UNIVERSIDADE DE PASSO FUNDO

Reinaldo Borges Salla

SISTEMA PARA CONTROLE DE ACESSO EM AMBIENTES RESTRITOS UTILIZANDO RECONHECIMENTO FACIAL ATRAVÉS DE REDES NEURAIS

Passo Fundo

2019

Reinaldo Borges Salla

SISTEMA PARA CONTROLE DE ACESSO EM AMBIENTES RESTRITOS UTILIZANDO RECONHECIMENTO FACIAL ATRAVÉS DE REDES NEURAIS

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Adriano Luís Toazza.

Passo Fundo

2019

Reinaldo Borges Salla

Sistema para Controle de Acesso em Ambientes Restritos Utilizando Reconhecimento Facial Através de Redes Neurais

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Adriano Luís Toazza.

Aprovado em ____ de _____ de_____.

BANCA EXAMINADORA

_____

Prof. Dr. Orientador Adriano Luís Toazza - UPF

_____

Prof. Dr. Blanca Rosa Maquera Sosa - UPF

_____

Prof. Me. Joan Michel Levandoski - UPF

RESUMO

Avanços no campo de detecção e identificação facial efetivamente passam pela criação de modelos que permitam a extração automática de representações úteis em imagens. Neste contexto, este trabalho procurou explorar modelos de inteligência artificial, capazes de extrair milhões e até bilhões de representações em imagens contendo características faciais. O modelo desenvolvido neste trabalho consistiu em uma rede neural siamesa, formada por um par de redes neurais convolucionais idênticas utilizadas para calcular a diferença entre imagens. O modelo foi desenvolvido através da biblioteca TensorFlow, recentemente lançada pelo Google, e foi treinado com aproximadamente 700.000 imagens. Foi possível obter 84,78% de acurácia no conjunto de dados MUCT. Porém, como a aplicação específica para este trabalho consistiu em habilitar o acesso de usuários em um ambiente restrito, mediante registro prévio, para um sistema em tempo real, também procurou-se utilizar uma rede neural pré-treinada, que consistiu em uma implementação do modelo ResNet treinado com 3 milhões de imagens, disponível em uma biblioteca para a linguagem de programação Python denominada *Face Recognition*. Através deste método, foi possível obter 97,81% de acurácia no conjunto de dados MUCT. Portanto, foram apresentadas duas abordagens diferentes referentes à identificação facial, e utilizando-se do microcomputador Raspberry Pi, este projeto resultou na simulação do acesso de diferentes indivíduos em um ambiente controlado, através da pesquisa e do desenvolvimento de redes neurais com aplicações em visão computacional e reconhecimento facial.

Palavras-Chave: Inteligência Artificial, Rede Neural Siamesa, Redes Neurais Convolucionais, TensorFlow, ResNet, Python, Visão Computacional, Reconhecimento Facial.

ABSTRACT

Advances in the field of facial identification and facial detection effectively go through the creation of models that allow the automatic extraction of useful representations in images. In this context, this work intended to explore models of artificial intelligence, that promote the extraction of millions and even billions of representations in images containing facial characteristics. The model developed in this work consisted in a siamese neural network, formed by a pair of identical convolutional neural networks used to calculate the difference between images. The model was developed through the TensorFlow library, recently released by Google, and it was trained with approximately 700,000 images. It was possible to obtain 84.78% accuracy in the MUCT dataset. However, as the specific application for this work consisted in enabling the access of users in a restricted environment, upon previous registration, for a real time system, this projects also sought to use a pretrained neural network, which consisted in a implementation of the ResNet model, trained with 3 million images, available in a library for the Python programming language called Face Recognition. Through this method, it was possible to reach a 97.81% accuracy on the MUCT dataset. Thus, it was presented two different approaches for facial identification, and using the Raspberry Pi microcomputer, this project resulted in the simulating of the access of different individuals in a controlled environment, through the research and the development of neural networks with applications in computer vision and facial recognition.

Keywords: Artificial Intelligence, Siamese Neural Network, Convolutional Neural Networks, TensorFlow, ResNet, Python, Computer Vision, Facial Recognition.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diferenças entre aprendizagem de máquina e inteligência artificial simbólica 19
Figura 2 – Exemplos de retângulos de características no algoritmo Viola-Jones 22
Figura 3 – Exemplo do cálculo da imagem integral. 22
Figura 4 – Representação de um sistema Boosting Adaptativo 23
Figura 5 – Classificador em cascata implementado no algoritmo Viola-Jones 24
Figura 6 – Modelo de um perceptron 25
Figura 7 – Representação de uma rede neural MLP com duas camadas ocultas 27
Gráfico 1 – Função sigmoide 28
Gráfico 2 – Função ReLU 29
Figura 8 – Função objetiva de uma rede neural com uma camada oculta 33
Figura 9 – Filtros utilizados para extrair características de baixo nível 34
Figura 10 – Filtros utilizados para extrair características de alto nível 34
Figura 11 – Classificação implementada através de uma rede neural convolucional 35
Figura 12 – Convolução de um filtro sobre o conjunto de pixels de uma imagem 36
Figura 13 – Resultado da convolução de um filtro sobre uma imagem 37
Figura 14 – Preenchimento de zeros nas bordas de uma imagem 37
Figura 15 – Operação Max Pooling 38
Figura 16 – Rede neural siamesa formada por um par de redes neurais convolucionais 39
Figura 17 – Representação de um gráfico de fluxo de dados no TensoFlow 41
Figura 18 – Exemplos de algumas imagens utilizadas para treinar a rede neural 45
Figura 19 – Representação simplificada da rede neural 46
Figura 20 – Estrutura do Raspberry Pi 3 47
Quadro 1 – Custo financeiro de diferentes microcomputadores 48
Quadro 2 – Especificações técnicas do Raspberry Pi 3 48
Figura 21 – Módulo da câmera Raspberry Pi 49
Fotografia 1 – Demarcação da região facial em uma imagem 51
Quadro 3 – Dados relativos aos conjuntos de dados 53
Figura 22 – Ilustração da detecção facial e do processo *image augumentation* 54
Quadro 4 – Especificações das camadas para cada RNC 55
Figura 23 – Diagrama da rede neural siamesa 56
Gráfico 3 – Erro da rede neural 56
Figura 24 – *Template* utilizado pela biblioteca *Face Recognition* para alinhas imagens 58

Figura 25 – Diagrama de blocos geral do projeto 59
Figura 26 – Fluxograma para o controle de teclas implementado no software 59
Figura 27 – Fluxograma para detecção e identificação facial em tempo real 60
Fotografia 2 – Ilustração para a condição de carregamento do software 61
Fotografia 3 – Ilustração para a condição de “usuário desconhecido” do software 61
Fotografia 4 – Interface gráfica para registro de um novo usuário 62
Fotografia 5 – Mensagem utilizada durante o cadastro de um novo usuário 62
Fotografia 6 – Ilustração para a condição de “usuário cadastrado” no software 63
Quadro 5 – Primeiros registros implementados pelo software 64
Fotografia 7 – Comportamento do Software para mais de um indivíduo 64
Fotografia 8 – Ajuste gamma implementado em imagens com baixo nível de luminosidade 65
Fotografia 9 – Fechadura eletrônica utilizada no neste projeto 66
Figura 28 – Esquemático do circuito desenvolvido neste projeto 66
Fotografia 10 – Placa de circuito impressa desenvolvida neste projeto 67
Fotografia 11 – Circuito inserido em uma caixa e conexão com cabo rollover 68
Fotografia 12 – Interligação de todos os componentes do projeto 69
Quadro 6 – Resultados para identificação facial 72
Figura 29 – Resultados para pares positivos obtidos através da 1ª abordagem 73
Figura 30 – Resultados para pares negativos obtidos através da 1ª abordagem 73
Figura 31 – Resultados para pares positivos obtidos através da 2ª abordagem 74
Figura 32 – Resultados para pares negativos obtidos através da 2ª abordagem 74

LISTA DE SIGLAS

API – *Application Programming Interface*

CPU – *Central Processing Unit*

CSV – *Comma Separated Values*

DARPA – *Defense Advanced Research Projects Agency*

FERET – *Facial Recognition Technology*

FPS – *Frames per Second*

GPU – *Graphics Processing Unit*

HDF – *Hierarchical Data Format*

MLP – *Multilayer Perceptron*

MUCT – Milborrow University of Cape Town

RBG – *Red, Blue and Green*

ReLU – *Rectified Linear Unit*

RMS – *Root Mean Square*

RNA – Rede Neural Artificial

RNC – Rede Neural Convolucional

URL - *Uniform Resource Locator*

XML - *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 16

1.1 OBJETIVO GERAL 17

1.2 OBJETIVOS ESPECÍFICOS 18

1.3 JUSTIFICATIVA 18

2 REVISÃO DA LITERATURA 19

2.1 APRENDIZAGEM DE MÁQUINA 19

2.1.1 Fundamentação Teórica 20

2.1.1.1 Aprendizagem supervisionada 20

2.1.1.2 Aprendizagem não supervisionada 20

2.1.1.3 Aprendizagem reforçada 21

2.1.2 Algoritmo Viola Jones 21

2.1.2.1 Haar features 21

2.1.2.2 Image integral 21

2.1.2.3 Algoritmo de aprendizagem 23

2.1.3 *Deep Learning* 24

2.2 REDES NEURAIS ARTIFICIAIS 24

2.2.1 Perceptron Multicamadas 25

2.2.2 Função de ativação 26

2.2.2.1 Função de ativação Sigmoide 27

2.2.2.2 Função de ativação ReLU 28

2.2.3 Algoritmo de retropropagação 28

2.2.3.1 Gradiente descendente em lote 31

2.2.3.2 Gradiente descendente estocástico 31

2.2.3.3 Gradiente descendente estocástico em mini lote 32

2.2.4 Redes Neurais Convolucionais 32

2.2.4.1 Benefícios *32*

2.2.4.2 Filtros em redes neurais convolucionais *33*

2.2.4.3 Camada de convolução *34*

2.2.4.4 Preenchimento (Padding) *37*

2.2.4.5 Camada de Subamostragem (Pooling) *38*

2.2.5 Redes Neurais Siamesas **38**

3 MATERIAIS E MÉTODOS **40**

3.1 SOFTWARE 40

3.1.1 Python **40**

3.1.2 TensorFlow **40**

3.1.3 Keras **42**

3.1.4 SciPy **42**

3.1.4.1 Numpy *42*

3.1.4.2 Matplotlib *42*

3.1.5 Colaboratory **43**

3.1.6 Conjuntos de dados **43**

3.1.7 Dlib **45**

3.1.8 Face Recognition **45**

3.1.9 OpenCV **46**

3.2 HARDWARE 47

3.2.1 Raspberry Pi **47**

3.2.2 Câmera **49**

4 DESENVOLVIMENTO **50**

4.1 DETECÇÃO FACIAL ATRAVÉS DO ALGORITMO VIOLA-JONES 50

4.2 IDENTIFICAÇÃO FACIAL: 1ª ABORDAGEM 50

4.2.1 Pré-processamento das imagens **50**

4.2.2 Desenvolvimento da rede neural **52**

4.3 IDENTIFICAÇÃO FACIAL: 2ª ABORDAGEM 57
4.4 ESTRUTURA FUNCIONAL DO PROJETO 58
4.5 FECHADURA ELETRÔNICA E HARDWARE 65
5 RESULTADOS E DISCUSSÃO 70
5.1 RESULTADOS PARA DETECÇÃO FACIAL 70
5.1.1 Velocidade de processamento 70
5.1.2 Detecção com relação à variação de luminosidade 70
5.1.3 Detecção de fraudes 71
5.2 RESULTADOS PARA IDENTIFICAÇÃO FACIAL 71
6 CONSIDERAÇÕES FINAIS 75
REFERÊNCIAS 77

1 INTRODUÇÃO

Tecnologias envolvendo reconhecimento facial foram inicialmente desenvolvidas por Woody Bledsoe, Charles Bisson e Helen Chan Wolf na década de 1960. Os sistemas implementados por estes pesquisadores promoviam a extração manual de características faciais. As principais dificuldades encontradas por estes pesquisadores consistiam em variações na luminosidade e variações espaciais nas representações dos pixels em imagens. Porém, mesmo com a limitação dos dispositivos eletrônicos da época, o sistema desenvolvido por Bledsoe e seus colegas constituiu um marco importante para o início de pesquisas na área de detecção e identificação facial (WEST, 2017).

Avanços significativos com relação à construção de métodos aplicados na extração de características faciais foram efetuados pelo cientista Takeo Kanade, na década de 1970 (KANADE, 1973). A ideias apresentadas por Kanade constituíam na construção de um sistema que permitisse extrair distâncias e ângulos de diferentes pontos em uma face. Sistemas que se baseiam apenas na extração de características geométricas são considerados modelos pobres atualmente, pois descartam informações importantes em representações faciais (CHIACHIA, 2013).

Posteriormente, nas décadas de 1980 e 1990, diversas entidades trabalharam para promover avanços na área de reconhecimento facial. Exemplificativamente, a organização militar DARPA (*Defense Advanced Research Projects Agency*), juntamente com outros órgãos governamentais dos Estados Unidos, permitiram o desenvolvimento do programa FERET (*Facial Recognition Technology*), que possuía o objetivo de criar um amplo conjunto de dados com aplicação em reconhecimento de faces (PHILLIPS, 1998). O conjunto de dados formado possuía mais de 14.000 imagens de aproximadamente 2000 usuários diferentes.

Na década de 2000, o algoritmo Viola-Jones foi essencial na área de detecção facial. Este algoritmo utilizou métodos de aprendizagem de máquina e se destacou pelo seu rápido processamento de imagens. Foi desenvolvido para implementar a detecção de diferentes tipos de objetos, mas possuía inspiração em problemas relacionados com detecção facial (VIOLA; JONES, 2001).

Paralelamente, em 2005, cientistas da Universidade de Nova Iorque apresentavam a ideia de implementar verificação facial através de métodos que permitiam calcular a similaridade entre representações extraídas de diferentes imagens (CHOPRA; HADSELL; LECUN, 2005). O modelo utilizado para extrair representações de imagens constituía em uma Rede Neural Convolucional (RNC). Este modelo permite aplicar um conjunto de filtros

adaptativos sobre uma imagem, modelando as informações extraídas como uma sequência de características invariantes com relação a distorções geométricas.

Assim, com o recente crescimento da pesquisa e da produção no campo de inteligência artificial (OCTAVIANO, 2017), redes neurais vêm se tornando uma tendência no desenvolvimento de sistemas com aplicações em reconhecimento facial. Por exemplo, o sistema desenvolvido pelo Google, denominado FaceNet, promove a utilização de redes neurais profundas (SCHOROFF, KALENICHENKO, PHILBIN, 2015), assim como o sistema desenvolvido pelo Facebook, (TAIGMAN et al., 2014) e o sistema desenvolvido pela Apple (RAJGOPAL, 2018).

A utilização de redes neurais no campo do reconhecimento facial ajudou a aumentar a utilização desta tecnologia em escala mundial. Por exemplo, sistemas de vigilância em massa estão sendo utilizados na China através de tecnologias envolvendo reconhecimento facial (SHIMIDT, 2019). Além disso, a empresa Baidu, em 2015, desenvolveu um sistema de detecção e identificação facial, utilizando redes neurais convolucionais (LIU et al., 2015), e substituiu a utilização de cartões magnéticos para habilitar a entrada de funcionários em suas dependências. Outro exemplo a ser considerado é a tecnologia desenvolvida pela empresa americana Mastercard, que passou a utilizar o reconhecimento facial para autorizar transações bancárias (BETH TECHNOLOGY, 2019).

Além disso, a promoção desta tecnologia também vem crescendo em âmbito nacional. Por exemplo, em São Paulo, o reconhecimento facial passou a ser utilizado para comprovar a identidade de condutores de veículos (FOLHA DE S.PAULO, 2019). No Rio de Janeiro, passará a ser utilizado em aeroportos e em estádios de futebol (GUIMARÃES, 2019).

1.1 OBJETIVO GERAL

O objetivo deste projeto consistiu em desenvolver um protótipo capaz de simular a entrada de usuários em um ambiente restrito, através do reconhecimento facial. O sistema deveria fazer a diferenciação entre usuários cadastrados e não cadastrados para habilitar a entrada no ambiente.

É importante ressaltar que o objetivo deste trabalho não foi efetuar uma ampla pesquisa sobre métodos tradicionais de identificação e detecção facial. O objetivo central consistiu em pesquisar e implementar modelos de inteligência artificial e redes neurais direcionadas ao reconhecimento facial. Desta forma, fazendo o uso de um conjunto apropriado de hardware e

câmera, a rede neural deveria trabalhar com as imagens extraídas do conjunto de frames gerado pelo sistema, operando em tempo real.

1.2 OBJETIVOS ESPECÍFICOS

a) Desenvolver uma rede neural para efetuar identificação facial através de imagens;

b) Implementar métricas para testar a acurácia da rede neural;

c) Desenvolver um algoritmo para implementar detecção facial em tempo real;

d) Desenvolver um algoritmo para efetuar o cadastro de usuários no sistema;

e) Desenvolver um sistema para acionar uma fechadura eletrônica, e simular a entrada em um ambiente controlado;

1.3 JUSTIFICATIVA

O desenvolvimento de novos sistemas com aplicações na identificação e verificação de usuários é imprescindível no atual contexto social. Assim, em comparação, por exemplo, com a utilização de métodos que fazem a requisição da digital, o desenvolvimento de um dispositivo de controle de acesso através do reconhecimento facial visa uma maior confortabilidade, pois não é necessário nenhum contato físico entre o usuário e o dispositivo. Além disso, em comparação com a utilização de cartões magnéticos, tecnologias envolvendo o reconhecimento facial também possibilitam promover uma maior segurança, pois o cartão magnético consiste em um material que pode ser desaproveitado pelo detentor do cartão.

A justificativa específica para a utilização de inteligência artificial e redes neurais orientadas ao reconhecimento facial consiste na capacidade destes modelos detectarem padrões relevantes nas características dos pixels de imagens, e, consequentemente, formarem sistemas com uma capacidade de desconsiderar padrões irrelevantes, como, por exemplo, a iluminação e a posição espacial da representação de uma face em uma imagem.

2 REVISÃO DA LITERATURA

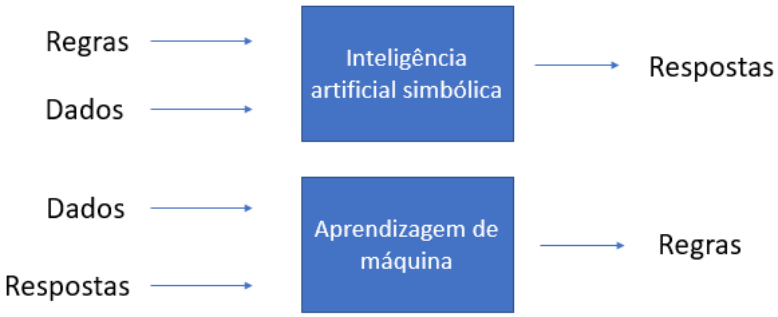
Este capítulo contém os conceitos teóricos necessários para o desenvolvimento deste trabalho. Em primeiro plano, serão exploradas as teorias relativas à aprendizagem de máquina. Este projeto implementará apenas a utilização de aprendizagem supervisionada, porém procura-se também definir, de forma resumida, aprendizagem não supervisionada e aprendizagem reforçada, pois estes algoritmos estão diretamente relacionados com aprendizagem supervisionada e possuem um alto potencial tecnológico no futuro (LECUN; BENGIO; HINTON, 2015; SUTTON; BARTO, 2018). Também é feita uma abordagem referente ao algoritmo Viola-Jones, que foi utilizado para implementar detecção facial neste projeto. Posteriormente é feita uma fundamentação teórica na área de redes neurais, com um destaque às RNCs.

2.1 APRENDIZAGEM DE MÁQUINA

Pode ser considerada uma ciência que propicia a automação de tarefas intelectuais, através de ações orientadas à processamento de dados, efetuadas por máquinas ou computadores (GOODFELLOW; BENGIO; COURVILLE, 2016).

Entretanto, ao contrário de inteligência artificial simbólica, que depende de regras construtivas previamente programadas, aprendizagem de máquina permite desenvolver sistemas que extraem as regras ou as características de dados a partir de exemplos (CHOLLET, 2017), conforme ilustrado na Figura 1.

Figura 1 – Diferenças entre aprendizagem de máquina e inteligência artificial simbólica



Fonte: Adaptado de Chollet (2017)

2.1.1 Fundamentação Teórica

Algoritmos que utilizam aprendizagem de máquina podem ser divididos em três categorias: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem reforçada (GÉRON, 2017), descritas a seguir.

2.1.1.1 Aprendizagem supervisionada

Consiste em aplicar um treinamento ou um reconhecimento de padrões em um sistema que possui as soluções (saídas) desejadas. Estes modelos também são chamados de aprendizagem com um "professor" (HAYKIN, 2001). Assim, o professor é responsável por fornecer os exemplos ao sistema.

Para cada dado de treinamento é associado um rótulo[1]. Desta forma, estes algoritmos permitem mapear um conjunto de entradas para um conjunto de saídas, utilizando como referência os rótulos associados à cada dado de entrada (GOODFELLOW; BENGIO; COURVILLE, 2016). Aprendizagem supervisionada consiste na forma mais comum de aprendizagem de máquina (LECUN; BENGIO; HINTON, 2015).

