UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

VICENTE COELHO LOBO NETO

RECONHECIMENTO DE SINAIS DO ALFABETO DA LIBRAS USANDO VISÃO COMPUTACIONAL

BAURU
Novembro de 2019

VICENTE COELHO LOBO NETO

RECONHECIMENTO DE SINAIS DO ALFABETO DA LIBRAS USANDO VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Assoc. João Paulo Papa
Coorientador: Prof. Assoc. Antonio Carlos Sementille

BAURU
Novembro de 2019

Vicente Coelho Lobo Neto

Reconhecimento de sinais do alfabeto da Libras usando Visão Computacional

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Assoc. Antonio Carlos Sementille
Coorientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profª. Drª. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, 18 de novembro de 2019.

Agradecimentos

Agradeço, primeiramente, à minha mãe, Edna, a meu pai, Silvio e irmã, Nichole, não apenas pelo suporte emocional e financeiro, mas principalmente por todo o amor.

Agradeço também a Daniele Fernandes e Daiane Tadeu pela ajuda na construção do conjunto de dados. A esta última devo também agradecer pelo companheirismo e amizade neste ano. Contrariando expectativas, não nos matamos.

Agradeço também aos meus orientadores e demais professores e funcionários da Unesp. Tanto pelos ensinamentos acadêmicos, como também por vários outros ensinamentos para a vida.

Não é possível deixar de agradecer também aos diversos amigos que encontrei durante essa trajetória. Os caminhos de muitos se separaram do meu nesse meio tempo, mas sempre levarei todos com muito carinho no meu lado esquerdo do peito.

Por último, mas certamente não menos importante, agradeço à Unesp. Universidade Pública na qual entrei por cotas e, depois de uma árdua jornada, agora estou saindo mais velho, mais maduro e com muito mais histórias. A experiência que vivi na Universidade é algo que me modificou de inúmeras formas e tenho muito orgulho disso. Em tempos de desmonte, meu desejo é que a Unesp possa continuar a existir e a ser pública por muito mais anos.

"Em tempos normais, nenhum indivíduo são pode concordar com a ideia de que os homens são iguais."

(Aldous Huxley)

Resumo

Naturalmente pautado pela coletividade, o ser humano é dependente de relações sociais. Nesse sentido, a comunicação é uma das ferramentas mais importantes para a criação de laços com outras pessoas. Apesar disso, ela pode representar um grande desafio para deficientes auditivos, quando desejam comunicar-se com falantes. A Libras, Língua Brasileira de Sinais, é a língua utilizada pelos deficientes auditivos no Brasil. Embora um dos idiomas oficiais do país, um grupo muito restrito de pessoas a conhece, o que dificulta o acesso aos deficientes auditivos a muitas coisas, como serviços públicos. Visando caminhar rumo à inclusão dessas pessoas, este trabalho apresenta um protótipo de tradutor de Libras para Português que contempla 20 letras do alfabeto. O mesmo utiliza ferramentas de Visão Computacional como técnicas de Processamento de Imagem e Aprendizado de Máquina para desenvolver um classificador capaz de identificar qual a letra sinalizada analisando uma imagem capturada por uma câmera RGB. O classificador desenvolvido obteve aproximadamente 88% de acurácia e 0.45 de *loss* na avaliação do conjunto de validação, no qual foram aplicadas técnicas de aumento de dados.

Palavras-chave: Libras, inclusão, deficiência auditiva, processamento de imagens, aprendizado de máquina, redes neurais convolucionais.

Abstract

Naturally guided by the collectivity, the humans are dependent on social relations. Considering this, communication is one of the most important tools for bonding with others. Nevertheless, it can pose a major challenge for the hearing impaired when they wish to communicate with speakers. Libras, the Brazilian Sign Language, is the language used by the hearing impaired in Brazil. Although one of the country's official languages, a very small group of people know it, which makes it difficult for deaf people to access many things, such as public services. Aiming to walk towards the inclusion of these people, this work presents a prototype translator from Libras to Portuguese that includes 20 letters of the alphabet. It uses Computer Vision tools such as Image Processing and Machine Learning techniques to develop a classifier that can identify which letter is signaled by analyzing an image captured by a RGB camera. The developed classifier obtained approximately 88% of accuracy and 0.45 of *loss* in the validation set evaluation, in which data augmentation techniques were applied.

Keywords: Libras, inclusion, hearing impairment, image processing, machine learning, convolutional neural networks.

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

2D	Duas dimensões
3D	Três dimensões
3DCNN	Rede Neural Convolucional 3D
CNN	Rede Neural Convolucional
HSV	*Hue-Saturation-Value*
IBGE	Instituto Brasileiro de Geografia e Estatística
Libras	Língua Brasileira de Sinais
LSTM	Rede de Memória de Longo Prazo
ORB	*Oriented FAST and Rotated BRIEF*
RGB	Red-Green-Blue
ROI	Região de Interesse
SIFT	*Scale Invariant Feature Transform*
SPA	*Single-Page Application*
SSD	*Single-Shot Detector*

Sumário

1 Introdução

O ser humano tende naturalmente a viver em coletividade, existindo a necessidade da criação de vínculos entre as pessoas para que o sujeito possa ser constituído e sentir-se pleno. Um importante meio para isso é através da comunicação que, por sua vez, é feita através das interações sociais usando algum tipo de linguagem.

> "A linguagem é responsável pela regulação da atividade psíquica humana, pois é ela que permeia a estruturação dos processos cognitivos. Assim, é assumida como constitutiva do sujeito, pois possibilita interações fundamentais para a construção do conhecimento" (VYGOTSKY, 2001).

Geralmente, a primeira forma de comunicar-se que vem à mente é através da língua oral. Um bebê, por exemplo, já desde as primeiras horas de vida é exposto à língua ouvindo a mãe e começa a ter sua comunicação estimulada. Quando chega o tempo, começa a arriscar tentar pronunciar algumas palavras. Depois disso, a cada dia essa habilidade vai desenvolvendo-se mais através das relações sociais. A questão é: e quando há algum problema nesse processo? Pessoas que possuem algum tipo de deficiência auditiva são seriamente prejudicadas nisso, dado que o problema na audição - sentido de grande relevância para a comunicação na língua oral - dificulta a interação com os grupos sociais nos quais essas pessoas estão inseridas.

O Censo de 2010, realizado pelo Instituto Brasileiro de Geografia e Estatística - IBGE, mostrou que 9,7 milhões de pessoas apresentam deficiência auditiva. "Desses, 2,1 milhões apresentam deficiência auditiva severa" (BRASIL, 2011). Cabe notar que cerca de um milhão são jovens de até 19 anos. É, desse modo, um contingente muito grande de pessoas que acaba sendo invisibilizado por sua dificuldade de comunicação.

Segundo Lacerda (2006), houve a difusão da defesa de uma política educacional de inclusão das pessoas com necessidades especiais no mundo todo a partir da década de 1990. No Brasil, a Língua Brasileira de Sinais (Libras) foi estabelecida como língua oficial das pessoas surdas em 2002, através da Lei nº 10.436. No Artigo 2º, ela determina que:

> "Deve ser garantido, por parte do poder público em geral e empresas concessionárias de serviços públicos, formas institucionalizadas de apoiar o uso e difusão da Língua Brasileira de Sinais - Libras como meio de comunicação objetiva e de utilização corrente das comunidades surdas do Brasil" (BRASIL, 2002).

Apesar disso, a acessibilidade para os deficientes auditivos ainda é difícil, dado que o número de pessoas que conhecem Libras é pequeno, assim como a quantidade reduzida de intérpretes; considerando a demanda. Aulas, palestras, consultas médicas, atendimento de

qualquer serviço público, pedir informações ou uma simples conversa casual são apenas alguns dos diversos cenários que podem se tornar grandes desafios para deficientes auditivos.

Um agravante é o fato de que a Libras, sendo uma língua, possui suas regras gramaticais próprias, que são diferentes da língua portuguesa. Isso foi mostrado por Lodi; Bortolotti e Cavalmoreti (2014) que apresentaram a fábula da "Galinha dos Ovos de Ouro" para estudantes deficientes auditivos. O texto foi enunciado em Libras e em português e, em seguida, foi solicitado que os alunos reproduzissem o enunciado em português. Parte dos resultados obtidos está demonstrada no Quadro 1.

Quadro 1 – Comparativo de estrutura gramatical entre Libras e Português

Português	Um pobre lenhador vivia com a mulher os filhos em sua roça longe de tudo Plantava milho e feijão e criava meia dúzia de galinhas (...)
Libras	HOMEM POBRE LENHADOR VIVER MULHER FILHOS ROÇA ISOLADO PLANTAR MILHO FEIJÃO CRIAR GALINHAS (...)

Fonte – Lodi; Bortolotti e Cavalmoreti (2014)

Dessa forma, o Português é uma segunda língua para os deficientes auditivos e seu aprendizado é complicado. Por conta disso, muitos não compreendem o Português, dificultando ainda mais a comunicação.

Na busca pela melhora desse cenário, algumas ferramentas foram desenvolvidas. HandTalk[1], Rybená[2] e VLibras[3] são sistemas que oferecem tradução de Português para Libras. Dependendo da opção, é possível digitar um texto ou até mesmo selecionar de uma página da internet e, em seguida, um intérprete virtual sinalizará o texto. Apesar de representarem um importante avanço, as soluções mencionadas contemplam apenas uma via na comunicação, que é da tradução de Português para Libras. Ou seja, embora permitam a transmissão de uma mensagem, não é possível receber uma resposta em Libras e traduzir para o Português.

Vê-se que, mesmo com as soluções apresentadas, não há a possibilidade de se ter um diálogo, dado que pessoas que não conhecem a Libras não conseguirão compreender alguma possível resposta que o deficiente auditivo pode vir a sinalizar. Isso representa uma falha séria no processo comunicativo. Assim, é preciso pensar em estratégias para combater isso.

Um meio é a utilização de sistemas de Visão Computacional. Segundo Huang (1996, tradução nossa), "a visão computacional visa construir sistemas autônomos que possam executar algumas das tarefas que o sistema visual humano pode executar (e até superá-lo em muitos casos)". Ela extrai informações do mundo real para atividades como o reconhecimento de pessoas, objetos, movimento ou mesmo reconhecimento de face e pose corporal.

[1] <https://www.handtalk.me/>
[2] <http://portal.rybena.com.br/site-rybena/>
[3] <http://www.vlibras.gov.br/>

Assim, uma forma de desenvolver um tradutor de Libras para Português poderia ser através da utilização de uma câmera simples para a captura dos sinais e fazer o reconhecimento através de visão computacional aliada a redes neurais profundas. Entretanto, essa tarefa apresenta certa complexidade, dado que possui diversos pontos problemáticos. Alguns exemplos são:

a) **Conjunto de Dados**: A Libras possui milhares de sinais e, apesar de já existir há alguns anos, não foi encontrado nenhum conjunto de dados organizado contendo todos os sinais. Dessa forma, é necessário estruturar e construir um conjunto de dados para o treinamento das redes neurais artificiais. Além disso, este conjunto demanda grande quantidade de dados;

b) **Diferentes tons de pele e iluminações**: Mesmo considerando a existência de um bom conjunto de dados, existem tons de pele muito variados. Ademais, as diversas possibilidades de iluminação também aumentam a dificuldade;

c) **Posição das mãos e interação com o corpo**: Além da posição das mãos, que já não é uma tarefa trivial, é necessário reconhecer também cada uma das partes do corpo para identificar com quais delas as mãos estão interagindo.

Dadas todas essas dificuldades que permeiam a criação de um tradutor de Libras para Português, foi avaliado que seria inviável o desenvolvimento desse sistema completo no período disponível para o Trabalho de Conclusão de Curso. Assim, foi pensado em um protótipo utilizando os sinais do alfabeto da Libras, mostrados na Figura 1.

