reconhecimento de Gestos

Esta etapa consiste em dois processos: Extrair coordenadas dos pontos do esqueleto e Classificar gesto com rede MLP treinada. No primeiro processo utilizou-se a API do *MediaPipe* para fazer a extração das coordenadas 3D dos pontos de referência da mão (assim como descrito na subseção 3.1.1.2.1). E, no processo seguinte, os arquivos gerados na fase anterior foram usados para compilar a rede MLP e classificar os sinais do banco de imagens do autor.

Em resumo, uma imagem de um dos sinais treinados da ASL é carregada e processada para extração das coordenadas. Então, o vetor de características com as coordenadas é submetido como entrada à rede MLP, que por sua vez, prevê (classifica) a qual gesto pertence a entrada em questão. A previsão da rede é então impressa na imagem carregada inicialmente junto com os pontos do esqueleto identificados pela solução do *MediaPipe*, como ilustra a Fig. (13).

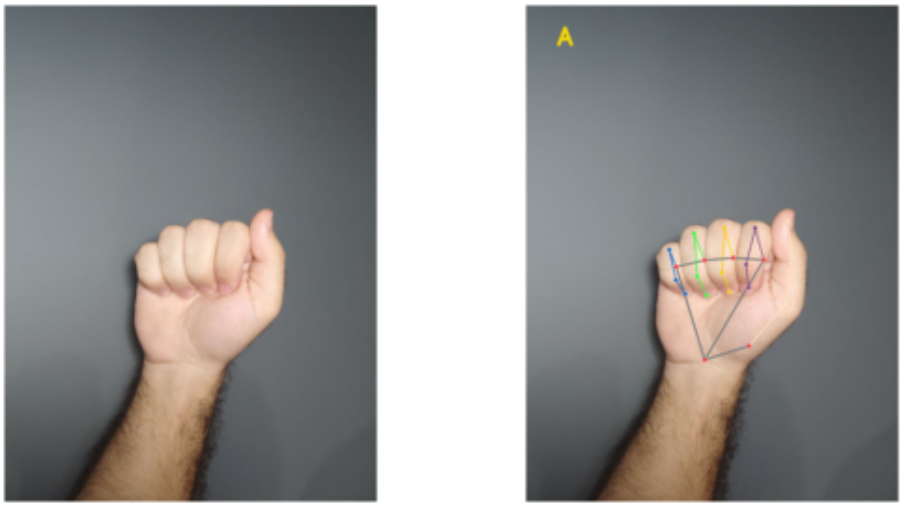


Figura 13 – Entrada e saída da etapa de reconhecimento de gestos.

4 Experimentos e Resultados

Nesta seção serão apresentados os experimentos realizados nas fases do processo metodológico e seus respectivos resultados.

4.1 Fase de Treinamento e Avaliação

Nesta fase, primeiramente, realizou-se um experimento de extração das coordenadas 3D a fim de calibrar os parâmetros da API do *MediaPipe*. E, posteriormente, o treinamento e avaliação da rede MLP.

4.1.1 Calibração da API do *MediaPipe*

Para a calibração da extrator de coordenadas criou-se um subconjunto experimental do *Dataset* da ASL contendo uma imagem para cada sinal do alfabeto com a finalidade de economizar tempo e recursos de processamento. As imagens foram selecionadas aleatoriamente no que se refere aos parâmetros: variação de iluminação, ângulos e diferentes indivíduos.

Realizou-se então um primeiro processamento no subconjunto experimental de imagens com as configurações da API conforme Tab. (1).

Tabela 1: Opções de configuração da API do *MediaPipe*.

Opções de Configuração	**Descrição**	**Valor**
stactic_hand_mode	Quando *false* a solução trata as imagens de entrada como um *stream* de vídeo	***True***
max_num_hands	Número máximo de mãos a serem detectadas	**1**
min_detection_confidence	Valor mínimo de confiança do modelo de detecção de mão [0.0 à 1.0] para que a detecção seja considerada bem-sucedida	**0.5 (*default*)**

Após o processamento, obteve-se as coordenas 3D dos pontos do esqueleto da mão apenas para 11 das 24 classes de sinais do subconjunto experimental. Os sinais a, c, e, h, i, m, n, q, s, t, u, w e y não foram detectados com sucesso. Observou-se então que o modelo detector de palma da solução do *MediaPipe* (descrito na seção 2.2.1.2.1) estava com dificuldades para delimitar o *bounding box* da mão, uma vez que as imagens do *dataset* da ASL utilizado já são cortadas. Nota-se na Fig. (14) a confusão do modelo na delimitação da área da palma da mão (linhas cor de cinza).

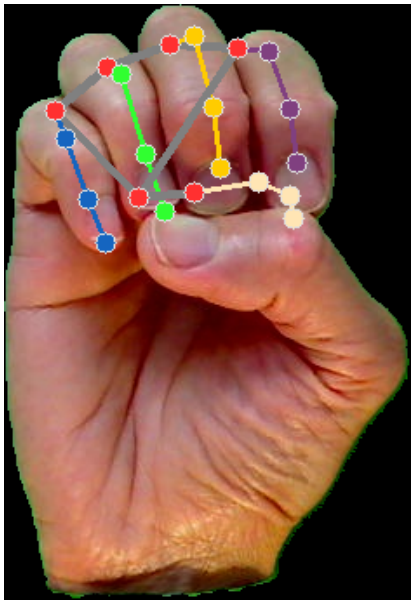


Figura 14 – Exemplo de confusão do modelo de detecção de palma do *MediaPipe*.