2.1.1.2 Aprendizagem não supervisionada

Consistem em algoritmos que não possuem um "professor". Usualmente são aplicados para codificar faixas de áudios ou conjuntos de vídeos, que posteriormente são passados para algoritmos de aprendizagem supervisionada (SHUMIDHUBER, 2015). Entretanto, é importante salientar que, mesmo quando os dados são inseridos automaticamente, o sistema ainda é considerado como um algoritmo de aprendizagem supervisionada (GOODFELLOW; BENGIO; COURVILLE, 2016), pois o "professor" pode ser um ser humano ou uma máquina (sistema de inteligência artificial) (GÉRON, 2017). Portanto, somente deve ser considerado um algoritmo não supervisionado quando as entradas não possuem nenhum tipo de rótulo ou pré-processamento capaz de dividir os dados de treinamento. Assim, o sistema deve reputar ou analisar os dados automaticamente (GOODFELLOW; BENGIO; COURVILLE, 2016).

---

[1] Os rótulos representam as saídas desejadas do sistema. Por exemplo, para um sistema de classificação de caracteres, um rótulo pode ser considerado o texto correspondente à um caracter. Em um sistema utilizado para prever valores de ações na bolsa de valores, os valores dos rótulos representariam os valores de uma ação em um determinado período de tempo.

2.1.1.3 Aprendizagem reforçada

Consistem em algoritmos que diferem de forma drástica de aprendizagem supervisionada ou aprendizagem não supervisionada (GÉRON, 2017). São formados por sistemas direcionados à objetivos, a partir de ações de interação (SUTTON; BARTO, 2018). O objeto de aprendizagem, denominado *agente*, interage com ambiente, recebendo recompensas ou penalidades de acordo com a ações tomadas. Desta forma, conforme o número de interações aumenta, o agente cria uma *política*, que possui o objetivo de reduzir a obtenção de penalidades e maximizar o recebimento de recompensas (SHUMIDHUBER, 2015).

2.1.2 Algoritmo Viola Jones

Desenvolvido por Paul Viola e Michael Jones em 2001, utiliza métodos de aprendizagem de máquina para detectar objetos em tempo real de forma eficiente (VIOLA; JONES, 2001). Este algoritmo foi utilizado neste projeto para detectar faces e, dessa forma, foi usado para auxiliar a identificação facial (foi utilizado para garantir que a rede neural implemente a comparação de imagens contendo apenas representações faciais).

2.1.2.1 Haar features

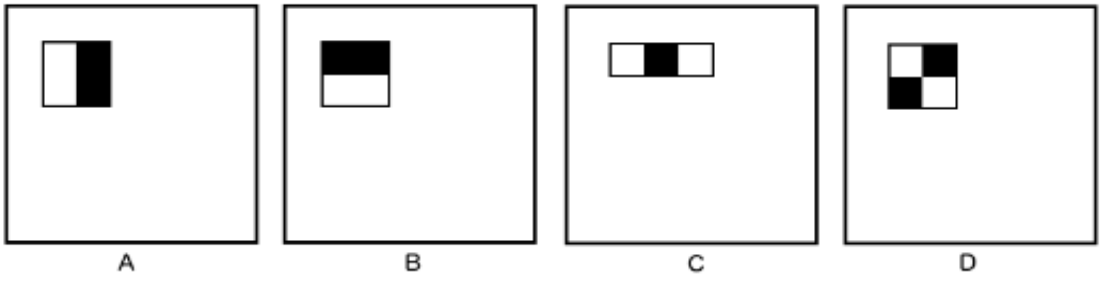
A técnica apresentada por Viola e Jones opera sobre características extraídas dos pixels, denominadas *HAAR features*, implementadas através de características retangulares. Estas características não promovem a extração de representações complexas de imagens (texturas, formas, contornos). Entretanto, possibilitam extrair representações simples (características relacionadas com a intensidade luminosa). É possível efetuar uma separação entre três tipos de características distintas (PRADO, 2018).

A primeira característica é formada por dois retângulos e permite computar a diferença entre a soma dos pixels contidos nas duas regiões retangulares (Figuras 2 A e 2 B). A segunda característica é formada por três retângulos e implementa o cálculo da soma dos pixels dos retângulos exteriores e subtrai o resultado com o valor dos pixels do retângulo interior (Figura 2 C). A terceira característica é formada por quatro retângulos e possibilita computar a subtração dos pares diagonais das regiões retangulares (Figura 2 D).

2.1.2.2 Image integral

Para evitar a operação de um número elevado de características sobre a imagem, o algoritmo Viola-Jones apresentou o conceito de imagem integral. Assim, implementa-se o uso de uma representação intermediária da imagem original (VIOLA; JONES, 2001), utilizada para

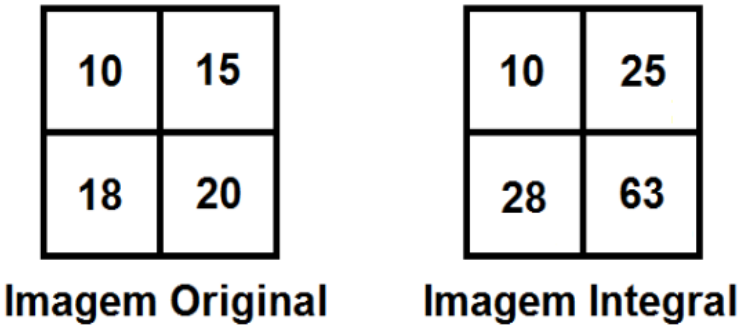
Figura 2 – Exemplos de retângulos de características no algoritmo Viola-Jones.



Fonte: Prado (2018)

elevar a velocidade de processamento do algoritmo durante a sua execução. A imagem integral pode ser calculada somando-se o valor atual de um pixel com os valores dos pixels acima e à esquerda, conforme a Figura 3. O cálculo para a determinação da imagem integral está descrito na Equação (1) (VIOLA; JONES, 2001).

Figura 3 – Exemplo do cálculo da imagem integral.



Fonte: Adaptado de Prado (2017).

$$II(m,n) = \sum_{m'=0}^{m' \leq m} \sum_{n'=0}^{n' \leq n} I(m',n') \tag{1}$$

Onde:

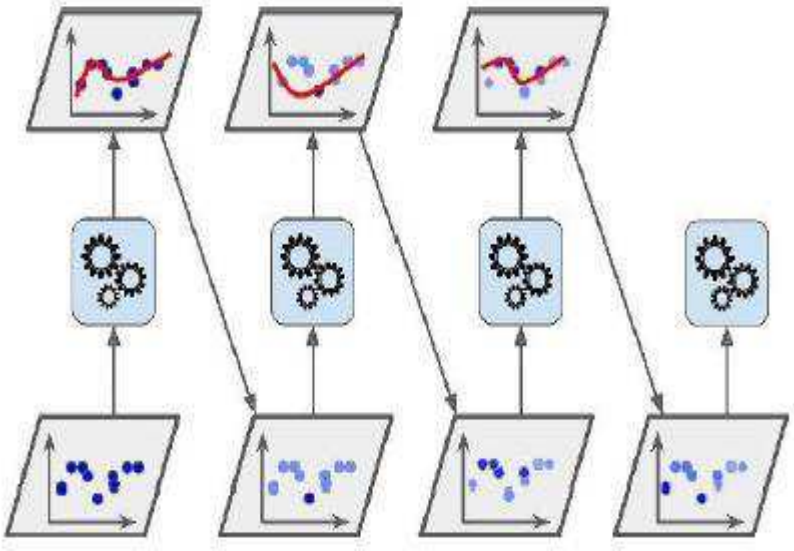
$II(m,n)$: representa um pixel da imagem integral nas posições (m,n)

$I(m',n')$: representa um pixel da imagem original nas posições (m',n')

2.1.2.3 Algoritmo de aprendizagem

A técnica utilizada para treinar o algoritmo Viola-Jones consistiu em um algoritmo de aprendizagem de máquina denominado Boosting Adaptativo. Este algoritmo consiste em um conjunto de classificadores, no qual cada classificador influencia e corrige o próximo (GÉRON, 2017). Assim, conforme a Figura 4, cada ajuste efetuado em um classificador permite elevar a acurácia do modelo. Logo, é possível verificar que a forma como o sistema determina as suas saídas a partir de suas entradas é adaptada e corrigida durante o treinamento.

Figura 4 – Representação de um sistema Boosting Adaptativo

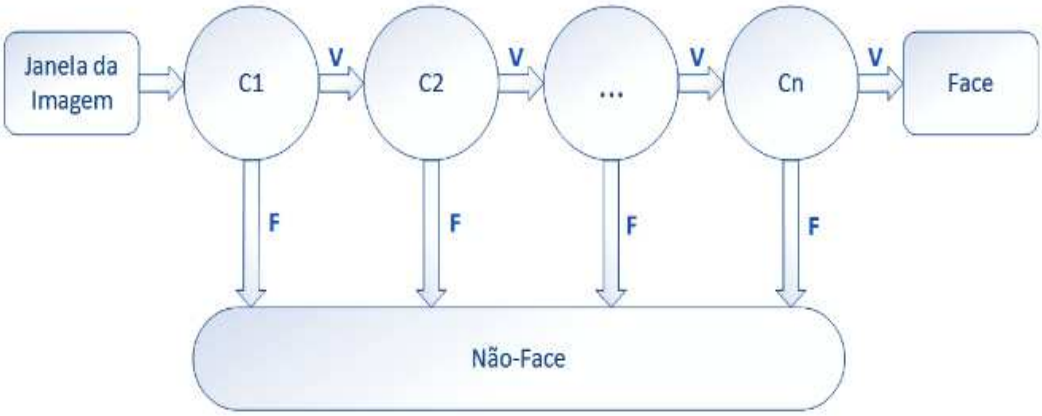


Fonte: Géron (2017)

Portanto, Boosting Adaptativo é utilizado para treinar um conjunto de classificadores em cascata no algoritmo Viola-Jones. Em primeiro plano, são selecionadas janelas com 24 linhas e 24 colunas na imagem. Janela positiva é considerada uma janela que possui representações faciais e janela negativa é considerada uma janela que não possui representações faciais. A seleção de uma janela da imagem só é feita se a detecção por todos os classificadores for considerada positiva (PRADO, 2018).

Assim, conforme pode ser verificado na Figura 5, a janela da imagem consiste na entrada do primeiro classificador (C1). Caso seja feita a detecção de uma imagem positiva, o segundo classificador (C2) é acionado. O processo é repetido até o último classificador (Cn). Caso algum classificador apresente como resultado uma imagem negativa, a janela é desconsiderada. O algoritmo Viola-Jones trabalha apenas com imagens monocromáticas e foi treinado com 4916 imagens contendo faces e 9544 imagens não contendo faces (VIOLA; JONES, 2001).

Figura 5 – Classificador em cascata implementado no algoritmo Viola-Jones



Fonte: Prado (2018)

2.1.3 *Deep Learning*

Deep learning consiste em uma técnica amplamente utilizada atualmente em algoritmos de aprendizagem de máquina (SHUMIDHUBER, 2015). Permite computar representações de dados e possui uma ênfase em aprender sucessivas camadas com representações cada vez mais significativas (CHOLLET, 2017). Redes neurais profundas são os principais sistemas utilizados em *deep learning* (SHUMIDHUBER, 2015), pois são formados por uma sucessão de camadas hierárquicas capazes de extrair informações e representações de dados.

2.2 REDES NEURAIS ARTIFICIAIS

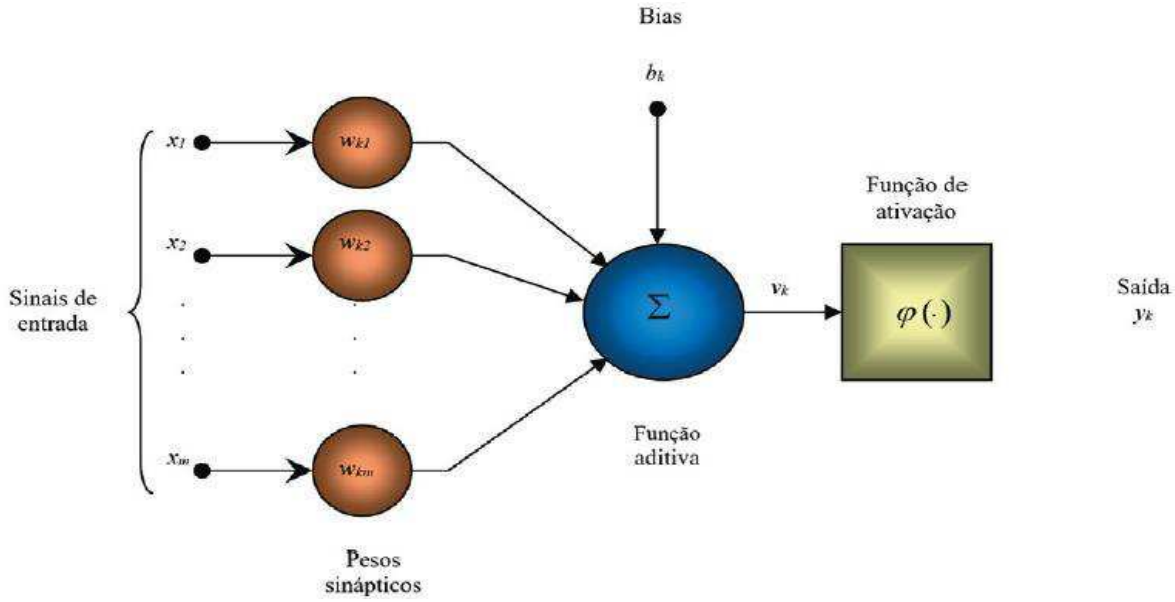
Redes Neurais Artificiais (RNAs) são modelos matemáticos que possuem aplicações em diversos cenários acadêmicos e comerciais atualmente (LECUN; BENGIO; HINTON, 2015). Algumas das áreas nas quais uma RNA se destaca incluem: classificação de dados, modelos de regressão, reconhecimento de padrões, controle de sistemas de automação e robótica, modelos de aprendizagem reforçada, linguagem de processamento natural, conversão de fala em texto, reconhecimento de fala, visão computacional e geração de dados. Estes sistemas possuem uma inspiração biológica, assumindo que "o cérebro é um *computador (*sistema de processamento de informações) altamente *complexo, não linear e paralelo".* (HAYKIN, 2001, p.27).

2.2.1 Perceptron Multicamadas

São RNAs estruturadas em uma sequência de camadas formadas por perceptrons. O modelo de um perceptron foi apresentado inicialmente por Frank Rosenblatt na década de 1950 e possuía o objetivo de ilustrar as propriedades fundamentais de sistemas inteligentes (ROSENBLATT, 1958). Portanto, os percetrons, também chamados de neurônios artificiais ou unidades processadoras, representam os elementos estruturais básicos de uma rede neural MLP (*Multilayer Perceptron*).

Cada unidade recebe um somatório dos valores das unidades anteriores multiplicados pelas respectivas forças de conexão (pesos sinápticos) entre os neurônios. O resultado é somado ou subtraído com um sinal de polarização, chamado de *bias*. Também é utilizada uma função matemática, usualmente não linear, chamada de função de ativação. O processo pode ser visualizado na Figura 6. A entrada "v_k" da função de ativação representa a soma ponderada dos estímulos de entrada.

Figura 6 – Modelo de um perceptron



Fonte: HAYKIN (2001)

Os pesos são adaptados durante o treinamento e, dessa forma, possibilitam parametrizar as transformações efetuadas pela rede neural (CHOLLET, 2017). O *bias* promove um *offset* e, portanto, possui "o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente". (HAYKIN, 2010, p.37). Em uma rede MLP, os dados fluem da entrada para a saída, passando por todos as unidades. A Equação (2) permite representar a dinâmica matemática de um único perceptron (HAYKIN, 2001).

$$y_k = \varphi\left(\sum_{j=1}^{m}(w_{kj}\, x_j) + \ b_k\right) \tag{2}$$

Onde:

y_k: saída do neurônio k

$\varphi(\cdot)$: função de ativação

m: número de sinapses

w_{kj}: peso associado ao neurônio k pela sinapse j

x_j: sinal na entrada da sinapse j

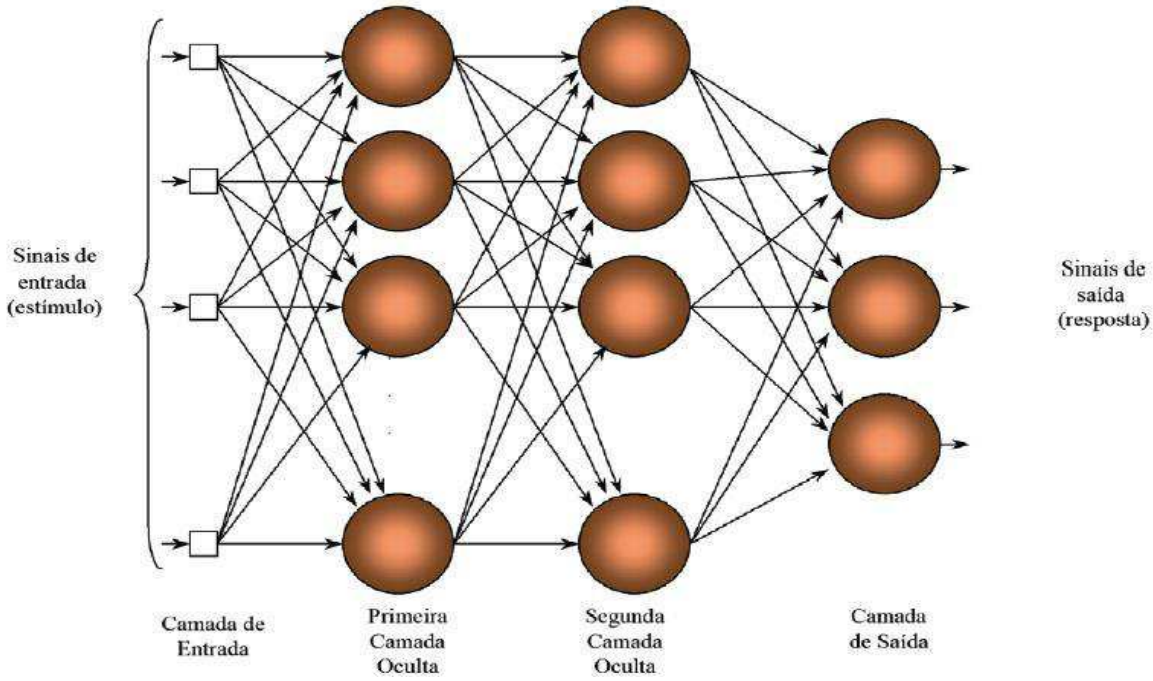
b_k: sinal de polarização do neurônio k

É possível visualizar uma rede neural MLP com duas camadas ocultas na Figura 7. Este tipo de modelo também é chamado de rede neural densa ou rede neural totalmente conectada. A camada de entrada é responsável por receber os dados de entrada. Por exemplo, se o dado de entrada for uma imagem, a entrada da rede neural recebe os valores dos pixels da imagem (CHOLLET, 2017), e se o dado de entrada for um sinal sonoro, representa a amplitude do sinal para um determinado período de tempo. A camada de saída é utilizada para fornecer as respostas do sistema (LECUN; BENGIO; HINTON, 2015). A diferença entre redes neurais rasas e redes neurais profundas é subjetiva (TABACOF, 2017). Entretanto, é importante salientar que modernos algoritmos de aprendizagem profunda utilizam redes neurais com múltiplos estágios (SHUMIDHUBER, 2015) e podem envolver dezenas ou centenas de camadas. (CHOLLET, 2017).

2.2.2 Função de ativação

Possibilita determinar um estado de ativação para um determinado neurônio (AMINI, 2019). Portanto, indica uma faixa de espaço de ativação de uma unidade em uma RNA. Além disso, também possibilita introduzir não-linearidade ao sistema. Duas funções amplamente difundidas atualmente consistem na função sigmoide e na função ReLU (*Rectified Linear Unit*) (LECUN; BENGIO; HINTON, 2015), descritas a seguir.

Figura 7 – Representação de uma rede neural MLP com duas camadas ocultas.



Fonte: HAYKIN (2001)

2.2.2.1 Função de ativação Sigmoide

A utilização desta função de ativação é promovida em problemas envolvendo uma previsão de probabilidades, principalmente em classificações ou distribuições de probabilidades binárias (NWANKPA et al., 2018). A função de ativação do tipo sigmoide está relacionada com o conceito de regressão logística, pois fornece uma lógica (probabilidade) de ocorrer um determinado resultado. Apresenta como resposta valores que variam da faixa de 0 até 1. Pode ser modelada pela função sigmoide ou função logística, representada pela curva sigmoide. A Equação (3) exemplifica a dinâmica matemática de uma função de ativação sigmoide. A curva sigmoide pode ser verificada no Gráfico 1.

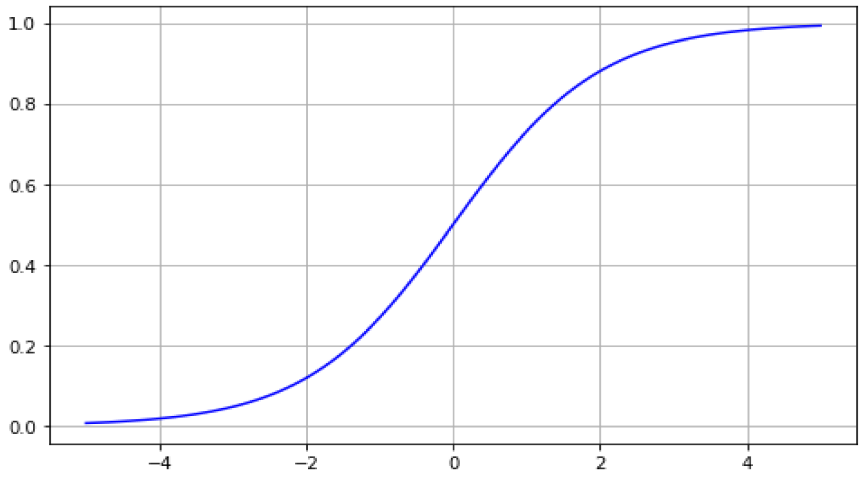
$$y_k = \frac{1}{1 + e^{-v_k}} \tag{3}$$

Onde:

y_k: saída da função de ativação

v_k: entrada da função de ativação

Gráfico 1 – Função sigmoide



Fonte: Próprio Autor

2.2.2.2 Função de ativação ReLU

Os benefícios da função de ativação ReLU foram apresentados inicialmente em 2010 pelos pesquisadores Geoffrey Hinton e Vinod Nair na Universidade de Toronto em suas pesquisas envolvendo máquinas de Boltzmann restritas (modelos generativos que possuem duas camadas e podem ser utilizados em aprendizagem não supervisionada) (NAIR; HINTON, 2010).