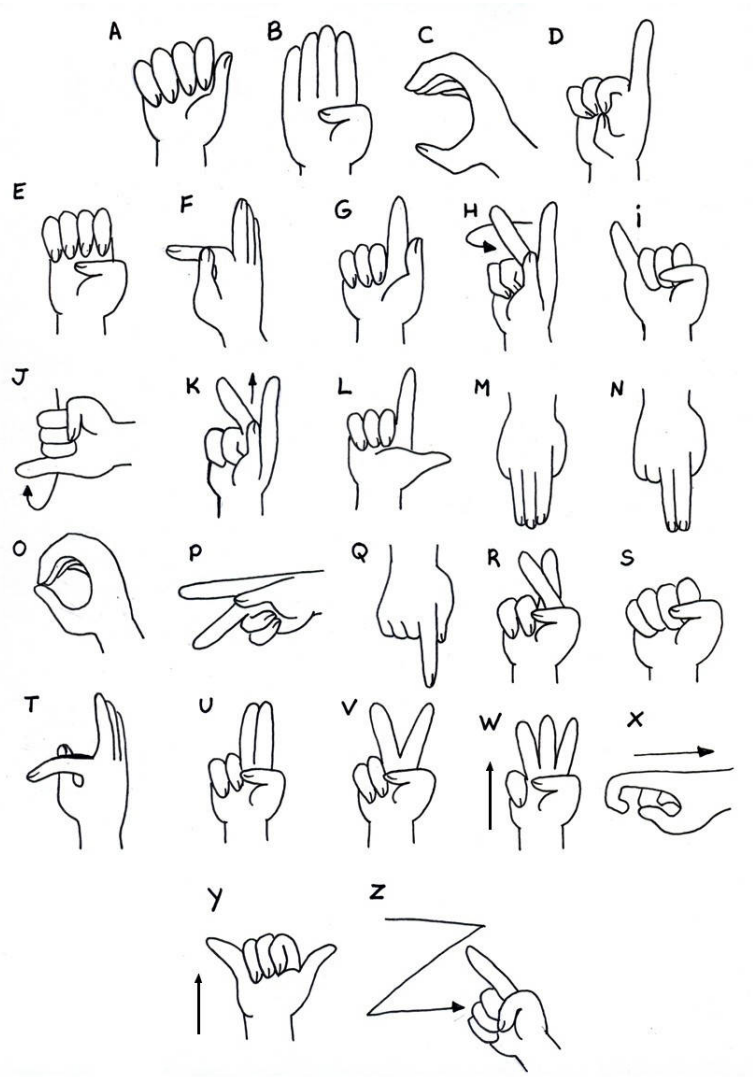
Para isso, foram tomadas imagens RGB como entrada; realizado pré-processamento para segmentação de pele através do espaço de cor HSV; remoção de plano de fundo e detecção de contorno através do algoritmo de *Canny* para posterior classificação usando uma Rede Neural Convolucional (CNN). Não foram utilizadas características extraídas manualmente, pois, embora testadas, não apresentaram adições significativas nos resultados.

1.1 Trabalhos Correlatos

Em Shahriar et al. (2018), os autores focaram no desenvolvimento de uma boa forma de extrair a pele em uma imagem, o que foi feito utilizando o espaço de cor YCbCr. Além disso, buscava-se separar apenas quatro classes, as letras A e V, palma da mão e rosto. A acurácia obtida foi de 94%. A classificação foi feita utilizando *Transfer Learning* através da rede *AlexNet* já pré-treinada.

Outro trabalho que utiliza redes pré-treinadas é o de Garcia e Viesca (2016). A rede utilizada foi a *GoogLeNet* treinada usando o conjunto de dados 2012 ILSVRC. Neste trabalho, foram utilizadas todas as letras paradas do alfabeto para a classificação, com o diferencial de que o resultado obtido pelo classificador passa por uma rede de processamento de texto. A

Figura 1 – Alfabeto dos sinais da Língua Brasileira de Sinais



Fonte – Biblioteca Virtual (2019)

acurácia considerando apenas a classificação de imagem foi, após vários testes, entre 65% e 72%. Passando pela rede de processamento de texto, os resultados obtidos melhoraram para aproximadamente 90% de acurácia.

Em Taskiran; Killioglu; Kahraman (2018) os autores treinaram a própria CNN. Ela classifica não apenas as letras do alfabeto, como também os números, utilizando outras técnicas de pré-processamento das imagens, como encontrar o formato da mão usando *Convex Hull*. Foi obtida acurácia de aproximadamente 98%, mas o artigo não detalha como foi construído o conjunto de validação.

Por fim, foi utilizado como base para o desenvolvimento deste trabalho o artigo de Prateek et al. (2018). Nele também é feito o treinamento da própria CNN, mas o pré-processamento das imagens inclui a detecção de borda usando o algoritmo de *Canny* e adição de características extraídas manualmente através do algoritmo *Scale-Invariant Feature Transform*

(SIFT). A acurácia obtida também foi de aproximadamente 98%, mas, assim como no trabalho anterior, poucas informações são fornecidas sobre o conjunto de validação.

O restante do trabalho está organizado como segue. Primeiramente, no Capítulo 2 são introduzidos alguns conceitos fundamentais que norteiam este trabalho. Em seguida, o Capítulo 3 detalha como foi feita a captura e tratamento das imagens utilizadas no conjunto de dados. Já o Capítulo 4 aprofunda-se no processo do desenvolvimento do classificador construído e as ferramentas utilizadas. O Capítulo 5 demonstra quais os passos necessários para utilizar o classificador. O Capítulo 6 apresenta a interface desenvolvida para a utilização do sistema. Por fim, o Capítulo 7 apresenta as conclusões do autor sobre o trabalho desenvolvido.

2 Fundamentação Teórica

Pode-se considerar como foco deste trabalho a resolução de um problema de reconhecimento de padrões que, para tal, utiliza aprendizado de máquina com redes neurais artificiais aliadas a técnicas de processamento de imagem.

Antes de prosseguir, cabe definir brevemente alguns conceitos.

2.1 Reconhecimento de Padrões

A observação de padrões sempre acompanhou os seres humanos durante seu desenvolvimento. Através da observação dos padrões de cheia do Rio Nilo, os antigos egípcios conseguiram desenvolver sua agricultura e sobreviver nas regiões desérticas do nordeste da África. Do mesmo modo, foi analisando os padrões de prole do cruzamento de diferentes tipos de ervilhas que o austríaco Gregor Mendel criou as leis fundamentais da genética.

Na Computação, o reconhecimento de padrões é baseado na análise de dados. Estes, por sua vez, podem ser puramente numéricos; sinais codificados; imagens ou até mesmo textos.

> "O campo do reconhecimento de padrões se preocupa com a descoberta automática de regularidades nos dados por meio do uso de algoritmos de computador e com o uso dessas regularidades para executar ações como classificar os dados em diferentes categorias" (BISHOP, 2006, tradução nossa).

As aplicações para o reconhecimento de padrões são bastante diversas, fazendo com que seja utilizado nas mais diversas áreas do conhecimento. Alguns exemplos que podem ser mencionados são a detecção de *e-mails* indesejados; análise de sinais em exames médicos para identificar doenças ou a busca de padrões de objetos em imagens, como será feito neste trabalho.

2.2 Aprendizado de Máquina

Uma das formas de realizar o reconhecimento de padrões, é fazendo uso do aprendizado de máquina. Ele consiste na utilização de métodos estatísticos através de algoritmos para que um sistema computacional seja possível de prever dados e/ou tomar decisões sem que seja necessário explicitamente programá-lo para isso (KOZA et al., 1996).

2.2.1 Tipos de Aprendizado de Máquina

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

- **Supervisionado**: É o método utilizado neste trabalho. Cada instância dos dados de entrada fornecidos ao computador conterá uma ou mais etiquetas associadas. Estas etiquetas devem ser definidas por especialistas. Dessa forma, o objetivo do sistema a ser desenvolvido é, ao receber uma instância de dados diferente das que foram fornecidas como entrada, ser capaz de definir qual a etiqueta que deve ser associada a este novo dado;
- **Não Supervisionado**: Os dados da entrada não possuirão nenhum tipo de etiqueta associada. Assim, cabe ao sistema identificar as relações entre os dados, geralmente agrupando os que possuam características semelhantes;
- **Semi-Supervisionado**: Pode ser visto como um híbrido das duas formas mencionadas anteriormente. Geralmente apenas uma pequena quantidade dos dados de entrada possuirá alguma etiqueta, como no aprendizado supervisionado, enquanto a maioria não possuirá nenhuma informação associada, como no aprendizado não supervisionado;
- **Aprendizado por Reforço**: Este método diferencia-se dos outros pelo fato de que nele 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.

2.2.2 Conjuntos de Dados

Independentemente do método escolhido, é necessário, primeiramente, possuir dados de entrada. Estes dados geralmente são divididos em dois conjuntos: treino e teste.

Os dados do conjunto de treino serão utilizados para a criação de um sistema computacional que consiga, de maneira aproximada, descrever de alguma forma esse conjunto. Este processo é chamado de treinamento. Em seguida, a qualidade deste sistema é avaliada utilizando o conjunto de testes. Em alguns casos, pode-se existir um terceiro conjunto, o de validação. Ele é utilizado como auxiliar no processo de treinamento.

2.2.3 Categorização das Tarefas

As tarefas associadas ao Aprendizado de Máquina podem ser divididas em diversas categorias. Dentre elas, cabe citar a regressão e a classificação.

A regressão visa o desenvolvimento de uma equação capaz de descrever a variação de uma variável dependente através da variação de uma ou mais variáveis independentes

(PETERNELLI, 2003). Desse modo, é esperado que um sistema de regressão apresente como saída valores numéricos. O sistema desenvolvido para realizar regressões é chamado de regressor.

Já na classificação, o objetivo do sistema é categorizar os dados em duas ou mais classes diferentes, dependendo de como foi construído o conjunto de treino. Assim, ao receber um novo dado, é esperado que o sistema consiga dizer a qual classe deste dado, mesmo que ele não tenha sido visto durante o treinamento. O sistema desenvolvido para realizar essas classificações é chamado de classificador. É a categoria do problema deste trabalho.

É importante mencionar que essas categorias são geralmente associadas ao aprendizado supervisionado. Existem outras categorias que se encaixam melhor a outros tipos de aprendizado de máquina, como *clustering* e redução dimensional.

2.2.4 Medidas de Avaliação

O aprendizado de máquina não visa encontrar soluções exatas para os problemas que se propõe a resolver. Seu objetivo é encontrar aproximações que apresentem resultados satisfatórios baseados em algum tipo de medida.

A medida de avaliação que foi utilizada neste trabalho é a acurácia. Ela calcula qual a distância entre o valor predito pelo classificador e o valor real presente nos conjuntos de treino e teste. A acurácia pode ser vista como sendo a medida da exatidão dos resultados fornecidos pelo classificador.

Outras formas de avaliação que podem ser citadas são a precisão, revocação, ou ainda medidas derivadas, como a F_1.

2.3 Redes Neurais Artificiais

Dentre as diversas maneiras de realizar aprendizado de máquina, uma que se destaca é a que utiliza redes neurais artificiais.

Inspiradas no funcionamento do cérebro, essas redes são compostas por neurônios. Estes, por sua vez, representam modelos matemáticos que aplicam algum tipo de função sobre os dados que recebem em sua entrada e fornecem uma saída para os próximos neurônios.

Essas redes geralmente possuem três tipos de neurônios: entrada, saída e neurônios internos. Os dois primeiros, como seus nomes sugerem, são os responsáveis, respectivamente, pela entrada e saída de dados na rede. Já os neurônios internos, também conhecidos como ocultos, são os responsáveis por realizar o processamento dos dados de entrada para extração de características que irão para a saída.

2.3.1 Redes Diretas (*Feedforward*)

Este tipo de rede neural artificial é o mais simples. Essas redes apresentam diversos neurônios densamente conectados entre si, formando um grafo. Este grafo não pode conter ciclos, ou seja, não há realimentação entre os neurônios internos. Um exemplo deste tipo de rede são as redes *Perceptron* e suas variações.

2.4 Aprendizado Profundo

O aprendizado profundo é um ramo do aprendizado de máquina que utiliza modelos hierárquicos para criar as abstrações contidas nos sistemas computacionais desenvolvidos.

Pode-se entender o aprendizado profundo como o empilhamento de diversas camadas mais simples para formar um sistema final mais robusto. Isso geralmente leva a uma melhor representação dos dados. Por conta disso, nos últimos anos, técnicas de aprendizado profundo têm recebido grande destaque, principalmente devido a seus excelentes resultados em diversas aplicações (LECUN; Y.BENGIO; HINTON, 2015).

Como efeito colateral, elas normalmente exigem uma grande quantidade de dados para fins de treinamento, uma vez que são modelos bastante complexos. Por conta disso, sob a falta de dados, estas técnicas podem sofrer com sobre-ajuste (*overfitting*), o que pode causar uma convergência prematura. Sobre-ajuste é quando o sistema aprende somente como representar os dados do conjunto de treino, levando, assim, a uma pobre generalização de dados não vistos durante o treinamento.

2.4.1 Redes Convolucionais

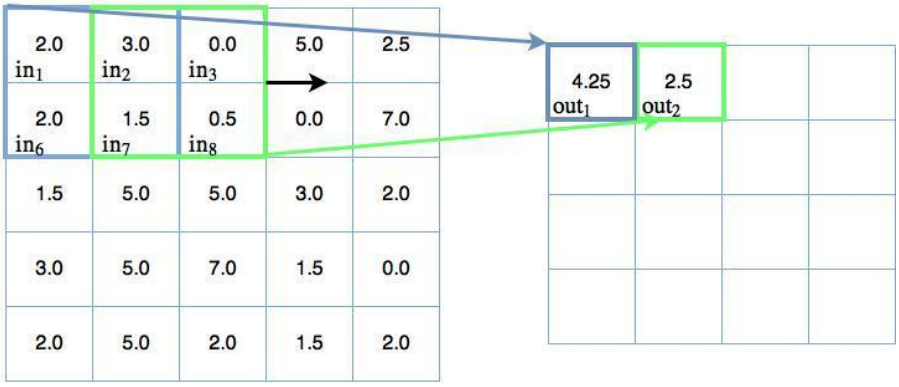
As redes convolucionais representam um dos tipos de modelos que podem ser utilizados para compor uma arquitetura de aprendizado profundo. Tendo como entrada, geralmente, imagens, essas redes apresentam neurônios que realizam, a operação de convolução.