Com o fim de corrigir isso, redimensionou-se as imagens adicionando um fundo de 500 *pixels* em todas as direções, como ilustrado na Fig. (15).

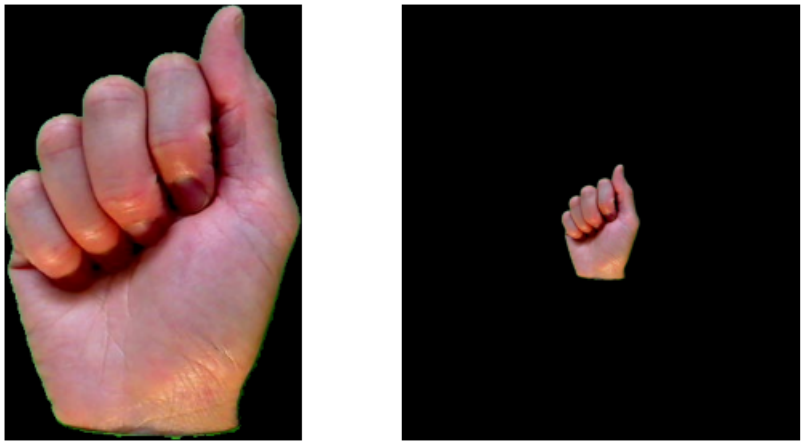


Figura 15 – Exemplo de uma imagem do sinal 'A' redimensionada.

Um segundo processamento foi feito com o subconjunto experimental redimensionado e, para uma *min_detection_confidence* de 0.5 (*default*), obteve-se sucesso na detecção de 20 classes de sinais, com exceção dos sinais c, m, o e w. Possivelmente, devido a oclusão parcial ou total dos dedos nestes sinais (Fig. 16).

Outras rodadas de processamento foram executadas variando-se o valor do parâmetro *min_detection_confidence*, até que para um valor de 0.3, todas as coordenadas para as classes de sinais do subconjunto experimental de imagens foram extraídas com êxito. Utilizou-se então os mesmos parâmetros (Tab. 2) para a extração final de coordenadas no *dataset* da ASL redimensionado.

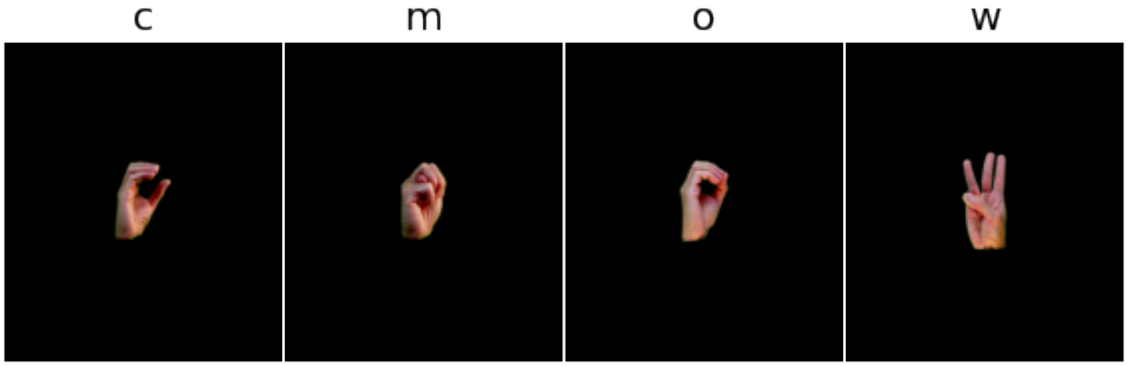


Figura 16 – Sinais não detectados no segundo processamento.

Tabela 2: Opções de configuração utilizada no dataset da ASL.

Opções de Configuração	**Valor**
stactic_hand_mode	***True***
max_num_hands	**1**
min_detection_confidence	**0.3**

O *dataset* da ASL foi então processado e obtiveram-se 1605 coordenadas 3D de pontos de referência da mão de um total de 1675 imagens de sinais. Os sinais com as quais a API do *MediaPipe* teve mais dificuldade foram: c, o, p, u e w. A Figura (17) mostra um gráfico com o número de imagens que tiveram as coordenadas extraídas com sucesso para cada classe de sinal da ASL (cada sinal possui um total de 70 imagens). Estas 1605 coordenadas compuseram o *dataset* final de coordenadas.