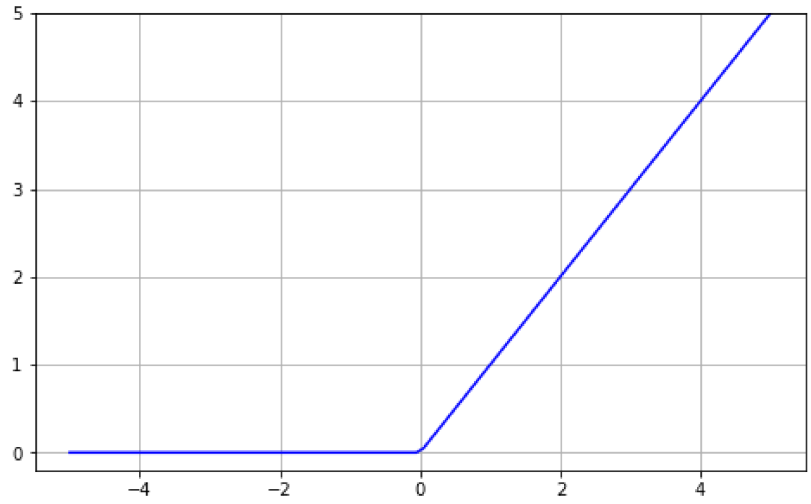
O principal benefício da utilização da função ReLU em redes neurais consiste na aceleração da convergência do modelo. (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Assim, permite anular valores negativos ou nulos e não alteram valores positivos, conforme a Equação (4). É possível verificar o comportamento da função ReLU no Gráfico 2.

$$y_k = \begin{cases} 0 & Se\ v_k \leq 0 \\ v_k & Se\ v_k \geq 0 \end{cases} \tag{4}$$

2.2.3 Algoritmo de retropropagação

Desenvolvido inicialmente nas décadas 1970 e 1980, foi essencial para a expansão de pesquisas relacionadas com RNAs. Este algoritmo permitiu, no âmbito de pesquisas relacionadas com inteligência artificial, implementar processos de treinamento em redes com múltiplas camadas (LECUN; BENGIO; HINTON, 2015). Em suma, é utilizado para adaptar os valores dos pesos de uma RNA, usando como base a redução do erro obtido sequencialmente pelo modelo.

Gráfico 2 – Função ReLU



Fonte: Próprio Autor

É possível quantificar o erro para problemas envolvendo regressão (previsão de valores) através da equação RMS (*Root Mean Square*), efetuando o somatório dos erros quadráticos entre o valor desejado e o valor obtido, conforme a Equação (5). Por outro lado, é possível calcular o erro através da entropia cruzada para problemas envolvendo uma classificação, conforme a Equação (6) (AMINI, 2019).

$$\boldsymbol{J(W)} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - f\left(x^{(i)}; \boldsymbol{W}\right)\right)^2 \tag{5}$$

$$\boldsymbol{J(W)} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(f\left(x^{(i)}; \boldsymbol{W}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - f\left(x^{(i)}; \boldsymbol{W}\right)\right)\right) \tag{6}$$

Onde:

$\boldsymbol{J(W)}$: função de custo, parametrizada pelo conjunto dos pesos $\boldsymbol{W}$, representados como uma matriz para cada camada

m: quantidade de amostras utilizadas para computar o erro

$y^{(i)}$: valor esperado da rede neural

$f\left(x^{(i)}; \boldsymbol{W}\right)$: valor obtido pela rede neural através do conjunto dos pesos $\boldsymbol{W}$

O modelo matemático utilizado para atualizar os pesos é denominado gradiente descendente, e permite diminuir a função de erro (também chamada de função de custo), parametrizada pelos pesos do modelo, atualizando os pesos na direção contrária do gradiente da função com relação aos pesos (RUDER, 2016). Desta forma, o gradiente é computado para o erro, obtido na camada de saída, com relação aos parâmetros da rede neural.

Os pesos são inicializados de forma aleatória (TENSORFLOW GUIDE, 2019). Desta forma, através do gradiente descendente, é possível adaptar os pesos na direção contrária da máxima ascensão da função de custo, ou seja, na direção contrária do gradiente da função, que pode ser representado pela derivada parcial da função com relação aos pesos, conforme a Equação (7) (RUDER, 2016).

$$\boldsymbol{W}_{(n+1)} = \boldsymbol{W}_{(n)} - \eta \frac{\partial J(\boldsymbol{W}_{(n)})}{\partial \boldsymbol{W}_{(n)}} \tag{7}$$

Onde:

$\boldsymbol{W}_{(n+1)}$: conjunto atualizado dos pesos

$\boldsymbol{W}_{(n)}$: conjunto antigo dos pesos

η: taxa de aprendizagem ($0 < \eta < 1$), cujo valor permite determinar a velocidade pela qual a rede neural deverá convergir para o ponto mínimo

$\frac{\partial J(\boldsymbol{W}_{(n)})}{\partial \boldsymbol{W}_{(n)}}$: máxima ascensão da função objetiva com relação ao conjunto dos pesos

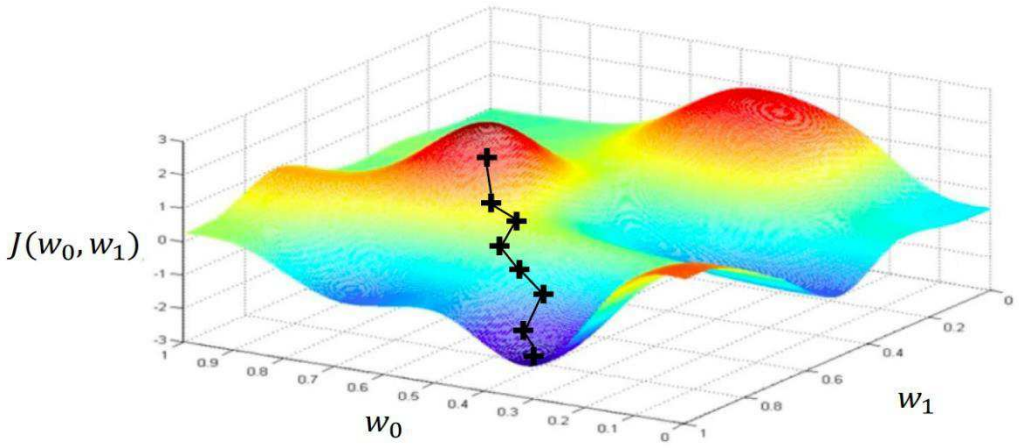
Assim, assumindo que uma rede neural é constituída por um conjunto sequencial de camadas, o algoritmo de retropropagação pode ser considerado uma aplicação prática da regra em cadeia (*chain rule*) para derivadas na função objetiva (AMINI, 2019).

Desta forma, os gradientes são propagados por todos as unidades, na direção da camada de saída para a camada de entrada (LECUN; BENGIO; HINTON, 2015). O treinamento é efetuado em múltiplas etapas ou épocas (*epochs*) até se obter o menor valor possível para a função objetiva (TENSORFLOW GUIDE, 2019).

Conforme ilustrado na Figura 8, a função objetiva "$J(w_0, w_1)$" para uma rede neural com uma camada oculta é parametrizada pelo conjunto de pesos "w_0" entre a camada de entrada e a camada oculta. Além disso, a função objetiva também é parametrizada pelo conjunto de pesos "w_1" entre a camada oculta e a camada de saída. Os pontos em destaque indicam os diferentes pesos obtidos através do gradiente descendente.

O gradiente descendente pode ser dividido em três classes distintas: gradiente descendente em lote, gradiente descendente estocástico e gradiente descendente estocástico em mini lote (RUDER, 2016), descritos a seguir.

Figura 8 – Função objetiva de uma rede neural com uma camada oculta



Fonte: Amini (2019).

2.2.3.1 Gradiente descendente em lote

Possibilita computar o gradiente da função de erro para o conjunto de treinamento completo. Em comparação com os outros métodos, pode ser considerado lento, pois só é possível calcular o gradiente e atualizar os pesos para o conjunto completo (em apenas um lote).

2.2.3.2 Gradiente descendente estocástico

Permite atualizar os pesos para cada dado de treinamento. Portanto, cada dado interfere no cálculo do gradiente da função objetiva, conforme a Equação (8) (RUDER, 2016).

$$\boldsymbol{W}_{(n+1)} = \boldsymbol{W}_{(n)} - \eta \frac{\partial J\left(\boldsymbol{W}_{(n)}; x^{(t)}; y^{(t)}\right)}{\partial \boldsymbol{W}_{(n)}} \tag{8}$$

Onde:

$\frac{\partial J\left(\boldsymbol{W}_{(n)}; x^{(t)}; y^{(t)}\right)}{\partial \boldsymbol{W}_{(n)}}$: derivada parcial da função objetiva com relação aos pesos, para cada dado de treinamento $x^{(t)}$. O parâmetro $y^{(t)}$ representa o rótulo do dado.

2.2.3.3 Gradiente descendente estocástico em mini lote

Consiste em atualizar os pesos para um conjunto de dados de treinamento, considerando uma quantidade de amostras igualmente definidas, conforme e Equação (9) (RUDER, 2016). Este método é adequado para redes neurais complexas com uma grande quantidade de camadas (por exemplo, mais de 5 camadas), pois permite gerar modelos estáveis com uma rápida convergência (RUDER, 2016).

$$\boldsymbol{W}_{(n+1)} = \boldsymbol{W}_{(n)} - \eta \frac{\partial J\left(\boldsymbol{W}_{(n)}; x^{(i:i+m)}; y^{(i:i+m)}\right)}{\partial \boldsymbol{W}_{(n)}} \quad (9)$$

Onde:

$\frac{\partial J\left(\boldsymbol{W}_{(n)}; x^{(i:i+m)}; y^{(i:i+m)}\right)}{\partial \boldsymbol{W}_{(n)}}$: derivada parcial da função objetiva com relação aos pesos para cado lote m.

2.2.4 Redes Neurais Convolucionais

A primeira implementação prática de redes neurais em um sistema de larga escala foi efetuada através de RNCs no final da década de 1980 (CHOLLET, 2017). Naquela época, este sistema foi utilizado pelo serviço postal dos Estados Unidos para reconhecer códigos numéricos.

Entretando, RNCs passaram a ser amplamente utilizadas pela comunidade científica apenas a partir de 2012, devido aos resultados obtidos por Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton na competição ImageNet. O modelo construído por Krizhevsky e seus colegas permitiu quebrar recordes na competição, possuindo um aumento percentual relativo de acurácia em mais de 10% com relação ao modelo vencedor na edição anterior (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Desde então, RNCs se tornaram o modelo mais comum utilizado em competições na área de visão computacional (CHOLLET, 2017). Além de imagens, RNCs também possuem aplicações envolvendo sequencias temporais (LECUN; BENGIO, 1998).

2.2.4.1 Benefícios

A utilização de RNCs no processamento de imagens é promovida pela sua capacidade de aprender padrões locais em um espaço multidimensional. Desta forma, este tipo de arquitetura

possui a característica de invariação espacial ou invariação translacional, ou seja, possibilita extrair representações de uma imagem independentes da posição espacial dos pixels da representação em questão (CHOLLET, 2017).

Outro benefício consiste na redução do número de parâmetros de uma rede neural. Por exemplo, a inserção de uma imagem em uma rede neural densa envolve a geração de uma grande quantidade de parâmetros (por exemplo, centenas de bilhões de parâmetros), tornando o modelo demasiadamente complexo.

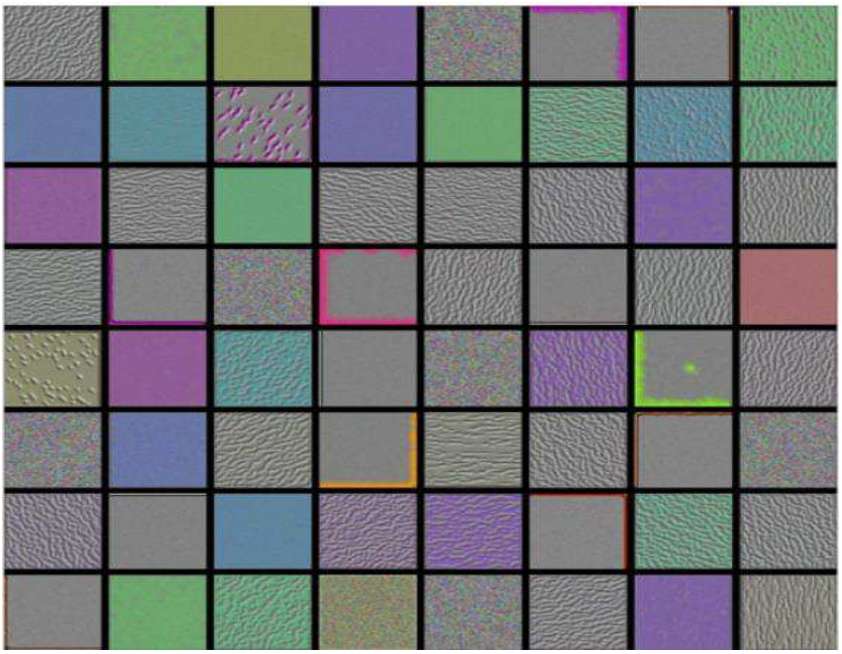
Uma RNC é formada por uma sequência de camadas de convolução e camadas de subamostragem (LECUN; BENGIO, 1998). As camadas de convolução promovem a característica de invariação translacional e as camadas de subamostragem promovem a redução de parâmetros.

2.2.4.2 Filtros em redes neurais convolucionais

Em Processamento Digital de Sinais, a convolução é definida como a multiplicação da energia de dois sinais, sendo que um dos sinais deve ser invertido em relação ao eixo das abcissas. Em RNCs, o processo da convolução consiste em convoluir filtros, também chamados de máscaras, com um sinal de entrada, como, por exemplo, uma imagem. Dessa forma, é possível extrair a resposta dos filtros sobre as diferentes representações dos pixels no espaço bidimensional ou no espaço tridimensional de uma imagem monocromática ou de uma imagem na codificação RGB (*Red Green and Blue*), respectivamente. O resultado da convolução de um filtro com uma imagem é denominado mapa de características.

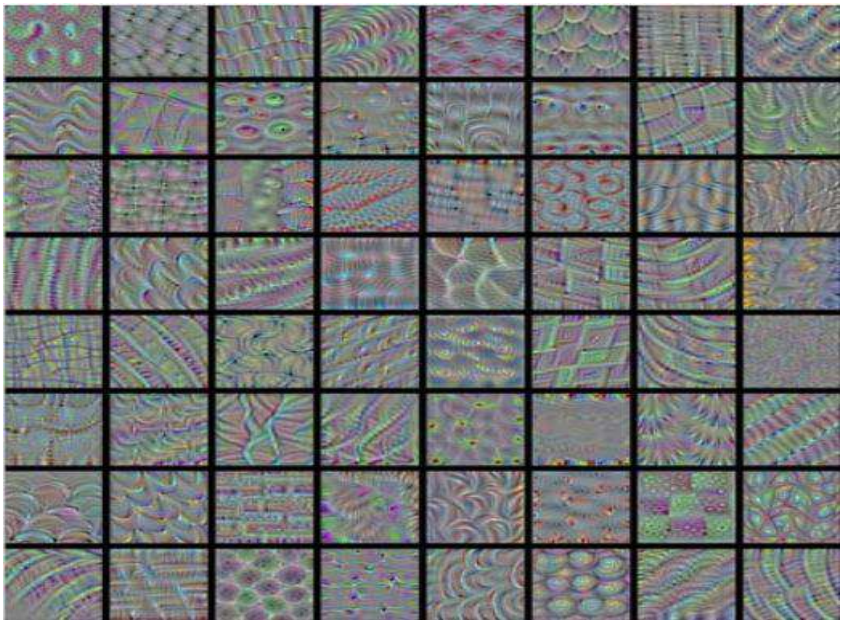
Os estágios iniciais de uma RNC permitem extrair características de baixo nível (níveis de intensidade luminosa) das imagens, porém, conforme o número de camadas aumenta, uma RNC é capaz de extrair características de alto nível (formas, texturas e contornos) (CHOLLET, 2017). Portanto, os filtros presentes nas camadas inicias podem ser considerados filtros passa-baixa, e os filtros nos estágios de saída de uma RNC são considerados filtros passa-alta. A Figura 9 exemplifica filtros capazes de extrair características de baixo nível, e a Figura 10 ilustra filtros capazes de extrair características de alto nível. Porém, é importante salientar que os filtros se ajustam durante o treinamento do modelo de acordo com o algoritmo de retropropagação, pois os valores dos filtros representam os pesos de uma RNC (LECUN; BENGIO, 1998). Este ajuste é implementado conforme abordado na secção 2.2.3.

Figura 9 – Filtros utilizados para extrair características de baixo nível



Fonte: Adaptado de Chollet (2017)

Figura 10 – Filtros utilizados para extrair características de alto nível



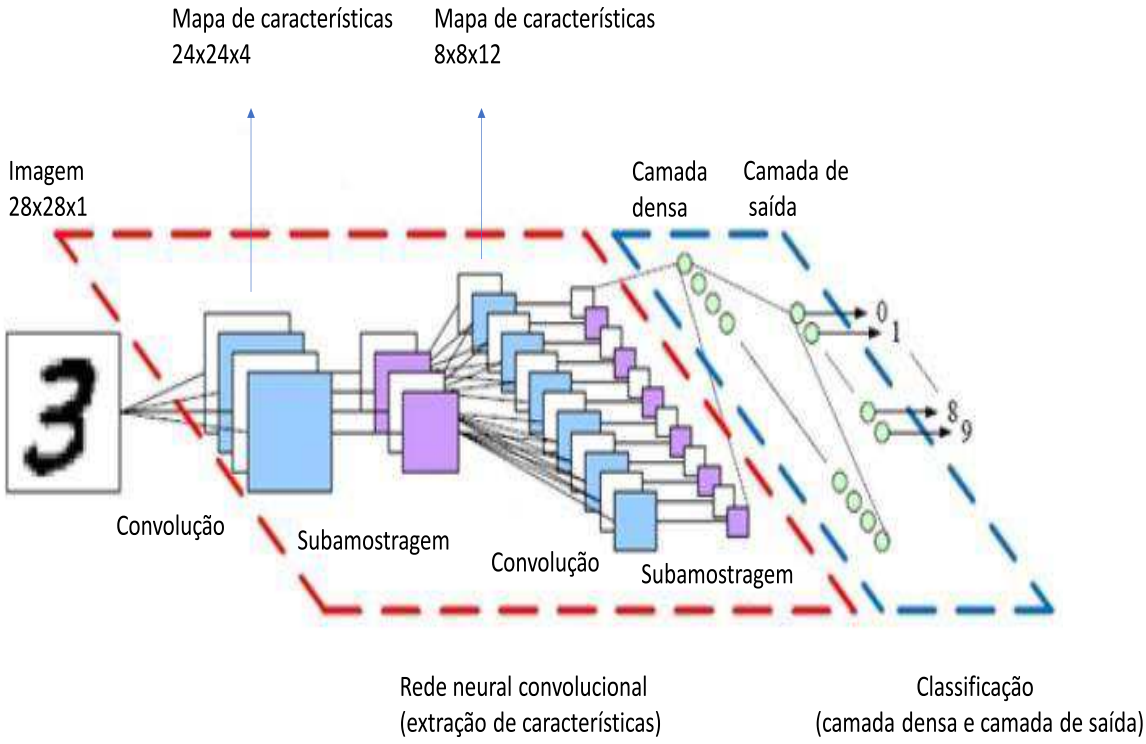
Fonte: Adaptado de Chollet (2017)

2.2.4.3 Camada de convolução

A Figura 11 permite verificar uma RNC utilizada para classificar dígitos manuscritos de 0 a 9. É possível notar que cada convolução permite aumentar a profundidade da imagem. Este parâmetro é controlado pelo número de filtros (LECUN; BENGIO, 1998). Por exemplo, na

primeira convolução são utilizados 4 filtros e na segunda convolução são utilizados 12 filtros, gerando uma imagem com 12 níveis de profundidade.

Figura 11 - Classificação implementada através de uma rede neural convolucional



Fonte: Adaptado de Lecun; Bengio (1992)

Desta forma, cada nível de profundidade representa o mapa de características resultante da convolução de um filtro com a imagem (KARPATHY, 2017). A camada densa e a camada de saída são utilizadas para efetuar a classificação. Isto é possível, por exemplo, atribuindo-se funções de ativação softmax na camada de saída. A função de ativação softmax permite efetuar uma distribuição de probabilidades nas unidades da última camada do modelo. Ou seja, cada unidade da última camada é responsável por armazenar uma classe, e a função softmax possibilita determinar a probabilidade da imagem de entrada pertencer a cada classe.