Conforme Ludwig (2007), 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*. Os filtros são matrizes de pesos geradas aleatoriamente que, durante o processo de treinamento, têm seus pesos ajustados.

Pode-se entender a convolução como o processo de mover esses filtros sobre a imagem, conforme mostrado na Figura 2.

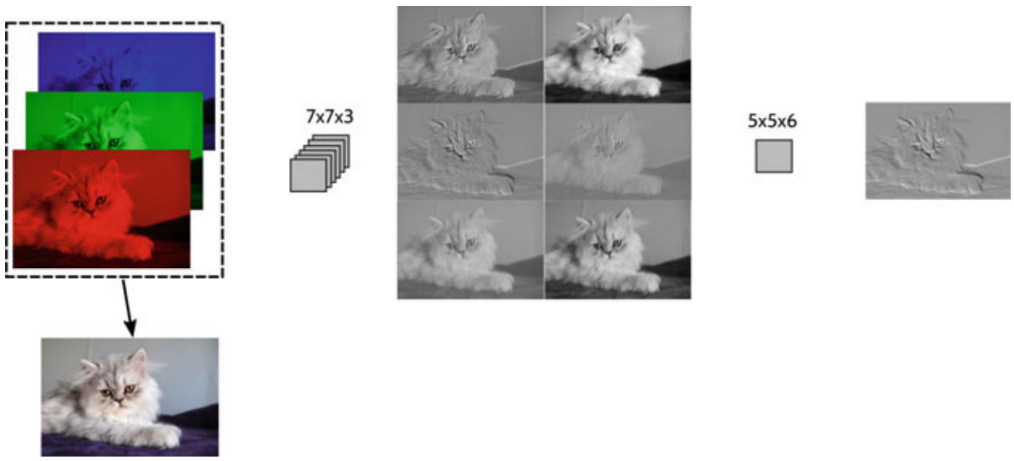
Através da convolução, são extraídas características das imagens de modo a descrevê-las em diversas outras imagens mais simples, como mostrado na Figura 3. Ela exemplifica a passagem de uma imagem por duas camadas convolucionais, a primeira com seis filtros de dimensão 7x7x3 e a segunda contendo apenas um filtro de 5x5x6.

Figura 2 – Exemplo do funcionamento de filtros convolucionais



Fonte – Adventures in Machine Learning (2019)

Figura 3 – Resultado da aplicação de filtros convolucionais em uma imagem



Fonte – Aghdam e Heravi (2017)

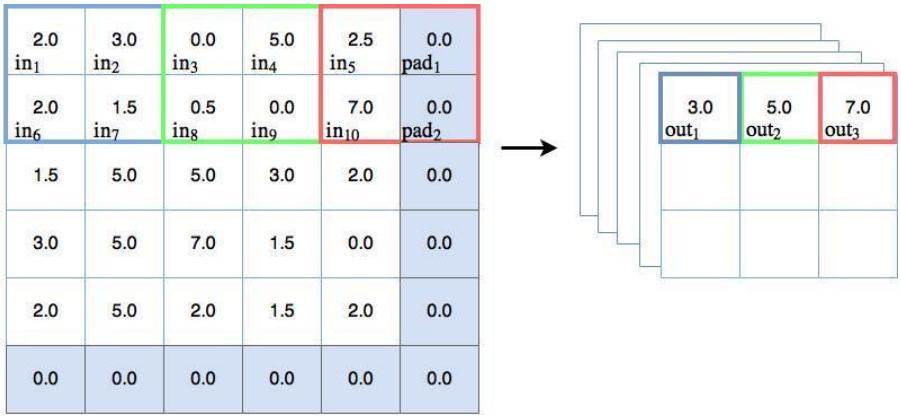
2.4.1.1 *Pooling*

Após a passagem por camadas convolucionais, é comum que seja aplicado algum tipo de *pooling* nas imagens resultantes. Ele funciona de maneira similar aos filtros convolucionais vistos anteriormente na subseção 2.4.1, mas, ao invés de multiplicar pesos aos pixels da imagem, estes filtros realizam algum tipo de operação estatística na vizinhança: usualmente encontrar o valor máximo ou a média.

Na Figura 4 é possível ver um exemplo da aplicação de um pooling máximo. Ou seja para cada vizinhança que o filtro percorrer, apenas o valor máximo será extraído. Dependendo do tamanho da imagem e do filtro, pode ser necessário adicionar bordas extras à imagem, representadas como os pixels azuis da Figura 4.

A aplicação de *pooling* ajuda a reduzir a quantidade de dados gerados através das camadas convolucionais, fazendo com que o modelo seja mais robusto e menos sensível a variações de escala e rotação.

Figura 4 – Exemplo de pooling máximo



Fonte – Adventures in Machine Learning (2019)

2.5 Processamento de Imagem

Derivada da área de processamento de sinais, o processamento de imagens, como seu nome sugere, tem como foco as imagens.

É uma área de estudo bastante grande que contempla diversos tipos de operações que podem ser categorizadas como sendo de baixo, intermediário ou alto nível (JUNIOR, 2018).

- **Baixo nível**: Contempla as operações primitivas que podem ser realizadas em imagens. Alguns exemplos são a redução de ruídos e aumento do contraste. Essas operações têm como entrada uma imagem e fornecem como saída também uma imagem;
- **Nível intermediário**: Essas operações, por sua vez, incluem a segmentação, análise e descrição da imagem. Alguns exemplos: detecção de borda, detecção de contornos e criação de histogramas. Essas operações tomam como entrada uma imagem e fornecem atributos como saída;
- **Alto nível**: Por fim, as operações de alto nível são as que buscam dar como saída algum sentido às imagens que recebem como entrada. Contemplam, por exemplo, as operações de detecção e localização de objetos.

2.5.1 Detecção de Borda

Como mencionado anteriormente na seção 2.5, uma das operações existentes na área de processamento de imagem é a da detecção de borda.

Ela consiste na identificação de variações bruscas na intensidade luminosa de uma imagem através da análise de gradientes. Com isso, é possível detectar uma mudança do objeto principal para o plano de fundo, por exemplo, o que caracteriza uma borda.

2.5.1.1 Algoritmo de Canny

Há diversas técnicas disponíveis para realizar essa tarefa, como métodos de Gradiente, filtro de *Sobel* e o algoritmo de *Canny*. Este último, por sua vez, é o que foi utilizado neste trabalho.

Desenvolvido por John F. Canny, ele é divido em quatro estágios principais, que serão descritos a seguir. A versão do algoritmo utilizada é a presente na biblioteca *OpenCV* e descrita na documentação da ferramenta (OPENCV, 2019).

1. **Redução de Ruído**: Primeiramente, é aplicado um filtro Gaussiano bidimensional (usualmente de dimensão 5x5) para suavizar a imagem de entrada. Isso é útil para remover quaisquer ruídos que possam estar presentes na imagem, dado que a detecção de borda é bastante suscetível a ruídos;

2. **Diferenciação**: Em seguida, para cada pixel da imagem, são determinadas a magnitude dos gradientes nas direções horizontal (G_x) e vertical (G_y). Tendo esses valores, é possível encontrar a magnitude gradiente de borda G e sua direção θ usando as Equações 2.1 e 2.2;

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2.2}$$

3. **Supressão de pontos não máximos**: Tendo G e θ, a imagem é submetida a uma verificação completa para remover os pixels que podem não constituir a borda, ou seja, que são indesejados. Isso é feito analisando, para cada pixel, se o mesmo é um máximo local em sua vizinhança na direção do gradiente;

4. **Limiarização (*Thresholding*)**: Por fim, é aplicado um método de limiarização chamado histerese para definir quais linhas realmente representam uma borda. Nele, são tomados um limitante superior e inferior. Quaisquer linhas acima do limitante superior, são consideradas como borda e mantidas na imagem. De maneira complementar, quaisquer linhas abaixo do limitante inferior são descartadas. As linhas presentes dentro do intervalo dos dois limitantes são consideradas como borda se estiverem conectadas a alguma linha que esteja acima do limitante superior.

3 Conjunto de Dados

Considerando o exposto na seção 2.4 a respeito do sobre-ajuste, a construção de um conjunto de dados apropriado é muito importante para a obtenção de um bom sistema. Desse modo, o autor optou por fazer manualmente a geração dos dados através da captura com uma câmera e posterior tratamento digital.

Os sinais do alfabeto foram classificados em dois grupos: os parados e os com movimento. O grupo dos com movimento contempla as letras H, J, K, X, Y, Z. Todas as outras letras possuem sinais parados. Essa divisão é importante pelo fato de que os sinais parados podem ser representados apenas com imagens, mas os com movimento demandam outras estratégias, como o uso de vídeos.

Este trabalho, particularmente, foi focado nas 20 letras paradas. Isso justifica-se pelo fato de que, através das letras disponíveis, já é possível formar uma grande quantidade de palavras ao mesmo tempo que simplifica o problema a ser resolvido, por não precisar tratar dos vídeos.

3.1 Captura

Primeiramente, foram capturadas imagens dos 20 sinais parados do alfabeto, em diversas posições e enquadramentos. Visando facilitar a remoção do plano de fundo, foi utilizado um fundo completamente verde, técnica conhecida como *Chroma Key*.

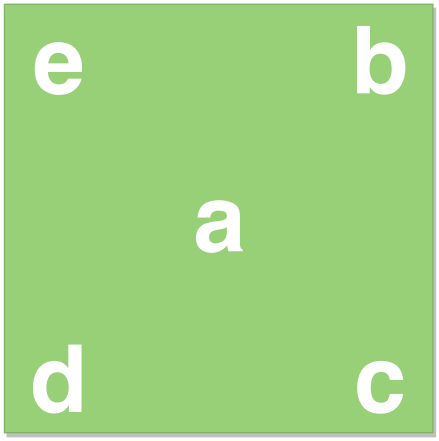
A captura das imagens foi feita usando uma câmera Nikon Coolpix P900, no modo de disparo contínuo. A resolução foi a máxima disponível (16M), na proporção 1:1, resultando em imagens quadradas de tamanho 3456x3456 pixels.

Foram definidos cinco diferentes enquadramentos para gerar maior variabilidade de imagens, conforme Figura 5.

Dessa forma, todos os sinais foram fotografados estando posicionado em cada um dos enquadramentos definidos, gerando imagens como as mostradas na Figura 6.

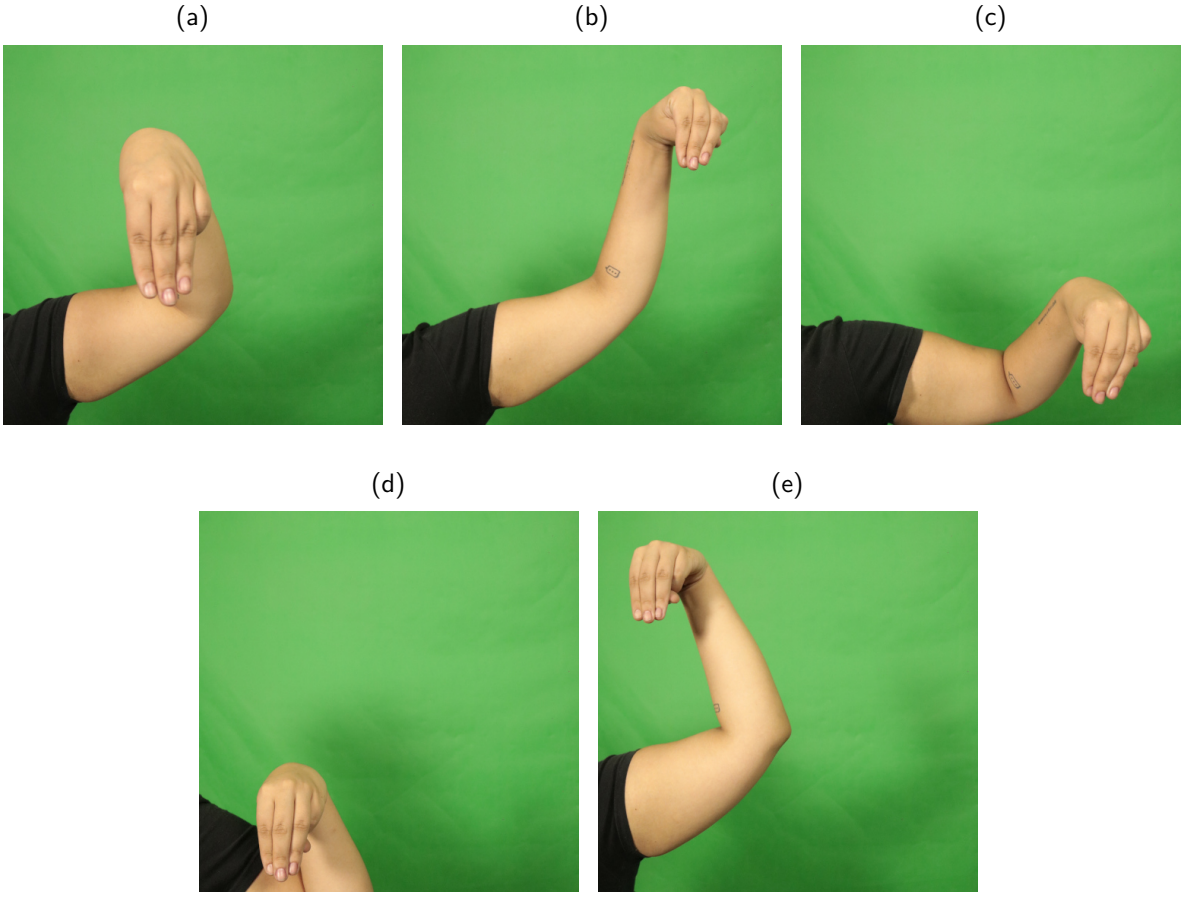
Além disso, para cada enquadramento, foi feita não apenas uma foto com a mão vista de frente, como também ela estando inclinada levemente para a esquerda e para a direita, como ilustrado na Figura 7. Com isso, considerando que foi utilizado o disparo contínuo, foram geradas aproximadamente 100 imagens para cada sinal do conjunto dos parados, totalizando 2274 imagens iniciais.

