

UNIVERSIDADE FEDERAL DE RONDÔNIA
DEPARTAMENTO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉIA DE OLIVEIRA ARAÚJO

**2DCNN: RECONHECIMENTO DE SINAIS DE LIBRAS
UTILIZANDO UMA REDE NEURAL CONVOLUCIONAL 2D**

PORTO VELHO
2023

ANDRÉIA DE OLIVEIRA ARAÚJO

**2DCNN: RECONHECIMENTO DE SINAIS DE LIBRAS
UTILIZANDO UMA REDE NEURAL CONVOLUCIONAL 2D**

Monografia submetida à Fundação Universidade Federal de Rondônia - UNIR, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Lucas Marques da Cunha
Universidade Federal de Rondônia

PORTE VELHO
2023

Dedico este trabalho ao meu amado filho, Caio Gabriel. Sua presença em minha vida trouxe luz, amor e motivação para alcançar novos desafios. Suas risadas e abraços são o combustível que impulsiona meu esforço diário. Te amo incondicionalmente.

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todos aqueles que estiveram presentes em minha jornada e contribuíram para o sucesso deste trabalho.

Agradeço a Deus por sua presença constante em minha vida, me guiando e abençoando em cada passo dessa jornada.

Ao meu filho Caio Gabriel, minha maior fonte de inspiração, agradeço por preencher meus dias com alegria e me motivar a buscar sempre o melhor.

À minha amada família, meu alicerce e fonte de amor incondicional, agradeço por estarem sempre ao meu lado, apoando e encorajando meus sonhos.

Ao meu companheiro Bruno Bordin, agradeço por ser meu apoio incondicional, por me motivar e me incentivar a ir além, mesmo nos momentos mais desafiadores. Sua presença ao meu lado tornou essa jornada mais significativa e especial.

Ao meu orientador Lucas Marques, minha gratidão por sua orientação precisa, paciência e dedicação ao longo dessa jornada acadêmica. Sua sabedoria e apoio foram fundamentais para o meu crescimento.

Ao meu chefe Anderson Kruse, que além de ser um amigo, foi um dos maiores incentivadores deste trabalho, agradeço por seu apoio, confiança e por acreditar em meu potencial.

Aos meus amigos, verdadeiros tesouros em minha vida, agradeço por estarem ao meu lado, compartilhando risadas, momentos inesquecíveis e por serem uma fonte constante de apoio e inspiração.

A todas as pessoas que, de alguma forma, cruzaram meu caminho e contribuíram para meu crescimento e desenvolvimento, meu sincero agradecimento. Vocês tornaram minha jornada mais rica e significativa.

Que todas essas pessoas saibam que sua presença e influência positiva em minha vida são eternamente valorizadas e lembradas. Meu coração transborda de gratidão por cada um de vocês.

"A educação é a arma mais poderosa que você pode usar para mudar o mundo." (Nelson Mandela).

RESUMO

ARAÚJO, Andréia de Oliveira. 2DCNN: RECONHECIMENTO DE SINAIS DE LIBRAS UTILIZANDO UMA REDE NEURAL CONVOLUCIONAL 2D. 2023. 62 f. – Curso de Bacharelado em Ciência da Computação, Universidade Federal de Rondônia. Porto Velho, 2023.

No Brasil, através do censo demográfico, foi possível avaliar que cerca 2,1 milhões de pessoas são consideradas surdas. A forma de comunicação e expressão utilizada pelas pessoas surdas é a Língua Brasileira de Sinais - LIBRAS, que é reconhecida como uma língua oficial do Brasil desde 2002, com a lei nº. 10.436. Embora existam meios para a inclusão de pessoas surdas, ainda não estão plenamente desenvolvidos para garantir a inclusão efetiva na sociedade, tendo assim, a pessoa surda, dificuldades em estabelecer uma comunicação plena com a sociedade. Um meio para solucionar esse problema pode ocorrer através da utilização da visão computacional, que permite que uma máquina "enxergue" e extraia informações significativas de imagens. Nesse contexto, o presente trabalho propõe o uso de uma rede neural convolucional 2D (CNN2D) para o reconhecimento de gestos estáticos em LIBRAS, por meio da detecção de imagens em tempo real. O objetivo é realizar três experimentos: o primeiro para a detecção das letras estáticas do alfabeto em LIBRAS, totalizando 21 letras; o segundo para a detecção dos números em sinais estáticos de LIBRAS, abrangendo os números de 0 a 9; e o terceiro para a detecção de 10 palavras em sinais estáticos de LIBRAS, sendo essas: 'Adulto', 'America', 'Casa', 'Gasolina', 'Juntos', 'Lei', 'Palavra', 'Pedra', 'Pequeno' e 'Verbo'. Os resultados dos treinamentos realizados com a CNN2D demonstram uma acurácia de 97.93% para o experimento I (detecção das letras do alfabeto), 91.39% para o experimento II (detecção dos números) e 91.19% para o experimento III (detecção das palavras), mostrando um bom desempenho na detecção em tempo real e equiparando-se aos trabalhos descritos na literatura.

Palavras-chave: Rede Neural Convolutinal 2D. CNN2D. Reconhecimento de Sinais. LIBRAS.

ABSTRACT

ARAÚJO, Andréia de Oliveira. 2DCNN: LIBRAS sign recognition using a 2D convolutional neural network. 2023. 62 f. – Curso de Bacharelado em Ciência da Computação, Universidade Federal de Rondônia. Porto Velho, 2023.

In Brazil, through the demographic census, it was possible to assess that about 2.1 million people are considered deaf. The form of communication and expression used by deaf people is the Brazilian Sign Language - LIBRAS, which has been recognized as an official language in Brazil since 2002, with law n°. 10,436. Although there are means for the inclusion of deaf people, they are not yet fully developed to guarantee effective inclusion in society, thus having the deaf person have difficulties in establishing full communication with society. A way to solve this problem can be through the use of computer vision, which allows a machine to "see" and extract meaningful information from images. In this context, the present work proposes the use of a 2D convolutional neural network (CNN2D) for the recognition of static gestures in LIBRAS, through the detection of images in real time. The objective is to carry out three experiments: the first for the detection of static letters of the alphabet in LIBRAS, totaling 21 letters; the second for the detection of numbers in static LIBRAS signs, covering numbers from 0 to 9; and the third for the detection of 10 words in static LIBRAS signs, namely: 'Adult', 'America', 'Home', 'Gasoline', 'Together', 'Law', 'Word', 'Stone', 'Small' and 'Verb'. The results of training performed with CNN2D demonstrate an accuracy of 97.93% for experiment I (detection of letters of the alphabet), 91.39% for the experiment II (detection of numbers) and 91.19% for experiment III (detection of words), showing a good performance in real-time detection and matching the works described in the literature.

Keywords: 2D Convolutional Neural Network. CNN2D. Signal Recognition. LIBRAS.

LISTA DE FIGURAS

Figura 1 – Alfabeto manual da LIBRAS.	6
Figura 2 – Matriz de confusão para um problema com duas classes.	10
Figura 3 – Rede Neural Simples e Rede Neural Profunda.	12
Figura 4 – Exemplos de (a) underfitting, (b) overfitting e (c) boa generalização.	13
Figura 5 – Função sigmoide.	14
Figura 6 – Função ReLU.	15
Figura 7 – Função Leaky ReLU.	15
Figura 8 – Exemplo de uma rede neural convolucional e suas diferentes camadas	17
Figura 9 – Exemplo de uma operação convolucional	17
Figura 10 – Exemplo de uma operação convolucional com imagem RGB	18
Figura 11 – Exemplo do funcionamento da camada de Pooling	19
Figura 12 – Conjunto de dados - sinal da letra A	24
Figura 13 – Conjunto de dados - variações sinal do número 0	26
Figura 14 – Conjunto de dados - variações sinal da palavra "pequeno"	26
Figura 15 – Conjunto de dados - Números	27
Figura 16 – Conjunto de dados - Palavras	27
Figura 17 – Arquitetura CNN2D - valores de entradas e saídas das camadas	29
Figura 18 – Arquitetura CNN2D - valores de entradas e saídas das camadas	30
Figura 19 – Reconhecimento em tempo real - Tela Principal	32
Figura 20 – Reconhecimento em tempo real - Tela da caixa delimitadora	32
Figura 21 – Reconhecimento em tempo real - Tela saída de texto	32
Figura 22 – Reconhecimento em tempo real - Telas Geradas	33
Figura 23 – Acurácia e Loss - Experimento I	35
Figura 24 – Acurácia e Loss - Experimento II	35
Figura 25 – Acurácia e Loss - Experimento III	36
Figura 26 – Matriz de confusão - Experimento I	37
Figura 27 – Matriz de confusão - Experimento II	38
Figura 28 – Matriz de confusão - Experimento III	39
Figura 29 – Captura em tempo real - Experimento I	40
Figura 30 – Variações na captura em tempo real - Experimento I, letra A	40
Figura 31 – Variações na captura em tempo real - Experimento I, letra N	41
Figura 32 – Variações na captura em tempo real - Experimento I, letra T	41
Figura 33 – Captura em tempo real - Experimento II	41
Figura 34 – Variações na captura em tempo real - Experimento II, Número 3	42
Figura 35 – Variações na captura em tempo real - Experimento II, Número 7	42
Figura 36 – Captura em tempo real - Experimento III	43

Figura 37 – Variações na captura em tempo real - Experimento III, Palavra 'juntos' . . .	43
Figura 38 – Variações na captura em tempo real - Experimento III, Palavra 'pedra' . . .	44
Figura 39 – Captura em tempo real - Identificação da mão direita e esquerda	44

LISTA DE QUADROS

Quadro 1 – Parâmetros de Libras e suas Manifestações.	5
---	---

LISTA DE TABELAS

Tabela 1 – Tabela comparativa dos trabalhos apresentados nesta seção	21
Tabela 2 – Tabela de versões	22
Tabela 3 – Tabela de informações do conjunto de dados do experimento I	25
Tabela 4 – Tabela de informações do conjunto de dados dos experimentos II e III	27
Tabela 5 – Acurácia e Loss dos conjuntos de dados	34

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
CNN	Convolutional Neural Network
CNN1D	Convolutional Neural Network 1D
CNN2D	Convolutional Neural Network 2D
CNN3D	Convolutional Neural Network 3D
FN	Falso Negativo
FP	Falso Positivo
IBGE	Instituto Brasileiro de Geografia e Estatística
LIBRAS	Língua Brasileira de Sinais
RNA	Rede Neural Artificial
RNAs	Redes Neurais Artificiais
TFN	Taxa de Falso Negativo
TFP	Taxa de Falso Positivo
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo
1D	Uma Dimensão
2D	Duas Dimensões
3D	Três Dimensões

SUMÁRIO

1 – INTRODUÇÃO	1
1.1 JUSTIFICATIVA	2
1.2 OBJETIVO GERAL	2
1.2.1 OBJETIVOS ESPECÍFICOS	2
1.3 ORGANIZAÇÃO DO TRABALHO	3
2 – REVISÃO DE LITERATURA	4
2.1 LÍNGUA BRASILEIRA DE SINAIS — LIBRAS	4
2.2 VISÃO COMPUTACIONAL	6
2.2.1 RECONHECIMENTO DE PADRÕES	6
2.3 APRENDIZAGEM DE MÁQUINA	7
2.3.1 TIPOS DE APRENDIZADO DE MÁQUINA E SUAS DEFINIÇÕES	8
2.3.2 CONJUNTO DE DADOS	8
2.3.3 MEDIDA DE DESEMPENHO	9
2.4 REDES NEURAIS ARTIFICIAIS	11
2.4.1 APRENDIZADO PROFUNDO	11
2.4.1.1 DATA AUGMENTATION	13
2.4.1.2 FUNÇÃO DE ATIVAÇÃO	13
2.5 REDE NEURAL CONVOLUCIONAL	16
2.5.1 CAMADA CONVOLUCIONAL	17
2.5.2 CAMADA DE POOLING	18
2.5.3 CAMADA TOTALMENTE CONECTADA	19
2.5.4 TÉCNOLOGIAS UTILIZADAS EM REDES NEURAIS CONVOLUCIONAIS	19
2.6 TRABALHOS RELACIONADOS	20
3 – METODOLOGIA	22
3.1 CONFIGURAÇÃO DO AMBIENTE	22
3.2 CONFIGURAÇÃO DAS MÁQUINAS UTILIZADAS	22
3.3 CONJUNTO DE DADOS	23
3.3.1 INFORMAÇÕES DO CONJUNTO DE DADOS	23
3.3.2 CRIAÇÃO DO CONJUNTO DE DADOS	25
3.4 PRÉ-PROCESSAMENTO DE DADOS	28
3.5 DESCRIÇÃO DA ARQUITETURA DA REDE NEURAL CONVOLUCIONAL	28
3.6 TREINAMENTO DA REDE NEURAL CONVOLUCIONAL	31
3.7 UTILIZAÇÃO DO MODELO EM TEMPO REAL	31

4 – ANÁLISE E DISCUSSÃO DOS RESULTADOS	34
4.1 RESULTADOS QUANTITATIVOS	34
4.2 RESULTADOS QUALITATIVOS	39
5 – CONCLUSÃO	45
5.1 TRABALHOS FUTUROS	45
Referências	47
Anexos	50
ANEXO A – Código em Python	51
ANEXO B – Código em Python	54
ANEXO C – Código em Python	55
ANEXO D – Código em Python	60

1 INTRODUÇÃO

Segundo o Instituto Brasileiro de Geografia e Estatística - [IBGE \(2010\)](#), de acordo com o censo demográfico realizado no Brasil em 2010, que é a pesquisa mais recente, há 9,7 milhões de brasileiros com algum grau de deficiência auditiva, dos quais 2,1 milhões possuem um grau mais avançado, considerando-os surdos. Para a maioria das pessoas com deficiência auditiva, a comunicação é realizada por meio da Língua Brasileira de Sinais (LIBRAS), considerada uma língua oficial do Brasil desde 24 de abril de 2002 através da Lei n° 10.436 [LEI... \(2002\)](#), onde a LIBRAS é definida como uma "[...] forma de comunicação e expressão, em que o sistema lingüístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema lingüístico de transmissão de idéias e fatos, oriundos de comunidades de pessoas surdas do Brasil".

Existe um estudo realizado sobre os aspectos da comunicação do sujeito surdo e a sua inclusão na sociedade de [Ribeiro e Festa \(2017\)](#), onde aponta que uma das grandes dificuldades das pessoas surdas é a comunicação plena com a sociedade. "Essa dificuldade decorre da falta de conhecimento dos ouvintes em relação aos surdos enquanto minoria linguística, visto que a língua majoritária do Brasil é a Língua Portuguesa" ([RIBEIRO; FESTA, 2017](#)). Conforme pesquisas realizadas para analisar as dificuldades que as pessoas surdas têm em se comunicar diante da sociedade atual, os estudos demonstraram que esse obstáculo vem desde as pessoas mais próximas, como a família, já que, sem esse conhecimento em LIBRAS por parte dos familiares, a pessoa surda acaba se limitando na sua comunicação, podendo não se expressar plenamente. Isso faz com que o surdo se afaste da sua família, tornando-se uma pessoa solitária. Outro meio muito importante na vida da comunidade surda é o ambiente de trabalho. Muitas vezes dentro das empresas, não há funcionários com conhecimento em LIBRAS, o que dificulta a interação com a equipe. Isso faz com que o surdo não se sinta acolhido, e mais uma vez cria uma barreira de comunicação que afasta as pessoas. Nesse caso, a falta de conhecimento dos ouvintes em LIBRAS, gera essa barreira ([RIBEIRO; FESTA, 2017](#)). Devido ao avanço nas tecnologias atuais, já existe algumas ferramentas que auxiliam as pessoas surdas na comunicação, como os aplicativos HandTalk¹ e o Vlibras² que traduz o português para LIBRAS. Por mais que esses aplicativos ajudem na inclusão dos surdos diante a sociedade, eles realizam apenas uma via de comunicação, sendo essa a tradução do português para LIBRAS ([NETO, 2019](#)), onde não se tem a outra via de comunicação que seria a tradução do sinal de LIBRAS para o português, o que poderia ajudar tanto na comunicação das pessoas surdas, quanto também na forma de aprendizagem da LIBRAS. Uma estratégia para tentar solucionar o problema por meios computacionais é a utilização da visão computacional, já que a visão computacional busca construir sistemas que executem algumas das tarefas que o sistema visual humano pode

¹Disponível em: <<https://www.handtalk.me/br/>>

²Disponível em: <<https://www.gov.br/governodigital/pt-br/transformacao-digital/ferramentas/vlibras>>

executar ([NETO, 2019](#)).

Visando a inclusão das pessoas surdas na sociedade, o presente trabalho irá demonstrar a viabilidade do reconhecimento de gestos estáticos de LIBRAS através de uma rede neural convolucional, por meio da captura de uma imagem através da câmera em tempo real, detectando o sinal que está sendo realizado, utilizando assim a visão computacional para auxiliar as pessoas na identificação dos sinais da LIBRAS.

1.1 JUSTIFICATIVA

Sabendo que a LIBRAS se difere das linguagens orais pelo fato dela utilizar a visão como um canal comunicativo ([FREIRE, 1999](#)), estudos utilizando a visão computacional podem ajudar a desenvolver meios tecnológicos que auxiliem a sociedade na aprendizagem de LIBRAS, já que a visão computacional é definida como "[...] a ciência responsável pela visão de uma máquina, pela forma como um computador enxerga o meio à sua volta, extraíndo informações significativas a partir de imagens capturadas por câmeras de vídeo, sensores, scanners, entre outros dispositivos" ([MILANO; HONORATO, 2010](#)). Podemos utilizar dessa ciência para obter uma ajuda na resolução do problema. Há diversos estudos realizados utilizando a visão computacional, voltados ao reconhecimento de gestos, como: "Redes Neurais Artificiais aplicadas no Reconhecimento de Gestos usando o Kinect" de [Alvarenga, Correa e Osório \(2012\)](#), "Um sistema de baixo custo para reconhecimento de gestos em LIBRAS utilizando visão computacional" de [Monteiro et al. \(2016\)](#), "Desenvolvimento de tecnologia baseada em redes neurais artificiais para reconhecimento de gestos da língua de sinais" de [Silva et al. \(2018\)](#), entre diversos outros. Nota-se cada vez mais relevância assuntos voltados à área de reconhecimento de gestos, o que pode contribuir para o desenvolvimento de meios que ajudem a sociedade a se desenvolver na LIBRAS, uma vez que um sistema de reconhecimento de sinais em LIBRAS baseado em visão computacional seria mais barato do que dispositivos mais específicos que utilizem luvas ou braceletes ([STEFANO et al., 2021](#)).

1.2 OBJETIVO GERAL

Utilizar a rede neural convolucional para o reconhecimento de gestos estáticos em LIBRAS por meio da captura de uma imagem através da câmera em tempo real, com intuito de aumentar o número de métodos computacionais que auxiliem a inclusão das pessoas surdas na sociedade.

1.2.1 OBJETIVOS ESPECÍFICOS

- Verificar como a visão computacional pode auxiliar pessoas surdas;
- Analisar aspectos relacionados a LIBRAS, tais como gestos estáticos e dinâmicos;
- Compreender os mecanismos e tecnologias relacionadas a Rede Neural Convolutional;
- Estudar a estrutura da Rede Neural Convolutional;

- Definir métricas e parâmetros para o treinamento de rede;
- Definir o banco de dados que irá ser utilizado para treinamento, validação e teste.