O processo de convolução de um filtro sobre uma imagem com duas dimensões está descrito na Equação (10) (FERREIRA; GIRARDI, 2016). A convolução de um filtro quadrado de ordem 3 com uma imagem que possui 8 linhas e 8 colunas pode ser verificado na Figura 12. É possível notar o “deslocamento” do filtro sobre uma imagem (MANSANO, 2018).

$$o(m,n) = \sum_{i=1}^{q}\sum_{j=1}^{q} f(i,j)I(m-i,n-j) \tag{10}$$

Onde:

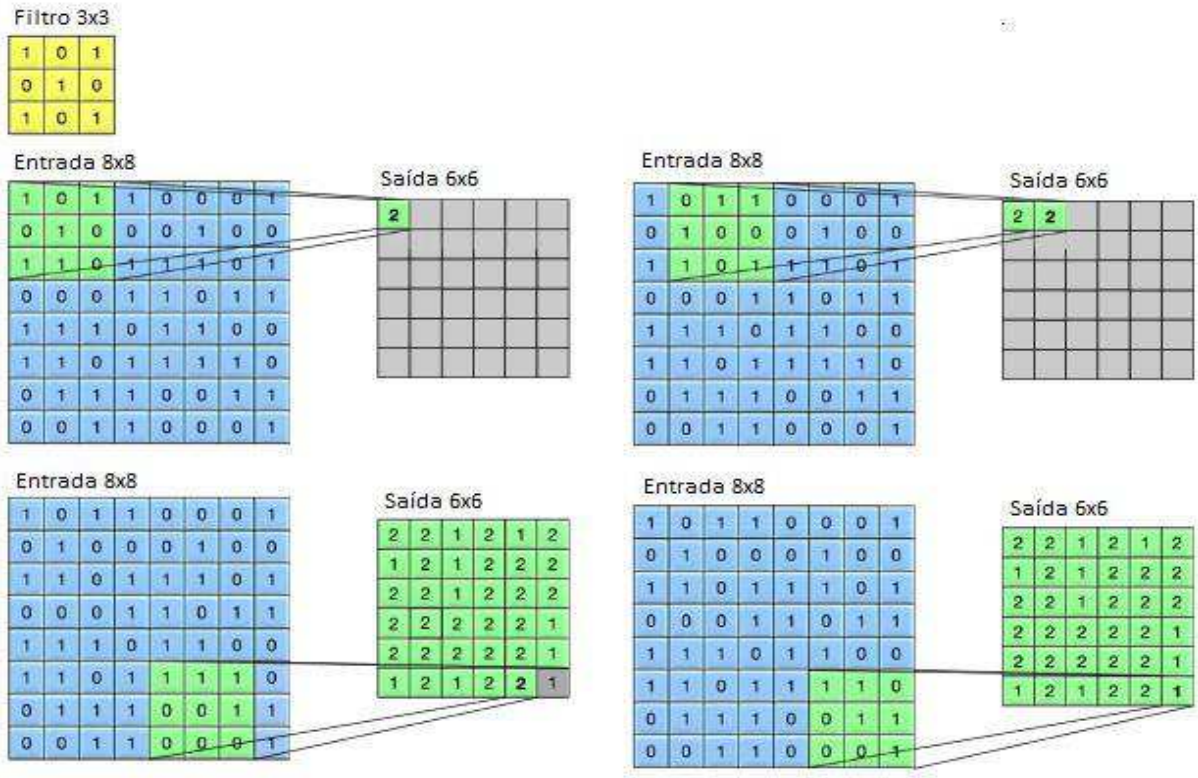
$o(m,n)$: resultado da convolução, nas posições (m,n)

q: dimensão do filtro, assumindo um filtro quadrado

$f(i,j)$: peso nas posições (i,j) do filtro

$I(m,n)$: valores dos pixels na imagem de entrada, nas posições (m,n)

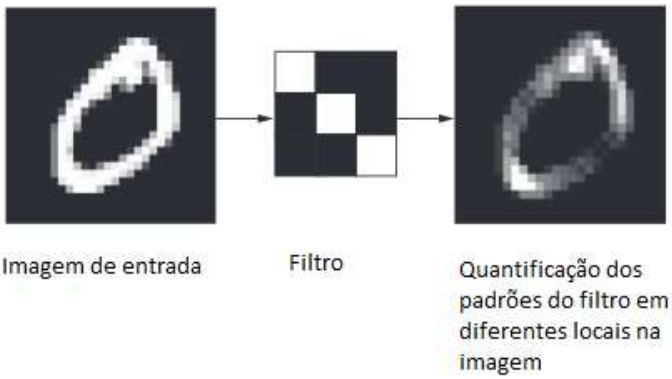
Figura 12 – Convolução de um filtro sobre o conjunto de pixels de uma imagem



Fonte: Adaptado de Mansano (2018)

A quantificação de um filtro sobre diferentes locais da imagem, por exemplo, do dígito manuscrito zero, pode ser visualizada na Figura 13. É possível verificar a capacidade do filtro destacar representações referentes à certos contornos da imagem.

Figura 13 – Resultado da convolução de um filtro sobre uma imagem



Fonte: Adaptado de Chollet (2017).

2.2.4.4 Preenchimento (Padding)

Preenchimento (*padding*) consiste em uma operação efetuada antes de uma convolução, necessária para impedir uma redução no número de linhas e colunas do dado resultante em comparação com a imagem de entrada. Esta redução é recorrente, pois a operação convolucional do filtro com a imagem é capaz de gerar efeitos de borda (*border effects*), reduzindo a altura e a largura da imagem resultante (CHOLLET, 2017) .

Uma técnica comum é denominada *Zero Padding.* Consiste em efetuar o preenchimento acrescentando zeros nas bordas da imagem, fazendo com que a informação perdida seja irrelevante (formada por zeros) e a informação relevante seja mantida. O processo de preenchimento de uma imagem com 7 linhas e 7 colunas, resultando uma imagem com 9 linhas e 9 colunas, pode ser visualizado na Figura 14.

Figura 14 – Preenchimento de zeros nas bordas de uma imagem

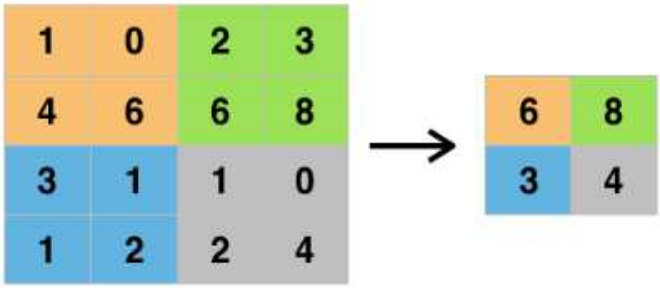
0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	0
0	7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7	0
0	7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7	0
0	7	6	5	4	3	2	1	0
0	1	2	3	4	5	6	7	0
0	0	0	0	0	0	0	0	0

Fonte: Adaptado de Mikulski (2019).

2.2.4.5 Camada de Subamostragem (Pooling)

Subamostragem (*pooling*) consiste em uma operação efetuada para impedir que uma RNC atinja um número excessivo de pesos. Em suma, permite reduzir a quantidade de dados dos mapas de características. Uma técnica usual consiste na utilização da operação Max Pooling (KARPATHY, 2017), que permite extrair os maiores valores em uma determinada janela. A demonstração da operação Max Pooling, efetuada por uma camada de subamostragem, pode ser verificada na Figura 15.

Figura 15 – Operação Max Pooling

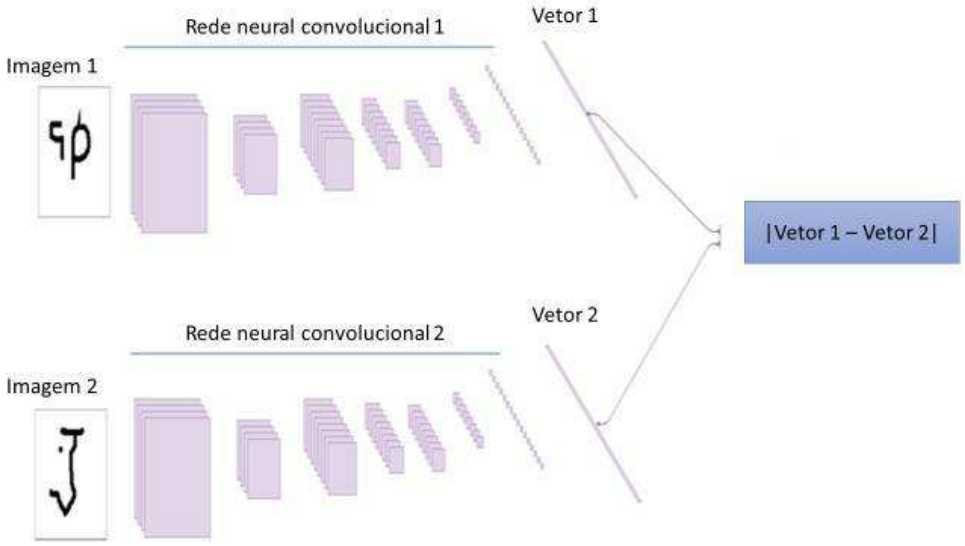


Fonte: Adaptado de Karpathy (2017)

2.2.5 Redes Neurais Siamesas

Desenvolvidas inicialmente em 1993 por Bromley e LeCun para efetuar verificações em assinaturas manuscritas (BROMPLEY et al., 1993), possuem uma ampla faixa de aplicações em sistemas de verificação ou identificação. Consistem em duas redes neurais idênticas utilizadas para determinar a distância de duas características de representações (BOUMA, 2017). Exemplificativamente, podem ser compostas por duas RNCs idênticas unidas por uma função responsável por determinar a distância entre as características extraídas de cada RNC. Uma possibilidade de treinamento para a arquitetura siamesa consiste em inserir sequencialmente pares de imagens pertencentes à uma mesma classe, no qual o resultado esperado é uma distância pequena, e pares de imagens pertencentes a classes distintas, do qual se espera um resultado formado por uma grande distância. É importante salientar que a RNC específica na qual cada imagem é inserida é irrelevante. Por serem simétricas, o resultado deve ser o mesmo independentemente da ordem de inserção da imagem específica dentro dos pares de dados de entrada (KOCH; ZEMEL; SALAKHUTDINOV, 2015). Um exemplo de uma rede neural siamesa pode ser verificado na Figura 16.

Figura 16 – Rede neural siamesa formada por um par de redes neurais convolucionais



Fonte: Adaptado de Bouma (2017)

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os principais materiais e métodos que foram utilizadas no desenvolvimento deste trabalho. É feito um destaque sobre a biblioteca (TensorFlow) utilizada para criar a rede neural, que representou a 1ª abordagem para implementar identificação facial, assim como sobre a biblioteca (*Face Recognition*) utilizada para substituir a rede neural criada neste projeto, que representou a 2ª abordagem para implementar identificação facial.

3.1 SOFTWARE

Esta secção contém as descrições das principais plataformas relacionadas com o desenvolvimento de software neste trabalho.

3.1.1 Python

A construção da rede neural, a utilização da biblioteca *Face Recognition* e a manipulação de vídeos e imagens em tempo real foram implementadas utilizando a linguagem de programação Python. Criada em 1992 por Guido Van Rossum, é considerada uma linguagem de programação de tipagem dinâmica (possui tipagem estática opcional através de funções de anotação), multiparadigma (suporta programação procedural e orientação à objetos), interpretada, imperativa, de alto nível e de propósito geral, ideal para aplicações dinâmicas em diversos tipos de plataformas (PYTHON SOFTWARE FOUNDATION, 2019).

O principal benefício da utilização desta linguagem de programação consiste na possibilidade da utilização de bibliotecas direcionadas ao desenvolvimento de algoritmos com aplicações em inteligência artificial e aprendizagem de máquina. Outro benefício fundamental consiste na promoção de bibliotecas direcionadas à computação científica que permitem, por exemplo, criar gráficos e manipular dados multidimensionais através de containers de dados.

3.1.2 TensorFlow

Foi utilizada para criar a rede neural neste trabalho. Lançada pelo Google em novembro de 2015 como uma biblioteca de código aberto, pode ser usada para a pesquisa e a produção de algoritmos relacionados com aprendizagem de máquina e redes neurais.

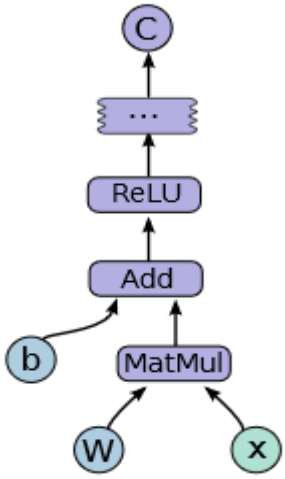
Em suma, possibilita definir gráficos de fluxo de dados em Python e executar estes gráficos em C++ (GÉRON, 2016). Ou seja, possibilita executar um gráfico de fluxo de dados

de uma rede neural em C++, que representa uma linguagem de programação rápida e precisa (STROUSTRUP, 2014), mesmo que a expressão e o controle do processo do modelo do gráfico seja efetuado em Python.

Um gráfico de fluxo de dados é composto por um conjunto de nós e cantos. Cada nó (*node*) representa uma operação matemática (soma, multiplicação, convolução). Os cantos (*edges*) representam os dados multidimensionais, também chamados de tensores. Tensores são sinônimos de arranjos. Exemplificativamente, vetores são tensores unidimensionais e matrizes são tensores bidimensionais (CHOLLET, 2017).

A Figura 17 ilustra operações em um gráfico de fluxo de dados. O nó MatMul efetua a multiplicação da matriz dos pesos (*W)* com o vetor de entrada (*x)*. Após isso, o resultado é somado com o vetor bias *(b)*. Posteriormente, o tensor resultante passa pela função de ativação *ReLU*. O resultado final é representado como a computação do custo (C) da função *ReLU.*

Figura 17 – Representação de um gráfico de fluxo de dados no TensoFlow



Fonte: Abadi (2015)

Dessa forma, os tensores são deslocados através do gráfico. Um núcleo (*kernel*) pode ser considerado uma operação matemática efetuada dentro de um dispositivo específico. Por exemplo, representa uma operação efetuada dentro de uma GPU (*Graphics Processing Unit*).

A criação de uma sessão (*session*) em Python permite criar e executar o gráfico de fluxo de dados. Os parâmetros de uma rede neural correspondem a tensores mantidos em variáveis que são atualizadas durante o treinamento da rede neural (ABADI, 2015).

3.1.3 Keras

Este trabalho também utilizou a API (*Application Programming Interface*) Keras no desenvolvimento da rede neural. Assumindo que a biblioteca TensorFlow pode ser considerada uma ferramenta relativamente de baixo nível (CHOLLET, 2017), Keras atua como uma ferramenta de alto nível, possibilitando implementar algoritmos de aprendizagem de máquina de forma modular. Porém, é importante salientar que, para utilizar a API Keras, é necessário possuir instalado no sistema uma biblioteca de computação numérica capaz de manipular dados multidimensionais em baixo nível, como, por exemplo, o Tensorflow. Além de simplificar a construção das camadas da rede neural, Keras também fornece um conjunto de módulos úteis para a implementação do algoritmo de retropropagação, com um extenso conjunto de funções de custo e otimizadores. Também, promove um conjunto de funções que podem ser utilizadas no pré-processamento dos dados de entrada.

3.1.4 SciPy

SciPy consiste em um conjunto de bibliotecas utilizadas na modelagem matemática de dados em Python. É considerado um software de código aberto com aplicações em engenharia, computação científica e análise de dados. Os principais segmentos do conjunto SciPy que serão utilizadas nesse trabalho consistem na biblioteca Numpy e na biblioteca Matplotlib, descritas a seguir.

3.1.4.1 Numpy

Numpy permite manipular dados de forma rápida e compacta, através de funções que permitem abstrair fundamentos de aritmética, números complexos, álgebra linear, cálculo diferencial e cálculo multivariável. Possibilita definir diferentes variáveis em um tipo de dado específico utilizado dentro da biblioteca, denominado *ndarray*. Portanto, promove a utilização de um eficiente container genérico de dados (NUMPY DEVELOPERS, 2019).

3.1.4.2 Matplotlib

Matplotlib será utilizada na criação de gráficos. Desenvolvida por Jonh Hunter em 2003 e amplamente utilizada pela comunidade de desenvolvedores de software em Python (HUNTER et al., 2007), possibilita criar gráficos e imagens em diversas interfaces e sistemas operacionais. Foi utilizada neste trabalho, por exemplo, para verificar o comportamento da rede neural.

3.1.5 Colaboratory

É constituído por um ambiente de programação Jupyter Notebook, composto por um conjunto de células interativas que permitem executar programas computacionais escritos em Python através de uma máquina virtual que utiliza o sistema de tecnologia em nuvem do Google (COLAB, 2019). A 1ª abordagem referente à identificação facial neste projeto foi implementada neste ambiente.

Um dos principais benefícios da utilização do ambiente Colaboratory consiste na disponibilidade de uma GPU. O dispositivo específico disponível muda constantemente, porém, no início de 2019, este ambiente possibilitava fazer o uso da placa de vídeo TESLA K80, da marca NVIDEA, que possui aproximadamente 8 teraflops de precisão simples e 24 GB de memória GDD5 (NVIDIA CORPORATION, 2019).

Uma GPU permite aumentar a velocidade dos cálculos envolvendo vetores e matrizes em um dispositivo computacional e, dessa forma, quando comparadas com uma CPU (*Central Processing Unit*), possibilitam aumentar a velocidade de treinamento de redes neurais em 50 vezes ou mais (SCHMIDHUBER, 2015).

A transferência de uma rede neural treinada no ambiente Colaboratory para, por exemplo, o Raspberry Pi, pode ser feita da seguinte forma: salvam-se os pesos da rede neural em um arquivo com extensão HDF (*Hierarchical Data Format*), e posteriormente é feita a leitura deste arquivo no Raspberry Pi. Arquivos HDF são utilizados para salvar massivas quantidades de dados, usualmente envolvendo dados multidimensionais, como vetores e matrizes. A API Keras possui funções orientadas a ler e escrever os pesos de uma rede neural em arquivos HDF.

3.1.6 Conjuntos de dados

O conjunto de dados utilizado para testar a acurácia da rede neural consistiu no MUCT (*Milborrow University Of Cape Town*), da Universidade da Cidade do Cabo, na África do Sul, possuindo um total de 3755 imagens divididas em 424 indivíduos diferentes. Este conjunto foi priorizado para implementar testes, pois apresenta variações de idade, gênero e etnia (MILBORROW; MORKEL; NICOLLS, 2010), representando desafios que são abordados em um sistema de reconhecimento facial em tempo real.

Para implementar o treinamento da rede neural, foram utilizados 10 conjuntos de dados distintos. Procurou-se um elevado número de conjunto de dados de treinamento para fornecer exemplos suficientes ao sistema, fazendo com que a rede neural obtivesse a capacidade de generalização, podendo identificar indivíduos, mesmo com variações faciais, geométricas ou

luminosas. Os 10 conjuntos de dados utilizados para treinar a rede neural estão descritos a seguir.

- *Label Faces in the Wild*: Fornecido pela Universidade de Massachusetts, nos Estados Unidos. Possui um total de 13.000 imagens divididas em 5.749 indivíduos diferentes (GARY B. HUANG et al., 2012).
- *Color Feret*: Fornecido pelo Instituto Nacional de Padrões e Tecnologia, nos Estados Unidos. Possui um total de 14.126 imagens divididas em 1.199 indivíduos diferentes (PHILLIPS, 2016).
- *AT&T*: Fornecido pela Universidade de Cambridge, na Inglaterra. Possui um total de 400 imagens divididas em 10 indivíduos diferentes (SAMARIA; HARTER, 1994; AT&T LABORATORIES CAMBRIDGE, 2002).
- *Carnegie Mellon*: Fornecido pela Universidade Carnegie Mellon, nos Estados Unidos. Possui um total de 3.660 imagens divididas em 90 indivíduos diferentes (CARNEGIE MELLON UNIVERSITY, 2019).
- *Cropped Yale B*: Fornecido pela Universidade de Yale, nos Estados Unidos. Possui um total de 2.535 imagens divididas em 39 indivíduos diferentes (UCSD COMPUTER VISION, 2019).
- *Caltech Faces*: Fornecido pela Instituto de Tecnologia da California, nos Estados Unidos. Possui um total de 450 imagens divididas em 27 indivíduos diferentes (CALTECH, 2019).
- Essex *94*: Fornecido pela Universidade de Essex, na Inglaterra. Possui um total de 3.060 imagens divididas em 153 indivíduos diferentes (ESSEX FACIAL IMAGES, 2019).
- Essex *95*: Fornecido pela Universidade de Essex, na Inglaterra. Possui um total de 1.440 imagens divididas em 72 indivíduos diferentes (ESSEX FACIAL IMAGES, 2019).
- Essex *96*: Fornecido pela Universidade de Essex, na Inglaterra. Possui um total de 3.040 imagens divididas em 152 indivíduos diferentes (ESSEX FACIAL IMAGES, 2019).
- *Essex Grimace*: disponibilizado pela Universidade de Essex, na Inglaterra. Possui um total de 360 imagens divididas em 18 indivíduos diferentes (ESSEX FACIAL IMAGES, 2019).

Os conjuntos de dados da Universidade de Essex foram separados, pois cada conjunto possui um banco de imagens com características diferentes (diferem na posição geométrica e no plano de fundo das imagens).

Os termos e condições de cada conjunto variam de acordo com a possibilidade de reprodução das imagens em trabalhos acadêmicos. Neste trabalho, serão exibidas apenas imagens que permitem a possibilidade de reprodução. A Figura 18 permite visualizar algumas imagens dos conjuntos de dados *Label Faces in the Wild, Caltech Faces* e MUCT.