Figura 5 – Esquema de enquadramentos para captura dos sinais



Fonte – Elaborada pelo autor

Figura 6 – Exemplos de imagens nos enquadramentos definidos



Fonte – Elaboradas pelo autor

Figura 7 – Exemplo de sinal rotacionado para a (a) esquerda, (b) frente e (c) direita.

(a) (b) (c)

Fonte – Elaboradas pelo autor

3.2 Tratamento

Finalizada a captura das imagens, estas foram tratadas digitalmente visando a remoção do plano de fundo para posterior anotação e corte da região de interesse.

3.2.1 Remoção do plano de fundo

Nesta primeira etapa, foi utilizado o *software Blender*[1] versão 2.79. Ele é gratuito, de código aberto, está disponível para *Windows*, *macOS* e *Linux* e realiza tarefas tanto em duas (2D) quanto em três dimensões (3D) para modelagem, animação, texturização, composição, renderização, e edição de imagens e vídeos.

Para este trabalho, foi utilizado o modo de composição. O mesmo demanda a criação de um grafo contendo todas as operações que deverão ser realizadas, sequencialmente, na imagem. Os nós utilizados estão descritos na Figura 8.

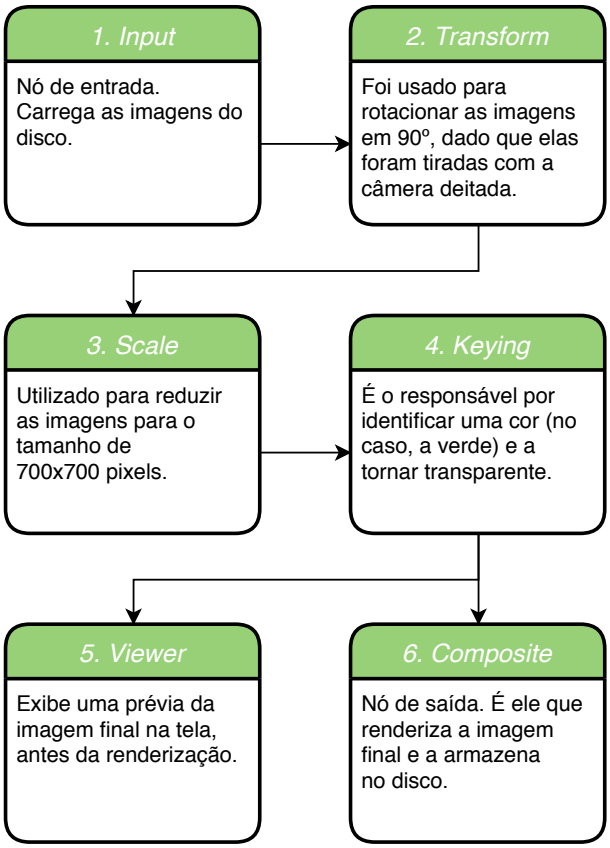
3.2.2 Região de Interesse

A posição das mãos e braços é de extrema importância na compreensão de um sinal em Libras. A mesma configuração de mão pode ter significados diferentes dependendo de onde as mãos estão posicionadas. Apesar disso, para este trabalho, a informação da posição dos braços não é relevante na identificação da letra individual.

Tendo isso em mente, foi realizado o corte apenas das mãos nas imagens. Isso facilita para o classificador, pois são informações a menos para processar. Em seguida, foi utilizada a máscara de transparência criada com o *Blender* na etapa anterior e tornado o plano de fundo inteiramente preto. Por fim, as imagens foram reduzidas para o tamanho padronizado de

[1] <https://www.blender.org/>

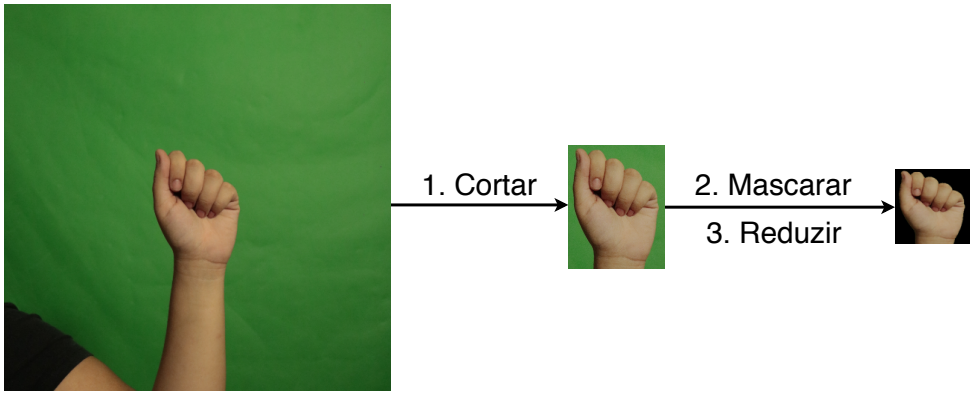
Figura 8 – Nós utilizados no pré-processamento feito com o Blender.



Fonte – Elaborada pelo autor

100x100 pixels. Exemplos de imagens obtidas em cada etapa desse processo estão mostradas na Figura 9.

Figura 9 – Etapas do tratamento das imagens



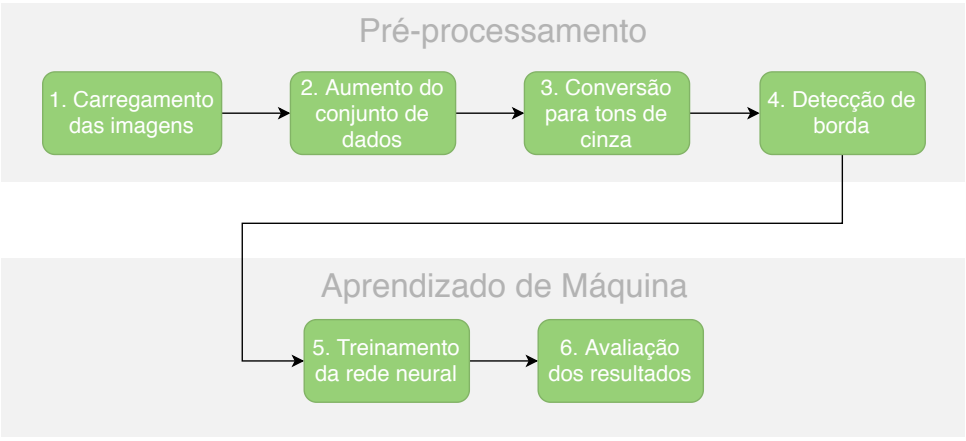
Fonte – Elaborada pelo autor

4 Desenvolvimento

Para realizar a classificação das letras, foram utilizadas técnicas de processamento de imagem e aprendizado de máquina profundo. Para tal, foi utilizada a linguagem *Python*, que possui diversas bibliotecas úteis já implementadas. Dentre estas, foram utilizadas a *OpenCV*, que trata de visão computacional e processamento de imagens; e a *Keras*, biblioteca para redes neurais artificiais. Ambas possuem código aberto e são amplamente utilizadas.

O desenvolvimento do classificador foi realizado seguindo a sequência de etapas ilustrada na Figura 10, sendo dividida em uma parte de pré-processamento e outra de aprendizado de máquina. Esta organização foi baseada no trabalho de Prateek et al. (2018), com algumas adaptações. Assim como no trabalho mencionado, as imagens do conjunto de dados foram convertidas para tons de cinza e foi realizada detecção de borda. Apesar disso, Prateek et al. (2018) adicionaram características extraídas manualmente através do algoritmo *Scale Invariant Feature Transform* (SIFT). Apesar de testada, a utilização deste e outros descritores como o *Oriented FAST and Rotated BRIEF* (ORB) não apresentou contribuições significativas nos resultados finais neste trabalho, quando comparada à versão sem o uso destes descritores.

Figura 10 – Sequência utilizada no desenvolvimento do classificador



Fonte – Elaborada pelo autor

4.1 Pré-processamento

O pré-processamento foi realizado usando principalmente a ferramenta *ImageDataGenerator* disponível na *Keras*. Ela cobre a maior parte das necessidades desta etapa.

4.1.1 Carregamento das Imagens

Um dos métodos desta ferramenta é o *flow_from_directory*. Ele faz o carregamento de imagens do disco em lotes, o que otimiza o uso de memória durante o treinamento da rede neural artificial, pois não é preciso manter todas as imagens do conjunto de dados carregadas na memória durante todo o tempo.

4.1.2 Aumento do conjunto de dados

A capacidade de generalização de um modelo de aprendizado profundo está diretamente relacionado à qualidade dos dados utilizados. Se esses dados não apresentarem muita variação, o classificador pode apresentar sobre-ajuste, isto é, apesar de possuir bons resultados na avaliação do conjunto de treinamento, o modelo não apresenta bons resultados quando testado com dados aos quais ele não foi exposto no treinamento.

Por conta disso, quando se dispõe de uma pequena quantidade de dados, como é o caso deste trabalho, realizar o aumento do conjunto de dados pode auxiliar no desenvolvimento de um bom classificador.

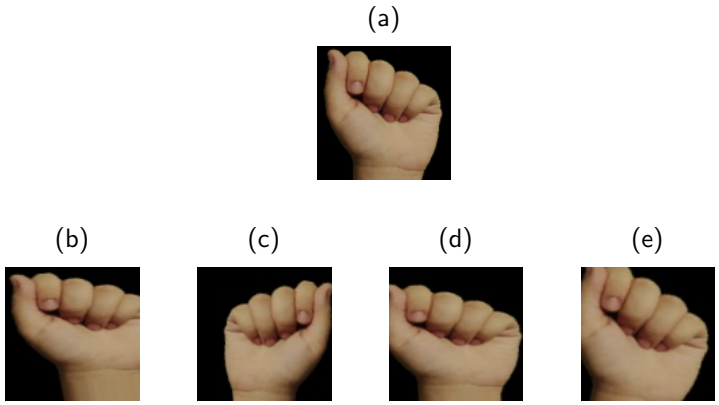
A classe *ImageDataGenerator* também apresenta ferramentas para realizar esse aumento. Dentre as diversas opções disponíveis, foram utilizadas as seguintes. Cabe notar que diferentes opções podem ser aplicadas concomitantemente. Uma lista de todas as opções pode ser encontrada na documentação da ferramenta[1].

- ***rotation_range* = 15**: Realiza rotações aleatórias na imagem de até no máximo o valor fornecido, em graus. Neste caso, as rotações serão de no máximo 15°;
- ***width_shift_range* = 0.1**: Realiza deslocamentos aleatórios na imagem no sentido da largura. Neste caso, os deslocamentos serão de no máximo 10%;
- ***height_shift_range* = 0.1**: Realiza deslocamentos aleatórios na imagem no sentido do comprimento. Assim como no item anterior, os deslocamentos serão de no máximo 10%;
- ***zoom_range* = 0.1**: Realiza ampliações aleatórios na imagem. As ampliações foram configuradas para ser de no máximo 10%;
- ***horizontal_flip* = *True***: Aleatoriamente espelha a imagem no eixo horizontal.

As imagens geradas são ligeiramente diferentes entre si, como pode ser visto na Figura 11. Apesar disso, essas pequenas diferenças já ajudam bastante no aumento da generalização do modelo que será aprendido.

[1] <https://keras.io/preprocessing/image/>

Figura 11 – Exemplo de (a) imagem do conjunto de dados e (b), (c), (d), (e) imagens derivadas geradas artificialmente.



Fonte – Elaboradas pelo autor

4.1.3 Conversão para tons de cinza

As imagens geradas devem, em seguida, ser convertidas para tons de cinza. Isso dá-se por dois motivos. Primeiramente, pelo fato de que não é necessário manter os 3 canais de cores, pois o classificador aprenderá melhor se a imagem possuir apenas um canal. Mais importante que isso, a próxima etapa, detecção de borda, exige que a imagem seja de apenas um canal.