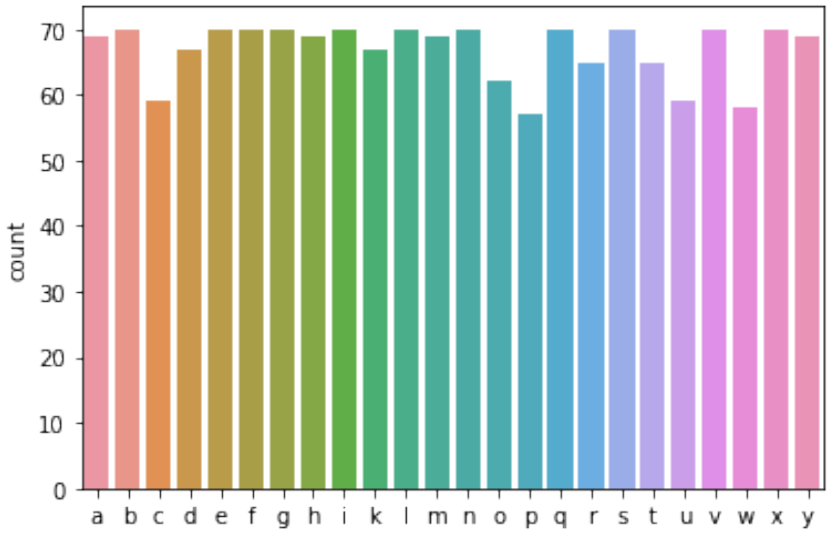
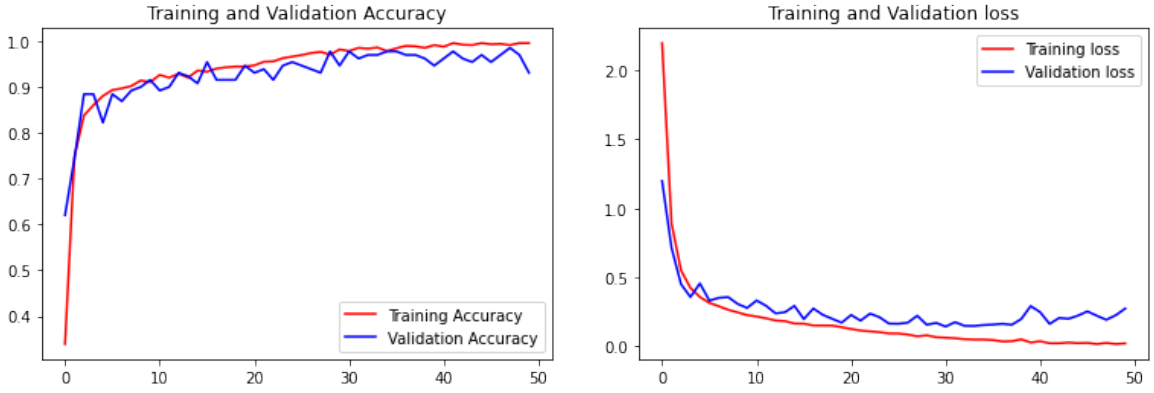


Figura 17 – Gráfico do número de imagens com os sinais detectados no *dataset* da ASL.

4.1.2 Treinamento e Avaliação da Rede MPL

Para o treinamento e avaliação da rede MLP utilizou-se o método de validação *Hold-Out*, que segundo Yadav e Shukla (2016), consiste em dividir os dados em duas partes que são usadas para treino e teste do modelo, respectivamente. Dividiu-se então o *dataset* de coordenadas em: base de treino/validação e base de teste. 80% do total de dados foram separados para a base de treino/validação (sendo, aproximadamente, 70% dos dados para treino e 10% para validação durante o processo de treinamento) e 20% do total de dados para a base de teste. Este método de divisão dos dados busca evitar o problema de *over-fitting*, que ocorre quando o modelo se adapta demais aos dados de treinamento e perde sua capacidade de generalização, tendo performance ruim quando exposto a um novo conjunto de dados.

Após a divisão dos dados, executou-se o treinamento da rede por 50 épocas. Ao fim do processo de treinamento (que durou cerca de três minutos) analisou-se os dados de perda e acurácia nas bases de treino e validação para determinar um critério de parada que, para este trabalho, foi definido como a época em que o modelo obteve o menor valor de perda (*loss*) na base de validação. Analisando-se os gráficos da Fig. (18), nota-se que no gráfico (b), por volta da época 30, a perda na base de dados de treino (*Training loss*) continua diminuindo, aproximando-se do zero, enquanto a perda na base de validação (*Validation loss*) começou a aumentar, o que indica que, a partir dessa época, o modelo começou um ajuste excessivo aos dados de treino (*over-fitting*). Além disso, também por volta da época 30, no gráfico (a), observa-se que o modelo alcançou um dos maiores níveis de acurácia na base de validação (*Validation Accuracy*).



(a) Gráfico Acurácia x Época

(b) Gráfico Perda x Época

Figura 18 – Gráficos do processo de treinamento da rede MLP.

Portanto, carregou-se os pesos correspondentes à época 31 (na qual o modelo apresentou o menor valor de perda na base de validação) no modelo. Os valores das métricas de acurácia (Acc.) e perda (Loss) das bases de treino e validação para a época

31 podem ser vistos na Tab. (3).

Tabela 3: Valores para as métricas de acurácia e perda na época 31.

Época	**Acc. Treino**	**Loss Treino**	**Acc. Validação**	**Loss Validação**
31	0.9775	0.0631	**0.9767**	**0.1438**