Figura 18 – Exemplos de algumas imagens utilizadas para treinar a rede neural

Fonte: Gary B. Huang et al. (2012); California Institute of Technology (2019); Milborrow S.; Morkel J.; Nicolls F. (2010)

3.1.7 Dlib

Corresponde à uma biblioteca orientada à solução de problemas na área de aprendizagem de máquina (DLIB C++ LIBRARY, 2019). Fornece um amplo conjunto de funções especializadas na detecção e identificação facial. Não foi utilizada nenhuma função para detecção facial neste trabalho, pois o algoritmo Viola-Jones apresentou resultados satisfatórios (conforme será discutido no capítulo referente aos resultados) sem atrasar a geração de frames no sistema. Porém, foram utilizadas funções para a identificação facial na 2ª abordagem deste trabalho. Tais funções estão inclusas na biblioteca *Face Recognition,* descrita a seguir.

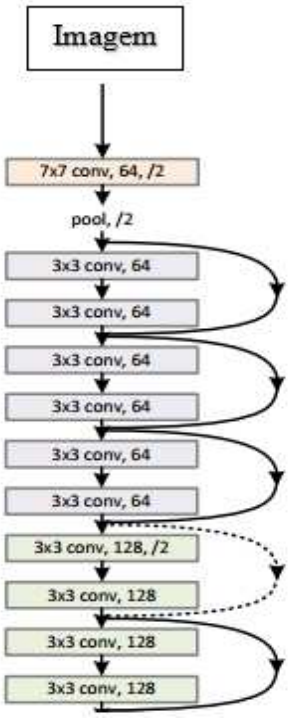
3.1.8 Face Recognition

Esta biblioteca implementa as mesmas funções de Dlib, porém, de forma mais rápida e concisa. Assim, atua como um envoltório sobre Dlib (GEITGEY, 2019). *Face Recognition*, e, portanto, Dlib, fazem o uso de uma rede neural denominada ResNet, introduzida em 2017 por

pesquisadores da Microsoft (HE et al., 2017). Reutilizar redes neurais é comum na área de aprendizagem de máquina (MENEGOLA, 2018), pois permite obter resultados satisfatórios (CHOLLET, 2017) sem a necessidade da criação inicial do modelo. Esta área é chamada de aprendizagem por transferência.

A rede neural ResNet foi desenvolvido para solucionar um problema denominado degradação de redes neurais, que resulta em uma saturação (diminuição de acurácia) de redes neurais com um número elevado de camadas (por exemplo, dezenas de camadas). A ideia apresentada para o modelo ResNet consiste em não conectar as camadas de forma direta, mas de forma residual. A Figura 19 permite ilustrar as 11 primeiras camadas de convolução da rede neural ResNet. A quantidade de camadas apresentada no artigo original introduzido pelos pesquisadores da Microsoft consistia em 34 camadas de convolução. O algoritmo implementado na biblioteca *Face Recognition* faz uso de uma versão do modelo ResNet com 29 camadas de convolução (KING, 2017).

Figura 19 – Representação simplificada da rede neural ResNet



Fonte: Adaptado de He et al. (2017)

3.1.9 OpenCV

A biblioteca OpenCV foi utilizada para controlar a geração de frames em tempo no algoritmo do projeto. Desenvolvida inicialmente em 1999 pela empresa Intel, atualmente consiste em software de código aberto e é amplamente utilizada em aplicações rápidas e

precisas que efetuam um processamento digital de imagens e vídeos em tempo real (OPENCV TEAM, 2019).

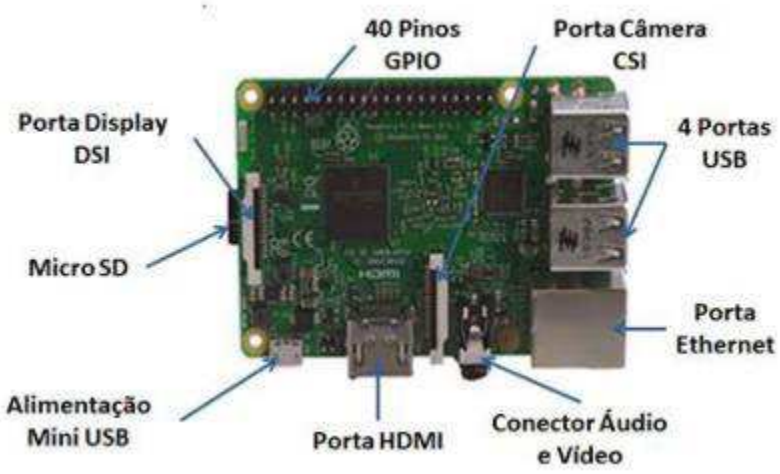
3.2 HARDWARE

Esta secção contém as descrições das principais plataformas relacionadas com o desenvolvimento de hardware neste trabalho.

3.2.1 Raspberry Pi

O microcomputador escolhido para o desenvolvimento deste trabalho consistiu no Raspberry Pi 3 modelo B. Este dispositivo pode ser visualizado na Figura 20.

Figura 20 - Estrutura do Raspberry Pi 3



Adaptado de Khandelwal (2016).

As principais características que promovem a utilização do Raspberry Pi em um sistema de reconhecimento facial incluem: pequeno tamanho, portabilidade, acessibilidade no mercado, baixo custo e, principalmente, permitir construir algoritmos com versões atualizadas das bibliotecas TensorFlow, Keras e OpenCV e *Face Recognition*.

Outros dispositivos semelhantes ao Raspberry Pi, apesar de possuírem uma capacidade de processamento mais elevada, possuem um alto custo comparativo, conforme pode ser visualizado no Quadro 1. As especificações técnicas do Raspbery Pi 3 modelo B estão descritas no Quadro 2.

Quadro 1 – Custo financeiro de diferentes microcomputadores

Dispositivo	Preço
Raspberry Pi 3	R$ 199,99
Odroid C2	R$ 409,99
Nvidia Jetson Nano	R$ 759,99
Beaglebone Black	R$ 799,99

Fonte: Próprio Autor (2019)

Quadro 2 – Especificações do Raspberry Pi versão 3 modelo B

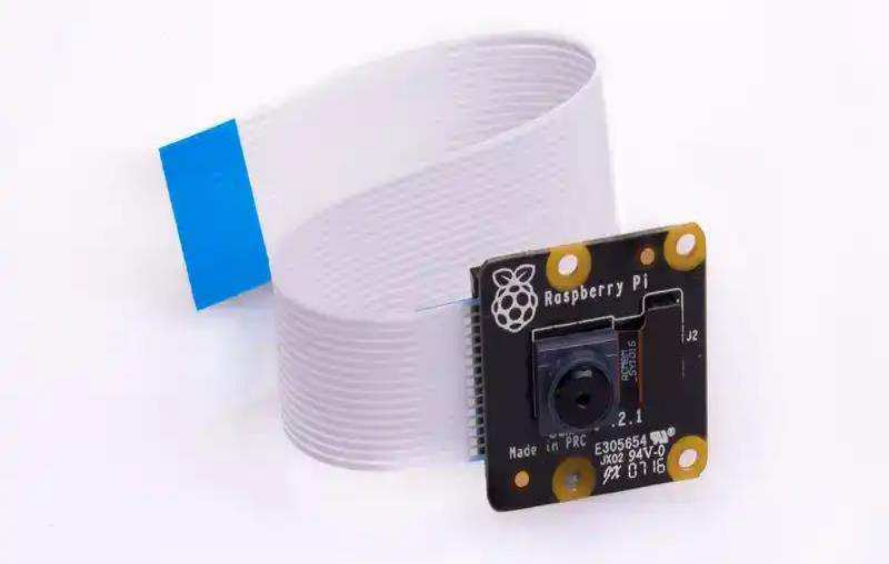
Marca	Raspberry Pi
Versão	3
Modelo	B
Ano de Lançamento	2016
CPU	Quad-core 64 bits ARMv8
Clock	1.2 GHz
Wireless	LAN 802.11n
Bluetooth	4.1 BLE
Memória RAM	1 GB
Portas USB 2.0	4
GPIO	40
Porta Ethernet	1
Porta HDMI	1
Conector Áudio e Vídeo	1
Interface para Câmera	1
Interface para Display	1
Micro SD Card	1
Corrente de Alimentação	2.5 A
Tensão de Alimentação	5 V
Dimensões	85 x 56 x 17 mm
Peso	42 g

Fonte: Adaptado de Raspberry Pi Fundation (2019).

3.2.2 Câmera

Foi utilizada a versão 2 do módulo da câmera oficial do Raspberry Pi para efetuar o controle do vídeo e a aquisição de imagens no local de desenvolvimento do projeto. A segunda versão do módulo oficial da câmera possui 8 megapixels (RASPBERRY PI FOUNDATION, 2019), permite capturar imagens em alta resolução e transmitir vídeos em uma taxa de 30 frames por segundo. Foi feito o uso da câmera “NoIR”, que não possui um filtro infravermelho, e pode ser utilizada em ambientes com diferentes níveis de luminosidade. O dispositivo pode ser visualizado na Figura 21.

Figura 21 – Módulo da câmera Raspberry Pi



Fonte: Raspberry Pi Fundation (2019)

4 DESENVOLVIMENTO

Este capítulo aborda como foi desenvolvido o trabalho referente à detecção e identificação facial. A versão final do projeto utilizou o algoritmo Viola-Jones para efetuar detecção facial e utilizou o modelo presente na biblioteca *Face Recognition* para implementar identificação facial (2ª abordagem). Entretanto, também procura-se esclarecer todos os passos relacionados com o desenvolvimento referente a rede neural utilizando a biblioteca Tensorflow (1ª abordagem), pois treinar uma rede neural artificial consistia no objetivo original deste trabalho para implementar identificação facial e consistiu em uma etapa fundamental deste projeto.

4.1 DETECÇÃO FACIAL ATRAVÉS DO ALGORITMO VIOLA-JONES

A implementação do algoritmo Viola-Jones foi feita através de um arquivo XML (*Extensible Markup Language*) interpretado pela biblioteca OpenCV. O arquivo consiste em um modelo previamente treinado para detectar regiões frontais de uma face em uma imagem. Assim, para o software desenvolvido neste trabalho, sendo executado em tempo real no Raspberry Pi, todos os frames gerados pelo sistema eram analisados pelo modelo. Entretanto, foi determinada uma dimensão mínima (80x80) para uma imagem ser aceita pelo modelo. Isto foi necessário para garantir uma resolução mínima para as imagens que são posteriormente identificadas. A resposta do algoritmo Viola-Jones consiste nas coordenadas de uma face na imagem. Desta forma, é possível implementar uma demarcação das características faciais em imagens, conforme demonstrado na Fotografia 1.

4.2 IDENTIFICAÇÃO FACIAL: 1ª ABORDAGEM

A 1ª abordagem para identificação facial consistiu na utilização da biblioteca Tensorflow, em conjunto com a API Keras, no ambiente Colaboratory. Também foram utilizadas as bibliotecas Numpy e Matplotlib.

4.2.1 Pré-processamento das imagens

Todas as imagens foram pré-processadas para o formato 56x56, convertidas em arranjos Numpy (*np.ndarray*) e selecionadas em pares. Foi utilizado o formato 56x56 para impedir um

Fotografia 1 - Demarcação da região facial em uma imagem

Fonte: Próprio Autor (2019)

tempo de treinamento excessivo (como o sistema possui um número relativamente elevado de imagens, uma alta resolução para cada imagem poderia implicar um treinamento com uma duração elevada, mesmo no ambiente Colaboratory).

Os pares de imagens foram gerados da seguinte forma: para cada imagem, selecionava-se outra imagem do mesmo indivíduo e outra imagem de outro indivíduo (aleatório), formando dois pares. Pares positivos são considerados pares de imagens contendo o mesmo sujeito e pares negativos são considerados pares de imagens contendo sujeitos diferentes. O algoritmo Viola-Jones também foi utilizado para selecionar quais imagens dos conjuntos de dados seriam adequadas para o treinamento (ou seja, seriam semelhantes à imagem capturada em tempo real no Raspberry Pi). Isso resultou em uma diminuição na quantidade de imagens utilizadas, pois algumas imagens não foram reconhecidas pelo modelo do algoritmo Viola-Jones (possuíam expressões faciais laterais e não possuíam expressões faciais frontais).

Assim, de forma experimental, foi desenvolvida a Equação (11), que permite verificar quantos pares foram gerados para cada conjunto de dados. O valor de sujeitos é reduzido por 1 para evitar a geração de pares repetidos (pares com as mesmas imagens, porém, com a ordem de cada imagem invertida).

$$Q = (N - 1) \cdot T \tag{11}$$

Onde:

- Q: Quantidade de pares gerados (positivos e negativos)
- N: Número de imagens por indivíduo (reconhecidas pelo algoritmo Viola-Jones)
- T: Número total de imagens (reconhecidas pelo algoritmo Viola-Jones)

O Quadro 3 ilustra a quantidade de imagens em cada conjunto de dados, e a quantidade de pares que foram selecionados para cada conjunto. Totalizaram-se 351.348 pares (702.696 imagens). Todos os conjuntos de dados foram inseridos no google drive, e posteriormente montados do google drive para o google colaboratory, pois não é possível inserir uma quantidade elevada de imagens de forma direta no google colaboratory.

Além disso, foram aplicadas transformações (perturbações) em todas as imagens. Estas transformações são utilizadas para aumentar as variações luminosas e geométricas nas imagens de treinamento, e, desta forma, aumentar o ruído das imagens, elevando a dificuldade de treinamento do modelo. Isto é necessário, pois quanto maior a dificuldade de treinamento da rede neural, desde que haja convergência, melhor será a acurácia do modelo.

As transformações poderiam ser aplicadas manualmente em cada imagem, utilizando, por exemplo, a biblioteca OpenCV ou o conjunto de bibliotecas no pacote SciPy. Porém, neste trabalho, foram implementadas através das funções orientadas ao pré-processamento de dados na API Keras. Assim, como a rede neural também é criada nesta API, é possível conectar estes processos e executar as transformações de uma forma que, para cada época, seja inserida uma imagem diferente, e a rede neural nunca receba uma imagem repetida, pois a API Keras gera transformações aleatórias e possui um *offset* interno e impede que seja aplicado a mesma transformação para uma imagem. Estas transformações envolvem rotações e alterações no brilho da imagem, processo chamado de *image augmentation.* Portanto, foram aplicados ruídos diretamente sobre as imagens resultantes do detector Viola-Jones, conforme pode ser verificado na Figura 22.

4.2.2 Desenvolvimento da rede neural

A rede neural foi desenvolvida com preceitos apresentados na literatura clássica (HAYKIN, 2001; LECUN; BENGIO, 1998) e na literatura moderna (CHOLLET, 2017; KARPATHY, 2017). Foram geradas 8 camadas de convolução e 3 camadas de subamostragem.

Quadro 3 – Dados relativos aos conjuntos de dados

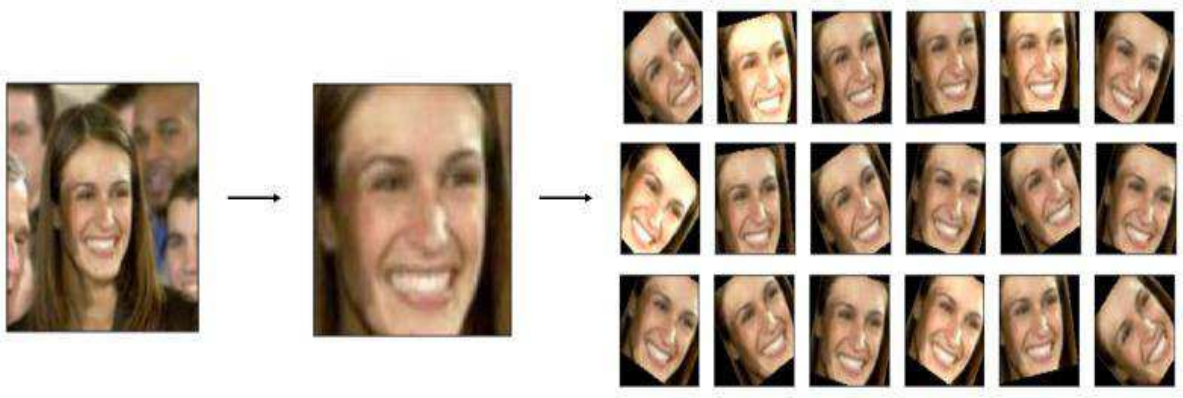
Conjunto de dados	Nº total de indivíduos	Nº de indivíduos reconhecidos[2]	N	T	Q
Label Faces in the Wild [3]	5.749	1.802	3	5.406	10.812
Color Feret	1.199	424	4	1.696	5.088
AT&T	40	40	10	400	3.600
Carnegie Mellon	90	85	30	2.550	73.950
Cropped Yale B	39	38	58	2.204	125.628
Caltech Faces	27	15	18	270	4.590
Essex 94	153	139	20	2.780	52.820
Essex 95	72	57	20	1.140	21.660
Essex 96	152	124	20	2.480	47.120
Essex Grimace	18	16	20	320	6.080
MUCT	424	225	10	2.250	20.250

Fonte: Próprio Autor (2019)

[2] Para alguns conjuntos de dados, a quantidade de indivíduos utilizados foi reduzida de forma significativa. Por exemplo, para o conjunto de dados *Label Faces in the Wild,* o número de indivíduos foi reduzido de 5749 para 1203, pois 4546 indivíduos possuem apenas uma imagem, impossibilitando a geração de pares positivos. Para o conjunto de dados *Color Feret,* o número de indivíduos foi reduzido de 1199 para 424, pois 775 indivíduos possuíam imagens com características laterais, impossibilitando a detecção de imagens pelo algoritmo Viola-Jones.

[3] Para o conjunto de dados *Label Faces in the Wild,* o número de imagens por indivíduo não era constante. Porém, a média consistia em 3 imagens por indivíduo. Para os outros conjuntos de dados, o número de imagens por indivíduo era constante.

Figura 22 – Ilustração da detecção facial e do processo *image augumentation*



Fonte: Próprio Autor (2019)

Todas as camadas de convolução possuem a função de ativação ReLU para acelerar a convergência do modelo. A saída da última camada de convolução de cada rede neural foi vetorizada. Além disso, também foi implementada uma camada de *Dropout*, que permite aleatoriamente anular determinadas conexões durante o treinamento do modelo (isto impede que a rede neural "memorize" os dados de treinamento, promovendo uma maior generalização sobre os dados). Posteriormente, foram adicionadas duas camadas densas com funções de ativação ReLU e sigmoide, ambas com 1024 unidades. Isto permitiu normalizar os valores dos pesos para o primeiro quadrante do plano cartesiano (valores positivos entre 0 e 1). O Quadro 4 permite verificar as especificações de cada camada. Cada rede neural convolucional possui 30.714.000 parâmetros.

A saída de cada rede neural é constituída por um vetor, representando a característica facial extraída de cada imagem. Estes conjuntos de dados também são chamado de *embeddings* ou *encodings,* pois representam uma redução de dimensionalidade (transformação de uma imagem com dimensão 56x56 para um vetor com 1.024 elementos). A saída do modelo completo consiste em uma camada responsável por calcular a distância euclidiana entre as representações extraídas de cada rede neural, conforme a Equação (12). Assim, o resultado deve constituir um número decimal representando a diferença entre o par de imagens que foi inserido no modelo. A Figura 23 permite visualizar a rede neural no formato de um diagrama.

Quadro 4 – Especificações das camadas para cada RNC

Camada	Dimensão	Função de ativação	Quantidade de parâmetros
Convolução 1	(56, 56, 32)	ReLU	320
Convolução 2	(56, 56, 32)	ReLU	9.248
Subamostragem 1	(28, 28, 32)	-	-
Convolução 3	(28, 28, 64)	ReLU	18.496
Convolução 4	(28, 28, 64)	ReLU	36.928
Subamostragem 2	(14, 14, 64)	-	-
Convolução 5	(14, 14, 128)	ReLU	73.856
Convolução 6	(14, 14, 128)	ReLU	295.168
Subamostragem 3	(7, 7, 256)	-	-
Convolução 7	(7, 7, 256)	ReLU	1.180.106
Convolução 8	(7, 7, 512)	ReLU	2.359.808
Vetorização (Flatten)	(25.088)	-	-
Dropout (taxa=50%)	(25.088)	-	-
Densa 1	(1.024)	ReLU	25.691.136
Densa 2	(1.024)	Sigmoide	1.049.600

Fonte: Próprio Autor (2019)

$$d(p,q) = \sum_{i=1}^{n} (q_i - p_i)^2 \qquad (12)$$

Onde:

$d(p,q)$: distância entre dois vetores

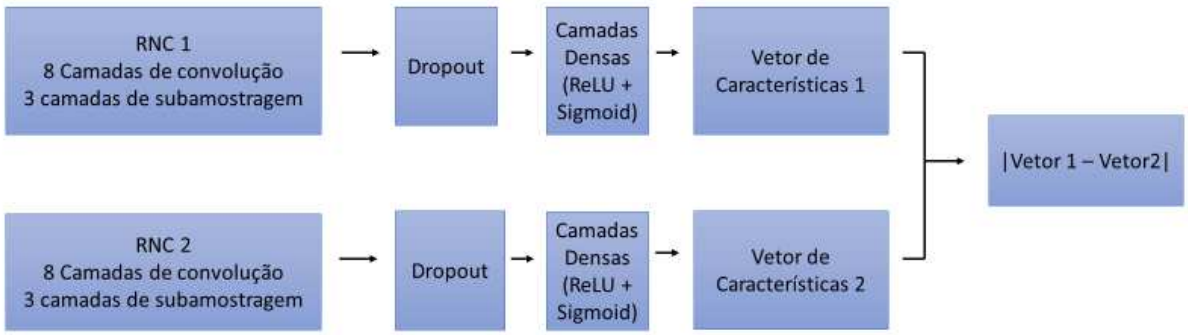
n: comprimento dos vetores

i: índice para cada elemento do vetor

$(q_i - p_i)$: distância entre cada elemento dos vetores.