A classe *ImageDataGenerator* também facilita nesta etapa, dado que só é necessário configurar a opção *color_mode* como *'grayscale'*.

4.1.4 Detecção de borda

A última etapa do pré-processamento é a detecção de borda. Essa detecção tem como objetivo reduzir uma imagem a uma versão binarizada que possui apenas as partes importantes, o que também contribui para a qualidade do classificador a ser treinado.

A técnica utilizada neste trabalho foi o algoritmo de Canny, descrito na subseção 2.5.1.1. Um exemplo de resultado de sua aplicação pode ser visto na Figura 12.

Figura 12 – Exemplo de (a) imagem em tons de cinza e (b) resultado da detecção de borda.

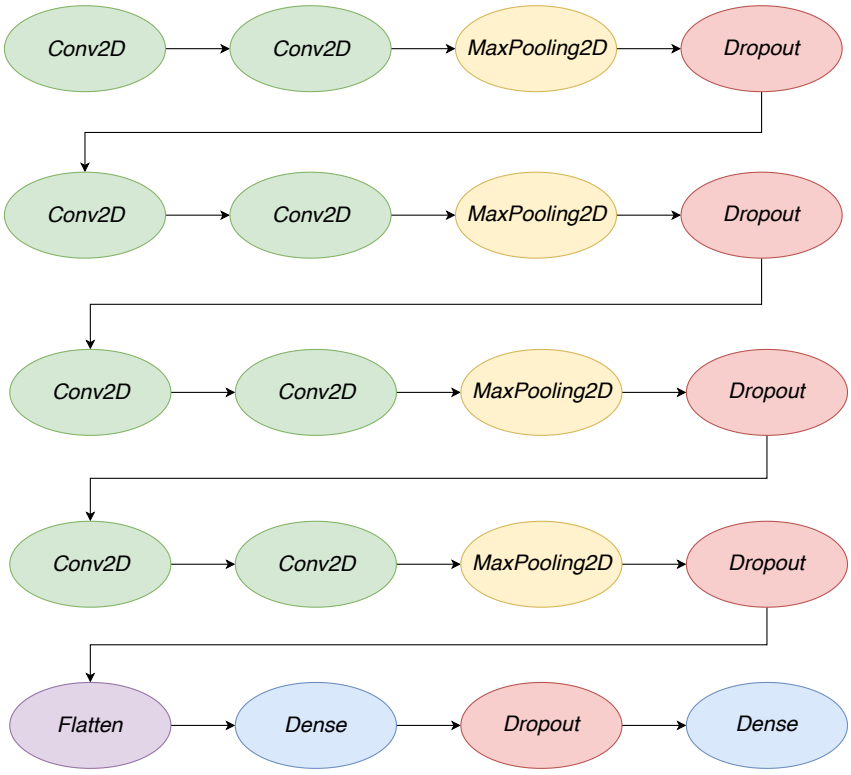
Fonte – Elaboradas pelo autor

Este algoritmo está implementado na biblioteca *OpenCV* e recebe três parâmetros: a imagem, limite inferior e limite superior. Os limites definem quais arestas continuarão na imagem final. Neste trabalho, foram utilizados os valores de 100 e 200 para os limites inferior e superior, respectivamente. Mais detalhes de sua utilização podem ser encontrados nos tutoriais da ferramenta[2].

4.2 Aprendizado de Máquina

Terminado o pré-processamento, as imagens estão prontas para iniciar a etapa de aprendizado de máquina. Para tal, foi construída uma arquitetura profunda utilizando as ferramentas da *Keras*. Foram utilizadas as camadas *Conv2D*, *MaxPooling2D*, *Dropout*, *Flatten* e *Dense*. Uma representação do modelo final pode ser vista na Figura 13.

Figura 13 – Representação do modelo construído



Fonte – Elaborada pelo autor

4.2.1 Conv2D

As duas primeiras camadas do modelo, *Conv2D*, são Convolucionais e trabalharão sobre dados 2D, neste caso, as imagens. O funcionamento de redes convolucionais foi descrito na subseção 2.4.1.

[2] <https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html>

4.2.2 *MaxPooling2D*

A camada a seguir, *MaxPooling2D* representa a realização de um *pooling* máximo nas imagens. O conceito de *pooling* foi definido na subseção 2.4.1.1.

4.2.3 *Dropout*

Esta camada é responsável por diminuir o tamanho da rede neural artificial ao eliminar aleatoriamente algumas conexões entre os neurônios. Isso serve como uma regularização e também ajuda reduzir o sobre-ajuste. Esta camada não possui parâmetros a serem treinados, atuando apenas como auxiliar no desenvolvimento da rede neural artificial.

4.2.4 *Flatten*

Assim como a anterior, a camada *Flatten* também não possui parâmetros a serem treinados. Sua função é, como o nome sugere, achatar os dados. Isto é, ela torna dados multidimensionais em unidimensionais. Essa etapa é importante pelo fato de que a próxima camada demanda dados unidimensionais.

4.2.5 *Dense*

A camada final, *Dense*, representa uma camada padrão de uma rede neural direta completamente conectada. Essa estrutura é utilizada em redes neurais artificiais mais simples, como mencionado na subseção 2.3.1.

Essa camada, por sua vez, será a responsável por pegar os dados obtidos das camadas anteriores e realizar a predição final da classe correspondente à imagem de entrada.

4.3 Avaliação dos Resultados do Treinamento

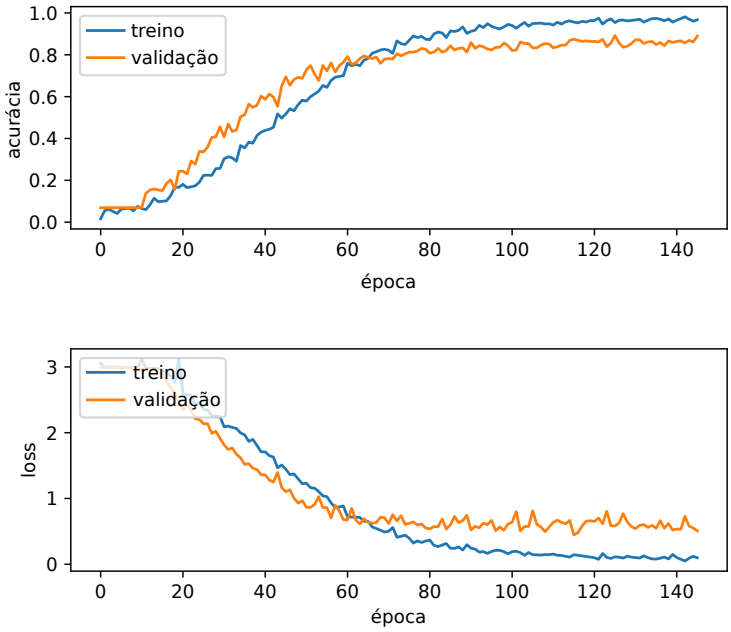
O treinamento foi realizado utilizando um servidor contendo dois processadores Intel Xeon E5-2603 v3 operando a 1.60GHz e 32GB de memória RAM.

A versão instalada da biblioteca *Keras* foi a 2.2.5 tendo como *backend* o *Tensorflow* 1.14.0. Já a versão da *OpenCV* utilizada foi a 4.1.1.26.

Terminado o treinamento, o classificador obteve aproximadamente 88% de acurácia e 0.45 de *loss* na avaliação do conjunto de validação. A evolução do treinamento pode ser vista na Figura 14.

Como pode ser visto nas imagens, o treinamento todo foi realizado em 140 épocas, isto é, foram necessárias 140 passagens por todo o conjunto de dados. Apesar disso, é possível observar que em apenas 60 épocas já se tinha atingido valores próximos de 80% para a acurácia. A partir deste ponto, a melhora dos valores ocorreu de maneira bastante lenta; o que já era

Figura 14 – Evolução da acurácia e *loss* durante treinamento



Fonte – Elaborada pelo autor

esperado, dado ser algo comum em Aprendizado de Máquina. Por fim, a diferença entre a evolução dos valores no treino e validação sugere que o classificador não está sofrendo de sobre-ajuste.

A partir dessa avaliação, foi elaborada uma matriz de confusão. Conforme Souza (2019), matriz de confusão "é uma tabela que mostra as frequências de classificação para cada classe do modelo". Dessa forma, é possível identificar quais classes do modelo estão se sobrepondo. A matriz final pode ser vista na Figura 15.

Analisando a matriz de confusão, é possível identificar que o erro mais comum do classificador ocorre na diferenciação das letras 'F' e 'T'. Isso é algo natural, dada a similaridade entre as letras, mostradas nas Figuras 1, 16, 22c e 22d; estando essas últimas duas na página 41. A posição das mãos é praticamente a mesma: dedo indicador deitado e demais dedos em pé, com a diferença na posição do polegar em relação ao indicador.

Figura 15 – Matriz de confusão resultante

Fonte – Elaborada pelo autor

Figura 16 – Comparação entre sinal da letra (a) F e (b) T

(a)

(b)

Fonte – Elaboradas pelo autor

5 Utilização do Modelo

Apesar de já possuir o modelo treinado, dado o pré-processamento efetuado nas imagens do conjunto de dados, não é possível pegar uma imagem qualquer e tentar realizar a predição da letra a que se refere. Assim, é necessário realizar algum tipo de pré-processamento nas imagens que serão utilizadas na predição.

5.1 Caixa Delimitadora

A primeira coisa a se fazer, é definir as coordenadas de uma caixa delimitadora onde deverá ser posicionada a mão antes de ter sua letra reconhecida. O objetivo dessa caixa é restringir a imagem apenas ao objeto de interesse.

> "Na detecção de objetos, geralmente usamos uma caixa delimitadora para descrever o local do alvo. A caixa delimitadora é uma caixa retangular que pode ser determinada pelas coordenadas dos eixos x e y no canto superior esquerdo e as coordenadas dos eixos x e y no canto inferior direito do retângulo" (DIVE INTO DEEP LEARNING, 2019, tradução nossa).

Existem algumas técnicas de aprendizado profundo que conseguem determinar essas coordenadas automaticamente, como as redes *Single Shot Detector* (SSD). Embora testada para este trabalho, por o conjunto de dados não ser muito grande, esta técnica não apresentou resultados satisfatórios. Dessa forma, foram definidas manualmente as coordenadas de uma caixa delimitadora.

5.2 Segmentação da Imagem

Como pode ser visto na Figura 11, as imagens utilizadas para fazer o treinamento do classificador tiveram seu plano de fundo removido. É necessário fazer um processo similar com as imagens de teste.

Na construção do conjunto de dados, conforme mencionado na Seção 3.1, foi utilizado um fundo verde. Dessa forma, sua remoção era baseada em identificar os tons de verde e mascará-los. Entretanto, para as imagens de teste, não necessariamente o fundo será completamente verde. Dessa forma, é necessário encontrar outra forma de encontrar a pele da imagem para remover o plano de fundo.

Um caminho lógico para fazer isso é, ao invés de tentar identificar os tons de verde, buscar as cores referentes ao tom de pele na imagem. O problema deste processo é determinar quais os valores de cor a se procurar, dado que a cor da pele em uma imagem pode variar muito;

não apenas pelo tom de pele particular de cada pessoa, como também pelas diferenças de luminosidade. Dessa forma, é preciso que se determine qual a cor a ser buscada na identificação da pele antes de cada utilização do sistema, atuando como uma calibração.

5.2.1 Determinação da cor da pele

Embora possa parecer simples em um primeiro momento, a determinação da cor da pele do usuário contém diversos pontos a serem considerados.

O primeiro deles é que é impossível determinar uma única cor para representar toda a pele. Diversos tons se sobrepõe nas mãos. Normalmente é possível observar que as cores das extremidades da palma da mão geralmente são mais escuras que as do centro.

Por conta disso, é necessário definir não uma cor específica, e sim, limitantes de cor. Isto é, definir um limitante inferior e superior nos quais qualquer cor dentro do intervalo formado por estes limitantes será considerada como pele. Um outro ponto a ser considerado é que geralmente as cores da palma da mão são diferentes das da costa. Desse modo, é necessário considerar esses intervalos também.

Visando englobar todas essas restrições, foi desenvolvido um algoritmo para realizar a determinação dos limitantes. O algoritmo completo pode ser visto na Figura 17.

Ele está dividido em duas partes principais. Na primeira, são determinados os maiores intervalos consecutivos de cor em cada canal das duas imagens, da palma e da costa da mão. Com isso, é possível determinar qual o intervalo de cor dominante para cada canal das imagens.