1.3 ORGANIZAÇÃO DO TRABALHO

Para organização do trabalho segue, no capítulo 2 são introduzidos alguns conceitos fundamentais para este trabalho, no capítulo 3 é desenvolvido a metodologia que foi utilizada no trabalho, no capítulo 4 encontra-se a análise e discussão dos resultados obtidos neste trabalho, por fim no capítulo 5 será apresentado as conclusões acerca do trabalho desenvolvido.

2 REVISÃO DE LITERATURA

Este capítulo tem como objetivo introduzir uma contextualização acerca de temas que são necessários para o desenvolvimento deste trabalho.

2.1 LÍNGUA BRASILEIRA DE SINAIS — LIBRAS

LIBRAS significa Língua Brasileira de Sinais. É uma língua de modalidade gestual-visual, pois é produzida pelas mãos e recebida pelos olhos, sendo utilizada pela comunidade surda como uma língua oficial no Brasil desde 24 de abril de 2002 através da Lei nº 10.436/2002, a qual diz:

"Art. 1º É reconhecida como meio legal de comunicação e expressão a Língua Brasileira de Sinais – Libras e outros recursos de expressão a ela associados. Parágrafo único. Entende-se como Língua Brasileira de Sinais - Libras a forma de comunicação e expressão, em que o sistema lingüístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema lingüístico de transmissão de idéias e fatos, oriundos de comunidades de pessoas surdas do Brasil" ([LEI..., 2002](#)).

A LIBRAS é muito importante para a comunicação e a inclusão das pessoas surdas no Brasil, pois permite que a comunidade surda expresse seus pensamentos e se comunique de forma efetiva, podemos verificar essa afirmação pela linguística Quadros (1997, apud [RIBEIRO; FESTA, 2011](#)) :

"Tais línguas são naturais internamente e externamente, pois refletem a capacidade psicobiológica humana para a linguagem e porque surgiram da mesma forma que as línguas orais - da necessidade específica e natural dos seres humanos de usarem um sistema linguístico para expressarem ideias, sentimentos e ações. As línguas de sinais são sistemas linguísticos que passaram de geração em geração de pessoas surdas. São línguas que não se derivaram das línguas orais, mas fluíram de uma necessidade natural de comunicação entre pessoas que não utilizam o canal auditivo-oral, mas o canal espaço-visual como modalidade linguística".

"Apesar da capacidade de expressão tão grande quanto a de línguas orais, as línguas de sinais se diferenciam na medida em que utilizam, para a transmissão de mensagens, o canal visual-espacial ao invés do canal oral-auditivo" ([BASTOS, 2016](#)). De acordo com [Caiafa et al. \(2020\)](#) a língua de sinais, é estabelecido por alguns parâmetros tais como a configuração da mão, locação, orientação da mão, movimento e expressões não manuais. Podemos ver no [Quadro 1](#), um resumo destes parâmetros e como são manifestados.

Quadro 1 – Parâmetros de Libras e suas Manifestações.

Parâmetros	Como são manifestados
Configuração da mão	Gesto feito com a mão
Movimento	Ação da mão
Locação	Local onde as mãos realizam ação
Orientação da mão	Direção da palma na mão
Expressões não manuais	Movimentos da face, cabeça ou tronco

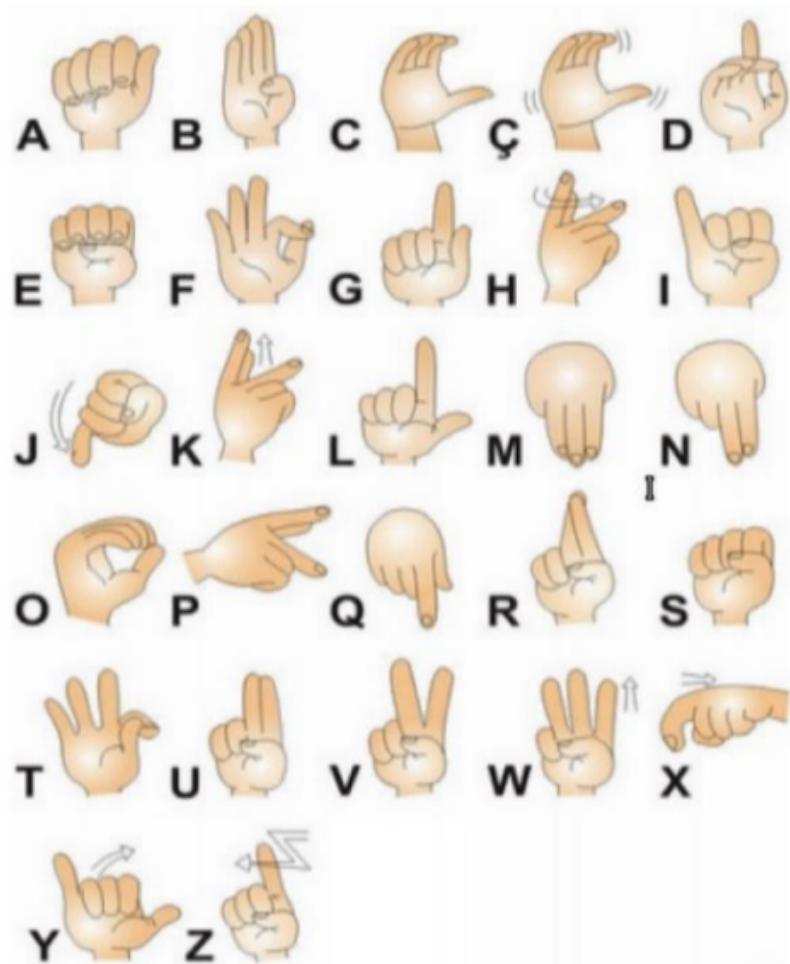
Fonte: CAIAFA et al. (2020).

De acordo com [Ramos \(2011\)](#), podemos definir esses parâmetros e manifestações da seguinte maneira:

- Configuração da Mão: É manifestado através de gesto feito com a mão, onde esse gesto pode ser da datilologia (alfabeto manual), como visto na Figura 1, podendo ser realizado pela mão predominante (mão direita para os destros) ou pelas duas mãos do emissor ou sinalizador;
- Movimento: É manifestado através da ação da mão, podendo ter movimentos ou não;
- Locação: É manifestado no local onde as mãos realizam a ação e onde incide a mão predominante configurada, podendo esta tocar alguma parte do corpo ou estar em um espaço neutro vertical (do meio do corpo até à cabeça) e horizontal (à frente do emissor);
- Orientação da mão: É manifestado através da direção da palma da mão fazendo com que os sinais tenham uma direção e a inversão desta pode significar idéia de oposição, contrariedade ou concordância número-pessoa;
- Expressões não manuais: É manifestado através de movimentos da face, cabeça ou tronco, são sinais que além dos quatro parâmetros mencionados acima, em sua configuração tem como traço diferenciador também a expressão facial e/ou corporal.

Segundo Strobel e Fernandes (1998, apud [PASSOS, 2019](#)), "a LIBRAS, assim como as demais línguas, possui variações linguísticas e uma estrutura gramatical própria. A Língua de Sinais Americana (ASL) difere da Língua de Sinais Britânica (BSL), que difere, por sua vez, da Língua de Sinais Brasileira". Além de também possuir variações regionais. No Brasil o alfabeto manual é composto por 27 sinais (considerando o ç), onde cada letra do alfabeto corresponde a um sinal de LIBRAS, conforme a Figura 1

Figura 1 – Alfabeto manual da LIBRAS.



Fonte: Pinheiro (2010, apud [PASSOS, 2019](#)).

2.2 VISÃO COMPUTACIONAL

Segundo [Passos \(2019\)](#) "A Visão Computacional é o processo de modelagem e replicação da visão humana que, utilizando software e hardware, é capaz de extrair informações do mundo real ao efetuar o processamento de imagens e o reconhecimento de padrões". Assim a Visão computacional busca auxiliar na resolução de problemas complexos, se baseando na cognição humana e a habilidade do ser humano nas tomadas de decisões de acordo com as informações contidas nas imagens ([PASSOS, 2019](#)).

2.2.1 RECONHECIMENTO DE PADRÕES

Para Haykin (2001, apud [PASSOS, 2019](#)), "o reconhecimento de padrões é definido como o processo pelo qual um padrão recebido é atribuído a uma classe dentre um número pré-determinado de classes". Para um sistema de análise de imagens de alto nível o reconhecimento de padrões é uma técnica fundamental, envolvendo identificar e interpretar padrões nas

imagens, extraindo assim, informações relevantes. Existem várias metodologias utilizadas no reconhecimento de padrões, entre elas:

- Casamento (matching) : "As técnicas baseadas em matching representam cada classe usando um vetor de características. Sendo que um novo padrão, desconhecido, é atribuído à classe mais próxima com base em uma métrica predefinida" ([PASSOS, 2019](#));
- Classificadores estatísticos: "representam cada amostra em termos de suas características ou atributos. Tais características são expressas por meio de medidas compondo um espaço, denominado espaço de características" ([PASSOS, 2019](#)), sendo um exemplo comum o classificador bayesiano, o qual utiliza do teorema de Bayes para calcular probabilidades de pertencer a determinada classe apartir das características observadas;
- Redes Neurais Artificiais: "é um modelo de grafo orientado em que os nós representam neurônios artificiais e as arestas orientadas denotam as conexões entre as entradas e as saídas dos neurônios" ([PASSOS, 2019](#)), as Redes Neurais Artificiais (RNAs) são treinadas para aprender padrões a partir de um conjunto de dados de treinamento e podem ser usadas para reconhecimento de padrões em imagens, especificamente esta é a utilizada neste trabalho.

Segundo Pedrini e Schartz (2008, apud [PASSOS, 2019](#)), alguns exemplos de tarefas executadas através de um sistema de Visão computacional utilizando o reconhecimento de padrões são: reconhecimento óptico de caracteres, identificação de impressões digitais, identificação de faces, reconhecimento de assinaturas, identificação de vasos sanguíneos de imagens da retina e o reconhecimento automático de placas de veículos.

O reconhecimento de padrões depende da extração de características da imagem, a qual extraia informações importantes para compor um vetor de características, onde posteriormente são utilizados como dados de entradas para o treinamento de um classificador, fazendo com que ocorra a identificação e o reconhecimento do padrão procurado na imagem ([PASSOS, 2019](#)).

2.3 APRENDIZAGEM DE MÁQUINA

Ao longo das últimas décadas, com as mais diversas complexidades de problemas que são tratados computacionalmente além do grande volume de dados que são gerados, houve a necessidade de ferramentas computacionais aprimoradas que diminuíssem a necessidade da mediação humana realizada por especialistas da área, as quais são capazes de resolver o problema que é desejado tratar a partir da experiência que foi passada, sendo esse processo conhecido como Aprendizado de Máquina (AM) ([FACELI et al., 2011](#)).

O AM vêm ganhando muito espaço com o passar do tempo, onde é realizado a utilização de algoritmos que extraem informações e as representam através de um modelo matemático, utilizando assim esse modelo para realizar inferências apartir de um conjunto de dados ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)). O objetivo é desenvolver algoritmos que possam aprender por conta própria e melhorar seu desempenho ao longo do tempo. Logo, segundo [Faceli et al. \(2011\)](#), podemos dizer que:

"A capacidade de aprendizado é considerada essencial para um comportamento inteligente. Atividades como memorizar, observar e explorar situações para aprender fatos, melhorar habilidades motoras/cognitivas por meio de práticas e organizar conhecimento novo em representações apropriadas podem ser consideradas atividades relacionadas ao aprendizado".

A partir dessa definição, podemos verificar as principais formas de aprendizado.

2.3.1 TIPOS DE APRENDIZADO DE MÁQUINA E SUAS DEFINIÇÕES

As principais formas de aprendizado são o supervisionado, semi-supervisionado, não supervisionado e o aprendizado por reforço.

- Aprendizado supervisionado: "[...] técnica para treinar um modelo a partir de dados de entrada e seus rótulos correspondentes" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)), sendo esse o método utilizado neste trabalho;
- Aprendizado não supervisionado: "[...] técnica para treinar um modelo para encontrar padrões em um conjunto de dados, normalmente um conjunto de dados não rotulado" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#));
- Aprendizado Semi-Supervisionado: Pode ser considerado uma abordagem híbrida, onde possui conjunto de dados com exemplos rotulados e não rotulados ([NETO, 2019](#));
- Aprendizado por reforço: "[...] o sistema continuará aprendendo ao receber feedbacks através de suas interações com usuário e/ou ambiente. Conforme essas interações vão ocorrendo, o sistema vai ajustando-se automaticamente para melhorar seus próximos resultados" ([NETO, 2019](#)).

2.3.2 CONJUNTO DE DADOS

Para realizar o aprendizado de máquina, é necessário obter um conjunto de dados como entrada. O AM depende desses dados para aprender e aprimorar seus modelos, permitindo que a máquina aprenda com os exemplos e padrões. Para que esse conjunto de dados se adeque ao algoritmo de AM é necessário muitas vezes realizar técnicas de pré-processamento, como Redução de dimensionalidade, Transformação de dados, Balanceamento de dados, entre outros ([FACELI et al., 2011](#)). Esse conjunto de dados podem ser divididos em três conjuntos principais, sendo eles o treinamento, teste e a validação.

- Dados de Treinamento: Utilizamos os dados de treinamento para treinar o algoritmo e então criar o modelo preditivo;
- Dados de Teste: "Usamos os dados de teste para validar a performance do modelo já treinado, ou seja, apresentamos ao modelo dados que ele não viu durante o treinamento, a fim de garantir que ele é capaz de fazer previsões" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)).
- Dados de Validação: Utilizamos os dados de validação para avaliar o modelo durante o treinamento.

Existem técnicas que são utilizadas para a divisão do conjunto de dados, podemos citar dois métodos:

- Validação cruzada holdout: É um método de validação que consiste em separar o conjunto de dados em duas partes, utilizando uma proporção delas como amostra de validação ([CUNHA, 2019](#)). Este método é o utilizado neste trabalho;
- Validação cruzada K-fold: Este método divide os dados em subconjuntos, que são chamados de folds. Ao ser treinado e validado em cada iteração um dos folds é utilizado como conjunto de validação e os restantes dos folds são utilizados como conjunto de treinamento ([CUNHA, 2019](#)).

2.3.3 MEDIDA DE DESEMPENHO

A matriz de confusão mostra uma visão completa do desempenho do modelo em termos de erros e acertos referentes às classificações, fornecendo assim uma análise mais detalhada das diferentes classes, sendo uma análise fundamental para avaliar a eficácia do modelo e tomar decisões sobre seu ajuste e otimização. Segundo [Goodfellow, Bengio e Courville \(2016\)](#) a matriz de confusão é:

"[...] uma tabela NxN que agrupa as suposições corretas e incorretas de um modelo de classificação. Um eixo de uma matriz de confusão é o rótulo que o modelo previu e o outro eixo é a verdade básica. N representa o número de classes. Por exemplo, N=2 para um modelo de classificação binária".

Um exemplo de problema com duas classes, onde temos como classe "uma categoria de um conjunto de valores de destino enumerados para um rótulo. Por exemplo, em um modelo de classificação binária que detecta spam, as duas classes são spam e não spam" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)), uma classe é denotada positiva (+) e a outra negativa (-), temos então a matriz de confusão ilustrada na Figura 2 , onde entende-se que:

- VP: Número de verdadeiros positivos, das quais foram corretamente classificados como positivos;
- VN: Número de verdadeiros negativos, das quais foram corretamente classificados como negativos;
- FP: Números de falsos positivos, das quais foram erroneamente classificados como positivos;
- FN: Números de falsos negativos, das quais foram erroneamente classificados como negativos.

Figura 2 – Matriz de confusão para um problema com duas classes.

		Classe predita	
		+	-
Classe verdadeira	+	VP	FN
	-	FP	VN

Fonte: [FACELI et al. \(2011\)](#).

De acordo com Monard e Baranauskas (2003, apud [FACELI et al., 2011](#)) A partir da matriz de confusão, existem várias outras medidas de desempenho que são derivadas, tais como:

- Taxa de erro na classe positiva: Proporção de exemplos da classe positiva incorretamente classificada, conhecida como taxa de falsos negativos (TFN), tendo sua fórmula da seguinte maneira:

$$\text{Taxa de erro na classe positiva} = \frac{\text{FN}}{\text{VP} + \text{FN}}$$

- Taxa de erro na classe negativa: Proporção de exemplos da classe negativa que foi incorretamente classificada, conhecido como taxa de falsos positivos (TFP), dada sua fórmula da seguinte maneira:

$$\text{Taxa de erro na classe negativa} = \frac{\text{FP}}{\text{FP} + \text{VN}}$$

- Taxa de erro total: "Dada pela soma dos valores da diagonal secundária da matriz, dividida pela soma dos valores de todos os elementos da matriz" ([FACELI et al., 2011](#)), tido como a proporção de exemplos classificados de forma incorreta em relação ao total de exemplos. Dada sua formula da seguinte maneira:

$$\text{Taxa de erro total} = \frac{\text{FP} + \text{FN}}{\text{FP} + \text{FN} + \text{VP} + \text{VN}} = \frac{\text{Predições incorretas}}{\text{Todas as predições}}$$

- Taxa de acerto ou acurácia total: "[...]a fração de previsões que um modelo de classificação acertou. Em um modelo com acurácia de 82% dizemos que o modelo acerta 82 previsões a cada 100 previsões realizadas" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)), dada sua formula da seguinte maneira:

$$\text{taxa de acerto} = \frac{\text{VP} + \text{VN}}{\text{FP} + \text{FN} + \text{VP} + \text{VN}} = \frac{\text{Predições corretas}}{\text{Todas as predições}}$$

- Precisão: "[...] é a razão entre os Verdadeiros Positivos e todos os Positivos. Para nossa declaração de problema, essa seria a medida de pacientes que identificamos corretamente como tendo uma doença cardíaca entre todos os pacientes que realmente

a têm" (GOODFELLOW; BENGIO; COURVILLE, 2016). A fórmula de precisão é dada da seguinte maneira:

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

- Sensibilidade: "Esta medida é uma representação direta da quantidade de falsos negativos em uma classificação. Se FN = 0, a Sensibilidade é máxima e igual a 1. Se FN for muito alto, a Sensibilidade diminui" (BEDUIN, 2021)

$$\text{Sensibilidade} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

- Logarithmic Loss (Log Loss) : Também conhecido como cross-entropy loss, é uma métrica onde a classe real e a probabilidade da classe prevista são comparadas, sendo que, quanto menor for a diferença entre a probabilidade e a classe real, menor é o loss. Então, quanto mais próximo de zero for o resultado, melhor é o modelo (SAVIETTO, 2021).

2.4 REDES NEURAIS ARTIFICIAIS

O cérebro humano é uma máquina muito poderosa e complexa capaz de processar uma enorme quantidade de informações em pouco tempo, sendo através dos neurônios que as informações são transmitidas e processadas. Muitas coisas sobre o cérebro ainda não foram respondidas, mas o que se sabe é que ele desenvolve suas regras através da experiência adquirida por situações anteriormente vivenciadas (GOODFELLOW; BENGIO; COURVILLE, 2016).