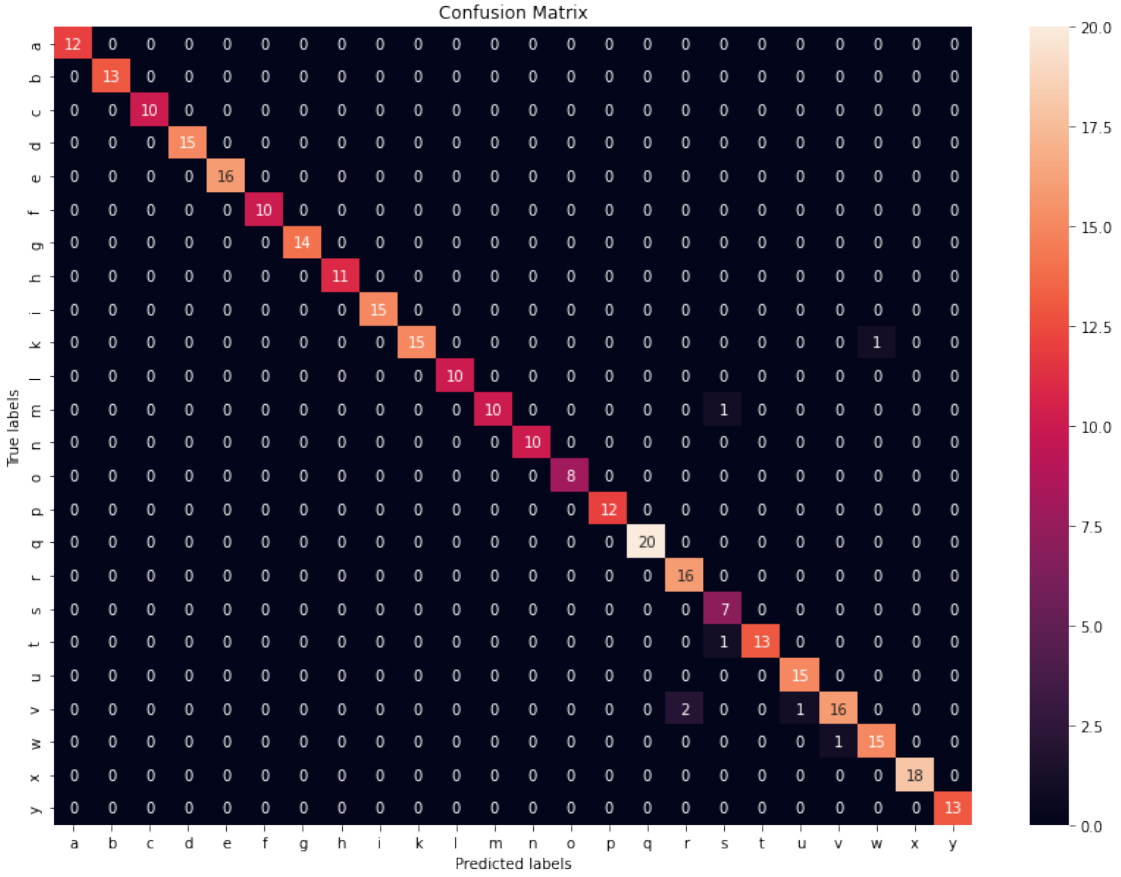
Com os pesos carregados, a base de teste foi submetida ao modelo para a avaliação dos resultados. O resultado das métricas de acurácia e perda na base de teste foram **0.9720** (97.2%) e **0.1378**, respectivamente, alcançando um resultado semelhante comparado à base de validação (ver Tab. 3), o que indica uma boa capacidade de generalização do modelo.

Analisando-se a matriz de confusão e o relatório de classificação (Fig. 19), percebe-se que as classes (sinais) com as quais o modelo mais se confundiu foram: m, r, s, t, u, v e w. Dentre estas, as classes com o pior desempenho, com base na métrica *f1-score*, foram s e v.

Embora o sinal 's' tenha recebido o menor valor de *f1-score* (0.88), é possível dizer que o modelo tem uma boa performance em sua classificação, uma vez que a métrica *recall* para este sinal possui valor máximo (1.0), o que significa que todas as ocorrências positivas foram classificadas corretamente. Porém, a precisão (*precision*) do classificador para este sinal foi baixa, sendo que os sinais 't' e 'm' foram classificados como 's' devido a grande similaridade em relação à posição do polegar nestes sinais, como visto na Fig. (20 (a)). Entretanto, é preciso levar em consideração a métrica *support* (número de ocorrências da classe na base de teste) que para o sinal 's' é a menor de todas, apenas 7, o que influenciou no baixo valor de precisão, apesar de ter tido o mesmo número de erros que a classe 'r', por exemplo.

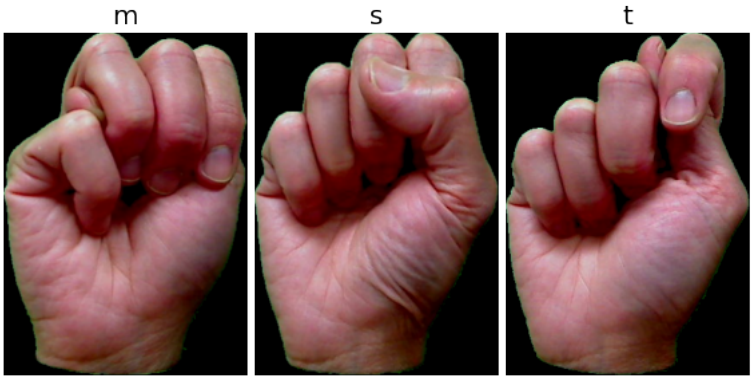
Por outro lado, o sinal 'v', mesmo que tenha um valor de *f1-score* (0.89) superior ao da classe 's', foi a classe com o pior resultado (levando em conta todas as métricas), um vez que três ocorrências positivas do sinal foram classificadas como 'u' e 'r', impactando no valor do *recall*, além de ter havido uma ocorrência de 'w' sendo classificada como 'v' o que afetou no valor de precisão (*precision*). Isso pode ser explicado pelo fato de que a diferença nestes sinais seja apenas uma pequena variação na posição e espaçamento entre os dedos indicador e médio (Fig. 20 (b)).

Outro dado interessante a se considerar é o fato de que o modelo teve um bom desempenho, levando em consideração a métrica *recall*, na classificação de sinais como c, o, p, e u mesmo estes possuindo um número de ocorrências menor que 60 no *dataset* de coordenadas (ver Fig. 17), o que sugere que isso não afetou de forma direta o aprendizado do modelo, exceto para o sinal 'w' (que teve uma de suas ocorrências positivas classificada como 'v') que pode ter sido influenciado por essa questão.