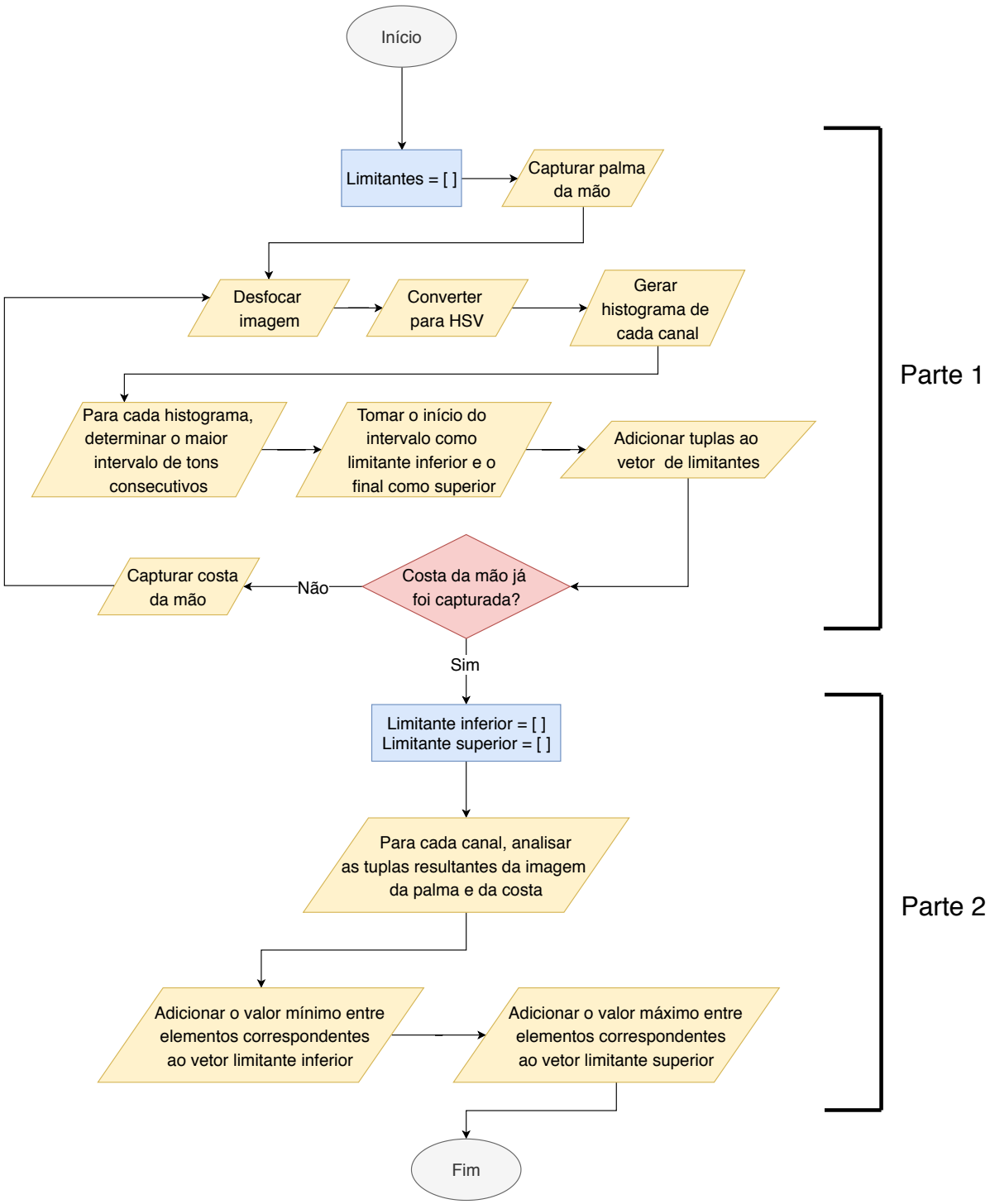
Primeiramente, são capturadas as imagens da palma e costa da mão. Para tomar essas fotos, é definida uma caixa delimitadora bem pequena, na qual o usuário deverá posicionar exatamente a parte desejada da mão, como vê-se na Figura 18. Em seguida, essa imagem é ligeiramente desfocada, visando diminuir possíveis ruídos.

Logo depois, as imagens devem ser convertidas para o espaço de cor HSV. Do inglês, *hue* - matiz, *saturation* - saturação e *value* - valor. A forma como esse espaço de cor representa as imagens é mais adequada para a definição destes limitantes.

Sabendo-se que os canais contém valores de 0 a 255 deve-se, primeiramente, encontrar o histograma de cada canal. Em seguida, verifica-se qual o maior intervalo do histograma no qual todos os elementos dentre desse intervalo possuem pixels que contém esses valores.

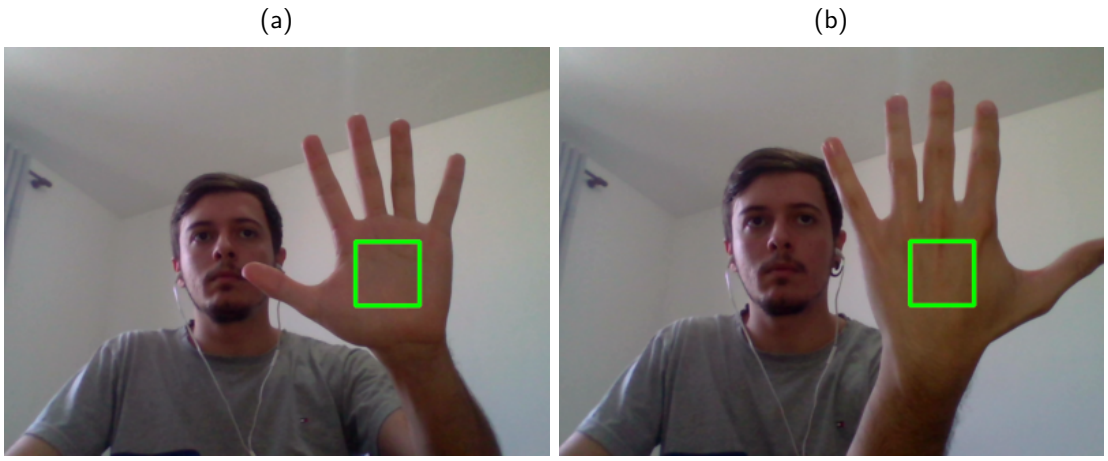
Como mostrado na Figura 19, ao final dessa etapa, obtém-se um vetor com dois elementos, ambos possuindo três tuplas, cada uma referente a um canal. Tendo todas as tuplas, é necessário combiná-las para formar os limitantes inferior e superior finais. Essa é a segunda parte do algoritmo. Isso foi feito através da adição do menor valor de elementos correspondentes ao vetor de limitante inferior e adição do maior valor ao limitante superior, conforme ilustrado na Figura 20.

Figura 17 – Algoritmo para determinação de limitantes



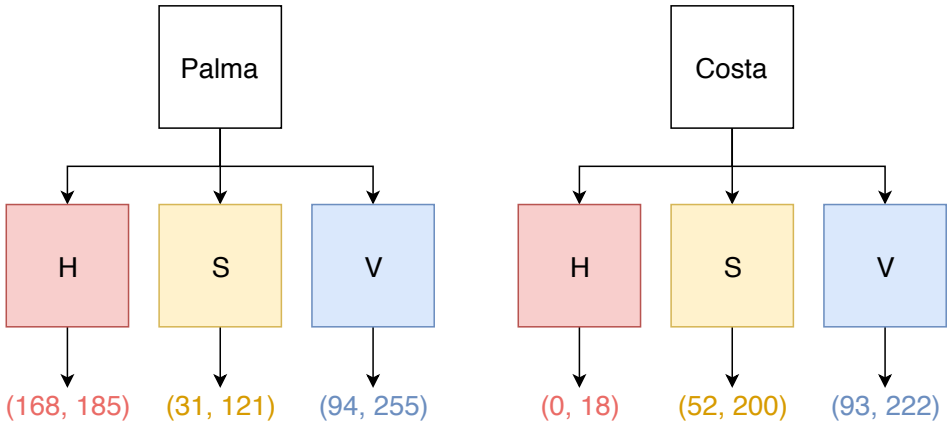
Fonte – Elaborada pelo autor

Figura 18 – Calibração do sistema: captura da (a) palma e (b) costa da mão



Fonte – Elaboradas pelo autor

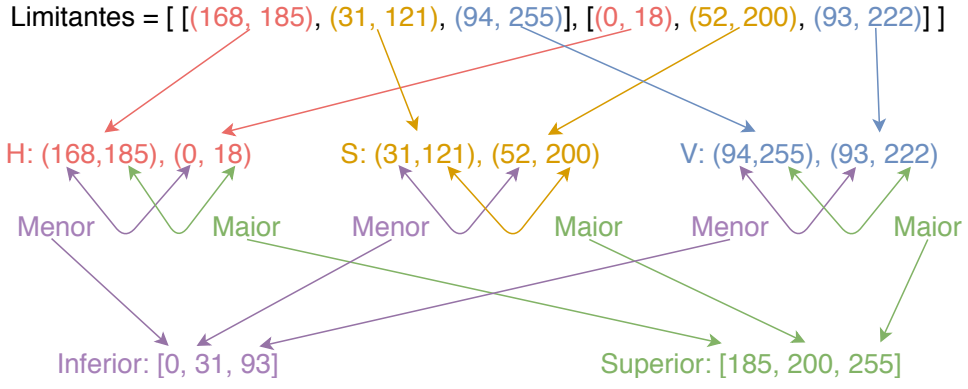
Figura 19 – Exemplo de resultado da primeira parte do algoritmo



Limitantes = [[(168, 185), (31, 121), (94, 255)], [(0, 18), (52, 200), (93, 222)]]

Fonte – Elaborada pelo autor

Figura 20 – Exemplo de resultado da segunda parte do algoritmo

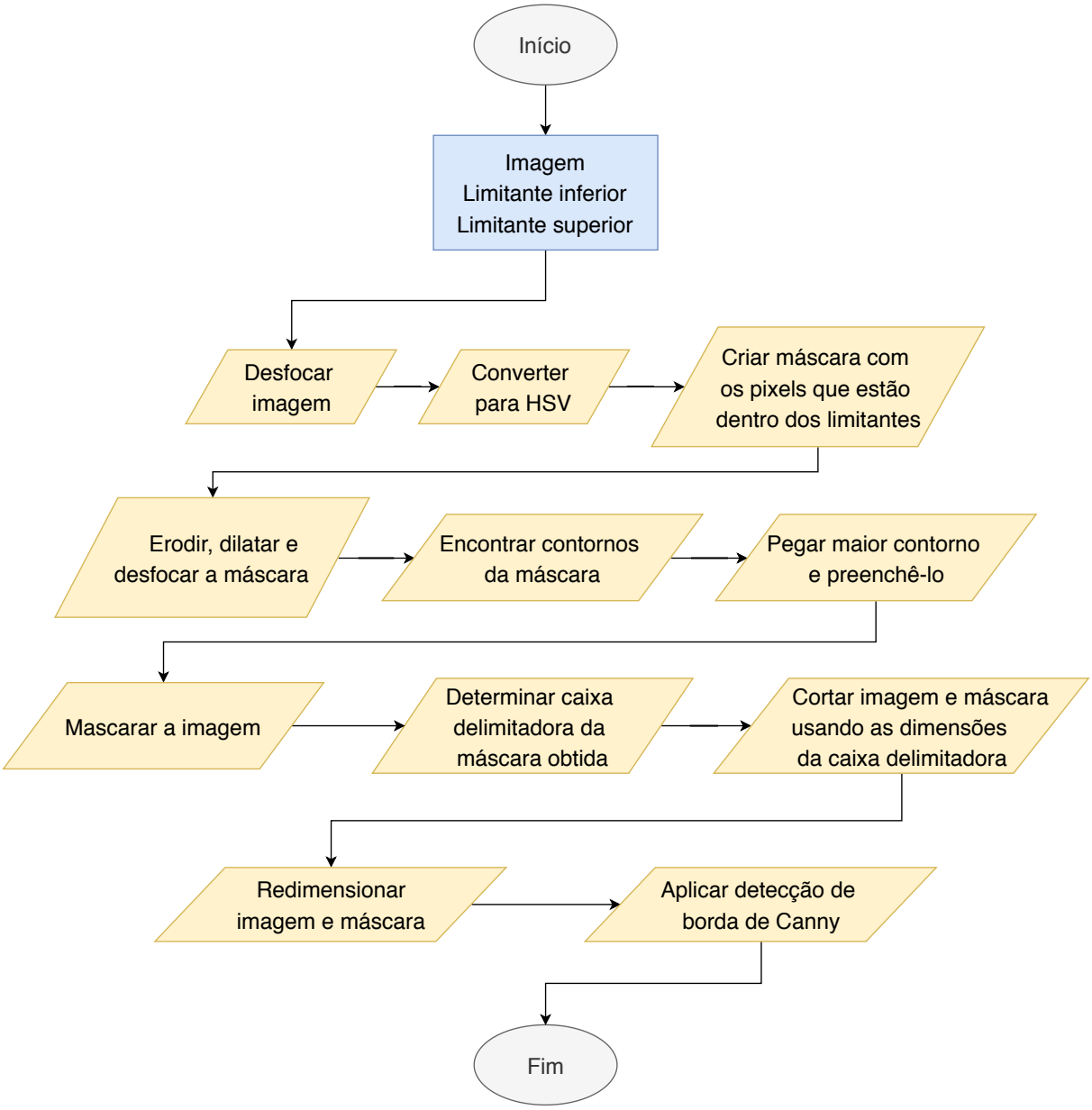


Fonte – Elaborada pelo autor

5.2.2 Região de Interesse

Tendo os valores dos limitantes, o próximo passo é determinar a região de interesse da imagem. Isto é, definir uma máscara que conterá os pixels referentes à mão, remover o plano de fundo, cortar e redimensionar a imagem. Para isso, foi desenvolvido o algoritmo da Figura 21.