A partir disso houve a motivação de desenvolver as Redes Neurais Artificiais (RNAs), tendo como inspiração a estrutura e o funcionamento do sistema nervoso, objetivando a simulação da capacidade de aprendizado do cérebro humano em adquirir conhecimentos (FACELI et al., 2011). Assim como o cérebro humano possui células chamadas de neurônios, as RNAs também possuem unidades chamadas de neurônios artificiais. Esses neurônios artificiais possuem camadas das quais trabalham juntas para processar as informações, cada camada possui uma função importante. Na Figura 3 pode-se visualizar as camadas das RNAs, a primeira camada é chamada de camada de entrada(Input Layer) onde recebe o conjunto de dados de entrada, realizam os cálculos matemáticos gerando assim uma saída de dados, e assim passam essa saída de dados dessa camada para a próxima camada, até chegar ao final da Rede Neural Artificial (RNA). A última camada da RNA é chamada de camada de saída (Output Layer), as camadas que ficam entre a camada de entrada e a camada de saída são referenciadas como camadas ocultas (Hidden Layer), onde cada camada possui um algoritmo simples contendo uma função de ativação (GOODFELLOW; BENGIO; COURVILLE, 2016).

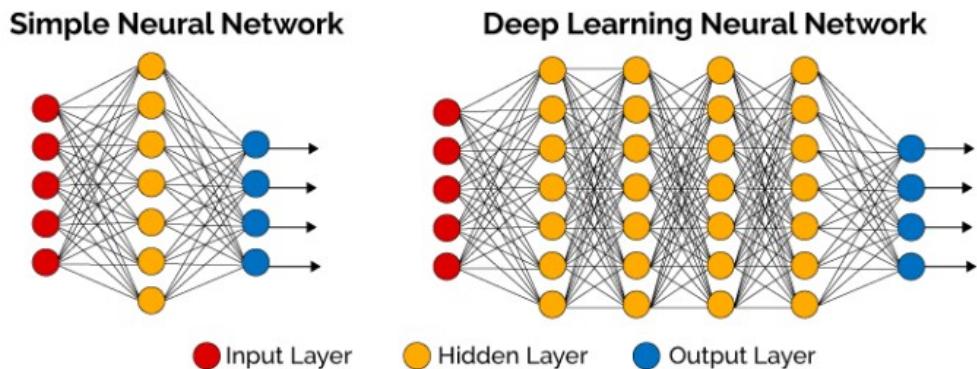
2.4.1 APRENDIZADO PROFUNDO

O aprendizado profundo, também conhecido como Deep Learning [...] usa camadas de neurônios matemáticos para processar dados, compreender a fala humana e reconhecer objetos

visualmente” (GOODFELLOW; BENGIO; COURVILLE, 2016), adquirindo grandes avanços recentes em visão computacional. O aprendizado profundo é baseado nos conceitos de RNAs onde tem como inspiração a maneira que o cérebro humano funciona.

“Pode-se entender o aprendizado profundo como o empilhamento de diversas camadas mais simples para formar um sistema final mais robusto” (NETO, 2019). A RNA é considerada profunda quando possui várias camadas intermediárias (ocultas) entre a camada de entrada e a camada de saída, permitindo o aprendizado de representações complexas e hierárquicas dos dados, sendo assim, o aprendizado profundo é uma evolução das RNAs (GOODFELLOW; BENGIO; COURVILLE, 2016), como podemos verificar de acordo com a Figura 3.

Figura 3 – Rede Neural Simples e Rede Neural Profunda.



Fonte: GOODFELLOW; BENGIO; COURVILLE (2016).

Segundo Goodfellow, Bengio e Courville (2016) a AM é um campo dentro da área de Inteligência Artificial. Uma de suas sub-área o aprendizado profundo (ou Redes Neurais Profundas), tem conseguido resultados excelentes nas tarefas de reconhecimento de padrões onde realiza o reconhecimento de objetos em cenas, palavras escritas, palavras faladas, identidades faciais, expressões faciais, detecção de anomalias onde detecta sequências incomuns em transações bancárias, previsões, onde prevê ações ou taxas de câmbio, recomenda quais filmes uma pessoa gostaria de assistir, entre outros exemplos. Mas para isso sabe-se que “elas normalmente exigem uma grande quantidade de dados para fins de treinamento, uma vez que são modelos bastante complexos” (NETO, 2019). Podendo assim a falta de dados acarretar ao modelo um sobreajuste (overfitting).

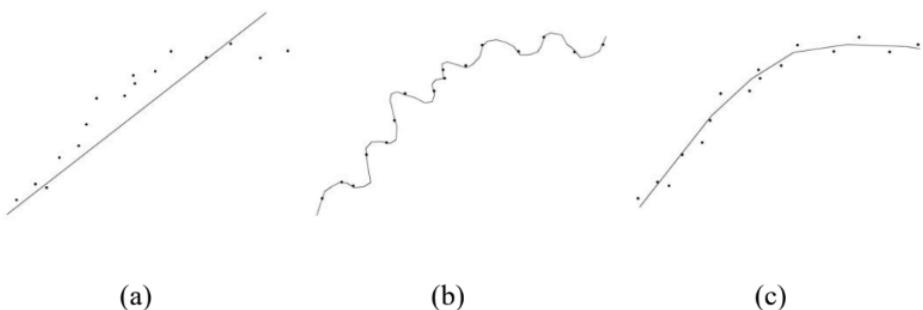
Existem diversas arquiteturas de RNAs, cada uma com sua própria forma de abordar diferentes tarefas. Algumas das arquiteturas mais comuns incluem: Redes Multilayer Perceptrons (Perceptrons Multicamadas), Redes Neurais Convolucionais, Redes Neurais Recorrentes, Long Short-Term Memory (LSTM), Redes de Hopfield, Máquinas de Boltzmann, Deep Belief Network, Deep Auto-Encoders, Generative Adversarial Network, Deep Neural Network Capsules. Essas são apenas algumas das muitas arquiteturas de RNAs existentes. Cada uma delas possui suas próprias características e é adequada para diferentes tipos de problemas e dados (GOODFELLOW;

BENGIO; COURVILLE, 2016).

2.4.1.1 DATA AUGMENTATION

Data augmentation é um método de regularização utilizado para impedir o overfitting. "[...] o overfitting ocorre quando o modelo aprende os detalhes nos dados de treino. Não é isso que queremos em Machine Learning. Em aprendizado de máquina buscamos a criação de um modelo que aprende a generalização dos dados, para então fazer previsões com novos dados" (GOODFELLOW; BENGIO; COURVILLE, 2016), ou seja, o modelo acaba aprendendo somente como representar os dados de treinamento, podendo levar a um desempenho ruim na generalização de novos dados, pois o modelo estará adaptado apenas ao conjunto de dados específicos do treinamento. No caso se a rede generalizar demais e não detectar os padrões de forma correta, chamamos de underfitting. Veremos na Figura 4 exemplos de underfitting, overfitting e de uma boa generalização.

Figura 4 – Exemplos de (a) underfitting, (b) overfitting e (c) boa generalização.



Fonte: RODRIGUES (2018).

Como nem sempre o conjunto de treinamento que se possui é grande o suficiente para que a rede aprenda os padrões desejados, "[...] no processo de data augmentation utilizamos diversas técnicas que aumentam os dados de forma artificial como: variação de brilho, saturação, zoom, reflexão horizontal e vertical, recorte entre outros métodos" (RODRIGUES, 2018). Fazendo assim, com que melhore os resultados da rede sem adicionar nenhuma característica nova a ser aprendida, reduzindo o overfitting, tornando assim a rede mais robusta.

2.4.1.2 FUNÇÃO DE ATIVAÇÃO

As funções de ativação são importantes por determinar a saída de cada neurônio, "Elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada" (GOODFELLOW; BENGIO; COURVILLE, 2016), sendo assim a função de ativação é a transformação não linear realizada ao longo do sinal de entrada, a saída transformada é enviada a próxima camada de neurônios de entrada. Quando não temos a função de ativação, os pesos e bias fazem a transformação linearmente, sendo uma equação simples de resolver, mas limitada

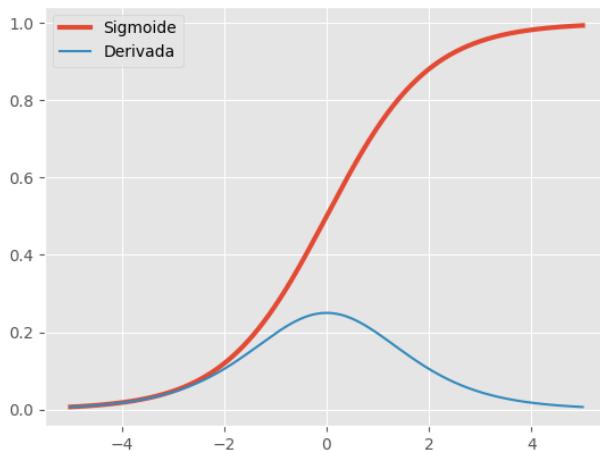
na capacidade de resolver problemas mais complexos. A função de ativação faz a transformação não-linear nos dados de entrada, fazendo com que a rede seja capaz de aprender e executar tarefas mais complexas ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)). As principais funções utilizadas podem ser vistas a seguir:

- Função sigmoide: Conhecida como função logística, sendo amplamente utilizada, podemos visualizá-la na Figura 5. A função é dada pela equação:

$$f(x) = \frac{1}{1 + e^{-x}}$$

"Ela assume valores apenas entre 0(não ativação) e 1(ativação). Comportamento bastante parecido com os próprios neurônios biológicos, que são ativados ou não dadas as entradas que recebe" ([RODRIGUES, 2018](#)).

Figura 5 – Função sigmoide.

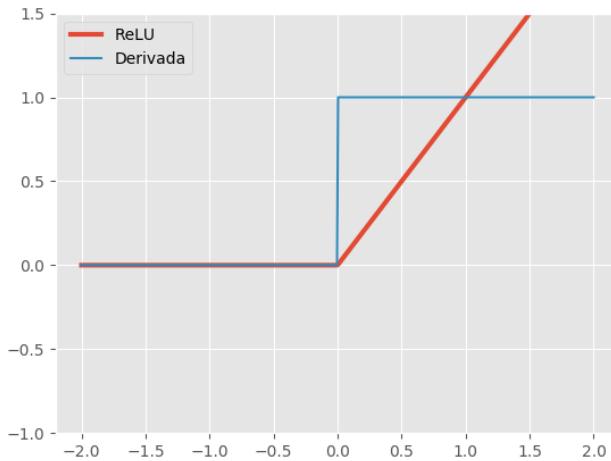


Fonte: [FACURE \(2017\)](#).

- Linear retificada (ReLU): "Retorna 0 ou um valor real maior que zero. Semelhante a função identidade, difere das funções anteriores, onde a propagação do gradiente desvanece nas regiões de causa, e se mostra bastante fácil de se otimizar" ([RODRIGUES, 2018](#)). É bastante utilizada na prática, podendo ser visualizada na Figura 6. Sua fórmula é dada por:

$$\text{ReLU}(x) = \max(0, x)$$

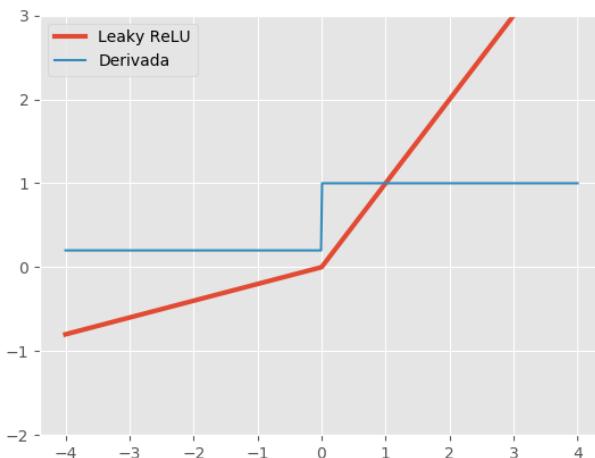
Figura 6 – Função ReLU.

Fonte: [FACURE \(2017\)](#).

- Leaky ReLU: Essa função é uma versão melhorada da função ReLU, "Na função ReLU, o gradiente é 0 para $x < 0$, o que fez os neurônios morrerem por ativações nessa região. Leaky ReLU ajuda a resolver este problema" [GOODFELLOW; BENGIO; COURVILLE](#). Podemos vizualizar de acordo com a Figura 7. Sua formula é dada por:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{se } x \geq 0 \\ ax, & \text{caso contrário} \end{cases}$$

Figura 7 – Função Leaky ReLU.

Fonte: [FACURE \(2017\)](#).

De acordo com [Goodfellow, Bengio e Courville \(2016\)](#), "substituímos a linha horizontal por uma linha não-zero, não horizontal. Aqui um é um valor pequeno como 0,01 ou algo

parecido. A principal vantagem de substituir a linha horizontal é remover o gradiente zero". A Leaky ReLU é uma variação da função ReLU que introduz uma pequena inclinação para valores negativos;

- Softmax: A função softmax é um tipo de função sigmoide, onde transforma as saídas de cada classe para valores entre 0 e 1, dividindo pela soma das saídas, dando assim uma probabilidade de a entrada estar em uma determinada classe, sendo preferencialmente usada na camada de saída do classificador, onde é gerado as probabilidades para definir a classe de cada entrada ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)).

2.5 REDE NEURAL CONVOLUCIONAL

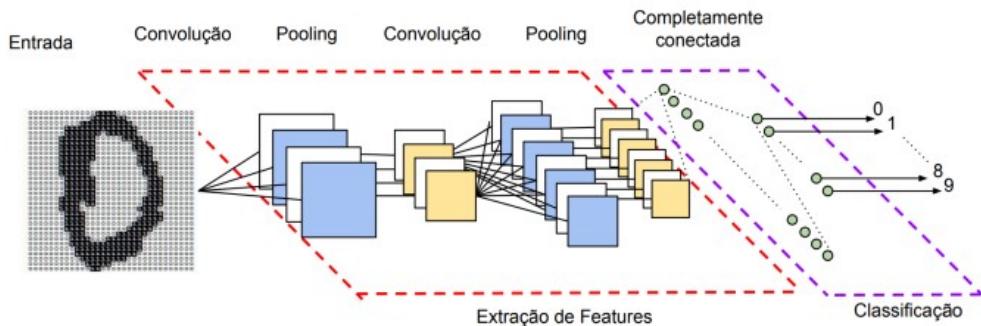
A Rede Neural Convolucional, conhecida como Convolutional Neural Network (CNN) é um tipo de rede neural profunda utilizada em visão computacional aplicada ao processamento de imagens e que "podem ser usadas para classificar imagens, agrupá-las por similaridade (busca de fotos) e realizar reconhecimento de objetos dentro de cenas. São algoritmos que podem identificar rostos, indivíduos, sinais de rua, cenouras, ornitorrincos e muitos outros aspectos dos dados" ([GOODFELLOW; BENGIO; COURVILLE, 2016](#)), sua arquitetura é formada por três tipos de camadas: a camada de convolução, a de subamostragem (pooling) e a totalmente conectada. Como podemos observar, de acordo com a Figura 8, as camadas de convolução e de pooling fazem parte da extração de características da imagem, enquanto que a camada totalmente conectada faz parte da classificação dos dados.

As CNNs lidam com três tipos diferentes de dados, dependendo das dimensões desses dados ([VERMA, 2019](#)). Podendo ser definidos da seguinte forma:

- 1D: Dados unidimensionais, sequência de textos ou dados sequenciais, podendo ser processados por modelos chamados de CNN1D ([KIRANYAZ et al., 2021](#));
- 2D: Dados de duas dimensões, altura e largura, sendo essas imagens, podendo ser processados por modelos chamados de CNN2D;
- 3D: Dados tridimensionais, além das dimensões de altura e largura também considera a dimensão de profundidade, podendo ser processados por modelos chamados CNN3D.

Essas estruturas podem ser conferidas diretamente na documentação do Keras ([Keras Team, 2023](#)).

Figura 8 – Exemplo de uma rede neural convolucional e suas diferentes camadas



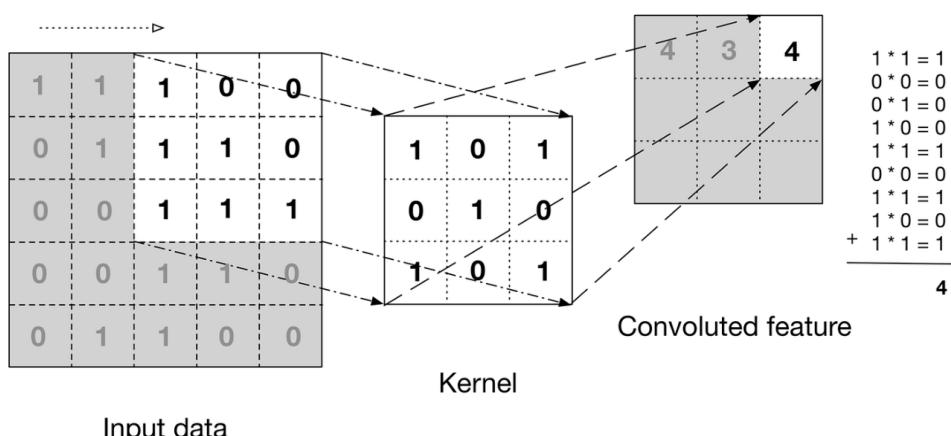
Fonte: VARGAS; PAES; VASCONCELOS (2016).

Veremos a seguir a arquitetura da rede neural convolucional de maneira mais detalhada.

2.5.1 CAMADA CONVOLUCIONAL

Segundo Queifer (2019), "As camadas convolucionais são responsáveis por extrair atributos dos volumes de entradas", ajudando na identificação de características importantes das imagens, como bordas, formas e cores. Conforme Ludwig (2007, apud NETO, 2019), "a convolução é um processo de filtragem feito tomando-se o valor central resultante da multiplicação de partes da imagem por outras matrizes menores. Estas últimas sendo os filtros, também conhecidos como kernels", entendendo assim, a convolução como um método de mover filtros sobre a imagem conforme a Figura 9, o qual esses filtros são matrizes de pesos, onde os pesos gerados são ajustados durante o processo de treinamento.

Figura 9 – Exemplo de uma operação convolucional



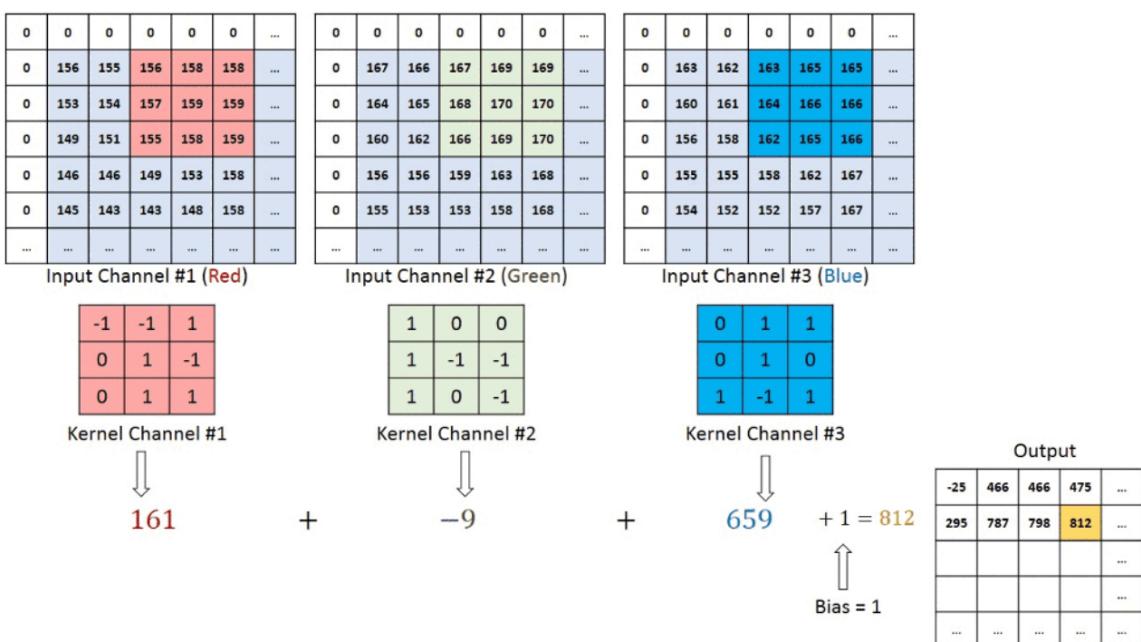
Fonte: LACERDA (2019b).