Para calcular o erro, foi utilizada a função de custo *Contrastive Loss*. Esta função permite calcular a distância entre pares de imagens anexando o rótulo “1” para pares positivos e “0” para pares negativos (CHOPRA; HADSELL; LECUN, 2005). Além disso, para adaptar os pesos a partir do valor do erro, foi utilizado o otimizador *Adadelta*. Este otimizador permite que

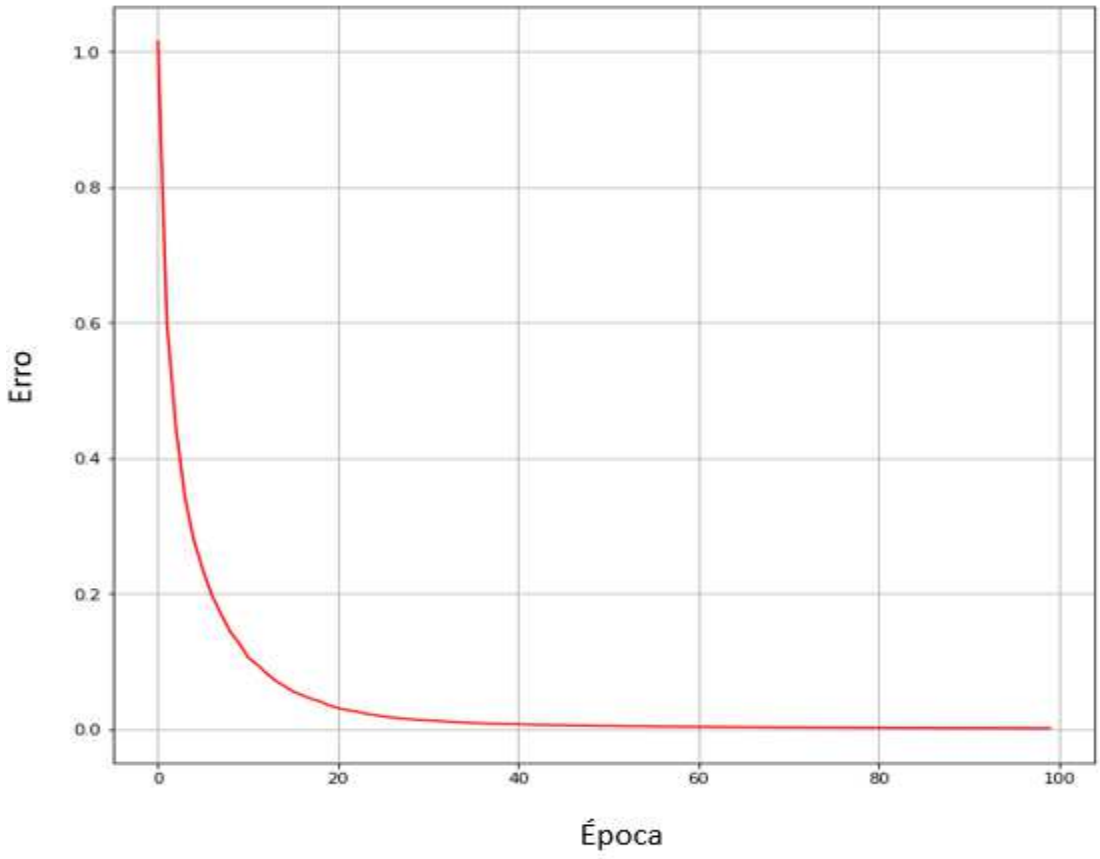
Figura 23 – Diagrama da rede neural siamesa



Fonte: Próprio Autor (2019)

o modelo adapte a taxa de aprendizagem durante o algoritmo de retropropagação, acelerando a convergência do modelo (ZEILER, 2012). O Gráfico 3 permite verificar o valor de erro para cada época entre 1 e 100. Rede neurais usualmente são treinadas em algumas dezenas de épocas (CHOLLET, 2017).

Gráfico 3 – Erro da rede neural



Fonte: Próprio Autor (2019)

Neste trabalho, o valor pelo qual a rede neural apresentou a maior acurácia no conjunto de dados MUCT foi para 18 épocas. O tempo de duração de treinamento de cada época foi de aproximadamente 5 minutos. A explicação para ter sido obtido um melhor resultado para 18 épocas, ao invés de um valor de épocas maior, consiste na dinâmica da rede neural passar a "memorizar" as imagens após um certo ponto de treinamento, mesmo sendo utilizada uma camada de *dropout*, conforme mencionado anteriormente. Assim, com mais de 18 épocas o modelo passou a sobre-ajustar o conjunto de dados de treinamento, fenômeno denominado *overfiting*. A rede neural foi treinada de ponta-a-ponta, ou seja, foi treinada da camada de entrada para a camada de saída, e permitiu utilizar somente a camada de saída como resposta.

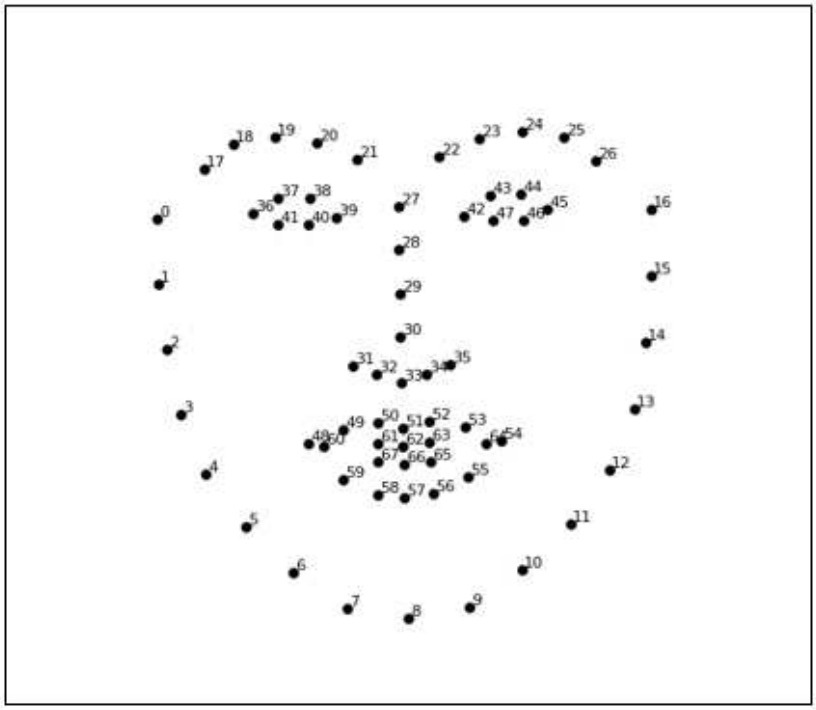
4.3 IDENTIFICAÇÃO FACIAL: 2ª ABORDAGEM

A 2ª abordagem para implementar identificação facial neste trabalho consistiu na utilização da biblioteca *Face Recognition*. Esta biblioteca foi usada para extrair representações faciais de imagens através do modelo da rede neural ResNet, conforme abordado na secção 3.1.8.

Um ponto fundamental implementado internamente na biblioteca *Face Recognition* para reduzir erros referentes a identificação facial consiste em suas funções orientadas ao pré-processamento facial. Assim, esta biblioteca implementa um alinhamento facial de imagens de acordo com regiões faciais previamente estabelecidas em um *template*, conforme ilustrado na Figura 24. É importante notar que, mesmo considerando que um dos principais benefícios da utilização de RNCs consistir na sua capacidade de invariação geométrica, implementaram-se algoritmos para reduzir estas variações na biblioteca *Face Recognition*, comprovando a dificuldade de gerar um sistema completamente invariante em termos geométricos na área de visão computacional. Além do alinhamento facial, outro ponto positivo da biblioteca *Face Recognition* consiste no treinamento do modelo utilizado pela biblioteca ter sido implementado com 3 milhões de imagens.

Outro benefício da biblioteca *Face Recognition* consiste na característica de não ser um sistema treinado de ponta-a-ponta. Desta forma, permite dividir o modelo completo em conjuntos de modelos menores e acessar saídas intermediárias. Assim, permite extrair, por exemplo, as representações de uma face em uma imagem, para posterior cálculos de distância entre diferentes representações, ao invés de apenas implementar o processo completo (determinação das representações e cálculo de distâncias em um único processo).

Figura 24 –*Template* utilizado pela biblioteca *Face Recognition* para alinhar imagens.



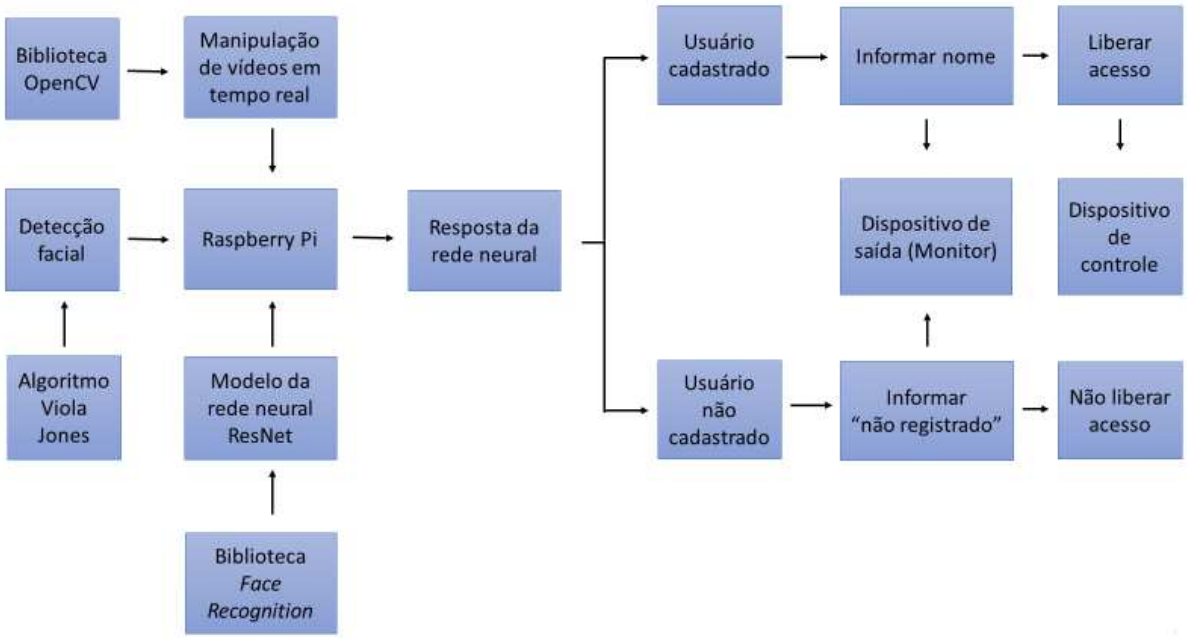
Fonte: (King, 2017)

Desta forma, o principal benefício desta dinâmica consiste na característica de que a extração de representações leva um tempo considerável em um microcomputador como o Raspberry Pi (0,7 s). Entretanto, o cálculo das distâncias é rápido (0,02 *u*s). Assim, para um programa de reconhecimento facial em tempo real, é possível salvar os valores das codificações (*encodings*) dos usuários cadastrados em um primeiro momento, e posteriormente extrair apenas as codificações da imagem capturada. O cálculo da distância entre duas codificações *encodings* também foi implementado através da distância euclidiana

4.4 ESTRUTURA FUNCIONAL DO PROJETO

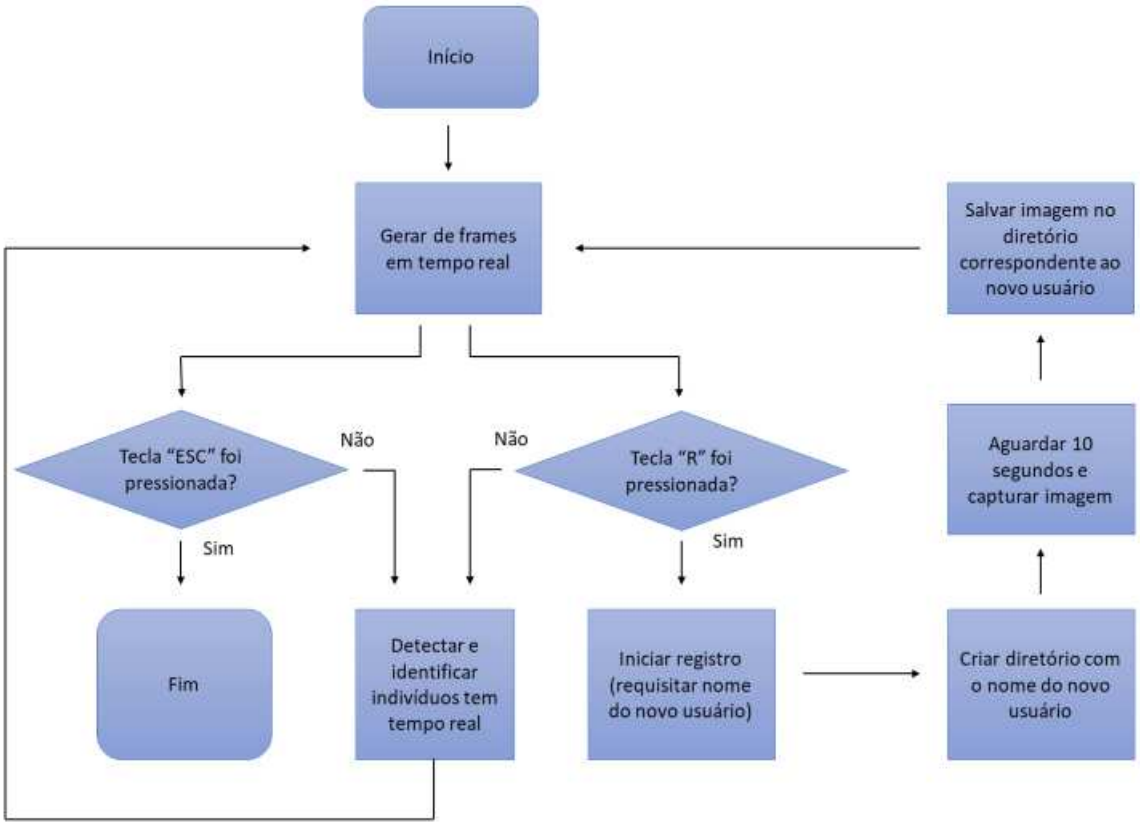
A Figura 25 ilustra um diagrama contendo diferentes segmentos do projeto. Conforme ilustrado na figura, caso o indivíduo seja reconhecido pelo programa, o acesso é liberado. Caso o indivíduo não seja reconhecido pelo algoritmo, nenhuma ação é implementada no dispositivo de controle de acesso. Foram utilizadas duas entradas no teclado para controlar o software, conforme demonstrado no Figura 26. Caso nenhuma tecla seja pressionada, o programa implementa detecção e identificação em tempo real, conforme ilustrado na Figura 27.

Figura 25 – Diagrama de blocos geral do projeto



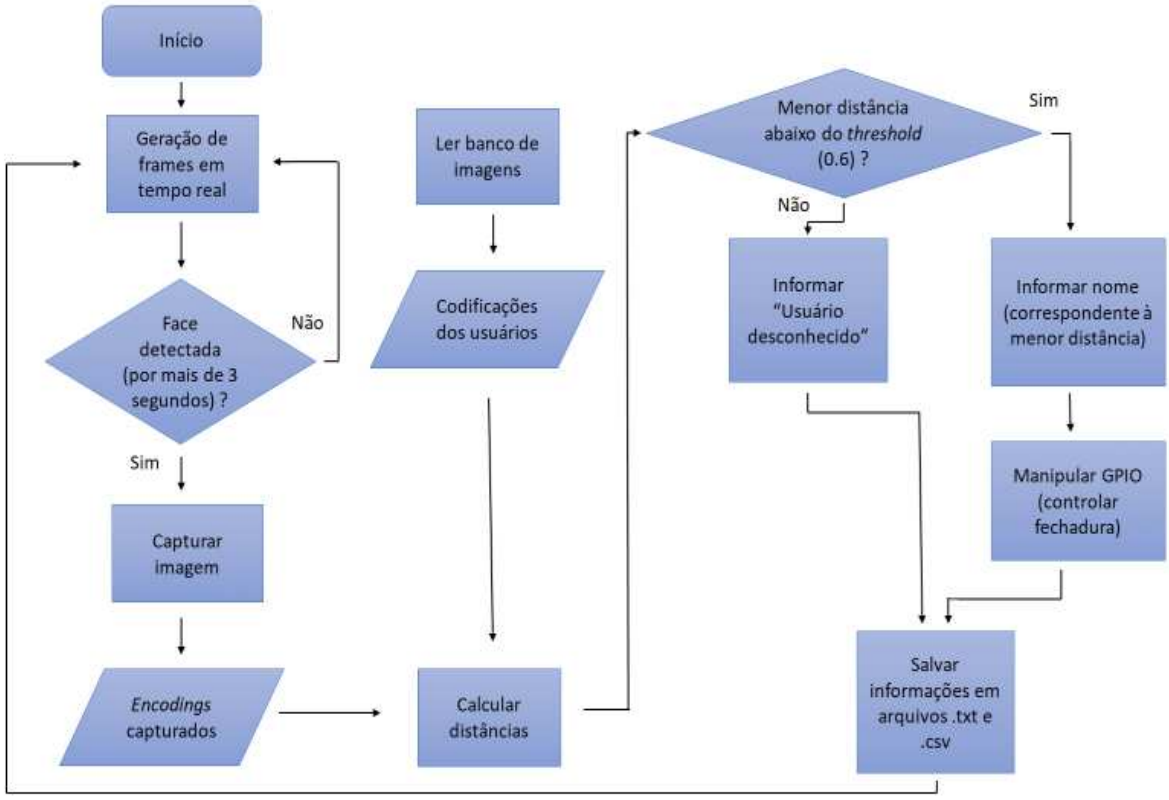
Fonte: Próprio autor

Figura 26 – Fluxograma para o controle de teclas implementado no software



Fonte: Próprio Autor

Figura 27 – Fluxograma para detecção e identificação facial em tempo real



Fonte: Próprio autor

A leitura das imagens dos usuários (banco de imagens), é feita apenas uma vez, pois, conforme abordado na secção 4.3, o programa demora 0,7 segundo para obter as codificações de uma imagem. Além disso, para habilitar a identificação, o usuário deve permanecer em frente a câmera por mais de 3 segundos, pois se o sistema implementasse a identificação em todos os momentos, o vídeo em tempo real apresentaria uma grande latência (para cada frame gerado, ocorreria um atraso de 0,7 segundo). O comportamento do sistema durante o tempo de carregamento pode ser verificado na Fotografia 2.

Após a captura da imagem, o programa calcula a distância entre as codificações da imagem capturada com as codificações dos usuários cadastrados. Assim, como o tempo do cálculo da distância entre um par de codificações pode ser considerado desprezível (0,02 us), a geração de frames em tempo real ocorre aparentemente sem atraso. A Fotografia 3 ilustra o comportamento do software para um indivíduo desconhecido.

A requisição do registro de um novo usuário encerra o vídeo, exibe uma solicitação do nome do novo usuário na tela, conforme ilustrado na Fotografia 4, e inicia um novo vídeo, com a mensagem “olhe para a câmera”, por 10 segundos, conforme pode ser verificado na Fotografia 5. Após o registro, a identificação em tempo real é retomada.

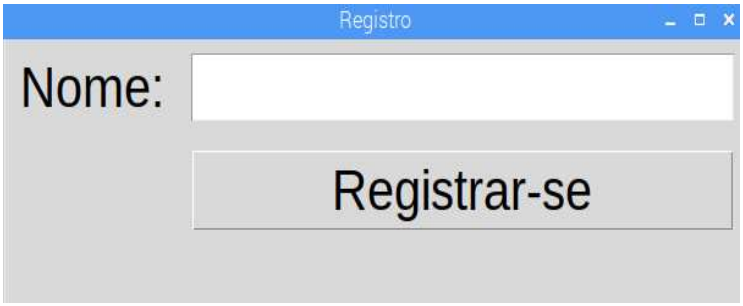
Fotografia 2 – Ilustração para condição de carregamento do software

Fonte: Próprio Autor (2019)

Fotografia 3 – Ilustração para condição de “indivíduo desconhecido” do software

Fonte: Próprio Autor (2019)

Fotografia 4 – Interface gráfica para registro de um novo usuário



Fonte: Próprio Autor (2019)

Fotografia 5 – Mensagem utilizada durante o cadastro de um novo usuário

Fonte: Próprio Autor (2019)

A Fotografia 6 ilustra o comportamento do software para um usuário previamente cadastrado. Foi utilizado um valor de limite igual à 0,6 para implementar a detecção de indivíduos desconhecidos. Este é o valor recomendado na documentação da biblioteca *Face Recognition*. Assim, se o resultado do cálculo da distância euclidiana entre dois conjuntos de codificações for maior do que 0,6, as imagens não são do mesmo indivíduo e o acesso não é liberado.