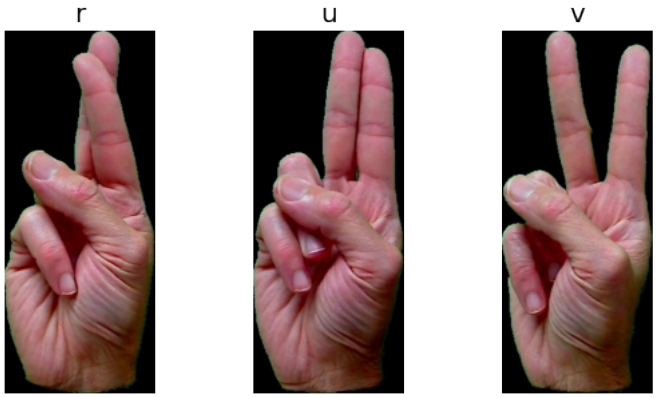


	precision	recall	f1-score	support
Class a	1.00	1.00	1.00	12
Class b	1.00	1.00	1.00	13
Class c	1.00	1.00	1.00	10
Class d	1.00	1.00	1.00	15
Class e	1.00	1.00	1.00	16
Class f	1.00	1.00	1.00	10
Class g	1.00	1.00	1.00	14
Class h	1.00	1.00	1.00	11
Class i	1.00	1.00	1.00	15
Class k	1.00	0.94	0.97	16
Class l	1.00	1.00	1.00	10
Class m	1.00	0.91	0.95	11
Class n	1.00	1.00	1.00	10
Class o	1.00	1.00	1.00	8
Class p	1.00	1.00	1.00	12
Class q	1.00	1.00	1.00	20
Class r	0.89	1.00	0.94	16
Class s	0.78	1.00	0.88	7
Class t	1.00	0.93	0.96	14
Class u	0.94	1.00	0.97	15
Class v	0.94	0.84	0.89	19
Class w	0.94	0.94	0.94	16
Class x	1.00	1.00	1.00	18
Class y	1.00	1.00	1.00	13
accuracy			0.98	321
macro avg	0.98	0.98	0.98	321
weighted avg	0.98	0.98	0.98	321

Figura 19 – Matriz de confusão e relatório de classificação do modelo na base de teste.



(a) Similaridade entre os sinais m, s e t.



(b) Similaridade entre os sinais r, u e v

Figura 20 – Similaridade entre sinais da ASL.

Contudo, de forma geral, o modelo apresentou um bom desempenho na base de teste, tendo em conta todos os resultados e métricas, alcançando uma acurácia de, aproximadamente, 98% na classificação de 321 ocorrências dos 24 sinais treinados da ASL.

4.2 Fase de Teste

Nesta fase executou-se um experimento de reconhecimento de gestos, onde o modelo da rede MLP treinado foi submetido a um banco de imagens produzidas pelo autor.

4.2.1 Reconhecimento de Gestos

Para o processo de reconhecimento de gestos, um *dataset* com os 24 sinais estáticos do alfabeto da ASL feitos com a mão do autor foram classificadas pelo modelo com o objetivo de testá-lo com um novo conjunto de dados, diferentes daqueles presentes no *dataset* da ASL utilizado para o treinamento.

As 24 imagens (uma para cada sinal) foram carregadas e processadas, uma a uma, para o reconhecimento do gesto (sinal) em questão. A Figura 21 mostra os resultados do

Figura 21 – Resultado do processo de reconhecimento de gestos.

reconhecimento de gestos. O texto acima das imagens (em preto) caracterizam a letra do alfabeto correspondente à imagem e o texto em amarelo (impresso na imagem) corresponde à previsão da rede para aquele sinal. Como se pode perceber, o modelo previu corretamente todos os 24 sinais, obtendo um total de 100% de acurácia na classificação, o que valida a capacidade de generalização e a performance do mesmo na tarefa de reconhecimento de gestos estáticos.

5 Conclusão e Trabalhos Futuros

A partir dos resultados experimentais obtidos no desenvolvimento deste trabalho, conclui-se que a abordagem da utilização dos dados de coordenadas 3D dos pontos do esqueleto da mão como entrada (vetor de características) para uma rede neural artificial MLP se mostra bastante pertinente na tarefa de reconhecimento de gestos estáticos da mão, proposta como objetivo geral do presente trabalho, visto que foi possível alcançar altos níveis de acurácia e capacidade de generalização com o modelo obtido como resultado da aplicação dessa abordagem.

Os dados do esqueleto provaram-se ser uma informação altamente semântica, uma vez que é possível obter característica complexas do formato e posicionamento das mãos e dedos em um conjunto relativamente pequeno de dados, o que permitiu o treinamento do modelo em um período curto de tempo, obtendo-se resultados satisfatórios.

Com o desenvolvimento de novos modelos de detecção/rastreamento de mãos ou aperfeiçoamento dos modelos existentes, no que tange a precisão das estimativas do esqueleto, será possível atingir resultados cada vez melhores no problema de reconhecimento de gestos, dado que esses modelos fornecerão dados mais acurados, facilitando o aprendizado dos algoritmos de *machine learning*, garantindo uma melhor classificação de gestos semelhantes ou gestos onde há oclusões parciais dos dedos.

Além disso, a qualidade das imagens e variações em parâmetros como iluminação e ângulos de rotação presentes no *dataset* da ASL utilizado neste trabalho, proporcionou uma maior eficácia do modelo de rede neural, no que diz respeito à sua capacidade de generalização.