Quando criado a estrutura de uma CNN, na convolução é fundamental a passagem de alguns parâmetros como o tamanho da matriz do kernel, o salto (stride) e o preenchimento de zero (zero padding), como visto na Figura 10

- Kernel: Refere-se a um filtro, o qual é formado por uma matriz de pesos.

- Stride: A Stride ou salto é a quantidade de colunas/linhas de pixel que o kernel irá saltar ao percorrer uma imagem.
 - Zero Padding: O Padding adiciona borda na imagem com valores de 0 (zero), para que o kernel analise toda a imagem sem interferências no resultado da convolução.

Figura 10 – Exemplo de uma operação convolucional com imagem RGB



Fonte: LACERDA (2019b).

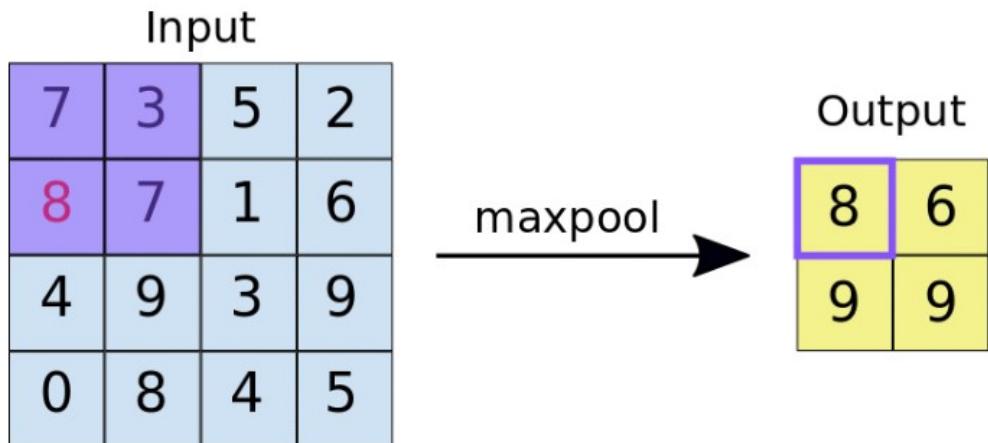
2.5.2 CAMADA DE POOLING

De acordo com Queifer (2019), "As camadas de pooling são responsáveis por reduzir a dimensionalidade do volume resultante após as camadas convolucionais e ajudam a tornar a representação invariante a pequenas translações na entrada". Geralmente após a camada convolucional existe a camada de pooling, "Cada mapa de características de uma camada de pooling está conectada ao correspondente mapa de características da camada convolucional anterior" (UTSCH, 2018), objetivando reduzir progressivamente a dimensão espacial do volume de entrada diminuindo assim, o custo computacional da rede. Ele funciona de maneira parecida com os filtros convolucionais visto anteriormente, mas ao invés de multiplicar os pixels, os filtros realizam operações para encontrar o valor máximo ou a média, onde para cada vizinhança que o filtro percorrer apenas o valor máximo será extraído (NETO, 2019).

- Max pooling: Essa função realiza operações onde o filtro passa e encontra o valor máximo da região. Esse máximo é então extraído, conforme podemos verificar na Figura 11. No exemplo apresentado, a imagem de entrada é de tamanho 4×4 e os dados são submetidos a um pooling de tamanho 2×2 . Podemos observar pelas cores que cada cor representa uma matriz de pooling passando pela imagem, resultando em um mapa de características (features) de tamanho 2×2 .

- Average pooling: Essa função calcula a média dos valores em uma região. Assim como o max pooling, ela também reduz a dimensionalidade dos dados. No entanto, em vez de selecionar o valor máximo, ela calcula a média dos valores.

Figura 11 – Exemplo do funcionamento da camada de Pooling



Fonte: [ALVES \(2018\)](#).

2.5.3 CAMADA TOTALMENTE CONECTADA

A camada totalmente conectada (ou Fully Connected(FC), em inglês), é responsável por conectar todos os neurônios da camada anterior a cada neurônio da camada atual. Segundo [Queifer \(2019\)](#), "As camadas totalmente conectadas são responsáveis pela propagação do sinal por meio da multiplicação ponto a ponto e o uso de uma função de ativação". Normalmente, os filtros da primeira camada convolucional são projetados para detectar características básicas, enquanto os filtros das próximas camadas detectam características mais complexas e abstratas. À medida que várias camadas convolucionais e de pooling são empilhadas, as características extraídas se tornam gradualmente mais abstratas, fazendo com que a camada totalmente conectada interprete essas características abstratas e realize funções de raciocínio complexo, podendo assim, por exemplo, ser utilizada para classificar objetos em uma imagem ou detectar pessoas em um vídeo ([UTSCH, 2018](#)).

2.5.4 TÉCNOLOGIAS UTILIZADAS EM REDES NEURAIS CONVOLUCIONAIS

Sabendo que as Redes Neurais Convolucionais (CNNs) são amplamente utilizadas na visão computacional em tarefas de processamentos de imagens, nesta seção serão tratadas tecnologias e bibliotecas populares que são utilizadas para implementar e treinar as redes neurais convolucionais.

- TensorFlow: É uma biblioteca de código aberto utilizada para computação numérica de alto desempenho, foi criada pela equipe do Google Brain para aprendizado de máquina e pesquisa de redes neurais profundas ([RIBEIRO; QUIMARÃES, 2018](#)), onde "Sua arquitetura flexível permite a fácil implantação de computação em várias plataformas

(CPUs, GPUs, TPUs) e de desktops a clusters de servidores para dispositivos móveis e periféricos" (BERTONI; FEDER, 2018).

- Keras: É uma biblioteca de código aberto que facilita a criação de modelos de redes mais profundas (RIBEIRO; QUIMARÃES, 2018). "Foi desenvolvida para tornar a implementação de modelos de deep learning o mais rápido e fácil possível para pesquisa e desenvolvimento. Compatível com Python 2.7 ou 3.5, pode ser executada perfeitamente em GPUs e CPUs, considerando as estruturas subjacentes" (BERTONI; FEDER, 2018).
- Python: A linguagem python é uma linguagem de fácil entendimento, possuindo uma estrutura de dados mais eficiente e de alto nível, além de possuir uma extensa biblioteca (ROSSUM; JR, 1995), tornando assim uma das escolhas mais populares para ser utilizada em aprendizado de máquina e aprendizado profundo;
- OpenCV: Sigla para Open Source Computer Vision Library (em português, Biblioteca de Computação Visual em Código Aberto), o OpenCV é uma das principais bibliotecas multiplataforma de código aberto contando com mais de 500 funções para o uso em visão computacional, tratamento de imagens, reconstrução 3D, Aprendizado de Máquina, processamento de imagens, entre outras (ABERTO, 2022).
- numpy: É uma biblioteca de cálculos fáceis, ajuda a trabalhar de maneira mais rápida com a manipulação de imagens.

2.6 TRABALHOS RELACIONADOS

Nesta seção serão apresentados os trabalhos referentes ao Reconhecimento de Sinais de Libras utilizando RNAs, tais quais foram baseados no desenvolvimento deste trabalho.

Em Neto (2019), o autor utiliza a detecção de borda usando o algoritmo de Canny, o conjunto de dados é formado pelas letras do alfabeto sendo o total de 20 letras, tido como sinais estáticos. Para o desenvolvimento do conjunto de dados o próprio autor relata a dificuldade de encontrar um conjunto apropriado para realizar o treinamento, sendo assim realizou o desenvolvimento do conjunto de dados utilizado em seu trabalho, utilizando uma técnica conhecida como Chroma Key, onde é utilizado um fundo verde. Além de realizar uma detecção de pele antes de realizar a detecção das letras. Foi desenvolvido uma interface para a utilização em tempo real. A rede neural utilizada nesse trabalho foi a CNN2D, onde a taxa de acerto(acurácia) do modelo desenvolvido foi de 88% .

Em Santos et al. (2019), o objetivo do trabalho é demonstrar a viabilidade da utilização de redes neurais convolucionais no reconhecimento de gestos estáticos de LIBRAS, tendo como gesto estáticos os numerais do 0 ao 9. Para o dataset foi criado uma base de dados com 4.000 (quatro mil) imagens, com tamanho 28x28, e fundo branco, sendo os sinais de 0 ao 9 da LIBRAS. O seguinte trabalho utilizou a arquitetura LeNet-5, usando a ferramenta WEKA para realizar o treinamento do modelo proposto. Resultando em uma taxa de acerto (acurácia) de 98%.

Em Bastos (2016), utilizou-se a redes Perceptron Multicamada, onde foi criado um

conjunto de dados com 40 sinais de LIBRAS, sendo esses: 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'I', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', '1', '2', '4', '5', '7', '9', 'Adulto', 'América', 'Avião', 'Casa', 'Gasolina', 'Identidade', 'Juntos', 'Lei', 'Palavra', 'Pedra', 'Pequeno' e 'Verbo. Neste trabalho não foi incluido as letras 'H', 'J' e 'Z', por apresentarem parâmetros de movimento. Os numeros '3' e '8' não foram inclusos por se tratarem de gestos similares aos sinais de 'W' e 'S', além do sinal '6' que se assemelha ao '9'. Ao todo, para cada um dos 40 sinais, foram tiradas 120 imagens com resolução de 50x50 pixels, tendo assim um total de 4.800 imagens. Neste trabalho utiliza-se detecção de pele. Como resultado final teve a taxa de acerto em 96,77%.

Na Tabela 1 mostra um resumo com as informações acerca dos trabalhos relacionados nesta seção de maneira comparativa.

Tabela 1 – Tabela comparativa dos trabalhos apresentados nesta seção

Trabalho	Base de Dados	Rede Neural	Sinais	Acurácia
(NETO, 2019)	Própria	CNN2D	20	88%
(SANTOS et al., 2019)	Própria	CNN2D	10	98%
(BASTOS, 2016)	Própria	Perceptron Multicamada	40	96.77%

Fonte: Elaborado pelo autor.

3 METODOLOGIA

Neste capítulo será apresentado o desenvolvimento de uma metodologia adaptada do trabalho de [Lacerda \(2019a\)](#). Desta maneira a metodologia será formada pelas seguintes etapas: Na seção [3.1](#) será descrito como foi feito a configuração do ambiente. Na seção [3.2](#) será informado as configurações da máquina que foi utilizada para desenvolver o projeto. Na seção [3.3](#) é descrito como que foi definido o conjunto de dados. Na seção [3.4](#) é descrito a etapa de pré-processamento dos dados. Na seção [3.5](#) fala sobre a estrutura da rede neural convolucional. Na seção [3.6](#) irá informar como foi realizado o treinamento da rede Neural Convolutinal. Na seção [3.7](#) será abordado a forma que ocorre o reconhecimento em tempo real.

3.1 CONFIGURAÇÃO DO AMBIENTE

Para o desenvolvimento da aplicação foi necessário a instalação de um conjunto de bibliotecas e pacotes, entre eles os principais utilizados é o Keras, TensorFlow, OpenCV e Numpy, suas definições encontram-se na subseção [2.5.4](#). A linguagem que será utilizada para realizar as implementações é a linguagem python utilizando a IDE PyCharm para realizar o desenvolvimento.

A seguir a tabela [2](#) mostra as versões que foram utilizados para a elaboração do trabalho:

Tabela 2 – Tabela de versões

Aplicação	Versão
Python	3.11
TensorFlow	2.12.0
Keras	2.12.0
OpenCV	4.7.0.72
NumPy	1.24.3
PyCharm	2022.3.3

Fonte: Elaborado pelo autor.

3.2 CONFIGURAÇÃO DAS MÁQUINAS UTILIZADAS

Para o treinamento dos experimentos foi utilizado um computador com as seguintes configurações:

- CPU: AMD Ryzen 7 2700X;
- Placa-mãe: ASUS TUF GAMING X570-PLUS-Chipset AMD X570

- Memória RAM: 16GB (2x8) 3000MHz
- GPU: AMD Radeon RX 5700 XT 8GB
- SSD: Gigabyte Aorus - NVMe M.2, Gen4 1TB
- Sistema Operacional: Windows 10 Pro

Para demais demandas do trabalho foi utilizado um notebook lenovo com as seguintes configurações:

- Processador: Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz 2.30 GHz;
- Memória RAM: 6,00 GB;
- Sistema Operacional: windows 10 Home

3.3 CONJUNTO DE DADOS

Nesta seção será abordado o conjunto de dados utilizados para o desenvolvimento deste trabalho. Foi utilizado um conjunto de dados pré-existente das letras do alfabeto em LIBRAS, disponível em [Lacerda \(2019a\)](#), para realizar o experimento I. No entanto, para os experimentos subsequentes, não foi encontrado conjuntos de dados adequados que correspondessem aos padrões desejados. Portanto, foi necessário criar um novo conjunto de dados com sinais e métricas relevantes. O estudo é realizado com 3 conjuntos de dados distintos, sendo esses: letras estáticas do alfabeto em LIBRAS, números estáticos em LIBRAS e palavras estáticas em LIBRAS.

3.3.1 INFORMAÇÕES DO CONJUNTO DE DADOS

Conforme o [Bastos \(2016\)](#) descreve, os sinais do alfabeto são semelhantes a alguns sinais de números, sendo esses a letra W e o número 3, e a letra S e o número 8, onde não é possível utilizar tais gestos por serem semelhantes. Sabendo que o [Santos et al. \(2019\)](#) realizou o trabalho apenas com os números de 0 a 9, neste trabalho optou-se por separar esses conjuntos de dados em três experimentos, sendo eles: sinais do alfabeto, números e palavras em LIBRAS.

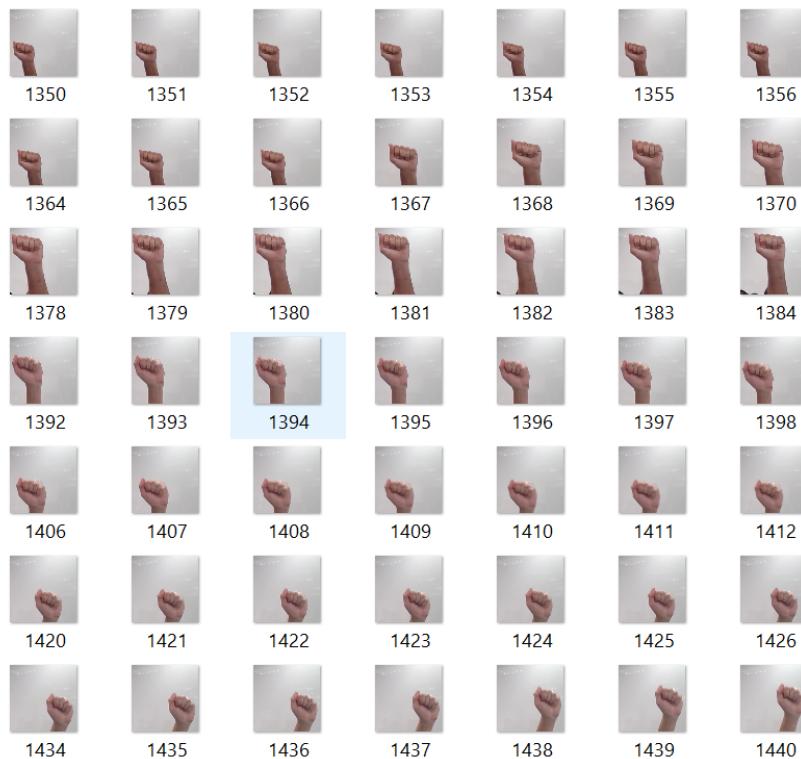
- Experimento I: O conjunto de dados utilizado são as letras do alfabeto em LIBRAS. Conforme mencionado por [Neto \(2019\)](#), os sinais de LIBRAS são divididos em estáticos e dinâmicos. A proposta deste trabalho é utilizar os gestos estáticos, o que exclui algumas letras do alfabeto em LIBRAS que requerem movimentos dinâmicos, como as letras H, K, J, X e Z. Para esse conjunto de dados, foi utilizado o banco de dados disponibilizado por [Lacerda \(2019b\)](#), com as letras: A, B, C, D, E, F, G, I, L, M, N, O, P, Q, R, S, T, U, V, W e Y.
- Experimento II: O conjunto de dados utilizados são os números em LIBRAS. A escolha desse conjunto se baseou no trabalho de [Santos et al. \(2019\)](#), o qual utilizou os sinais de números do 0 ao 9, sendo eles sinais estáticos. Por não encontrar um banco de dados disponível com as métricas adequadas, foi então necessário criar o banco de dados desse conjunto com os sinais dos números de 0 a 9, conforme Figura 13, baseado nas métricas

do banco de dados das letras, para manter essa padronização dos dados.

- Experimento III: O conjunto de dados utilizado consiste nas palavras em LIBRAS. A escolha dessas palavras foi baseada no trabalho de [Bastos \(2016\)](#), onde os gestos foram selecionados com a ajuda de profissionais da área, para captar sinais realizados exclusivamente pelas mãos. As palavras selecionadas para compor esse banco de dados são: "Adulto", "América", "Casa", "Gasolina", "Juntos", "Lei", "Palavra", "Pedra", "Pequeno" e "Verbo". Por não encontrar um banco de dados disponível com as métricas adequadas foi necessário criar um novo banco de dados para esse conjunto, conforme Figura 14, seguindo as métricas padronizadas.

A padronização dos conjuntos de dados foi realizada a partir do conjunto de dados das letras. Esse conjunto possui imagens com tamanho de 64x64 pixels e fundo branco. Os sinais captam principalmente a mão e, em alguns casos, parte do braço. Esses sinais são realizados em diferentes distâncias, com variações de iluminação e por diferentes pessoas. Além disso, foram consideradas as variações entre a mão direita e a mão esquerda. Essas características podem ser observadas na Figura 12.

Figura 12 – Conjunto de dados - sinal da letra A



Fonte: [Lacerda \(2019b\)](#) adaptado pelo autor.

O conjunto de aproximadamente 2.200 imagens possui variações para cada letra, conforme a Tabela 3, onde descreve exatamente a quantidade dos dados. Por se tratar de um conjunto de dados já existente, as informações contidas nele não foram alteradas. Assim, a partir desses valores, foi definido um valor aproximado para a criação do conjunto de dados dos experimentos II e III.