Fotografia 6 – Ilustração para condição de “usuário cadastrado” do software

Fonte: Próprio Autor (2019)

O software também permitiu gerar arquivos de texto e arquivos CSV (*Comma-Separated Values*), possibilitando salvar um conjunto de informações relevantes. O arquivo de texto é utilizado para possibilitar uma visualização rápida e o arquivo CSV é utilizado para a visualização dos dados em uma planilha. Os dados salvos consistem em: número do registro efetuado (para contar quantos registros foram feitos), nome do usuário, dia, horário e menor distância registrada entre a imagem capturada e o banco de imagens dos usuários cadastrados. Caso o usuário seja desconhecido (menor distância maior do que 0,6) ao invés de registrar o nome do usuário é registrada a mensagem “Indivíduo desconhecido”. Para evitar e geração de arquivos com um número demasiado de registros, um novo arquivo é gerado à cada 1000 registros. Para título de exemplificação, o Quadro 5 ilustra os 5 primeiros registros implementados pelo software.

Caso mais de um indivíduo permaneça em frente a câmera (por mais de 3 segundos) o sistema implementa a identificação do indivíduo centralizado na imagem e implementa uma demarcação com coloração diferenciada para o(s) outro(s) indivíduo(s), conforme a Fotografia 7. Caso todos os indivíduos permaneçam nas extremidades da imagem, o sistema implementa

Quadro 5 – Primeiros registros implementados pelo Software

Id	Dia	Hora	Usuário	Distância
001	23/10/2019	11:10	Reinaldo Salla	0,43
002	23/10/2019	11:13	Lourdes Borges	0,41
003	23/10/2019	11:15	Flavio Salla	0,47
004	23/10/2019	11:45	Reinaldo Salla	0,39
005	23/10/2019	12:01	Indivíduo Desconhecido	0,72

Fonte: Próprio Autor (2019)

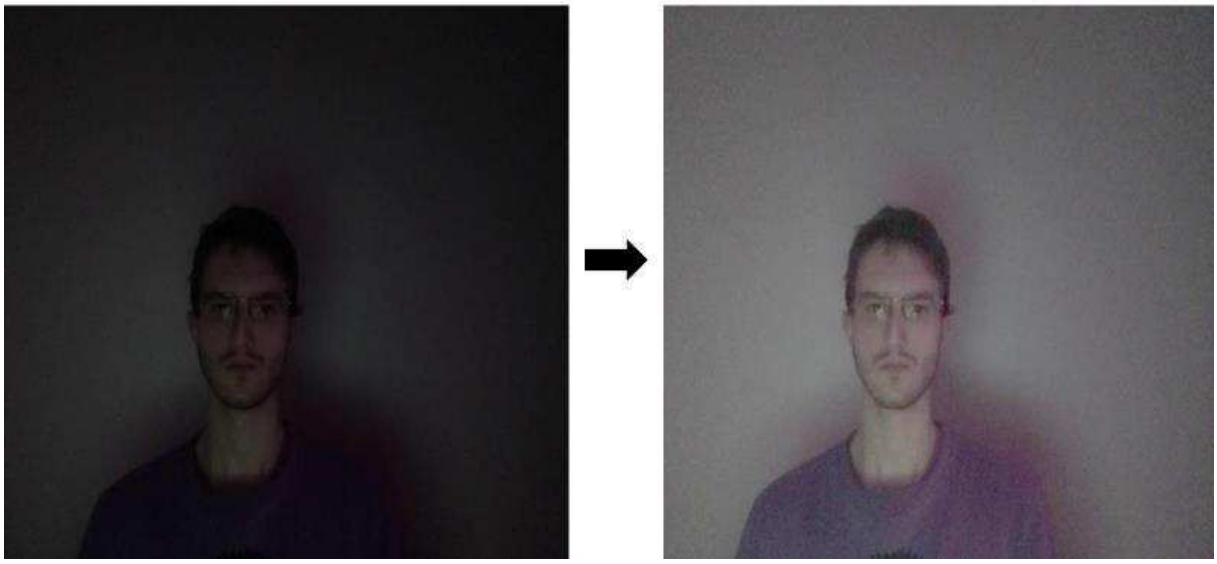
a demarcação diferença para todos os indivíduos. Assim, para múltiplos usuários, foi efetuada a identificação de apenas um indivíduo, e este indivíduo deveria estar centralizado na imagem pois, para controlar o acesso de um ambiente restrito, espera-se que apenas um indivíduo acesse o ambiente por vez, e este indivíduo esteja próximo da fechadura eletrônica.

Fotografia 7 – Comportamento do Software para mais de um indivíduo

Fonte: Próprio Autor (2019)

É importante salientar que foi implementada uma correção gamma na imagem, quando a imagem era capturada em um ambiente com um baixo nível de luminosidade. A correção gamma permite compensar as propriedades associadas à percepção e otimiza a distribuição de intensidades da imagem (SOSA, 2018). A detecção de uma imagem com baixo nível de luminosidade foi feita calculando o valor médio dos pixels da imagem. Assim, obtendo-se um valor médio de pixels abaixo de 20, efetuava-se um ajuste gamma com fator de 2,5, conforme pode ser verificado na Fotografia 8. Desta forma, o sistema também poderia ser utilizado em ambientes com um baixo nível de luminosidade.

Fotografia 8 – Ajuste gamma implementado em imagens com baixo nível de luminosidade



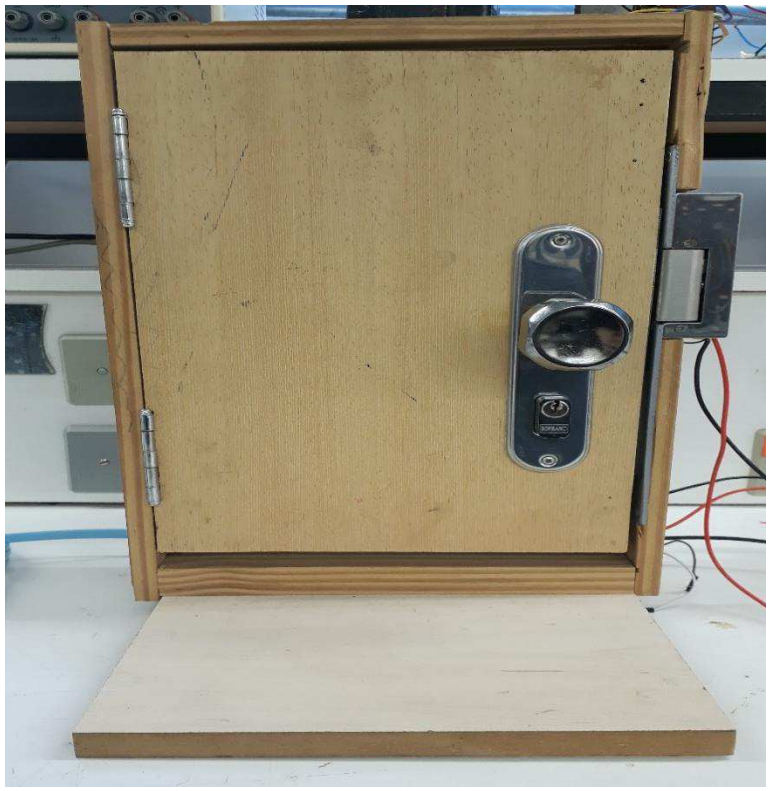
Fonte: Próprio Autor (2019)

4.5 FECHADURA ELETRÔNICA E HARDWARE

A fechadura eletrônica utilizada neste projeto pode ser verificada na Fotografia 9 Foi necessária a geração de um pulso com período de 2 segundos e amplitude de 12 volts para acionar esta fechadura.

Assim, assumindo que a tensão na saída dos pinos dos Raspberry Pi consiste em 3,3 volts, foi utilizado um conversor Boost para conectar o Raspberry Pi à fechadura eletrônica. Além disso, também foi utilizado um optoacoplador para isolar os componentes. Adotou-se o optoacoplador TLP250. Considerando que a corrente de entrada deste circuito integrado possui um valor máximo de 20 mA, se fazia necessário utilizar um resistor com resistência mínima de 165 Ω, conforme pode ser verificado na Equação (12). Logo, foi usado um resistor de 330 Ω.

Fotografia 9 – Fechadura eletrônica utilizada neste projeto



Fonte: Próprio Autor (2019)

$$R > \frac{V}{I_{max}} \tag{12}$$

$$R > \frac{3{,}3}{0{,}020}$$

$$R > 165\ \Omega$$

Onde:

R: Resistor na entrada do optoacoplador

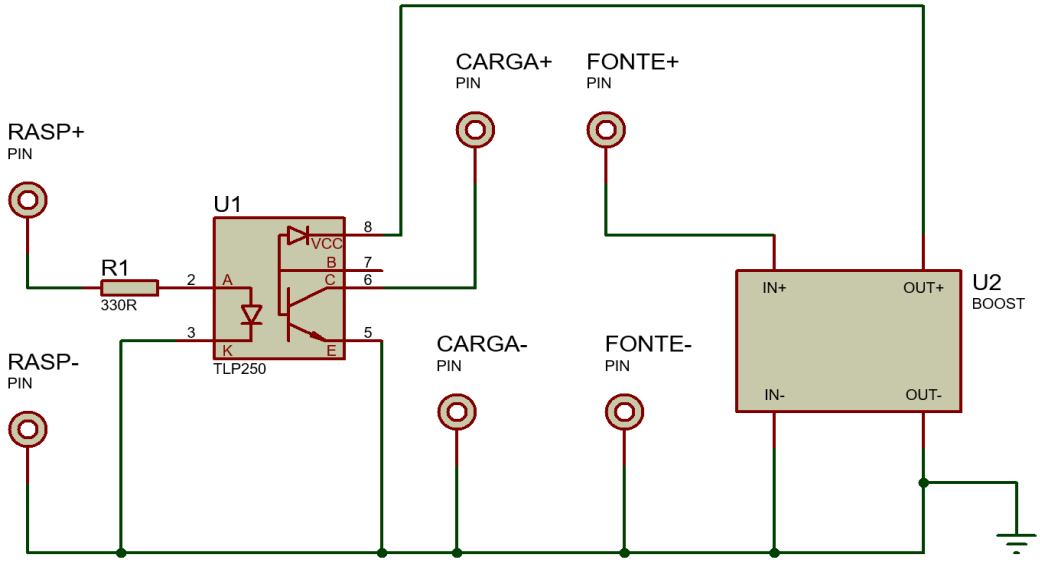
I_{max}: Máxima corrente na entrada do optoacoplador

V: Tensão na saída do Raspberry Pi

O esquemático do circuito desenvolvido pode ser verificado na Figura 28 e o placa de circuito impressa pode ser verificada na Fotografia 10. Foi necessário implementar o uso de uma fonte isolada para alimentar o conversor Boost. O Raspberry Pi possui 5 volts de alimentação, e foram feitas tentativas para utilizar o próprio Raspberry Pi e evitar utilizar uma

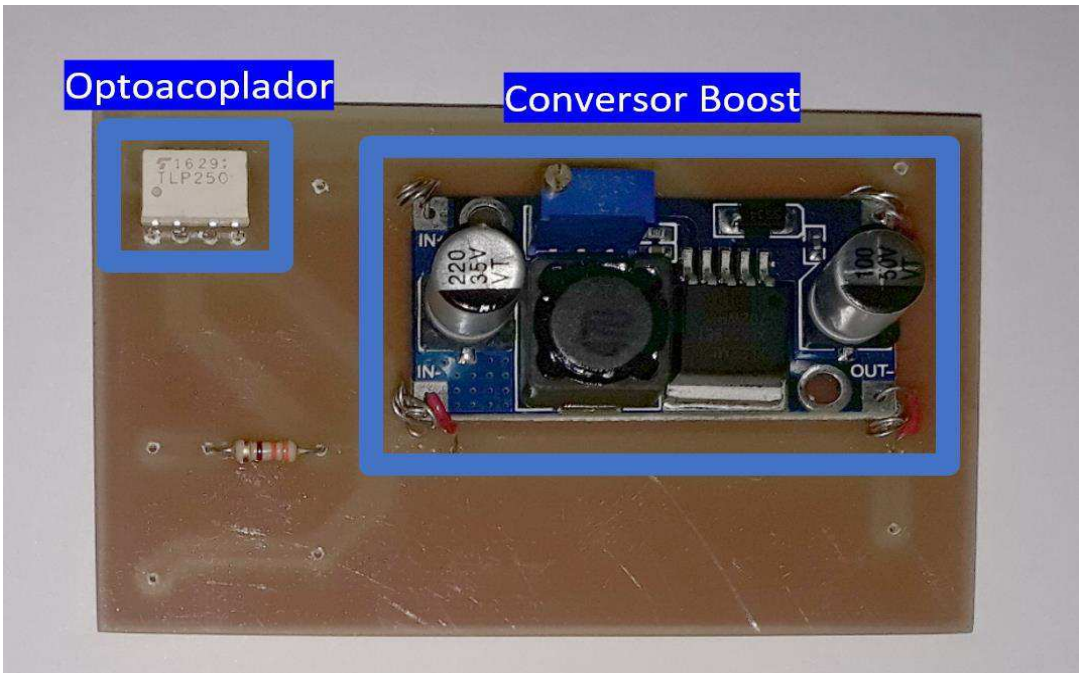
fonte separada, porém, neste caso, quando a fechadura eletrônica era acionada, a tensão de alimentação do Raspberry Pi era reduzida (passava de 5 volts para 3,5 volts) e o microcomputador entrava em modo de hibernação. Desta forma, foi utilizado um carregador de celular para promover a utilização de uma fonte isolada neste projeto.

Figura 28 – Esquemático do circuito desenvolvido neste projeto



Fonte: Próprio Autor (2019)

Fotografia 10 – Placa de circuito impressa desenvolvida neste projeto

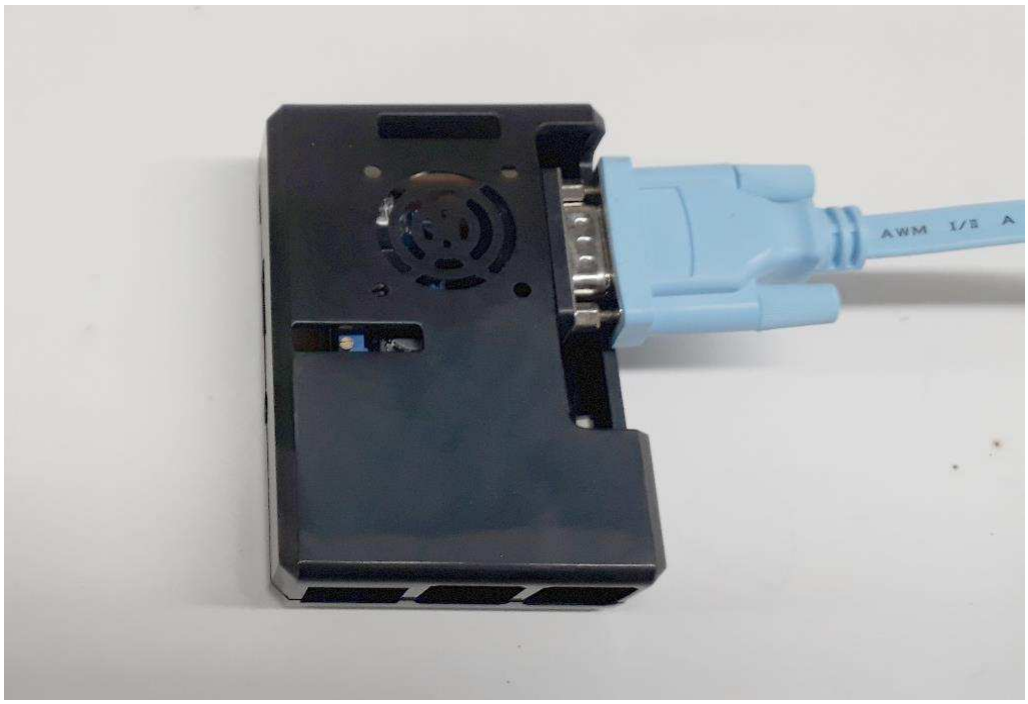


Fonte: Próprio Autor (2019)

É importante salientar que, para aumentar a performance do Raspberry Pi, a velocidade de clock do microcomputador foi elevado de 700MHz (valor padrão do dispositivo) para 1,3 GHz (máximo valor recomendado na documentação oficial do fabricante). Esta operação é chamada de *overclock*.

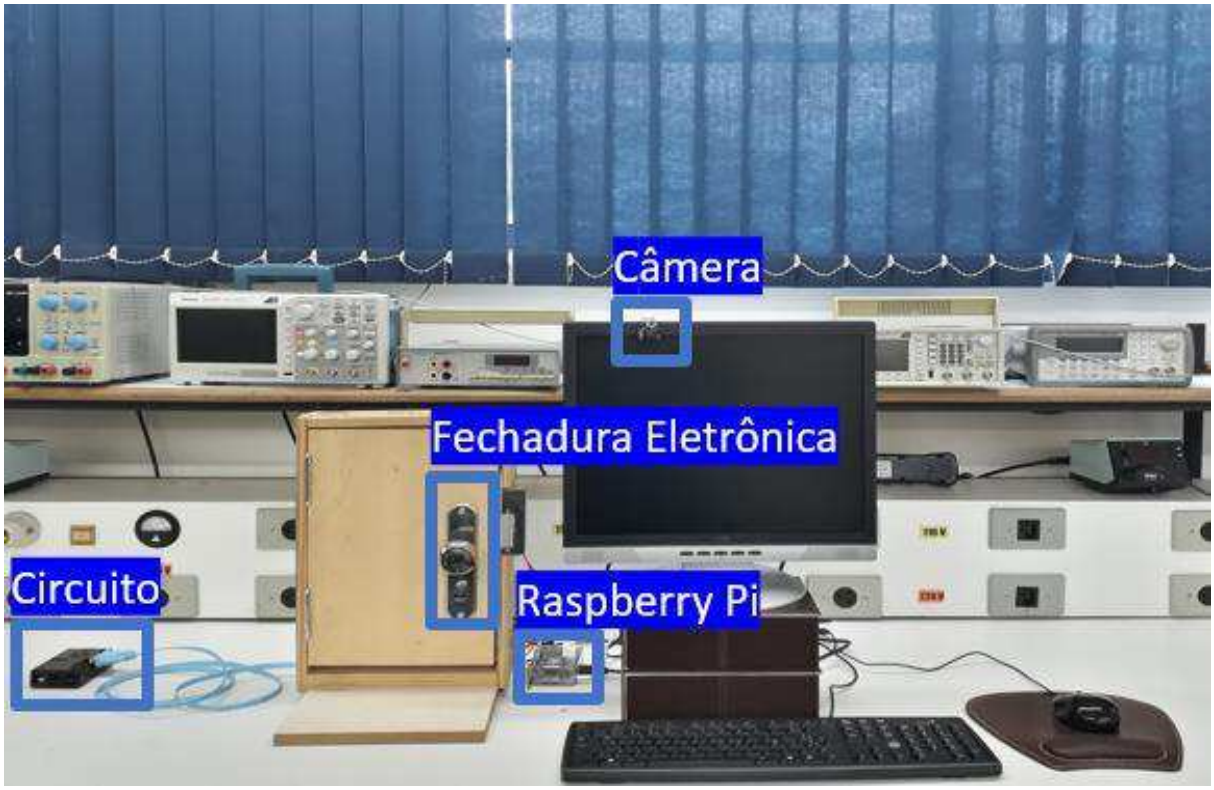
A Fotografia 11 permite observar que o circuito desenvolvido foi inserido em uma caixa e foi utilizado um cabo rollover para conectar o circuito com os diferentes componentes do sistema. O sistema completo, com todos os componentes interligados, pode ser verificado na Fotografia 12.

Fotografia 11 – Circuito inserido em uma caixa e conexão com cabo rollover



Fonte: Próprio Autor (2019)

Fotografia 12 – Interligação de todos os componentes do projeto



Fonte: Próprio Autor (2019)

5 RESULTADOS E DISCUSSÃO

Este capítulo contém os resultados para a detecção facial utilizando o algoritmo Viola-Jones, para a identificação facial implementada através da rede neural desenvolvida com a biblioteca Tensorflow e para a identificação facial utilizando a biblioteca *Face Recognition*.

5.1 RESULTADOS PARA DETECÇÃO FACIAL

Para avaliar o desempenho do algoritmo Viola-Jones, foram utilizados 3 critérios: velocidade de processamento, detecção com relação à variação de luminosidade e detecção de fraudes (fotos em papel ou celular) descritos a seguir.

5.1.1 Velocidade de processamento

O sistema projetado demorava aproximadamente 0,05 segundo para gerar 1 frame, quando o algoritmo estava detectando uma face. Assim, através da Equação (13), foi possível obter uma taxa em frames de aproximadamente 20 FPS (*Frames per Second*). Desta forma, é possível concluir que o algoritmo Viola-Jones é apropriado para aplicações em tempo real, e a velocidade de processamento do sistema foi satisfatória, apresentando uma latência desprezível quando o algoritmo está implementando detecção facial.

$$frame\ rate = \frac{1}{T} \tag{13}$$

$$frame\ rate = \frac{1}{0.05} = 20\ FPS$$

Onde:

T: Tempo para a geração de um frame

5.1.2 Detecção com relação à variação de luminosidade

A utilização do algoritmo Viola-Jones permitiu implementar a detecção em ambientes com diferentes níveis de luminosidade. Entretanto, dependendo do ambiente, a detecção ocorreu com maior ou menor facilidade. Por exemplo, em ambientes com nível de luminosidade muito alto ou muito baixo, a detecção ocorre com maior dificuldade. É importante salientar que esta dinâmica não impediu a detecção facial, porém, dificultou o processo. Desta forma, é

possível concluir que o algoritmo Viola-Jones não se comportou de forma satisfatória neste quesito, pois desejava-se que a detecção ocorresse de forma uniforme em todos os locais.