Por fim, como sugestões para trabalhos futuros têm-se:

- A aplicação dessa abordagem em sistemas de reconhecimento de gestos em tempo real com o propósito de avaliar sua performance neste cenário;
- A utilização dos dados de coordenadas 3D dos pontos do esqueleto como entrada para modelos temporais com intuito de reconhecer gestos dinâmicos, avaliando sua relevância para esse tipo de tarefa;
- A aplicação de uma abordagem híbrida, onde os dados das coordenadas 3D são concatenados aos dados de intensidade das imagens, como por exemplo aqueles obtidos através de redes neurais convolucionais, para avaliar se há alguma melhora em relação ao uso de apenas um ou outro dado.

Referências

AL-SAEDI, A. K. H.; AL-ASADI, A. H. H. Survey of hand gesture recognition systems. *Journal of Physics: Conference Series*, IOP Publishing, v. 1294, n. 4, p. 042003, sep 2019. Disponível em: <https://doi.org/10.1088/1742-6596/1294/4/042003>. Citado 2 vezes nas páginas 25 e 28.

BARCZAK, A. L. C. et al. A new 2d static hand gesture colour image dataset for asl gestures. *Research Letters in the Information and Mathematical Sciences*, v. 15, p. 12–20, 2011. Disponível em: <http://www.massey.ac.nz/massey/fms/Colleges/College%20of%20Sciences/IIMS/RLIMS/Volume%2015/GestureDatasetRLIMS2011.pdf>. Citado na página 33.

BOURKE, A.; O'BRIEN, J.; ÓLAIGHIN, G. Evaluation of threshold-based tri-axial accelerometer fall detection algorithm. *Gait & posture*, v. 26, p. 194–9, 08 2007. Disponível em: <https://doi.org/10.1016/j.gaitpost.2006.09.012>. Citado 2 vezes nas páginas 15 e 27.

BRAGA, A. d. P.; LUDERMIR, T. B.; CARVALHO, A. C. P. d. L. F. *Redes neurais artificiais*: teoria e aplicações. 1ª. ed. Rio de Janeiro: LTC, 2000. Citado 2 vezes nas páginas 30 e 31.

CHANG, C.-C. et al. New approach for static gesture recognition. *J. Inf. Sci. Eng.*, v. 22, p. 1047–1057, 09 2006. Disponível em: <https://www.researchgate.net/publication/220587991_New_Approach_for_Static_Gesture_Recognition>. Citado na página 25.

CHOUDHURY, A.; TALUKDAR, A.; SARMA, K. A review on vision-based hand gesture recognition and applications. In: _____. 1. ed. PA, USA: IGI Global, 2015. cap. 11, p. 261–286. ISBN: 13: 9781466684935. Citado 4 vezes nas páginas 15, 29, 30 e 31.

CYBENKO, G. Continuous valued neural networks with two hidden layers are sufficient. Medford, MA, 1988. Citado na página 38.

DAVIES, E. R. *Computer and Machine Vision*: Theory, algorithms, practicalities. Fourth edition. University of London, UK: Elsevier, 2012. ISBN: 978-0-12-386908-1. Citado na página 23.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification (2nd Edition)*. 2. ed. [S.l.]: Wiley-Interscience, 2000. Hardcover. ISBN 0471056693. Citado 2 vezes nas páginas 31 e 32.

GARG, P.; AGGARWAL, N.; SOFAT, S. Vision based hand gesture recognition. *International Journal of Computer and Information Engineering*, v. 3, n. 1, p. 186–191, 2009. Disponível em: <https://publications.waset.org/vol/25>. Citado 2 vezes nas páginas 26 e 28.

HAYKIN, S. *Redes neurais*: princípios e práticas. 2ª. ed. Porto Alegre, RS: Bookman, 2007. Citado 2 vezes nas páginas 31 e 32.

HEATON, J. *Introduction to Neural Networks for Java*. 2nd. ed. [S.l.]: Heaton Research, Inc., 2008. ISBN 1604390085. Citado na página 38.

KAUR, H.; RANI, J. A review: Study of various techniques of hand gesture recognition. In: *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*. [s.n.], 2016. p. 1–5. Disponível em: <https://doi.org/10.1109/ICPEICES.2016.7853514>. Citado 3 vezes nas páginas 15, 26 e 28.

LI, Y. et al. Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition. *EURASIP Journal on Image and Video Processing*, v. 2019, n. 78, p. 1–2, 2019. Disponível em: <https://doi.org/10.1186/s13640-019-0476-x>. Citado na página 23.

MITRA, S.; ACHARYA, T. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, v. 37, n. 3, p. 311–324, 2007. Disponível em: <https://ieeexplore.ieee.org/document/4154947>. Citado 2 vezes nas páginas 25 e 26.

National Research Council. *How People Learn*: Brain, mind, experience, and school. Expanded edition. Washington, DC, USA: The National Academies Press, 2000. ISBN 0-309-07036-8. Citado na página 23.

OUDAH, M.; AL-NAJI, A.; CHAHL, J. Hand gesture recognition based on computer vision: A review of techniques. *Journal of Imaging*, v. 6, n. 8, p. 3–21, 2020. Disponível em: <https://www.mdpi.com/2313-433X/6/8/73>. Citado 5 vezes nas páginas 15, 23, 26, 27 e 28.

PAVLOVIC, V.; SHARMA, R.; HUANG, T. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 19, n. 7, p. 677–695, 1997. Disponível em: <https://doi.org/10.1109/34.598226>. Citado 2 vezes nas páginas 27 e 28.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition, Fourth Edition*. [S.l.]: Academic Press, 2009. Hardcover. ISBN 9781597492720. Citado na página 30.

YADAV, S.; SHUKLA, S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. [s.n.], 2016. p. 78–83. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7544814>. Citado na página 44.