Figura 21 – Algoritmo para determinação da região de interesse



Fonte – Elaborada pelo autor

Primeiramente, a imagem deve ser desfocada e convertida para HSV, como na etapa anterior. Em seguida, é criada uma máscara com os pixels que estão dentro dos limitantes definidos anteriormente.

Apesar de já ser uma boa aproximação, esta máscara pode conter alguns ruídos, dado que foi criada unicamente considerando as cores. Geralmente a máscara apresentará alguns buracos em certos pontos da mão. Dessa forma, algumas operações podem ser aplicadas a ela para diminuição de ruídos. Uma dessas operações é também aplicar um desfoque à máscara.

Outras técnicas são as operações morfológicas. Para este trabalho, foram utilizadas a erosão e dilatação.

> "Operações morfológicas são um conjunto de operações que processam imagens com base em formas. Eles aplicam um elemento estruturador a uma imagem de entrada e geram uma imagem de saída. As operações morfológicas mais básicas são duas: Erosão e Dilatação" (GEEKSFORGEEKS, 2018, tradução nossa).

A erosão reduz a área do objeto, o que diminui os detalhes da imagem. Já a dilatação tem o efeito contrário, expande a área da imagem, acentuando os detalhes. A forma morfológica utilizada para realizar essas operações foi uma circunferência.

Em seguida, deve-se encontrar os contornos da imagem. Tendo os contornos, o próximo passo é pegar o contorno que cubra a maior área. Dado que a imagem já foi pré-mascarada, o maior contorno corresponderá à linha externa da mão. Por fim, preenche-se completamente o interior deste contorno.

Esta sequência de passos reduz em grande parte a quantidade de ruídos presentes na máscara. Desse modo, ela já pode ser aplicada na imagem. Depois disso, o plano de fundo já terá sido removido da imagem.

Um método bastante útil da biblioteca *OpenCV* é o que, dada uma máscara, ele retorna uma caixa delimitadora dos objetos presentes. Assim, é possível identificar a posição exata da mão e centralizá-la, eliminando quaisquer eventuais bordas pretas que a imagem poderia conter.

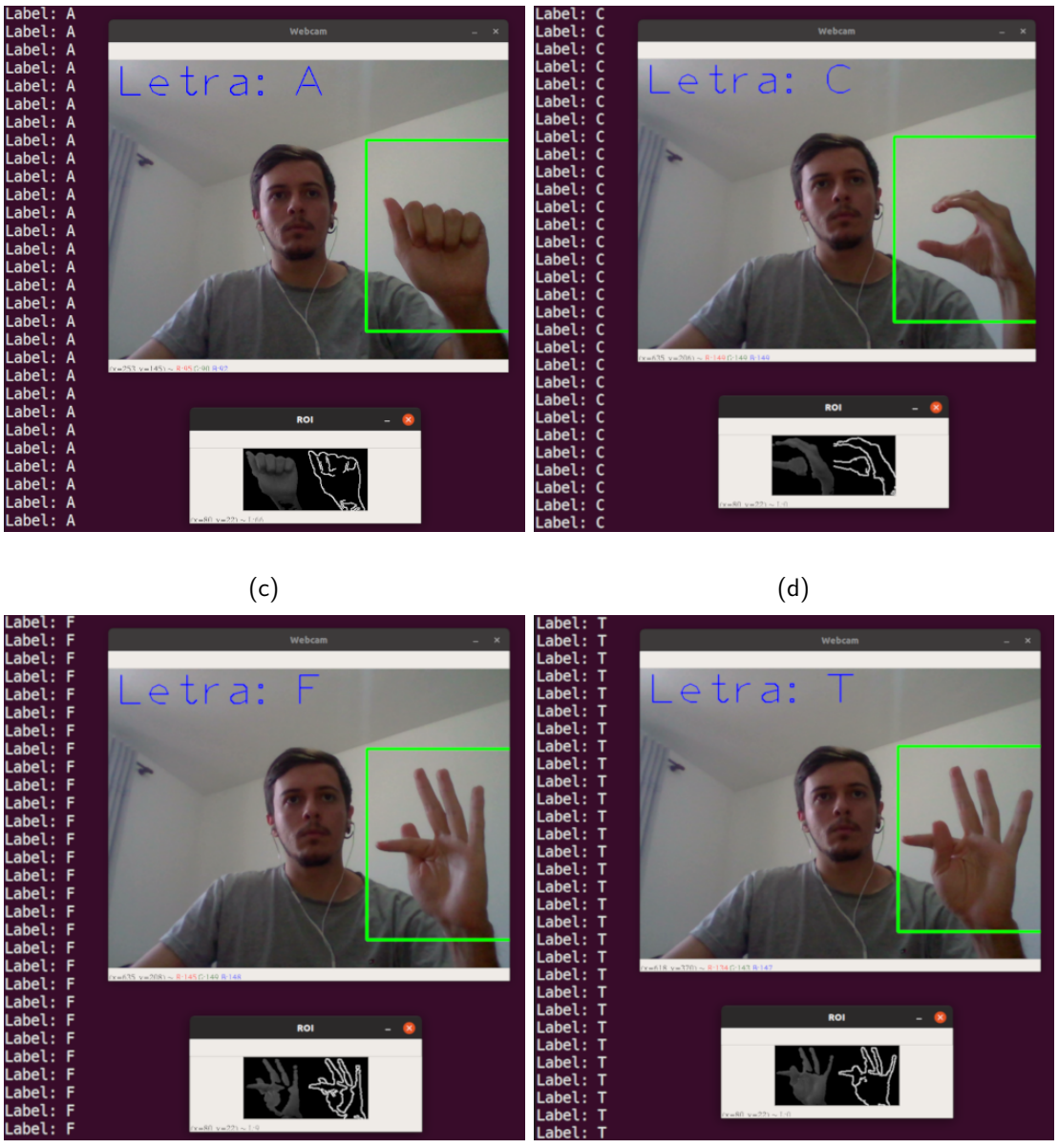
Após determinar a caixa delimitadora, a imagem e a máscara serão cortadas usando as coordenadas da caixa. Obtém-se, assim, uma imagem já mascarada e com a mão preenchendo quase que completamente a imagem. Por fim, imagem e máscara são redimensionadas para 100x100 pixels, tamanho das imagens do conjunto de dados.

O último passo é a aplicação do algoritmo de detecção de borda de Canny. Nesse ponto, a imagem já pode ser fornecida ao classificador para predição da classe.

Após a passagem por todos esses passos, foi desenvolvido o classificador mostrado na Figura 22. Nela, é possível ver em destaque a imagem da câmera com a indicação da letra reconhecida. A seguir, é possível ver a imagem da região de interesse (ROI) de duas formas: à esquerda, apenas com a pele segmentada e, à direita, essa mesma imagem após a aplicação do algoritmo de *Canny*. Ou seja, esta última imagem é a que será fornecida ao classificador. Por fim, vê-se na janela do terminal qual a letra reconhecida no momento.

Figura 22 – Sistema em uso para reconhecimento das letras (a) A, (b) C, (c) F e (d) T

(a) (b)



(c) (d)

Fonte – Elaboradas pelo autor

6 Interface Desenvolvida

De modo a facilitar a utilização do sistema, foi desenvolvida também uma interface com o usuário. Ela consiste em uma aplicação *web* de uma página. Do inglês, *Single-Page Application* ou apenas SPA.

6.1 Conceito

A aplicação foi desenvolvida tendo em mente a ideia de diferenças e que, ao juntar formas, cores e pessoas diferentes, se criam coisas boas.

Desse modo, a identidade visual inclui pessoas diferentes, sejam elas brancas, negras ou ainda de cores não convencionais como amarelo ou cinza. Além disso, há pessoas com deficiências, sendo representadas por pessoas com óculos, membros faltando ou utilizando cadeira de rodas.

Para compor o *design*, foram adicionadas também diversas formas irregulares em padrões que se encaixam com harmonia. Essas formas normalmente não estão contidas em apenas uma página, tendo continuidade na página seguinte.

A paleta de cores das formas, botões e *links* é baseada em tons azuis, os quais são os que possuem menor distorção dentre as pessoas com algum tipo de deficiência na visualização de cores, o daltonismo. Os desenhos das pessoas não seguem essa paleta.

Além disso, todas as páginas possuem um cabeçalho com *links* para navegação pela aplicação.

6.2 Página Inicial

A página inicial é bastante simples. Ela contém uma breve frase sobre do que se trata a aplicação e apresenta dois botões para acessar as principais funções propostas: reconhecimento de letra e a ferramenta de escrita. A tela desenvolvida pode ser vista na Figura 23.

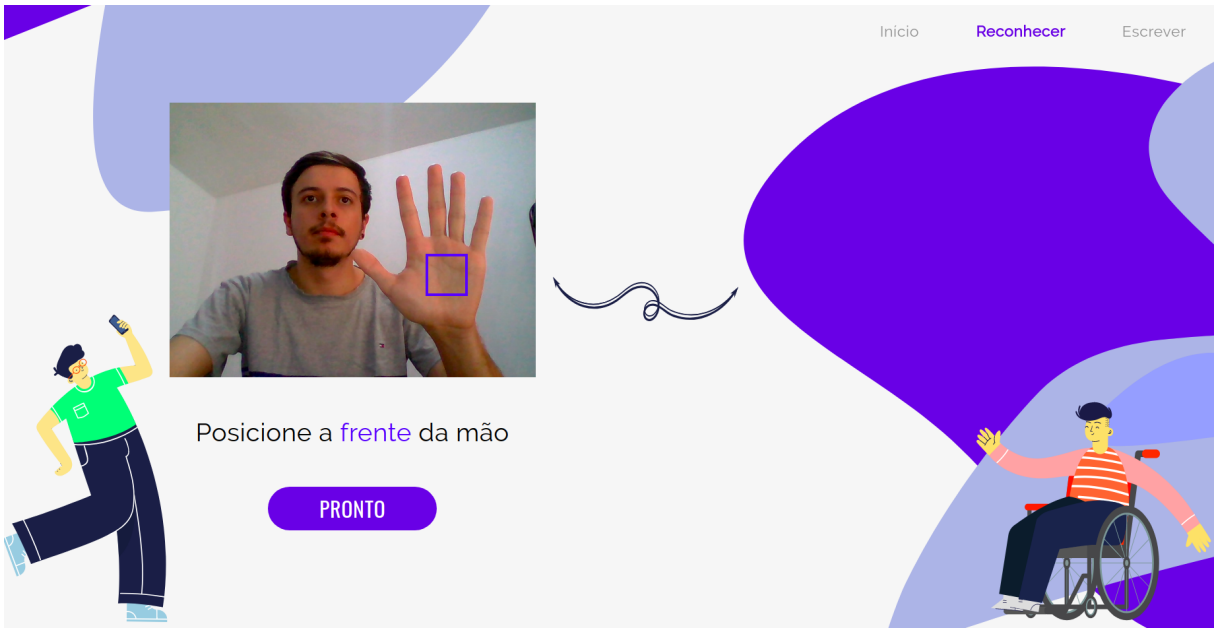
6.3 Calibração

Como mencionado na subseção 5.2.1, antes de utilizar o sistema, é necessário realizar a calibração do mesmo capturando as imagens do centro da frente e da costa da mão para que seja possível realizar a segmentação de pele. Na interface desenvolvida, a calibração é mostrada na Figura 24.