Tabela 3 – Tabela de informações do conjunto de dados do experimento I

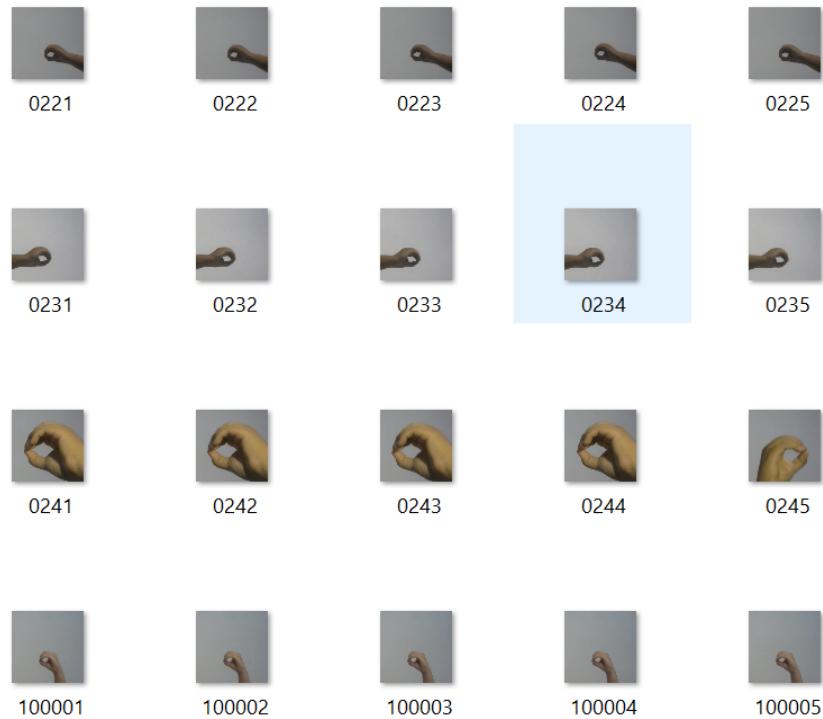
Classe	Treinamento	Teste	Total
A	1.686	579	2.265
B	1.662	562	2.224
C	1.686	583	2.269
D	1.650	550	2.200
E	1.670	574	2.244
F	1.647	450	2.097
G	1.650	550	2.200
I	1.650	550	2.200
L	1.650	550	2.200
M	1.650	550	2.200
N	1.650	550	2.200
O	1.650	550	2.200
P	1.650	550	2.200
Q	1.650	550	2.200
R	1.650	550	2.200
S	1.650	550	2.200
T	1.614	550	2.164
U	1.650	550	2.200
V	1.650	550	2.200
W	1.649	550	2.199
Y	1.650	550	2.200
Total	34.714	11.548	46.262

Fonte: Elaborado pelo autor.

3.3.2 CRIAÇÃO DO CONJUNTO DE DADOS

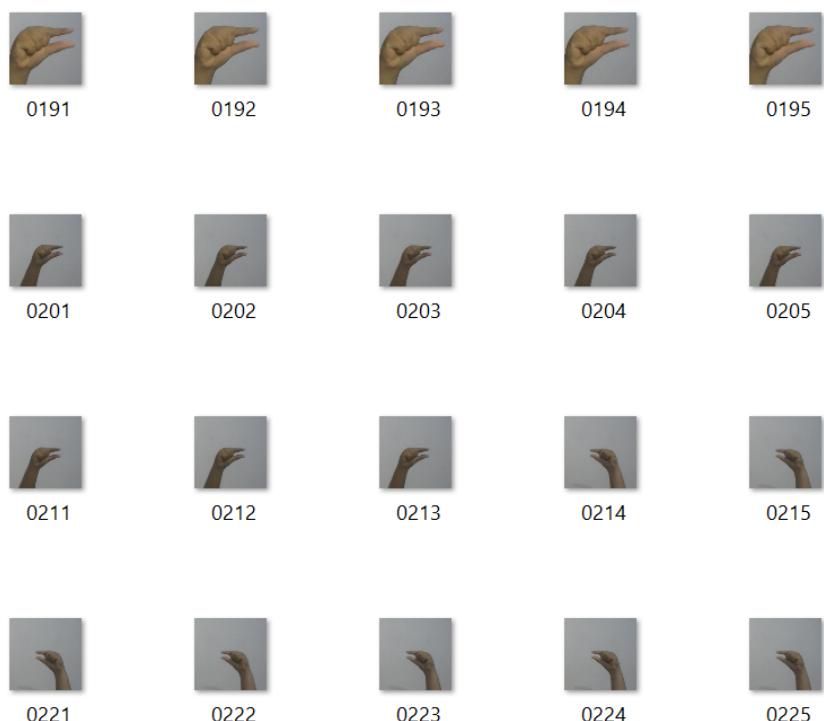
Para a criação do conjunto de dados dos experimentos II e III, foi utilizado um código em Python que captura imagens por meio de uma câmera, e as redimensiona para o tamanho de 64x64 pixels, já realizando a separação automática do conjunto de treinamento e o conjunto de teste. Todas as imagens do conjunto foram obtidas com fundo branco, e os sinais das mãos foram capturados em diferentes momentos, como de perto, um pouco longe e longe, a fim de diversificar o conjunto. Também foi considerada a variação entre duas pessoas para realizar os gestos, um homem e uma mulher, em ambientes com diferentes níveis de iluminação. Essa padronização dos dados foi adotada para garantir a diversidade do conjunto, conforme Figura 13 e Figura 14. Para captura das imagens foi utilizada a webcam Logitech HD 720p e a Webcam integrada do notebook.

Figura 13 – Conjunto de dados - variações sinal do número 0



Fonte: Elaborado pelo autor.

Figura 14 – Conjunto de dados - variações sinal da palavra "pequeno"



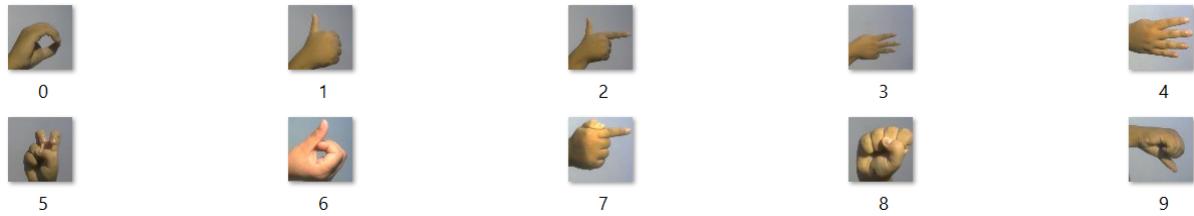
Fonte: Elaborado pelo autor.

Com base no trabalho de Júnior (2019), para todos os experimentos, foi utilizado o método de validação cruzada holdout, conforme descrito na subseção 2.3.2. Nesse método, foi

adotada uma divisão de 75% das amostras para treinamento e 25% para teste. Considerando que o experimento I utiliza para algumas classes valores variados, para essas classes a porcentagem se difere da proposta apresentada ocorrendo uma pequena variancia. Quanto aos dados de validação, as imagens serão geradas a partir do pré-processamento dos dados.

Podemos conferir as classes dos sinais realizados no conjunto de dados do experimento II e III, conforme Figura 33 e Figura 16.

Figura 15 – Conjunto de dados - Números



Fonte: Elaborado pelo autor.

Figura 16 – Conjunto de dados - Palavras



Fonte: Elaborado pelo autor.

Os dados dos conjuntos criados podem ser verificados conforme a Tabela 4.

Tabela 4 – Tabela de informações do conjunto de dados dos experimentos II e III

Descrição	Experimento II	Experimento III
Obtenção dos dados	Webcam integrada do notebook e a Webcam Logitech HD 720p	Webcam integrada do notebook e a Webcam Logitech HD 720p
Número de classes	10	10
Classes	0, 1, 2, 3, 4, 5, 6, 7, 8 e 9	'Adulto', 'América', 'Casa', 'Gasolina', 'Juntos', 'Lei', 'Palavra', 'Pedra', 'Pequeno' e 'Verbo'
Teste	550 imagens	550 imagens
Treinamento	1.650 imagens	1.650 imagens
Conjunto de dados	22.000 imagens	22.000 imagens
Dimensão	64x64 pixels	64x64 pixels

Fonte: Elaborado pelo autor.

3.4 PRÉ-PROCESSAMENTO DE DADOS

Utilizando a classe `ImageDataGenerator` é possível realizar um aumento no número de dados, por meio de pequenas modificações, conforme seção 2.4.1.1, além de obter uma porcentagem dessas imagens separadas para a validação. Uma lista mais detalhada de todas as opções do `ImageDataGenerator` pode ser encontrada em sua documentação¹. Assim, neste trabalho foi utilizado:

- `rescale`: Fator de reescalamento da imagem;
- `shear_range`: Intensidade de cisalhamento;
- `zoom_range`: Alcance para zoom;
- `horizontal_flip`: Inverte aleatoriamente as entradas horizontalmente;
- `validation_split`: Uma variação das imagens serão reservadas para a validação, sendo 0.25 das imagens de treino e 0.05 das imagens de teste.

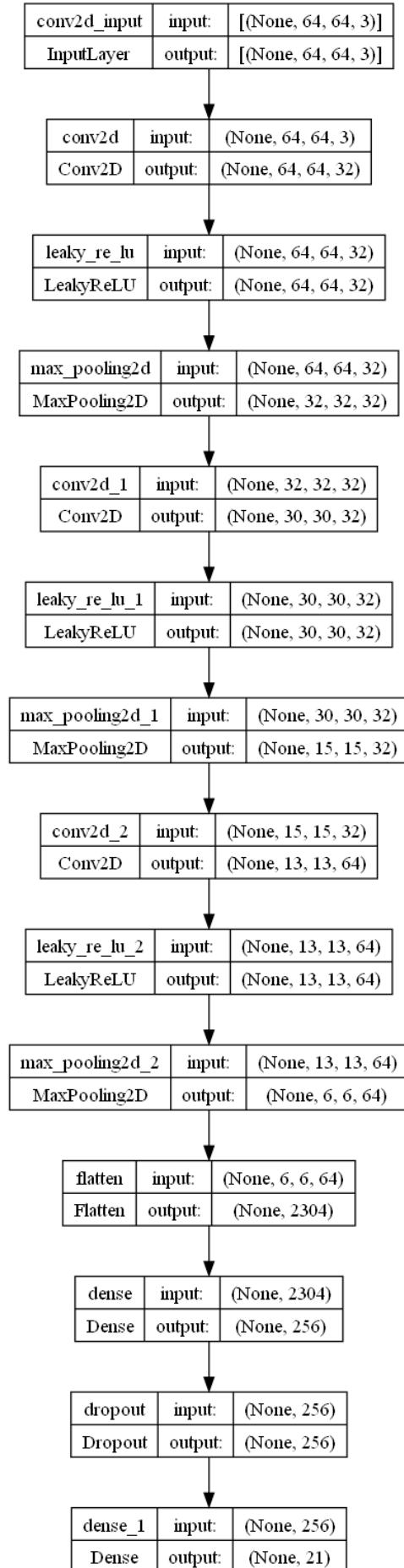
A partir disso as imagens são geradas com pequenas variações fazendo com que ajude no aumento da generalização do modelo a ser treinado.

3.5 DESCRIÇÃO DA ARQUITETURA DA REDE NEURAL CONVOLUCIONAL

Após realizar o pré-processamento, as imagens estão prontas para serem utilizadas no treinamento da rede. A estrutura da rede neural convolucional foi gerada por meio da biblioteca Keras, a qual é utilizada para montar a estruturas das redes usando as funções `Conv2D`, `LeakyReLU`, `MaxPooling2D`, `Flatten`, `Dense`, `Dropout`. Conforme a Figura 17, podemos ver a representação do modelo com a arquitetura final da rede, onde mostra os valores de entradas e saídas de cada camada, ao fim podemos verificar na última camada `Dense` que seu valor de saída corresponde aos número de classes, sendo essa a representação do experimento I. Assim, a arquitetura com suas respectivas entradas e saídas na Figura 18 se refere aos experimentos I e II, onde ambos possuem 10 classes.

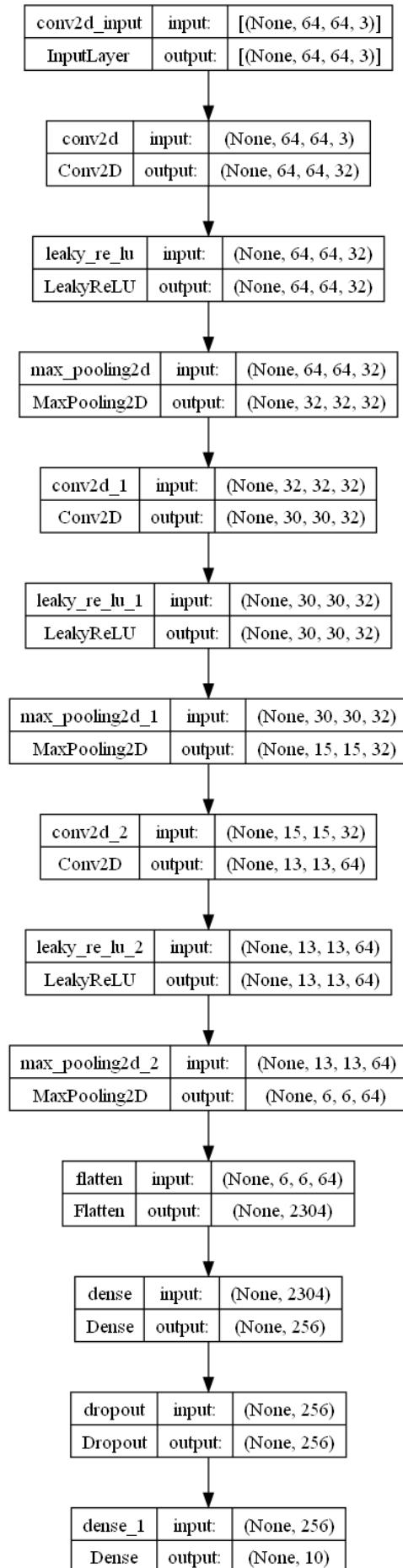
¹Disponível em: <https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator>

Figura 17 – Arquitetura CNN2D - valores de entradas e saídas das camadas



Fonte: Elaborado pelo autor

Figura 18 – Arquitetura CNN2D - valores de entradas e saídas das camadas



Fonte: Elaborado pelo autor

- Conv2D: Essa função é responsável por aplicar operações de convolução em imagens de entrada, trabalhando com dados 2D, extraíndo informações importantes. A definição da camada de convolução encontra-se na seção [2.5.1](#);
- LeakyReLU: É uma função de ativação, definida na seção [2.4.1.2](#) ;
- MaxPooling2D: É a função que representa a camada de Pooling, onde reduz a dimensionalidade, mas mantém as informações mais importantes, é definido na seção [2.5.2](#);
- Flatten: Nessa função os valores gerados com as características obtidas nas camadas anteriores são redimensionadas para se tornar um array linear de uma única dimensão. É usada como uma etapa de transição entre as camadas convolucionais e camadas totalmente conectadas, sendo extremamente importante já que a proxima camada demanda dados unidimensionais.
- Dense: É implementada a rede neural totalmente conectada, essa função é responsável por pegar os dados obtidos das camadas anteriores e realizar a predição final da classe correspondente à imagem de entrada.
- Dropout: Essa função é usada em redes neurais como uma técnica para evitar overfitting durante o treinamento. Nessa função faz com que seja "desativado" alguns neurônios da rede, fazendo com que a rede seja treinada de maneira mais independente, fazendo com que se obtenha um modelo mais robusto.

3.6 TREINAMENTO DA REDE NEURAL CONVOLUCIONAL

Depois da etapa de implementação da rede é realizado o treinamento, onde serão carregados os pesos da rede e em seguida escolhidos parâmetros de treinamento, como números de épocas e números de classes para o treinamento. O número de épocas é referente a quantidade de passadas completas pelos dados de treinamento. Cada época ajuda a melhorar a precisão e o desempenho do modelo. Para cada experimento a rede realizou 30 épocas, sendo 30 passadas completas pelo conjunto de treinamento.

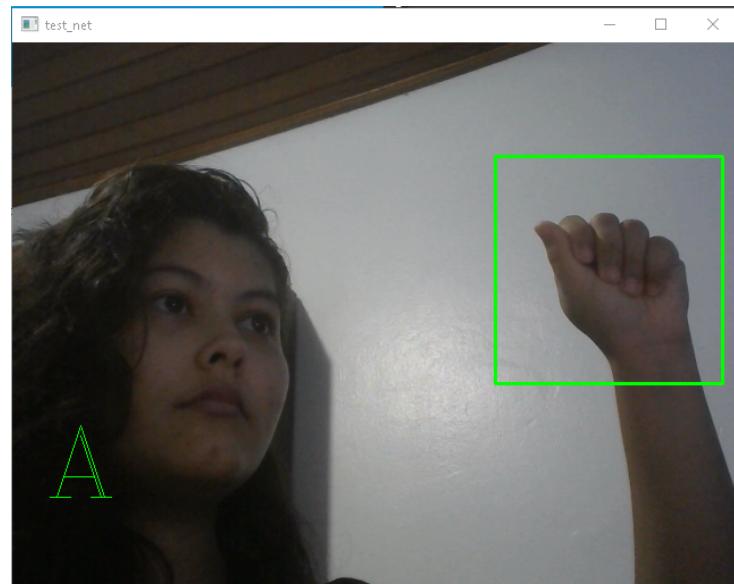
3.7 UTILIZAÇÃO DO MODELO EM TEMPO REAL

Para a utilização do modelo, foi gerada uma janela que utiliza a câmera para captar imagens em tempo real e classificar o sinal realizado, mostrando-o diretamente na tela. Nessa janela, há uma caixa delimitadora que restringe a imagem apenas ao objeto de interesse, no caso, o sinal realizado. A detecção do sinal só é gerada quando o sinal é feito dentro da caixa delimitadora.

Ao realizar o sinal de LIBRAS, o nome do sinal realizado aparecerá na tela, como podemos verificar na Figura [19](#). Além da tela principal, foram geradas mais duas telas auxiliares. Na Figura [20](#), podemos ver apenas a área da caixa delimitadora, e na Figura [21](#), a terceira tela

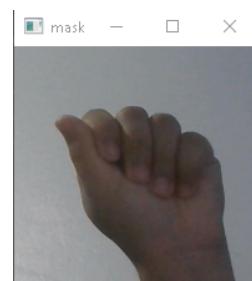
exibe a saída de texto com a classificação das imagens. Dessa forma, temos a visualização das três telas conforme a Figura 22.