ZHANG, F. et al. MediaPipe Hands: On-device Real-time Hand Tracking. *arXiv e-prints*, p. arXiv:2006.10214, 2020. Disponível em: <https://ui.adsabs.harvard.edu/abs/2020arXiv200610214Z>. Citado na página 28.

Apêndices

APÊNDICE A – Código extrator de coordenadas do *dataset*

O código abaixo foi escrito e executado no ambiente de desenvolvimento do *Google Colab*. Link para o arquivo original: <https://colab.research.google.com/drive/1XRMqtVSDUtmKCHRVMbqRJfi7ovuNQLGb>

```
pip install mediapipe # instalando biblioteca de soluções MediaPipe do Google

import cv2
import os
import mediapipe as mp
import zipfile

from google.colab import drive
drive.mount('/content/drive')

path = '' # caminho para o arquivo de imagens zipado
zip_object = zipfile.ZipFile(file = path, mode = 'r')
zip_object.extractall('./')
zip_object.close()

# configuração de parâmetros da solução de rastreamento de mãos
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=True,
    model_complexity=1,
    max_num_hands=1,
    min_detection_confidence=0.3)

directory = '' #caminho para o diretório das imagens
files = [os.path.join(directory, f) for f in sorted(os.listdir(directory))]

export = 'WRISTx,WRISTy,WRISTz,'
export+= 'THUMB_CMCx,THUMB_CMCy,THUMB_CMCz,'
export+= 'THUMB_MCPx,THUMB_MCPy,THUMB_MCPz,'
```

```
export+= 'THUMB_IPx,THUMB_IPy,THUMB_IPz,'
export+= 'THUMB_TIPx,THUMB_TIPy,THUMB_TIPz,'
export+= 'INDEX_FINGER_MCPx,INDEX_FINGER_MCPy,INDEX_FINGER_MCPz,'
export+= 'INDEX_FINGER_PIPx,INDEX_FINGER_PIPy,INDEX_FINGER_PIPz,'
export+= 'INDEX_FINGER_DIPx,INDEX_FINGER_DIPy,INDEX_FINGER_DIPz,'
export+= 'INDEX_FINGER_TIPx,INDEX_FINGER_TIPy,INDEX_FINGER_TIPz,'
export+= 'MIDDLE_FINGER_MCPx,MIDDLE_FINGER_MCPy,MIDDLE_FINGER_MCPz,'
export+= 'MIDDLE_FINGER_PIPx,MIDDLE_FINGER_PIPy,MIDDLE_FINGER_PIPz,'
export+= 'MIDDLE_FINGER_DIPx,MIDDLE_FINGER_DIPy,MIDDLE_FINGER_DIPz,'
export+= 'MIDDLE_FINGER_TIPx,MIDDLE_FINGER_TIPy,MIDDLE_FINGER_TIPz,'
export+= 'RING_FINGER_MCPx,RING_FINGER_MCPy,RING_FINGER_MCPz,'
export+= 'RING_FINGER_PIPx,RING_FINGER_PIPy,RING_FINGER_PIPz,'
export+= 'RING_FINGER_DIPx,RING_FINGER_DIPy,RING_FINGER_DIPz,'
export+= 'RING_FINGER_TIPx,RING_FINGER_TIPy,RING_FINGER_TIPz,'
export+= 'PINKY_MCPx,PINKY_MCPy,PINKY_MCPz,'
export+= 'PINKY_PIPx,PINKY_PIPy,PINKY_PIPz,'
export+= 'PINKY_DIPx,PINKY_DIPy,PINKY_DIPz,'
export+= 'PINKY_TIPx,PINKY_TIPy,PINKY_TIPz,CLASS\n'

alphabet = ['a','b','c','d','e','f','g','h','i','k','l','m','n','o','p','q','r'
,'s','t','u','v','w','x','y']

for image_path in files:

  image = cv2.imread(image_path)
  image_name = os.path.basename(os.path.normpath(image_path))
  image_coordinates = []

  for idx, letter in enumerate(alphabet):
    if image_name.startswith(letter):
      letter_class = idx

  results = hands.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

  if not results.multi_hand_world_landmarks:
    continue

  for hand_world_landmarks in results.multi_hand_world_landmarks:
    for i in range(21):
```

```
      image_coordinates.append(hand_world_landmarks.landmark[i].x)
      image_coordinates.append(hand_world_landmarks.landmark[i].y)
      image_coordinates.append(hand_world_landmarks.landmark[i].z)

  image_coordinates.append(letter_class)
  f = (",".join([str(item) for item in image_coordinates]))
  export += f + '\n'

output_file = '' # nome do arquivo de saída (.csv)
with open(output_file, 'w') as file:
  for line in export:
    file.write(line)
file.closed
```