Figura 23 – Página inicial da interface proposta

Fonte – Elaborada pelo autor

Figura 24 – Calibração do Sistema

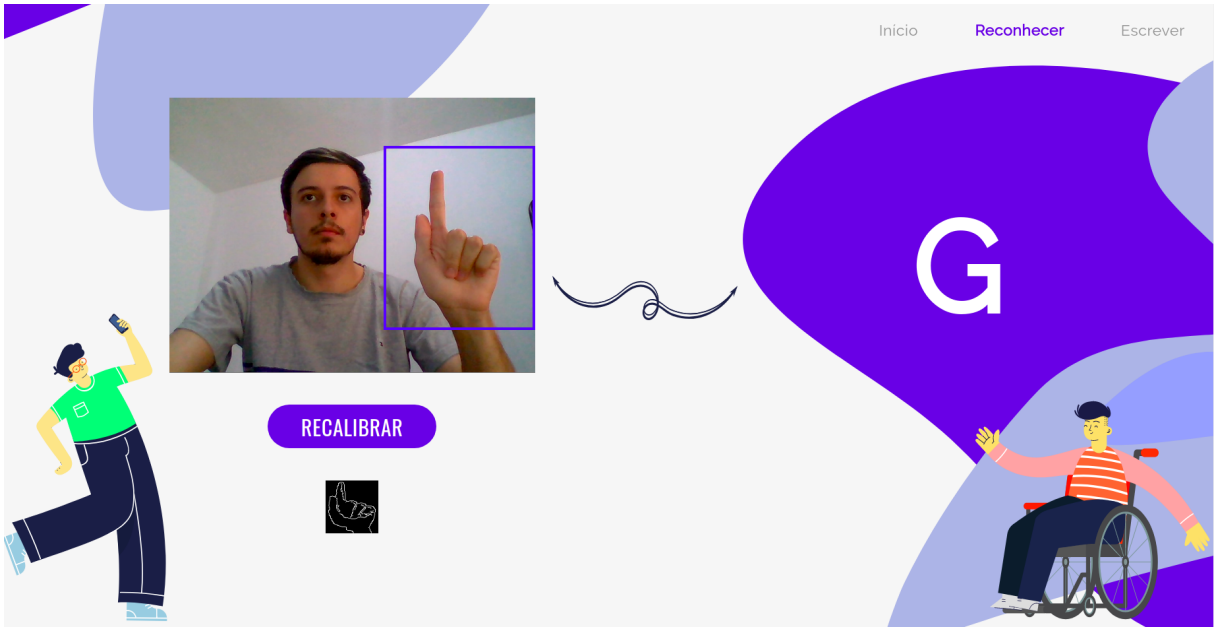


Fonte – Elaborada pelo autor

6.4 Reconhecimento de Letra

A segunda página apresenta a ferramenta de reconhecimento de letra. Do lado esquerdo, representada por um quadrado cinza, será exibida a imagem da *webcam* do usuário para que este possa se posicionar para começar a sinalizar. A letra reconhecida será exibida do lado direito, representada na Figura 25 pela letra 'G'.

Figura 25 – Ferramenta de reconhecimento



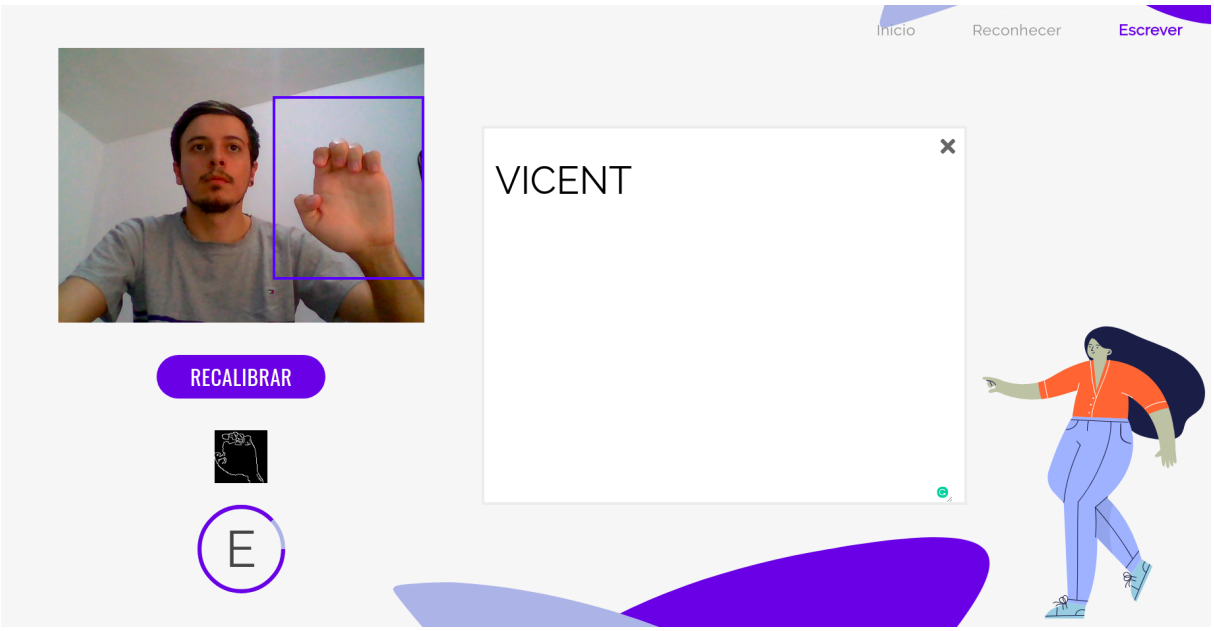
Fonte – Elaborada pelo autor

6.5 Ferramenta de Escrita

A última página da interface é uma ferramenta proposta para que possa escrever utilizando o sistema desenvolvido. Assim como na página anterior, há um quadrado cinza representando a *webcam*. Logo abaixo há a letra que está sendo reconhecida neste momento. Por fim, do lado direito há uma caixa de texto com as palavras já formadas. Esta página está ilustrada na Figura 26.

O funcionamento seria feito da seguinte forma: o usuário faz um sinal de uma letra na *webcam* e deve permanecer com o mesmo sinal tempo suficiente para preencher o contorno circular da área que mostra a letra reconhecida. Passado esse tempo, a letra seria considerada correta e adicionada à caixa de texto do lado direito.

Figura 26 – Ferramenta de escrita



Fonte – Elaborada pelo autor

7 Conclusão

Com base no trabalho desenvolvido, vê-se que o sistema atende de maneira bastante satisfatória ao objetivo inicial de reconhecer as letras paradas do alfabeto da Libras.

Durante o processo, foram abordados diversos tópicos que não estão presentes na grade de matérias eletivas do curso, como Processamento de Imagens e Aprendizado de Máquina. Contribuindo, desse modo, para a formação pessoal do autor.

É fato que, em comparação com os impressionantes resultados obtidos nos trabalhos de Prateek et al. (2018) e Taskiran; Killioglu; Kahraman (2018), o sistema desenvolvido apresentou resultados inferiores no treinamento; já que as acurácias obtidas nestes trabalhos correlatos foram superiores a 90%, enquanto a deste trabalho é de 88%. Apesar disso, a principal contribuição deste trabalho fica por conta do enfoque da Língua Brasileira de Sinais, utilizada pelos deficientes auditivos no Brasil. Sendo, então, um tema de importante relevância social.

Além disso, por conta da calibração inicial, a detecção de pele, em comparação com os outros trabalhos, é bem menos suscetível a problemas com diferentes tonalidades de pele. Em adição, a técnica de captura de imagens da frente e costa da mão para definição dos limitantes de cor não foi vista em nenhum outro trabalho.

Cabe notar que o classificador treinado realizou diversas predições erradas quando considerado o conjunto de validação, conforme mostrado na matriz de confusão da Figura 15. Um outro ponto de melhora são as formas de disponibilização do sistema.

Pelo exposto vê-se que, em trabalhos futuros, seria interessante buscar métodos para melhorar a separação das letras similares que fazem o classificador se confundir. Uma opção seria a utilização de classificadores em cascata. Isto é, fazer um classificador inicial que consideraria as letras parecidas como sendo a mesma classe e, quando uma imagem for identificada com essa classe, ela ser passada por um segundo classificador treinado especificamente para diferenciar essas letras.

Outra importante adição seria a das letras restantes do alfabeto. Dado que elas possuem movimento, seria necessário alterar a estrutura do classificador e, possivelmente, utilizar redes neurais artificiais diferentes. O tipo de dado de entrada para esse novo classificador poderia ser em formato de vídeos, que seriam classificados utilizando Redes Neurais Convolucionais 3D (3DCNN) ou Redes de Memória de Longo Prazo (LSTM).

Por fim, para difundir a utilização do sistema seria necessário melhorar sua disponibilidade. Assim, aperfeiçoamentos da interface desenvolvida e a possível criação de uma aplicação móvel são indispensáveis para aumentar a comodidade de uso e alcançar mais usuários.

Para finalizar este trabalho, é importante mencionar que a inclusão de todas as pessoas, independentemente de suas diferenças e necessidades especiais, é dever de todos. Esse pensamento deve se mostrar ainda mais forte quando dá-se conta do fato de que a Universidade Pública deve servir ao público. Desse modo, dar a devida atenção a esses assuntos é de extrema importância para que se possa trilhar um caminho em busca de uma sociedade mais justa, inclusiva e igualitária.

Referências

ADVENTURES IN MACHINE LEARNING. *Convolutional Neural Networks Tutorial in TensorFlow*. 2019. Disponível em: <https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/>. Acesso em: 20 out. 2019.

AGHDAM, H.; HERAVI, E. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. [S.l.: s.n.], 2017. ISBN 978-3-319-57549-0.

BIBLIOTECA VIRTUAL. *Cursos de Libras*. 2019. Disponível em: <http://www.bibliotecavirtual.sp.gov.br/temas/pessoa-com-deficiencia/cursos-de-libras.php>. Acesso em: 13 mar. 2019.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0-387-31073-8.

BRASIL. *Lei nº 10.436, de 24 de abril de 2002. Dispõe sobre a Língua Brasileira de Sinais - Libras e dá outras providências*. 2002. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>. Acesso em: 11 mar. 2019.

BRASIL. *Apesar de avanços, surdos ainda enfrentam barreiras de acessibilidade*. 2011. Disponível em: <http://www.portalfns.saude.gov.br/slideshow/2251-dia-nacional-do-surdo-promove-luta-pela-inclusao-dos-surdos-na-sociedade>. Acesso em: 13 ago. 2019.

DIVE INTO DEEP LEARNING. *12.3. Object Detection and Bounding Boxes*. 2019. Disponível em: <https://d2l.ai/chapter_computer-vision/bounding-box.html>. Acesso em: 21 out. 2019.

GARCIA, B.; VIESCA, S. A. Real-time american sign language recognition with convolutional neural networks. 2016.

GEEKSFORGEEKS. *Erosion and Dilation of images using OpenCV in python*. 2018. Disponível em: <https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/>. Acesso em: 21 out. 2019.

HUANG, T. Computer vision: Evolution and promise. *19th CERN School of Computing*, p. 21–25, 1996.

JUNIOR, J. C. S. J. *Processamento de Imagens: fundamentos*. Pontifícia Universidade Católica do Rio Grande do Sul, 2018. Disponível em: <https://www.inf.pucrs.br/~smusse/Simulacao/PDFs/aula_02_Fundamentos_PI.pdf>. Acesso em: 23 nov. 2019.

KOZA, J. R.; BENNETT, F. H.; ANDRE, D.; KEANE, M. A. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In: GERO, J. S.; SUDWEEKS, F. (Ed.). *Artificial Intelligence in Design '96*. Dordrecht: Springer Netherlands, 1996. p. 151–170. ISBN 978-94-009-0279-4.

LACERDA, C. B. F. d. A inclusão escolar de alunos surdos: o que dizem alunos, professores e intérpretes sobre esta experiência. *Cadernos CEDES*, v. 26, n. 69, p. 163–184, 2006. ISSN 0101-3262.

LECUN, Y.; Y.BENGIO; HINTON, G. E. Deep learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 521, n. 7553, p. 436–444, 2015.

LODI, A. C. B.; BORTOLOTTI, E. C.; CAVALMORETI, M. J. A. Z. Letramentos de surdos: práticas sociais de linguagem entre duas línguas/culturas. *Bakhtiniana: Revista de Estudos do Discurso*, 12 2014. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2176-45732014000200009&nrm=iso>. Acesso em: 13 ago. 2019.

LUDWIG, J. *Image Convolution*. Portland State University, 2007. Disponível em: <http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf>. Acesso em: 20 out. 2019.

OPENCV. *Canny Edge Detection*. 2019. Disponível em: <https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html>. Acesso em: 23 nov. 2019.

PETERNELLI, L. A. *Regressão linear e correlação*. Universidade Federal de Viçosa, 2003. Disponível em: <http://www.dpi.ufv.br/~peternelli/inf162.www.16032004/materiais/CAPITULO9.pdf>. Acesso em: 23 nov. 2019.

PRATEEK, S.; JAGADEESH, J.; SIDDARTH, R.; SMITHA, Y.; HIREMATH, P. G. S.; PENDARI, N. T. Dynamic tool for american sign language finger spelling interpreter. In: *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. [S.l.: s.n.], 2018. p. 596–600.

SHAHRIAR, S.; SIDDIQUEE, A.; ISLAM, T.; GHOSH, A.; CHAKRABORTY, R.; KHAN, A. I.; SHAHNAZ, C.; FATTAH, S. A. Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning. In: *TENCON 2018 - 2018 IEEE Region 10 Conference*. [S.l.: s.n.], 2018. p. 1168–1171.

SOUZA, E. G. de. *Entendendo o que é Matriz de Confusão com Python*. Data Hackers, 2019. Disponível em: <https://medium.com/data-hackers/entendendo-o-que-é-matriz-de-confus~ao-com-python-114e683ec509>. Acesso em: 21 out. 2019.

TASKIRAN, M.; KILLIOGLU, M.; KAHRAMAN, N. A real-time system for recognition of american sign language by using deep learning. In: *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. [S.l.: s.n.], 2018. p. 1–5.

VYGOTSKY, L. S. *A construção do pensamento e da linguagem*. 1ª. ed. São Paulo: WMF Martins Fontes, 2001.