Figura 19 – Reconhecimento em tempo real - Tela Principal



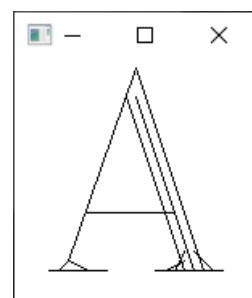
Fonte: Elaborado pelo autor

Figura 20 – Reconhecimento em tempo real - Tela da caixa delimitadora



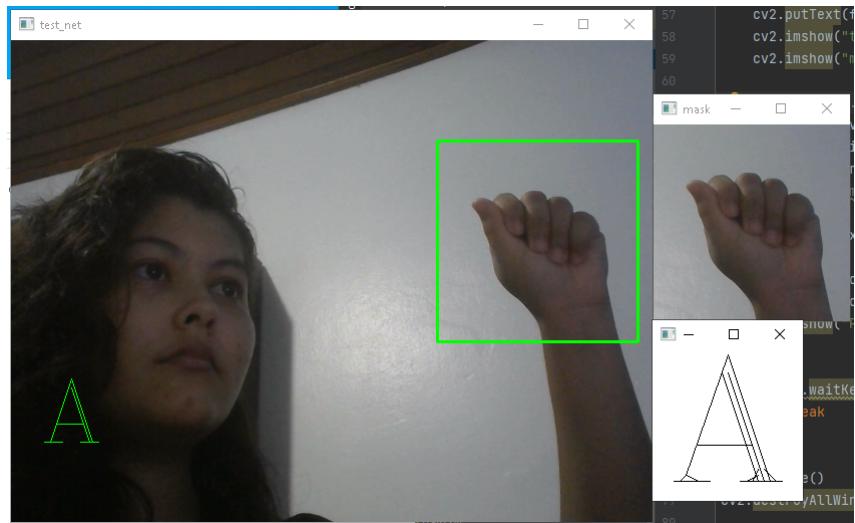
Fonte: Elaborado pelo autor

Figura 21 – Reconhecimento em tempo real - Tela saída de texto



Fonte: Elaborado pelo autor

Figura 22 – Reconhecimento em tempo real - Telas Geradas



Fonte: Elaborado pelo autor

4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Neste trabalho foram apresentados três experimentos para treinar a rede neural convolucional 2D, sendo que, para o experimento I, foi utilizado um conjunto de dados já criado, e nos experimentos II e III, foi desenvolvido o conjunto de dados.

4.1 RESULTADOS QUANTITATIVOS

Para a análise de desempenho da rede neural convolucional é utilizado as métricas matriz de confusão, acurácia e o loss, definidas na seção 2.3.3. Considerando a Tabela 5 podemos verificar os valores de Acurácia e Loss para cada conjunto.

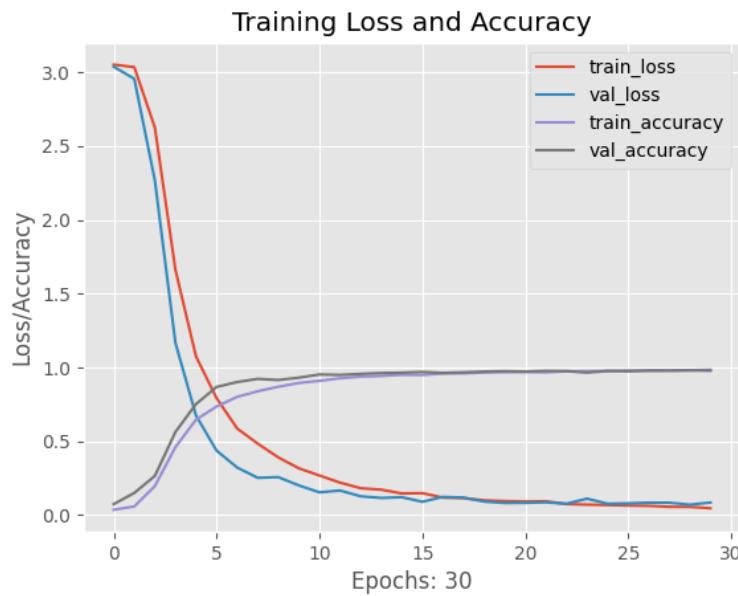
Tabela 5 – Acurácia e Loss dos conjuntos de dados

Conjunto de dados	Épocas	Acurácia (%)	Loss	Tempo de treinamento
Experimento I	30 épocas	97.93%	0.08522	41.9 min
Experimento II	30 épocas	91.39%	0.24582	19.8 min
Experimento III	30 épocas	91.19%	0.46170	19.6 min

Fonte: Elaborado pelo autor.

A seguir é apresentada uma síntese com a análise dos valores de acurácia e perda ao longo das épocas para cada experimento (Figura 23, Figura 24 e Figura 25), proporcionando assim uma visão geral do desempenho do modelo. Através dessa análise é possível avaliar se o modelo foi capaz de aprender os padrões relevantes do conjunto de dados treinados e se as previsões são mais precisas. Conforme a Figura 23, podemos verificar que a partir da 15^a época os valores se tornam constantes, sendo isso um indicador de que o modelo aprendeu os padrões dos dados de treinamento, fazendo assim previsões mais precisas.

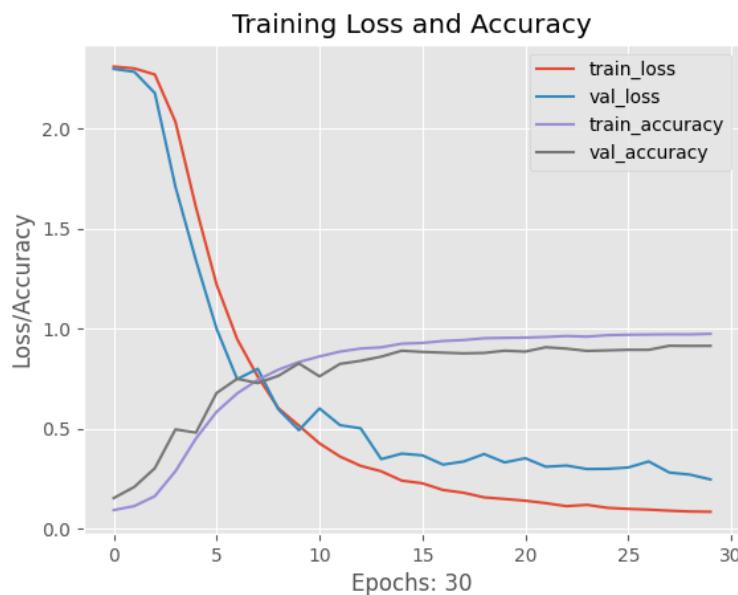
Figura 23 – Acurácia e Loss - Experimento I



Fonte: Elaborado pelo autor.

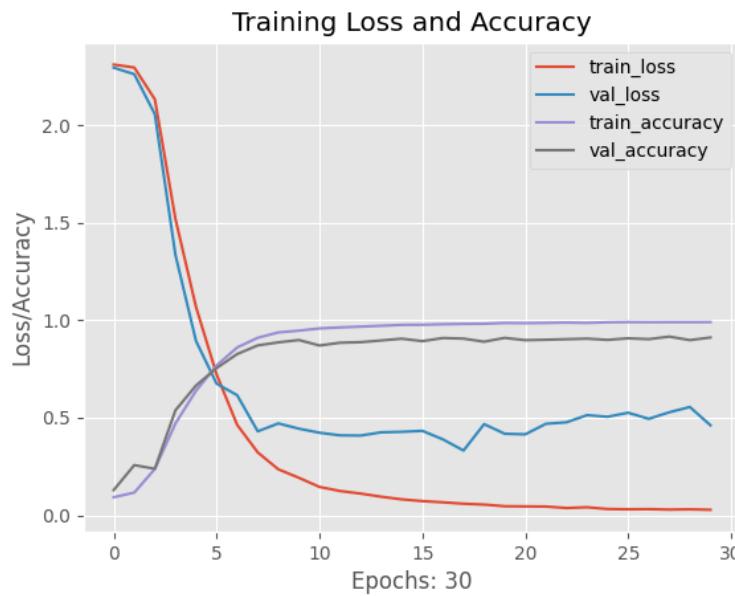
Para a Figura 24 e Figura 25, podemos observar a presença de uma linha de altos e baixos no valor de loss, o que indica uma certa irregularidade no desempenho do modelo durante o treinamento. Essa situação pode requerer ajustes e uma investigação adicional para alcançar valores mais estáveis e consistentes.

Figura 24 – Acurácia e Loss - Experimento II



Fonte: Elaborado pelo autor.

Figura 25 – Acurácia e Loss - Experimento III

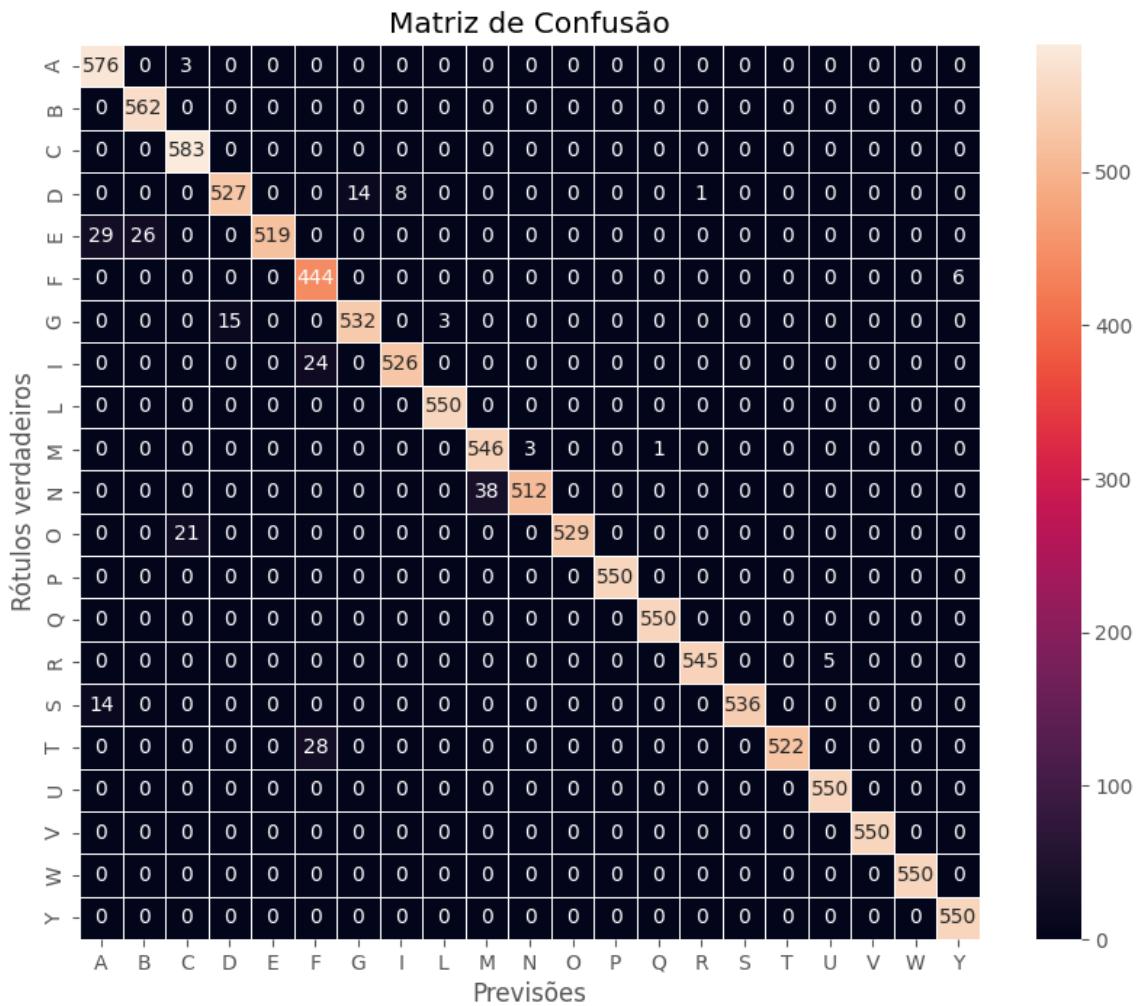


Fonte: Elaborado pelo autor.

A partir disso, podemos analisar a matriz de confusão, que em algumas classes obteve números de previsões nas classes erradas. Essa dificuldade na distinção entre algumas classes ocorre pelo fato de serem semelhantes tendo características visuais ou contextuais parecidas, tornando difícil para o modelo distingui-las corretamente, nos resultados qualitativos 4.2 veremos que ouve algumas detecções geradas erroneamente em determinado momento durante a detecção de alguns sinais.

Analisando a matriz de confusão do experimento I, apresentada na Figura 26, nos permite avaliar os erros e acertos relacionados às classes do experimento. Observamos que, das 21 classes, 10 tiveram todas as previsões corretas. Em relação às previsões erradas, a matriz revela valores baixos. A classe que apresentou o maior número de erros de previsão foi a classe E, com 29 previsões incorretas para a letra A e 26 para a letra B. Além disso, a classe N teve 38 previsões errôneas para a classe M.

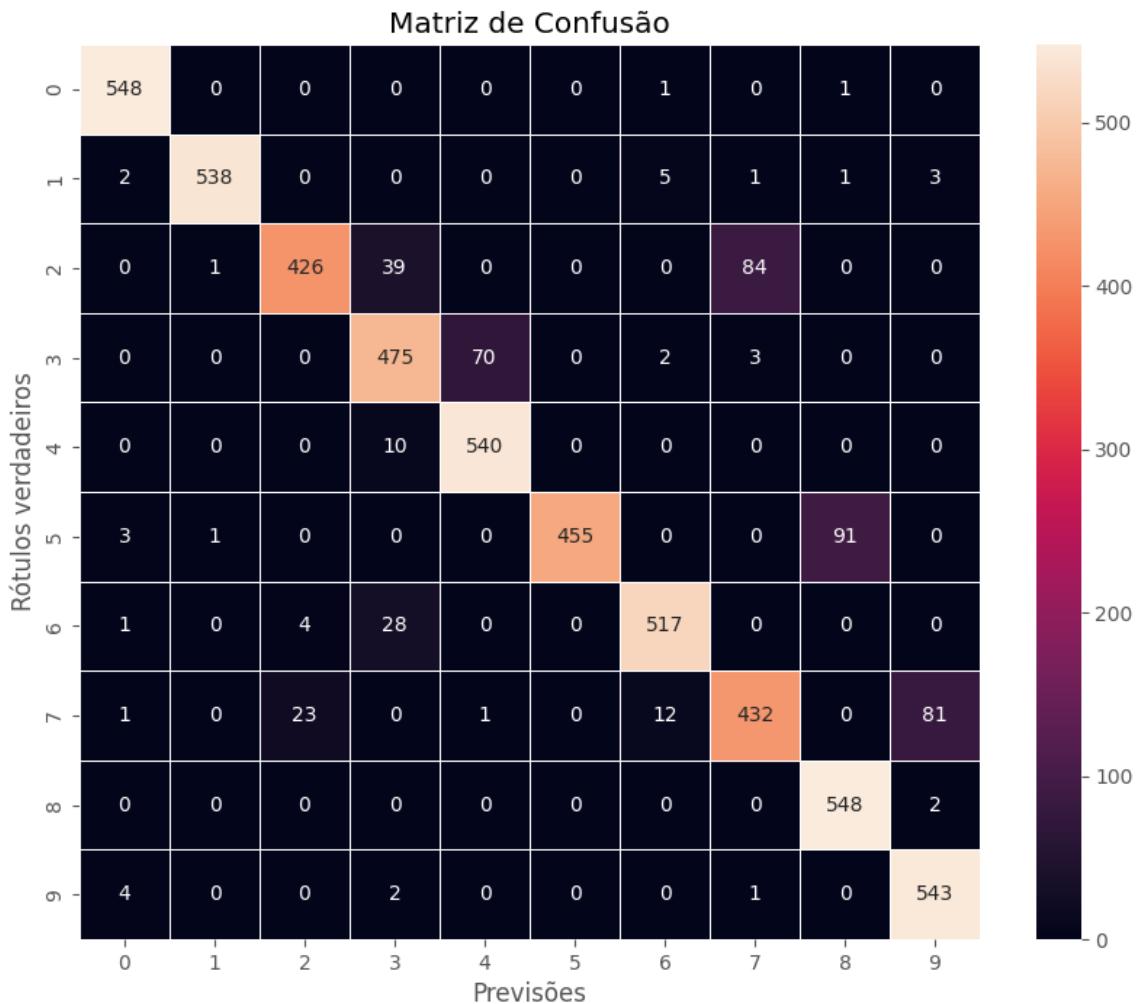
Figura 26 – Matriz de confusão - Experimento I



Fonte: Elaborado pelo autor.

Analisando a Figura 27, que representa a matriz de confusão do experimento II, podemos identificar padrões interessantes nas previsões das diferentes classes. Destacam-se as classes 0 e 8, que apresentaram os maiores índices de acertos, com apenas duas previsões incorretas. Por outro lado, a classe 2 registrou o maior número de erros, com 84 previsões equivocadas para o número 7, 39 previsões errôneas para o número 3 e apenas 1 para o número 1. Esses resultados ressaltam a importância de avaliar o desempenho do modelo em cada classe individualmente.

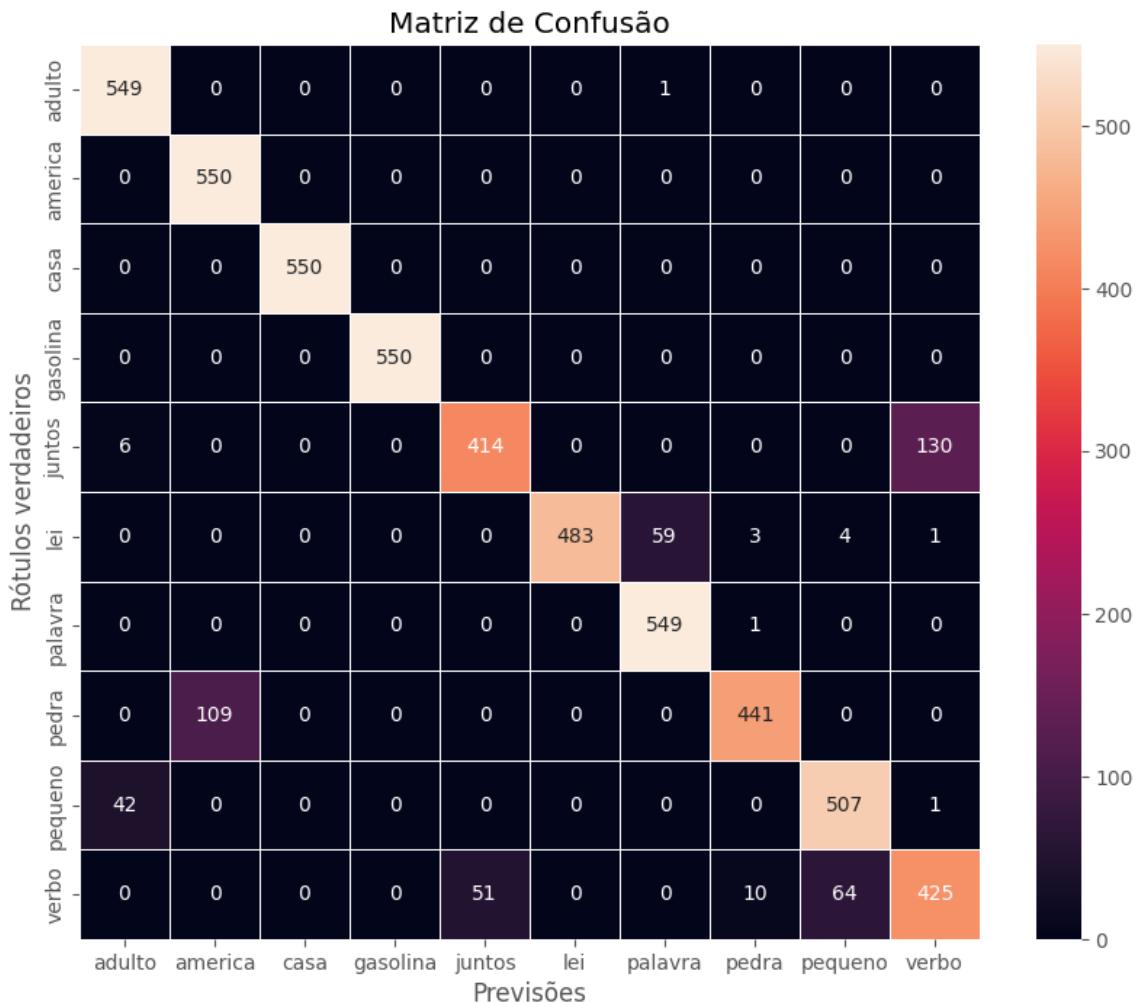
Figura 27 – Matriz de confusão - Experimento II



Fonte: Elaborado pelo autor.