APÊNDICE B – Código de treinamento da rede MLP

O código abaixo foi escrito e executado no ambiente de desenvolvimento do *Google Colab*. Link para o arquivo original: <https://colab.research.google.com/drive/152mtqgky1hHeROpj3vtfXqEIqZUjgojo>

```
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import sklearn

from google.colab import drive
drive.mount('/content/drive')

dataset_csv_file_path = '' # caminho para o arquivo de dados csv
dataset = pd.read_csv(dataset_csv_file_path)

# X - vetores de características
X = dataset.iloc[:, 0:63].values

# Y - classes
y = dataset.iloc[:, 63].values

# divisão dos dados em bases de treinamento/validação e teste
from sklearn.model_selection import train_test_split
X_train_val, X_test, y_train_val, y_test = train_test_split(
  X,
  y,
  test_size = 0.2,
  random_state = 1
)
```

```
X_train, X_val, y_train, y_val = train_test_split(
  X_train_val,
  y_train_val,
  test_size = 0.1,
  random_state = 1
)

# construção da arquitetura da rede MLP
mlp = tf.keras.Sequential()
mlp.add(tf.keras.layers.Dense(input_shape = (63,), units = 66, activation='relu'))
mlp.add(tf.keras.layers.Dense(units = 66, activation='relu'))
mlp.add(tf.keras.layers.Dense(units = 24, activation='softmax'))

# compilando rede
mlp.compile(loss = 'sparse_categorical_crossentropy', optimizer='Adam',
            metrics = ['accuracy'])

from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint(
  "mlp_weights.hdf5",
  save_weights_only=True,
  monitor='val_loss',
  save_best_only=True,
  mode='auto',
  verbose = 1
)

# treinamento da rede
epochs = 50
history = mlp.fit(X_train, y_train, batch_size = 1, epochs = epochs,
                  validation_data = (X_val, y_val),
                  callbacks=[checkpoint])


# avaliação da rede
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
# plotando gráfico de accuracy
gfx_epochs = range(len(accuracy))
plt.plot(gfx_epochs, accuracy, 'b', color = 'r', label='Training Accuracy')
plt.plot(gfx_epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend();

# plotando gráfico de loss
plt.plot(gfx_epochs, loss, 'b', color = 'r', label='Training loss')
plt.plot(gfx_epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend();

# carregando pesos
weights_file_path = '' # caminho para o arquivos de pesos
mlp.load_weights(weights_file_path)

# avaliando a base de teste
evaluation = mlp.evaluate(X_test, y_test)

predictions = mlp.predict(X_test)
predicted_classes = np.argmax(predictions,axis=1)

classes = ['a','b','c','d','e','f','g','h','i','k','l','m','n','o','p','q','r'
,'s','t','u','v','w','x','y']

# plotando matriz de confusão
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predicted_classes)
plt.figure(figsize = (14,10))
ax = plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(classes); ax.yaxis.set_ticklabels(classes);
```

```
# plotando relatório de classificação
from sklearn.metrics import classification_report
target_names = ["Class {}".format(letter) for letter in classes]

print(classification_report(y_test, predicted_classes, target_names=target_names))

# salvando arquivos da rede
model_json = mlp.to_json()
with open('mlp.json','w') as json_file:
  json_file.write(model_json)

from keras.models import save_model
output_file_path = '' # caminho para o arquivo de saída ([path]/nome_do_arquivo.hdf5)
save_model(mlp, output_file_path)
```

APÊNDICE C – Código de classificação de gesto

O código abaixo foi escrito e executado no ambiente de desenvolvimento do *Google Colab*. Link para o arquivo original: <https://colab.research.google.com/drive/1vYmTamlBxnwncegPg1j8tvgjp8MsV7J1>

```
pip install mediapipe # instalando biblioteca de soluções MediaPipe do Google

import cv2
import mediapipe as mp
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

# configuração de parâmetros da solução de rastreamento de mãos
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=True,
    model_complexity=1,
    max_num_hands=1,
    min_detection_confidence=0.3)

# leitura da imagem
image_path = '' # caminho para a imagem
image = cv2.imread(image_path)
# comentar esse trecho se não for presciso espelhar a imagem
image = cv2.flip(image, 1)

# processamento da imagem
```

```
results = hands.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

# imprime pontos do esqueleto na imagem
for hand_landmarks in results.multi_hand_landmarks:
      mp_drawing.draw_landmarks(
          image,
          hand_landmarks,
          mp_hands.HAND_CONNECTIONS,
          mp_drawing_styles.get_default_hand_landmarks_style(),
          mp_drawing_styles.get_default_hand_connections_style()
      )

# extração de coordenadas 3D do esqueleto
image_coordinates = []
for hand_world_landmarks in results.multi_hand_world_landmarks:
    for i in range(21):
      image_coordinates.append(hand_world_landmarks.landmark[i].x)
      image_coordinates.append(hand_world_landmarks.landmark[i].y)
      image_coordinates.append(hand_world_landmarks.landmark[i].z)

# criação do vertor de características
df = pd.DataFrame(image_coordinates).iloc[:].values
image_test = df.reshape(1,-1)

mlp_file_path = '' # caminho para arquivo de estrutura da rede
with open(mlp_file_path, 'r') as json_file:
  json_saved_model = json_file.read()

# carregamento do modelo a partir dos arquivos da rede treinada
mlp_loaded = tf.keras.models.model_from_json(json_saved_model)
weights_file_path = '' # caminho para arquivo de pesos da rede
mlp_loaded.load_weights(weights_file_path)
mlp_loaded.compile(loss = 'sparse_categorical_crossentropy', optimizer='Adam',
            metrics = ['accuracy'])

# classificação do gesto
prediction = mlp_loaded.predict(image_test)
predicted_class = np.argmax(prediction, axis=1)
```

```
alphabet = ['A','B','C','D','E','F','G','H','I','K','L','M','N','O','P','Q','R'
,'S','T','U','V','W','X','Y']

# adiciona a classe prevista na imagem
for idx, letter in enumerate(alphabet):
    if predicted_class == idx:
      cv2.putText(image, ' ' + letter, (50,200), cv2.FONT_HERSHEY_SIMPLEX, 5, (10,
                  8, lineType=cv2.LINE_AA)

# printa a imagem com a previsão da rede
plt.figure(figsize= [14,10])
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.axis('off')
```