Ao analisarmos a matriz de confusão do experimento III, representada pela Figura 28, podemos observar que, das 10 classes avaliadas, 3 tiveram acertos em todas as previsões. Entretanto, algumas classes apresentaram um maior número de erros nas previsões. Destaca-se a classe 'juntos', que teve 130 imagens erroneamente previstas como 'verbo', dentre as 550 imagens analisadas. Além disso, a classe 'verbo' teve 51 previsões equivocadas, classificadas como 'juntos', 64 como 'pequeno' e 10 como 'pedra'. Essas classes foram as que menos obtiveram acertos nas previsões.

Figura 28 – Matriz de confusão - Experimento III



Fonte: Elaborado pelo autor.

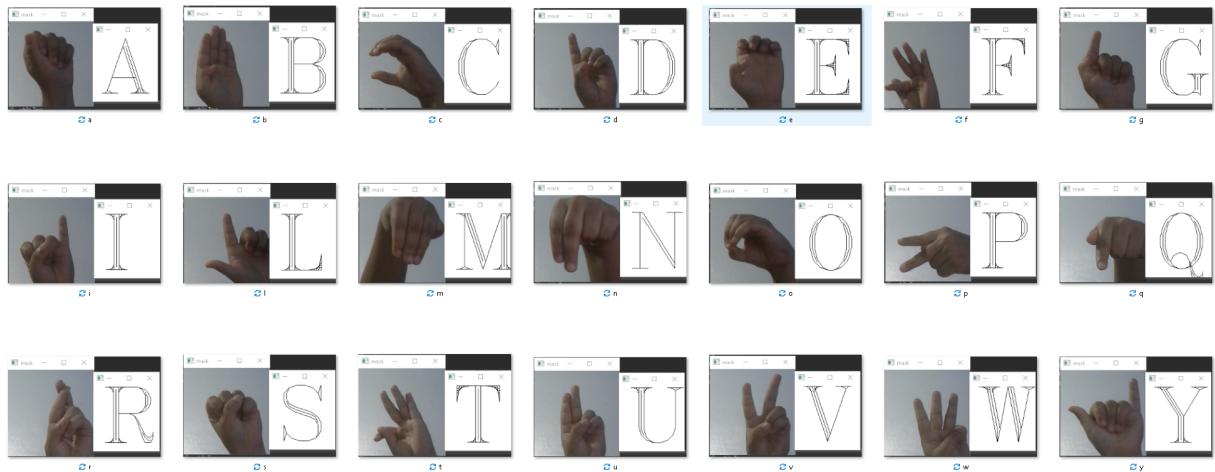
Por fim, considerando que para o experimento I, como podemos verificar na Tabela 5, o valor da acurácia foi de 97.93%, já para o experimento II o valor da acurácia foi de 91.39% e para o experimento III a acurácia foi de 91.19%. Esses valores são indicadores positivos de que o modelo realiza precisões de maneira correta para a maioria dos casos. Em comparação aos resultados obtidos nos trabalhos relacionados na seção 1 podemos ver que os valores dos experimentos são valores que se enquadram dentro da margem dos trabalhos relacionados.

4.2 RESULTADOS QUALITATIVOS

O modelo treinado foi testado em tempo real através da câmera do notebook e a camera Logitech HD 720, podendo assim avaliar diretamente o desempenho do modelo ao realizar capturas em tempo real. Desta maneira é possível analisar como o modelo se comporta,

além da capacidade de realizar previsões precisas, determinando assim se o modelo atende aos requisitos da aplicação. Para o experimento I, podemos notar na Figura 29, que ele realiza a predição de todas as letras, conforme proposto.

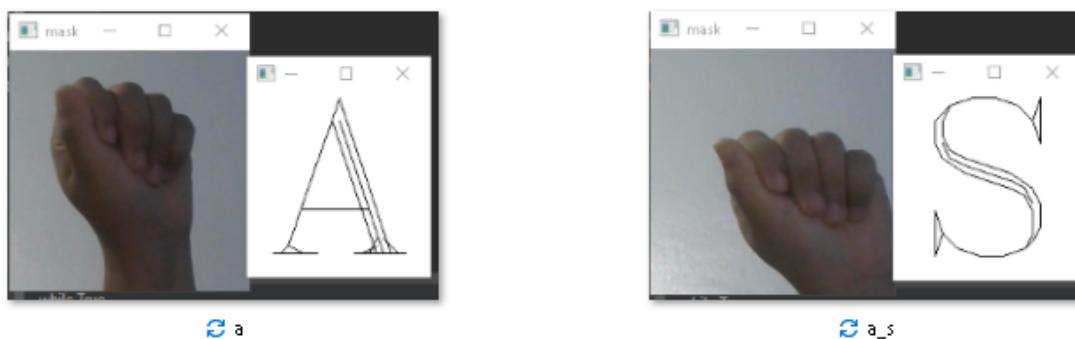
Figura 29 – Captura em tempo real - Experimento I



Fonte: Elaborado pelo autor

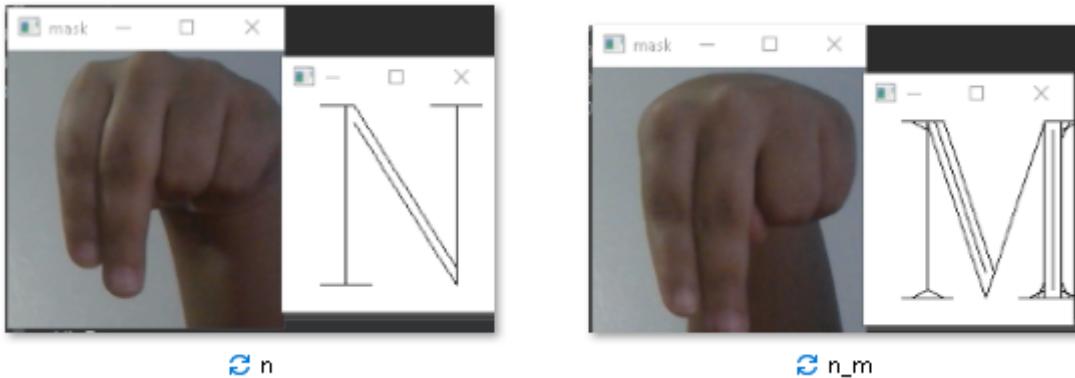
Pode-se verificar que em alguns momentos ocorreram previsões equivocadas, conforme ilustrado na Figura 30, onde o sinal da letra A foi erroneamente identificado como a letra S em alguns momentos. Na Figura 31, observa-se que houve variação entre o sinal da letra N e a letra M. Similarmente, na Figura 32, foi observada uma variação entre o sinal da letra T e a letra F. Esses exemplos indicam que, para sinais semelhantes, há uma pequena variação na previsão entre o sinal correto e o sinal semelhante.

Figura 30 – Variações na captura em tempo real - Experimento I, letra A



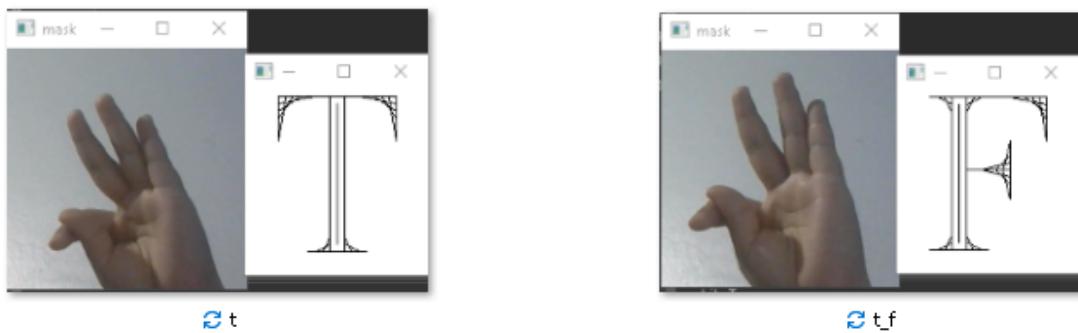
Fonte: Elaborado pelo autor

Figura 31 – Variações na captura em tempo real - Experimento I, letra N



Fonte: Elaborado pelo autor

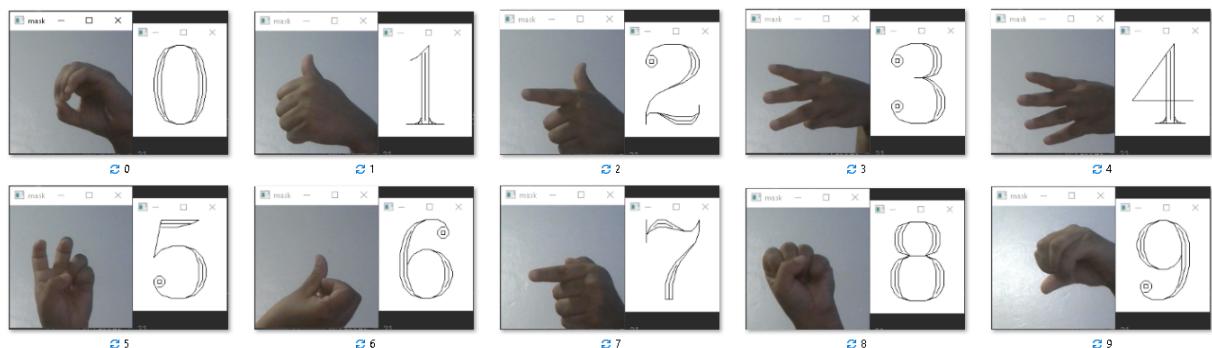
Figura 32 – Variações na captura em tempo real - Experimento I, letra T



Fonte: Elaborado pelo autor

Para o experimento II, podemos notar na Figura 33, que ele realiza a predição de todos os números, conforme proposto neste trabalho.

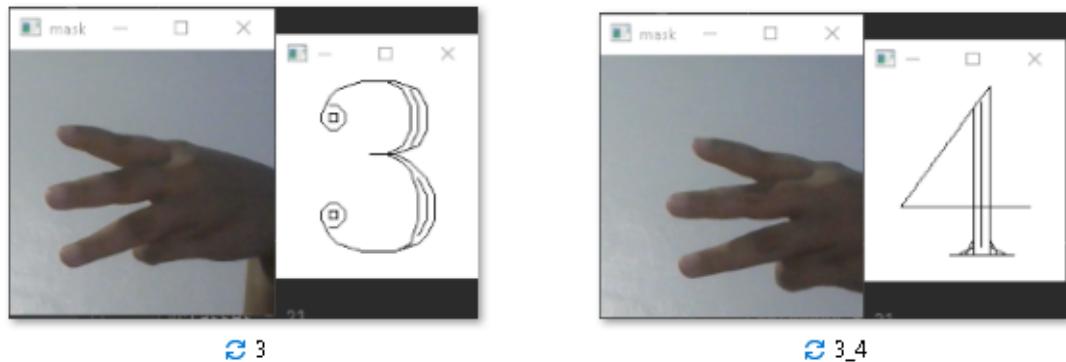
Figura 33 – Captura em tempo real - Experimento II



Fonte: Elaborado pelo autor

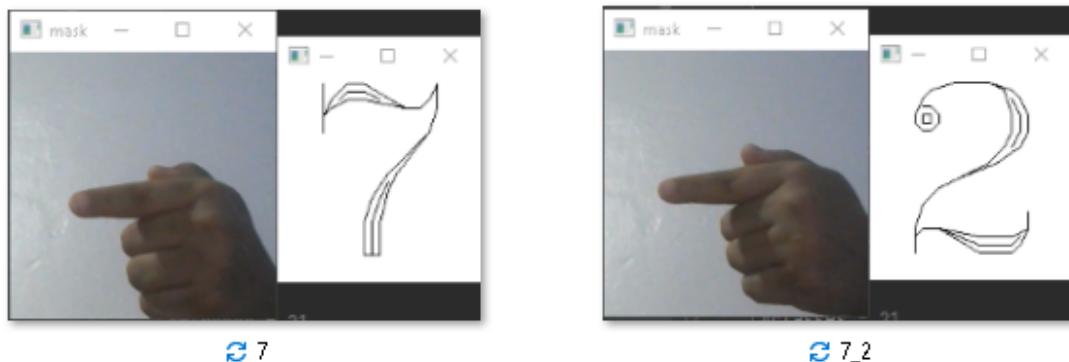
Podemos observar que ocorreram algumas variações no experimento II, conforme ilustrado na Figura 34, onde o sinal do número 3 às vezes foi erroneamente identificado como o número 4. Da mesma forma, na Figura 35, foi observada uma variação entre o número 7 e o número 2. É possível notar que esses sinais são semelhantes.

Figura 34 – Variações na captura em tempo real - Experimento II, Número 3



Fonte: Elaborado pelo autor

Figura 35 – Variações na captura em tempo real - Experimento II, Número 7



Fonte: Elaborado pelo autor

Para o experimento III, podemos notar na Figura 36, que ele realiza a predição de todas as palavras, conforme proposto neste trabalho. Ocorrendo apenas algumas variações entre algumas palavras, conforme podemos ver na Figura 37 e Figura 38.

Figura 36 – Captura em tempo real - Experimento III



Fonte: Elaborado pelo autor

Figura 37 – Variações na captura em tempo real - Experimento III, Palavra 'juntos'



Fonte: Elaborado pelo autor

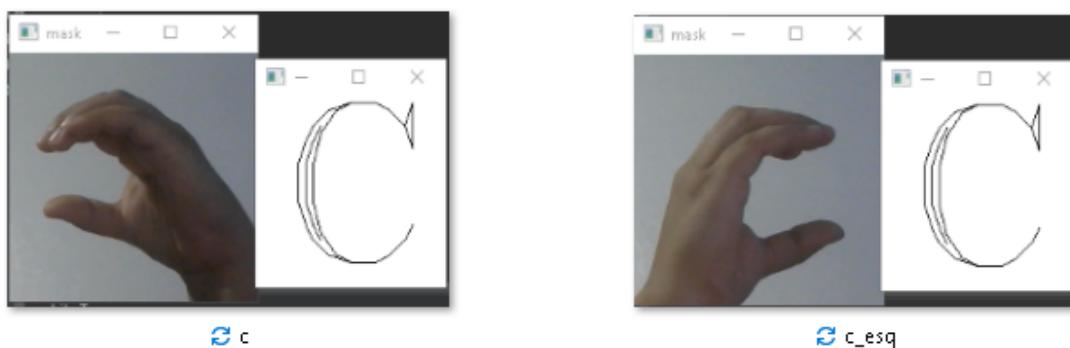
Figura 38 – Variações na captura em tempo real - Experimento III, Palavra 'pedra'



Fonte: Elaborado pelo autor

Com base nas análises realizadas, podemos concluir que, embora o modelo apresente algumas detecções equivocadas para classes semelhantes em alguns momentos, as experiências realizadas atendem à proposta do trabalho, que é a identificação dos sinais de LIBRAS por meio da captura em tempo real. Além disso, o modelo é capaz de identificar o sinal realizado tanto com a mão direita quanto com a mão esquerda, como ilustrado na Figura 39.

Figura 39 – Captura em tempo real - Identificação da mão direita e esquerda



Fonte: Elaborado pelo autor

Observou-se também durante a avaliação que a captura em tempo real requer ambientes mais iluminados para obter detecções mais precisas. Além disso, a utilização da caixa delimitadora é essencial para definir a área exata onde o sinal deve ser realizado, a fim de evitar interferências do fundo que possam impactar na detecção correta do sinal. Portanto, para obter resultados de predição mais precisos, é recomendável utilizar o sistema em ambientes com boa iluminação e fundo de cor branca. Além disso, é importante contar com uma câmera de boa qualidade para captar adequadamente os sinais realizados.

5 CONCLUSÃO

A motivação por esse estudo é promover a aprendizagem de Libras para melhorar a comunicação entre pessoas surdas e ouvintes, buscando a inclusão, o acesso à informação, o desenvolvimento pessoal e profissional, e o enriquecimento cultural. Esses benefícios se estendem não apenas às pessoas surdas, mas também à sociedade como um todo, criando um ambiente mais inclusivo e igualitário para todos.

Portanto, este trabalho teve como objetivo apresentar um sistema que classifica sinais estáticos de LIBRAS em tempo real através da câmera do computador. Para isso, foram treinados 3 conjuntos distintos de dados com gestos estáticos de LIBRAS, utilizando uma CNN2D. Como visto na seção 4.1, apresentou-se um ótimo desempenho, para o experimento I foi alcançado 97.92% de acurácia, para o experimento II, foi alcançado 91.39% de acurácia e para o experimento III, foi alcançado 91.19% de acurácia, realizando corretamente a classificação dos sinais em tempo real, proporcionando uma ferramenta eficiente e acessível para a aprendizagem dos sinais de LIBRAS.

Assim, a utilização dessa abordagem tecnológica demonstra um enorme potencial para auxiliar na inclusão e acessibilidade da comunidade surda, auxiliando na quebra de barreiras comunicativas, podendo ser utilizado na educação infantil proporcionando o aprendizado básico de crianças surdas e ouvintes. Podendo também, auxiliar no aprendizado de pessoas ouvintes que estão aprendendo LIBRAS, impulsionando e estimulando várias formas de aprendizagem em LIBRAS, de forma interativa, criando assim uma experiência de aprendizagem enriquecedora e facilitando a comunicação entre pessoas surdas e ouvintes.

Para facilitar o acesso e o compartilhamento deste trabalho, foi disponibilizado no repositório do GitHub, onde interessados poderão obter mais informações, acessar o código-fonte e colaborar com o desenvolvimento contínuo dessa pesquisa, Araujo (2023).

5.1 TRABALHOS FUTUROS

Embora os resultados tenham sido satisfatórios se comparados com os demais trabalhos relacionados, para atender de forma completa um sistema que traduza os sinais de LIBRAS para o português, ainda faltam muitas variantes, podemos dizer que os gestos estáticos é apenas uma parte inicial para conseguir desenvolver uma ferramenta completa, com a inclusão de gestos dinâmicos, expressões faciais e gestos que utilizam demais partes do corpo. Visando solucionar essas limitações encontradas no decorrer do trabalho e dado o interesse em propor que o trabalho seja utilizado em sistemas que auxiliem na inclusão dos surdos em meio à sociedade, para sugestões de trabalhos futuros:

- Encontrar mecanismos de identificação de expressões faciais e movimentos corporais;
- Inclusão de gestos dinâmicos, uma das possíveis soluções é a utilização da CNN3D+LSTM;

- Desenvolvimento e disponibilização de mais conjuntos de dados, visando contribuir com o avanço de estudos mais específicos no Brasil, considerando que a disponibilidade de um conjunto com essas especificidades buscadas, dificilmente é encontrado.

Referências

- ABERTO, O. V. computacional de código. **Introdução - OpenCV**. 2022. Disponível em: <<https://docs.opencv.org/4.x/d1/dfb/intro.html>>. Acesso em: 21 de jan. de 2023. Citado na página 20.
- ALVARENGA, M. L. T.; CORREA, D. S. O.; OSÓRIO, F. S. Redes neurais artificiais aplicadas no reconhecimento de gestos usando o kinect. **Anais do Computer on the Beach**, p. 347–356, 2012. Citado na página 2.
- ALVES, G. **Entendendo Redes Convolucionais (CNNs)**. 2018. Disponível em: <<https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>>. Acesso em: 20 de mai. de 2023. Citado na página 19.
- ARAUJO, A. d. O. **Projeto_Libras**. 2023. Disponível em: <https://github.com/Andreia-oliv/Projeto_Libras>. Citado na página 45.
- BASTOS, I. L. O. Reconhecimento de sinais da libras utilizando descritores de forma e redes neurais artificiais. Instituto de Matemática. Departamento de Ciência da Computação, 2016. Citado 5 vezes nas páginas 4, 20, 21, 23 e 24.
- BEDUIN, I. R. O. Detecção da covid-19 em imagens de raio-x: construindo um novo modelo de aprendizado profundo utilizando automl. 2021. Citado na página 11.
- BERTONI, A. L.; FEDER, D. V. d. S. **Rede neural convolucional aplicada à visão computacional para detecção de incêndio**. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2018. Citado na página 20.
- CAIAFA, E. G. et al. Aprendizado profundo no reconhecimento de sinais estáticos de libras. In: **Proc. 38th Simpósio Brasileiro de Telecomunicações e Processamento de Sinais**. [S.I.: s.n.], 2020. p. 1–5. Citado 2 vezes nas páginas 4 e 5.
- CUNHA, J. P. Z. **Um estudo comparativo das técnicas de validação cruzada aplicadas a modelos mistos**. Tese (Doutorado) — Universidade de São Paulo, 2019. Citado na página 9.
- FACELI, K. et al. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. [S.I.]: LTC, 2011. Citado 5 vezes nas páginas 4, 7, 8, 10 e 11.
- FACURE, M. **Funções de Ativação Entendendo a importância da ativação correta nas redes neurais**. 2017. Disponível em: <<https://matheusfacure.github.io/2017/07/12/activ-func/>>. Acesso em: 20 de mai. de 2023. Citado 2 vezes nas páginas 14 e 15.
- FREIRE, P. Introdução aos estudos sobre libras. 1999. Citado na página 2.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Aprendizado Profundo**. [S.I.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>. Citado 10 vezes nas páginas 7, 8, 9, 10, 11, 12, 13, 14, 15 e 16.
- IBGE, I. B. de Geografia e E. **Censo demográfico 2010 - População residente por tipo de deficiência, segundo a situação do domicílio, o sexo e os grupos de idade - Amostra - Características Gerais da População**. 2010. Disponível em: <<https://sidra.ibge.gov.br/tabela/3425#resultado>>. Acesso em: 10 de jan. de 2023. Citado na página 1.

- JÚNIOR, R. F. P. Reconhecimento de gestos estáticos utilizando redes neurais convolucionais. 2019. Citado na página 26.
- Keras Team. **Keras - API documentation: Convolution layers**. 2023. Disponível em: <<https://keras.io/api/layers/>>. Citado na página 16.
- KIRANYAZ, S. et al. 1d convolutional neural networks and applications: A survey. **Mechanical Systems and Signal Processing**, v. 151, p. 107398, 2021. ISSN 0888-3270. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0888327020307846>>. Citado na página 16.
- LACERDA, L. **CNN-Libras: Implementação de redes neurais convolucionais para reconhecimento de sinais em LIBRAS**. 2019. Disponível em: <<https://github.com/lucaaslb/cnn-libras>>. Citado 2 vezes nas páginas 22 e 23.
- LACERDA, L. **Deep Learning Visão Computacional — REDES NEURAIS CONVOLUCIONAIS**. 2019. Disponível em: <<https://medium.com/@lucaaslb/deep-learning-vis%C3%A3o-computacional-redes-neurais-convolucionais-c21f19f5ec34>>. Acesso em: 10 de mai. de 2023. Citado 4 vezes nas páginas 17, 18, 23 e 24.
- LEI N° 10.436, DE 24 DE ABRIL DE 2002. 2002. Disponível em: <https://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>. Acesso em: 10 de jan. de 2023. Citado 2 vezes nas páginas 1 e 4.
- MILANO, D. de; HONORATO, L. B. Visão computacional. **Faculdade de Tecnologia, Universidade Estadual de Campinas**, 2010. Citado na página 2.
- MONTEIRO, C. H. d. A. et al. Um sistema de baixo custo para reconhecimento de gestos em libras utilizando visão computacional. In: **Proc. 34th Simpósio Brasileiro de Telecomunicações e Processamento de Sinais**. [S.l.: s.n.], 2016. p. 349–352. Citado na página 2.
- NETO, V. C. L. Reconhecimento de sinais do alfabeto da libras usando visão computacional. Universidade Estadual Paulista, 2019. Citado 9 vezes nas páginas 1, 2, 8, 12, 17, 18, 20, 21 e 23.
- PASSOS, B. T. Reconhecimento de gestos do alfabeto da língua brasileira de sinais utilizando técnicas de visão computacional. Universidade do Vale do Itajai, 2019. Citado 3 vezes nas páginas 5, 6 e 7.
- QUEIFER, H. d. S. Reconhecimento de gestos para acionamento de objetos usando CNN e o algoritmo Haar cascade. Universidade Estadual da Paraíba, 2019. Citado 3 vezes nas páginas 17, 18 e 19.
- RAMOS, E. O. M. O papel da libras no aprendizado da língua portuguesa pelo aluno surdo não oralizado. 2011. Citado na página 5.
- RIBEIRO, M. da M.; QUIMARÃES, S. S. Redes neurais utilizando tensorflow e keras. **RE3C-Revista Eletrônica Científica de Ciência da Computação**, v. 13, n. 1, 2018. Citado 2 vezes nas páginas 19 e 20.
- RIBEIRO, R. d. O. C.; FESTA, P. S. V. Aspectos da comunicação do sujeito surdo e a sua inclusão na sociedade. **Memorial TCC Caderno da Graduação**, v. 3, n. 1, p. 529–539, 2017. Citado 2 vezes nas páginas 1 e 4.

- RODRIGUES, D. A. Deep learning e redes neurais convolucionais: reconhecimento automático de caracteres em placas de licenciamento automotivo. Universidade Federal da Paraíba, 2018. Citado 2 vezes nas páginas 13 e 14.
- ROSSUM, G. V.; JR, F. L. D. **Python tutorial.** [S.I.]: Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995. v. 620. Citado na página 20.
- SANTOS, A. V. et al. Rede neural artificial convolucional aplicada ao reconhecimento de configuração de mão nos símbolos de 0 a 9 da língua brasileira de sinais (libras). In: SBC. **Anais Estendidos do XV Simpósio Brasileiro de Sistemas de Informação.** [S.I.], 2019. p. 21–24. Citado 3 vezes nas páginas 20, 21 e 23.
- SAVIETTO, J. V. **Machine Learning: Metricas, validação cruzada, Bias e Variância.** 2021. Disponível em: <<https://medium.com/@jvsavietto6/machine-learning-métricas-validação-cruzada-bias-e-variação-380513d97c95>>. Citado na página 11.
- SILVA, B. C. R. et al. Desenvolvimento de tecnologia baseada em redes neurais artificiais para reconhecimento de gestos da língua de sinais. Universidade Federal de Goiás, 2018. Citado na página 2.
- STEFANO, G. dos S. et al. Um sistema de reconhecimento de sinais em libras usando cnn e lstm. 2021. Citado na página 2.
- UTSCH, K. G. Uso de redes neurais convolucionais para classificação de imagens digitais de lesões de pele. **Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica)-Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo. Espírito Santo,** 2018. Citado 2 vezes nas páginas 18 e 19.
- VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. **Proceedings of the xxix conference on graphics, patterns and images.** [S.I.], 2016. v. 1, n. 4. Citado na página 17.
- VERMA, S. **Understanding 1D and 3D Convolution Neural Network (Keras).** 2019. Disponível em: <<https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610>>. Citado na página 16.

Anexos

ANEXO A – Código em Python

Listing A.1 – Código em Python para a captura das imagens

```

import cv2
import time
import numpy as np
import os

image_x , image_y = 64 , 64
ESC = 27
CAPTURE = 32
dir_img_training = './treinamento_palavra/'
dir_img_test = './teste_palavra/'
QTD_TRAIN = 150
QTD_TEST = 50

def create_folder(folder_name):
    if not os.path.exists(dir_img_training + folder_name):
        os.mkdir(dir_img_training + folder_name)
    if not os.path.exists(dir_img_test + folder_name):
        os.mkdir(dir_img_test + folder_name)

def capture_images(letra , nome):
    create_folder(str(letra))
    cam = cv2.VideoCapture(0)

    img_counter = 0
    t_counter = 1
    training_set_image_name = 1
    test_set_image_name = 1
    folder = ''

    while True:
        ret , frame = cam.read()
        frame = cv2.flip(frame , 1)

```

```
img = cv2.rectangle(frame, (425, 100), (625, 300),
(0, 255, 0), thickness=2, lineType=8, shift=0)
result = img[102:298, 427:623]
cv2.putText(frame, folder +":_"
+str(img_counter), (30, 400),
cv2.FONT_HERSHEY_TRIPLEX, 1.5, (127, 127, 255))
cv2.imshow("frame", frame)
cv2.imshow("result", result)
if cv2.waitKey(1) == CAPTURE:

    if t_counter <= QTD_TRAIN:
        img_name = dir_img_training + str(letra) +
        "/" + nome + "{}.png".format(training_set_image_name)
        save_img = cv2.resize(result, (image_x, image_y))
        cv2.imwrite(img_name, save_img)
        print("{} written!".format(img_name))
        training_set_image_name += 1
        img_counter = training_set_image_name
        folder = "TRAIN"

    if t_counter > QTD_TRAIN and t_counter <= (QTD_TRAIN+
QTD_TEST):
        img_name = dir_img_test + str(letra) + "/" +
        nome + "{}.png".format(test_set_image_name)
        save_img = cv2.resize(result, (image_x, image_y))
        cv2.imwrite(img_name, save_img)
        print("{} written!".format(img_name))
        test_set_image_name += 1
        img_counter = test_set_image_name
        folder = "TEST"

    t_counter += 1

    if t_counter > (QTD_TRAIN+QTD_TEST):
        print('[INFO] FIM')
        break

if cv2.waitKey(1) == ESC:
    break
```

```
cam.release()
cv2.destroyAllWindows()

letra = input("LETRA:")
nome = input("NOME:")
capture_images(letra, nome)
```

ANEXO B – Código em Python

Listing B.1 – Código em Python, implementação da arquitetura da CNN2D

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import LeakyReLU

class Convolucao(object):
    @staticmethod
    def build(width, height, channels, classes):

        inputShape = (height, width, channels)

        model = Sequential()

        model.add(Conv2D(filters = 32, kernel_size = (3,3),
                         padding = 'same', input_shape = inputShape))
        model.add(LeakyReLU(alpha=0.1))
        model.add(MaxPooling2D((2,2)))

        model.add(Conv2D(filters = 32, kernel_size = (3,3)))
        model.add(LeakyReLU(alpha=0.1))
        model.add(MaxPooling2D((2,2)))

        model.add(Conv2D(filters = 64, kernel_size = (3,3)))
        model.add(LeakyReLU(alpha=0.1))
        model.add(MaxPooling2D((2,2)))

        model.add(Flatten())
        model.add(Dense(256, activation = 'relu'))
        model.add(Dropout(0.5))
        model.add(Dense(classes, activation = 'softmax'))

    return model
```

ANEXO C – Código em Python

Listing C.1 – Código em Python, implementação do treinamento da CNN2D

```
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.25)

test_datagen = ImageDataGenerator(rescale=1./255,
                                   validation_split=0.05)

# produz base de dados de treino gerada pelo modelo #
training_set = train_datagen.flow_from_directory(
    './dataset/treinamento_palavra',
    target_size=(64, 64),
    color_mode = 'rgb',
    batch_size=32,
    shuffle=False,
    class_mode='categorical')

#Pega o caminho para o diretorio e gera lotes de dados aumentados.
test_set = test_datagen.flow_from_directory(
    './dataset/teste_palavra',
    target_size=(64, 64),
    color_mode = 'rgb',
    batch_size=32,
    shuffle=False,
    class_mode='categorical')

# inicializar e otimizar modelo
print("[INFO] Inicializando e otimizando a CNN...")
start = time.time()

early_stopping_monitor = EarlyStopping(monitor='val_loss',
                                        mode='min', verbose=1, patience=15)

model = Convolucao.build(64, 64, 3, CLASS)

model.compile(optimizer=SGD(0.01), loss="categorical_crossentropy",
              metrics=["accuracy"])
```

```
# treinar a CNN
print("[INFO] - Treinando a CNN...")
classifier = model.fit(
    training_set,
    steps_per_epoch=(training_set.n // training_set.batch_size),
    epochs=EPOCHS,
    validation_data = test_set,
    validation_steps= (test_set.n // test_set.batch_size),
    verbose=1,
    callbacks = [early_stopping_monitor]
)

EPOCHS = len(classifier.history["loss"])

print("[INFO] - Salvando modelo treinado...")

file_date = getDateStr()
model.save('./models/' + FILE_NAME + file_date + '.h5')
print('[INFO] - modelo: ./models/' + FILE_NAME + file_date + '.h5 salvo!')

end = time.time()

print("[INFO] - Tempo de execu    o da CNN: %.1f min"
      %(getTimeMin(start, end)))

print('[INFO] - Summary:')
model.summary()

print("\n[INFO] - Avaliando a CNN...")
score = model.evaluate(test_set,
                       steps=(test_set.n // test_set.batch_size),
                       verbose=1)

print('[INFO] - Accuracy: %.2f%%' % (score[1]*100),
      '| Loss: %.5f' % (score[0]))

print("[INFO] - Sumarizando loss e accuracy para os datasets"
      "'train_net' e 'test_net'")
```

```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0,EPOCHS), classifier.history["loss"],
          label="train_loss")
plt.plot(np.arange(0,EPOCHS), classifier.history["val_loss"],
          label="val_loss")
plt.plot(np.arange(0,EPOCHS), classifier.history["accuracy"],
          label="train_accuracy")
plt.plot(np.arange(0,EPOCHS), classifier.history["val_accuracy"],
          label="val_accuracy")
plt.title("Training_Loss_and_Accuracy")
plt.xlabel(f"Epochs: {EPOCHS}")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig('./models/graphics/' + FILE_NAME + file_date + '.png',
            bbox_inches='tight')

print('[INFO] Gerando imagem do modelo de camadas da CNN')
plot_model(model, to_file='./models/image/' + FILE_NAME + file_date +
            '.png', show_shapes=True)

print('[INFO] Gerando matriz de confusao')

Y_pred = model.predict(test_set)
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = test_set.classes
cm = confusion_matrix(Y_true, Y_pred_classes)

plt.figure(figsize=(10, 8))
#CLASSE=['A', 'B', 'C', 'D', 'E', 'F',
# 'G', 'I', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'Y']
#CLASSE=['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
CLASSE=['adulto', 'america', 'casa', 'gasolina', 'juntos',
        'lei', 'palavra', 'pedra', 'pequeno', 'verbo']

sns.heatmap(cm, annot=True, linewidth=.5, fmt='d')
plt.title('Matriz_de_Confusao')
plt.xlabel('Previsoes')
plt.ylabel('Rotulos_verdadeiros')
```

```
tick_marks = np.arange(len(CLASSE))
plt.xticks(tick_marks + 0.5, CLASSE)
plt.yticks(tick_marks + 0.5, CLASSE)

plt.show()
plt.savefig('./models'+FILE_NAME+file_date+'.png')

print('\n[INFO] [FIM]: ' + getDateStr())
print('\n\n')
```

ANEXO D – Código em Python

Listing D.1 – Código em Python, implementação das janelas na captura em tempo real

```

import cv2
import numpy as np
from keras.models import load_model
import tensorflow as tf

def nothing(x):
    pass

image_x, image_y = 64,64

#informamos o modelo a ser usado
classifier=load_mode(
    'models/cnn_model_LIBRAS_palavras_28_05_2023_20_25.h5')

#classes = 21
classes = 10
''

modelo = { '0' : 'A', '1' : 'B', '2' : 'C' , '3': 'D',
    '4': 'E', '5': 'F', '6': 'G', '7': 'I', '8': 'L', '9': 'M',
    '10': 'N', '11': 'O', '12': 'P', '13': 'Q', '14': 'R',
    '15': 'S', '16': 'T', '17': 'U', '18': 'V', '19': 'W', '20': 'Y'}

modelo = { '0' : '0', '1' : '1', '2' : '2' , '3': '3',
    '4': '4', '5': '5', '6': '6', '7': '7', '8': '8', '9': '9'}
''

modelo = { '0' : 'adulto', '1' : 'america', '2' : 'casa' ,
    '3': 'gasolina', '4': 'juntos', '5': 'lei' ,
    '6': 'palavra', '7': 'pedra', '8': 'pequeno', '9': 'verbo' }

#le a imagem da camera e converte para o
# formato que o keras/tensorflow entendem para executar

```

```
def predictor():
    test_image = tf.keras.utils.load_img('./temp/img.png',
                                         target_size=(64, 64))
    test_image = tf.keras.utils.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis = 0)
    result = classifier.predict(test_image)

    maior, class_index = -1, -1

    #esse for informa a probabilidade de qual
    #a letra a ser visualizada na webcam
    #dado o maior valor, ele determina qual letra e
    for x in range(classes):
        if result[0][x] > maior:
            maior = result[0][x]
            class_index = x

    return [result, modelo[str(class_index)]]


#inicia a webcam
cam = cv2.VideoCapture(0)

img_counter = 0

img_text = [' ', ' ']

#a partir daqui o while cria as caixas e
#personaliza o terminal para mostrar o resultado
while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame,1)
    img = cv2.rectangle(frame, (425,100),(625,300), (0,255,0),
                        thickness=2, lineType=8, shift=0)

    imcrop = img[102:298, 427:623]

    cv2.putText(frame, str(img_text[1]), (30, 400),
               cv2.FONT_HERSHEY_COMPLEX, 3, (0, 255, 0))
```

```
cv2.imshow("test_net", frame)
cv2.imshow("mask", imcrop)

img_name = "./temp/img.png"
save_img = cv2.resize(imcrop, (image_x, image_y))
cv2.imwrite(img_name, save_img)
img_text = predictor()
#print(str(img_text[0]))

print(img_text[0])
#output = np.ones((150, 150, 3)) * 255
output = np.ones((150, 600, 3)) * 255
cv2.putText(output, str(img_text[1]), (15, 130),
            cv2.FONT_HERSHEY_TRIPLEX, 4, (0, 0, 0))
cv2.imshow("PREDICT", output)

if cv2.waitKey(1) == 27:
    break

cam.release()
cv2.destroyAllWindows()
```