5.1.3 Detecção de fraudes

Para um sistema de reconhecimento facial, principalmente em um contexto envolvendo segurança, é desejável que o software seja capaz de diferir imagens reais de imagens em papel ou imagens provenientes de celulares (neste trabalho, imagens reais são consideradas imagens que não são provenientes de papel ou de celulares).

O algoritmo Viola-Jones, neste projeto, não implementou a detecção de imagens em papel. Portanto, o sistema não foi vulnerável neste quesito. Entretanto, o algoritmo implementou a detecção de imagens em celulares.

Uma tentativa de solução para este problema consistiu em analisar a intensidade luminosa de imagens, pois imagens provenientes de celulares possuem uma intensidade luminosa maior do que imagens reais. Assim, convertendo uma imagem RBG para uma imagem monocromática e obtendo um valor médio dos pixels, procurou-se implementar uma separação entre imagens reais e imagens provenientes de celulares.

Porém, esta solução não é ideal, pois depende da luminosidade de um ambiente. Por exemplo, caso o celular seja ajustado para exibir um nível de baixa luminosidade, o sistema se tornaria mais vulnerável. Além disso, em um ambiente, por exemplo, com lâmpadas industriais, a diferença de intensidade luminosa entre imagens reais e imagens de celulares pode ser pequena. Desta forma, é possível concluir que a análise da intensidade luminosa consistiu em uma solução demasiadamente simples para um problema complexo. Conforme vêm sendo abordada na literatura moderna de reconhecimento facial (XU et al., 2016; CHAKRABORTY; DAS, 2014), a detecção de fraudes não é um problema trivial e necessita de métodos avançados de processamento sobre a imagem para ser solucionado.

5.2 RESULTADOS PARA IDENTIFICAÇÃO FACIAL

Ambos os modelos (rede neural siamesa desenvolvida com TensorFlow e modelo disponibilizado pela biblioteca F*ace Recognition*) foram testados sobre os 20.250 pares gerados através do conjunto de dados MUCT. Não foram aplicados ajustes manuais de luminosidade ou alinhamento facial para a 1ª abordagem. Também, não foi aplicado nenhum ajuste referente a luminosidade para testar a 2ª abordagem (o ajuste gamma foi aplicado somente nas imagens para o sistema no Raspberry Pi, operando em tempo real). Porém, é importante salientar que a

biblioteca *Face Recognition* implementa um alinhamento facial, conforme abordado na ítem 4.3.

Assim, o Quadro 6 permite verificar os resultados para as duas abordagens. A 2ª abordagem possui uma acurácia 13,03% maior do que a 1ª abordagem. Assumindo que um sistema em tempo real pode ser propenso a diversos tipos de problemas (luminosidade, variação geométrica, expressões faciais, distância com relação à câmera, utilização de acessórios), a 2ª abordagem foi mais apropriada para um sistema funcional prático.

Quadro 6 – Resultados para identificação facial

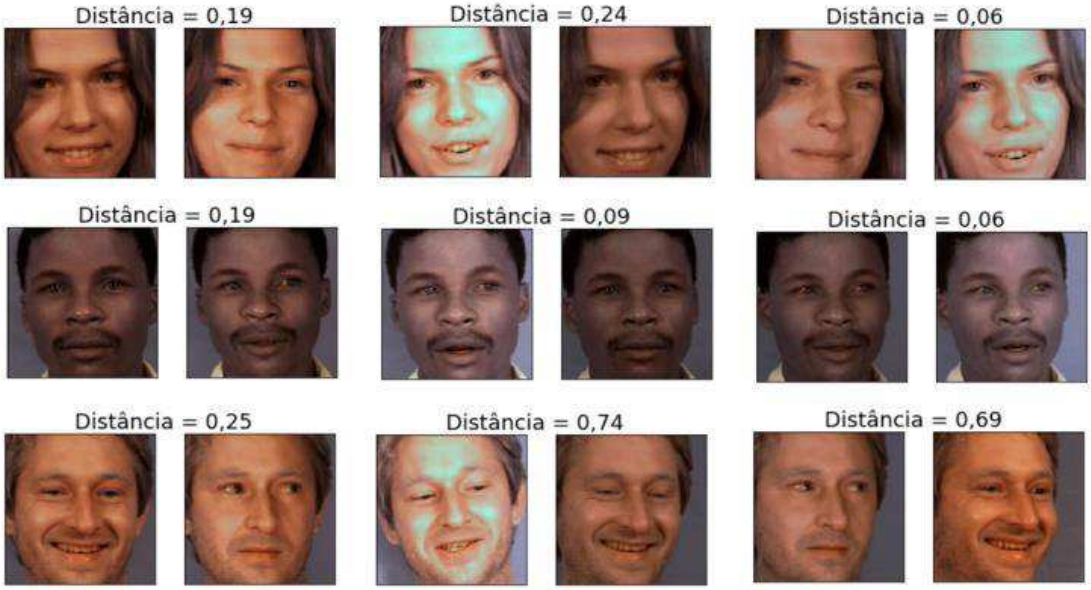
Método	1ª abordagem	2ª abordagem
Nº total de pares	20.250	20.250
Nº total de pares positivos	10.125	10.125
Nº de pares falsos positivos	572	163
Nº total de pares negativos	10.125	10.125
Nº de pares falsos negativos	2.510	279
Nº total de erros	3.082	442
Acurácia	84,78%	97,81%

Fonte: Próprio Autor (2019)

Referente ao sistema implementado em tempo real, utilizando o Raspberry Pi, foram cadastrados 8 usuários entre alunos e professores da universidade, assim como 15 indivíduos de outros locais além da universidade. Dentre todos os 23 usuários, o sistema não cometeu nenhum erro. Devido a questões legais, não é possível demonstrar imagens destes indivíduos (PRADO, 2018). Desta forma, para possibilitar visualizar os resultados da 1ª abordagem, de forma direta, utilizando o conjunto de dados MUCT, foram geradas as Figura 29 e 30. É possível observar que há uma distinção entre os valores das distâncias obtidas para pares positivos e para pares negativos. Alguns pares positivos chegaram a obter um valor menor do que 0,1, e alguns pares negativos chegaram a obter um valor maior do que 2. A distância média para pares positivos (avaliando todos os 10.125 pares) foi de 0,27, e a distância média avaliando todos os pares negativos foi de 1,38. Porém, foi possível verificar que alguns pares, mesmo semelhantes, obtiveram valores distintos. Por exemplo, na Figura 30, o par da 1ª linha com a 1ª

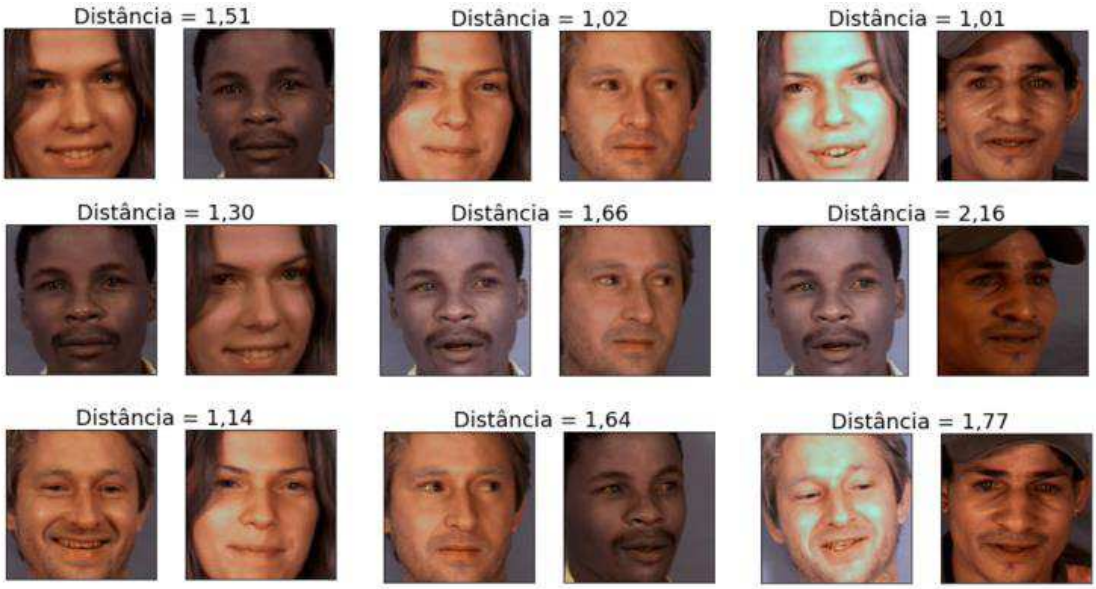
coluna e o par da 2ª linha com a 1ª coluna obtiveram um valor diferente para a distância, mesmo considerando que possuem uma imagem igual e outra imagem levemente modificada.

Figura 29 – Resultados para pares positivos obtidos através da 1ª abordagem



Fonte: Próprio Autor (2019)

Figura 30 – Resultados para pares negativos obtidos através da 1ª abordagem

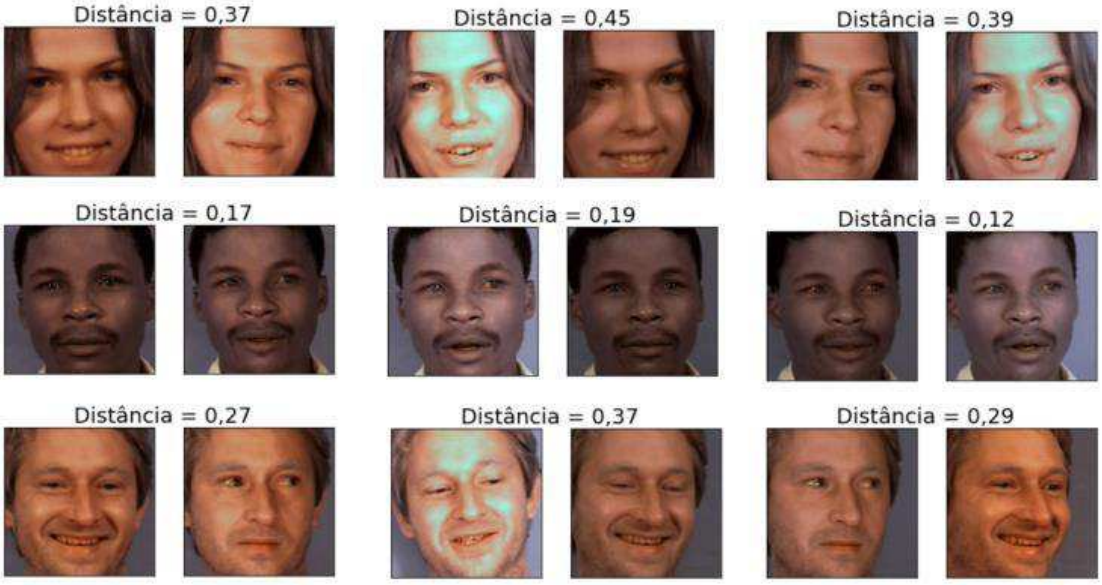


Fonte: Próprio Autor (2019)

As Figuras 31 e 32 permitem verificar os resultados para a 2ª abordagem. O valor médio obtido, avaliando todos os pares positivos, foi de 0,41. O valor médio para todos os pares negativos foi de 0,98. É possível verificar que, para a 2ª abordagem, a faixa de distância entre

pares positivos e pares negativos foi menor, em comparação com a 1ª abordagem. Entretanto, conforme demostrado no Quadro 6, esta abordagem cometeu menos erros. Além disso, pares de imagens semelhantes obtiveram valores iguais ou próximos, conforme pode ser verificado na Figura 32 (é possível observar que o par da 1ª linha com a 1ª coluna apresentou a mesma distância do par da 2ª linha com a 1ª coluna). Portanto, a 2ª abordagem foi mais estável para implementar identificação facial.

Figura 31 – Resultados para pares positivos obtidos através da 2ª abordagem



Fonte: Próprio Autor (2019)

Figura 32 – Resultados para pares negativos obtidos através da 2ª abordagem

Fonte: Próprio Autor

6 CONSIDERAÇÕES FINAIS

Em primeiro plano, o desenvolvimento deste projeto permitiu verificar a importância da quantidade de dados de treinamento em uma rede neural. O modelo desenvolvido neste trabalho foi treinado com 702.960 imagens. Levando em conta que o modelo desenvolvido pela biblioteca *Face Recognition* foi treinado com 3 milhões de imagens, o modelo desenvolvido pelo Google (FaceNet) foi treinado com 200 milhões de imagens (SCHOROFF, KALENICHENKO, PHILBIN, 2015) e o modelo desenvolvido pela empresa Baidu, considerado um sistema de identificação facial excelente (LIU et al., 2015), foi treinado com 260 milhões de imagens, é possível notar que há uma clara correlação entre quantidades de imagens e performance de redes neurais, principalmente RNCs.

Assim, uma possível melhoria para o trabalho consistiria na obtenção de uma quantidade maior de imagens para o treinamento da rede neural. Entretanto, não seria possível formar um conjunto de dados com centenas de milhões de imagens implementando a utilização de conjuntos de dados pré-definidos de diversas universidades (como foi utilizado neste trabalho). Desta forma, seria necessário o desenvolvimento de softwares orientados a extração ou a minagem de dados, que fossem, por exemplo, capazes de extrair imagem contendo faces de forma automática, de milhões de URLs (*Uniform Resource Locators*) da internet. A própria linguagem de programação Python possui diversas bibliotecas orientadas à minagem de dados que poderiam ser utilizados para esta aplicação, como, por exemplo, as bibliotecas Urllib, Beautiful-Soup, Requests, Selenium e Scrapy.

Outra possível melhoria consistiria em solucionar o problema relacionado com a detecção de fraudes, principalmente com relação a detecção de fotos em celulares, implementando uma solução que não dependesse da luminosidade, para aumentar o grau de segurança do sistema. Uma solução comum para este problema consiste na detecção do olho piscar (CHAKRABORTY; DAS, 2014), tecnologia denominada detecção de vivacidade (se o olho piscar é uma pessoa real, se não piscar é uma imagem proveniente de um celular e, desta forma, é uma fraude). Entretanto, esta não consiste em uma solução perfeita, pois poderia ser apresentado um vídeo em celular ao invés de uma imagem, tornando, desta forma, o sistema vulnerável a vídeos em celular. Uma solução mais avançada para este problema consistiria em analisar a textura da imagem (XU et al.), ou até mesmo utilizar uma RNC para detectar faces (GUO et al. 2018). Entretanto, é importante salientar que o principal motivo da utilização do algoritmo Viola-Jones neste trabalho foi pela necessidade de implementação do software no microcomputador Raspberry-Pi, e para utilizar um algoritmo de detecção mais complexo,

possivelmente seria necessário utilizar um computador propriamente dito, ou pelo menos um microcomputador com uma capacidade de processamento mais elevada.

De qualquer forma, mesmo considerando que o reconhecimento facial é uma tecnologia extremamente difícil de ser aplicada de forma prática (UNIVERSITY OF MASSACHUSETTS, 2018), foi possível, neste trabalho, com a utilização do algoritmo Viola-Jones para a detecção e do modelo disponibilizado pela biblioteca *Face Recognition* para identificação, desenvolver um sistema funcional de reconhecimento facial, adequado para uma aplicação em tempo real. O software desenvolvido no microcomputador Raspberri Pi[4] e o software desenvolvido no ambiente Colaboratory[5] podem ser verificados em seus respectivos repositórios.

[4] https://github.com/ReinaldoSalla/real-time-facial-recognition
[5] https://github.com/ReinaldoSalla/siamese-neural-network

REFERÊNCIAS

ABADI, Martín et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **arXiv preprint arXiv:1605.08695**, 2016. Disponível em: <https://arxiv.org/abs/1605.08695>. Acesso em: 15 mar. 2019, 06:19:55.

AMINI, Alexander. **MIT 6.S191: Introduction to Deep Learning.** 2019. 96 slides. Dísponível em: <http://introtodeeplearning.com/materials/2019_6S191_L1.pdf>. Acesso em: 01 mar. 2019, 06:21:07

AT&T LABORATORIES CAMBRIDGE, **The Database of Faces**. 2002. Dísponível em: < https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html >. Acesso em: 15 mai. 2019, 06:33:01.

BETH TECHNOLOGY. **Artificial Intelligence Podcast: Experts Discuss Critical Trends**. 2019. Disponível em: < https://beth.technology/artificial-intelligence-podcast>. Acesso em: 04 mar. 2019, 06:05:02.

BOUMA, Soren. **One Shot Learning and Siamese Networks in Keras.** 2017. Disponível em: <https://sorenbouma.github.io/blog/oneshot>. Acesso em: 09 mar. 2019, 10:01:23.

CARNEGIE MELLON UNIVERSITY. **Face Datasets**. 2019. Disponível em: <http://www.cs.cmu.edu/afs/cs/project/vision/vasc/idb/www/html/face/>. Acesso em: 08 set. 2019, 11:30:23.

CALTECH. **Computational Vision at Caltech**. 2019. Disponível em: < http://www.vision.caltech.edu/archive.html>. Acesso em: 11 set. 2019, 06:01:01.

CHAKRABORTY, Saptarshi; DAS, Dhrubajyoti. An Overview of Face Liveness Detection **arXiv preprint arXiv: 1405.2227**. 2014. Disponível em: <https://arxiv.org/abs/1405.2227>. Acesso em: 22 set. 2019, 09:01:28.

CHIACHIA, Giovani. **Learning Person-Specific Face Representations**. Tese (Doutorado em Ciência da Computação) – Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2013.

CHOLLET, François. **Deep Learning with Python**. Shelter Island: Manning Publications, 2017.

CHOPRA, Sumit; HADSELL, Raia; LECUN, Yann. In: **IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. vol. 1, p. 536-546, 2005.

COLAB. **Welcome to Colaboratory**. 2019. Disponível em: <https://colab.research.google.com/notebooks/welcome.ipynb>. Acesso em: 01 mar. 2019, 06:02:02.

DLIB C++ LIBRARY, **Features**. 2019. Dísponível em: < http://dlib.net/>. Acesso em: 01 set. 2019, 06:01:02.

ESSEX FACIAL IMAGES, **Description of the Collection of Facial Images**. 2019. Dísponível em: < https://dces.essex.ac.uk/mv/allfaces/ >. Acesso em: 17 mai. 2019, 06:39:01.

FERREIRA Anselmo; GIRALDI, Gilson. Convolutional Neural Network approaches to granite tiles classification. **Expert systems with applications**, vol. 84, p. 11, 2017.

FLORINDO, João. **Redes Neurais Convolucionais – Deep Learning.** 2018. 70 slides. Dísponível em: < https://www.ime.unicamp.br/~jbflorindo/Teaching/2018/MT530/T10.pdf> Acesso em: 08 mar. 2019, 07:01:23.

FOLHA DE S.PAULO, **99 Passa a Usar Reconhecimento Facial para Confirmar Identidade de Motoristas**. 2019 Dísponível em: < https://www1.folha.uol.com.br/tec/2019/05/99-passa-a-usar-reconhecimento-facial-para-confirmar-identidade-de-motoristas.shtml>. Acesso em: 22 mai. 2019, 07:03:01.

GEITGEY, Adam. **Face Recognition**. 2019. Disponível em: <https://github.com/ageitgey/face_recognition>. Acesso em: 12 set. 2019, 06:12:55.

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn and TensorFlow**: concepts, tools, and techniques to build intelligent systems. Sebastopol: O'Reilly Media, 2017.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge: MIT Press, 2016.

GUIMARÃES, Saulo Pereira. **Maracanã e Santos Dumont Receberão Câmeras de Reconhecimento Facial, diz Witzel**. 2019. Dísponível em: <https://www1.folha.uol.com.br/tec/2019/05/99-passa-a-usar-reconhecimento-facial-para-confirmar-identidade-de-motoristas.shtml>. Acesso em: 03 mai. 2019, 12:45:05.

GUO, Guanjun et al. A Fast Face Detection Method via Convolutional Neural Network. **arXiv preprint arXiv: 1803.10103**. 2018. Disponível em: <https://arxiv.org/abs/1803.10103>. Acesso em: 18 set. 2019, 06:00:12.

HAYKIN, Simon. **Redes Neurais:** Princípios e prática. 2ª edição, Porto Alegre: Bookman, 2001. Tradução de Paulo Martins Engel.

HE, Kaiming et al. Deep Residual Learning for Image Recognition. **arXiv preprint arXiv: 1512.03385**. 2017. Disponível em: <https://arxiv.org/abs/1512.03385>. Acesso em: 02 set. 2019, 06:03:22.

HUANG, Gary B. et al. Learning to Align from Scratch. **Advances in Neural Information Processing Systems (NIPS)**, 2012.

HUNTER, Jonh. Matplotlib: A 2D Graphics Environment. **Computing in Science and Engineering**, vol 9, p.90-95, 2007.

KANADE, Takeo. **Picture Processing by Computer Complex and Recognition of Human Faces**. Teste (Doutorado em Ciência da Informação) – Departamento da Ciência da Informação, Universidade de Kioto, Kioto, 1973.

KARPATHY, Andrej. **Convolutional Neural Networks for Visual Recognition.** 2017. <http://cs231n.github.io/convolutional-networks/>. Acesso em: 08 mar. 2019, 08:00:29.

KING, David. **High Quality Face Recognition with Deep Metric Learning**. 2017. Disponível em: < http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>. Acesso em: 03 set. 2019, 06:04:59

KHANDELWAL, Swati. **The Hacker News**. Disponível em: <http://thehackernews.com/2016/02/raspberry-pi-3-microcomputer.html>. Acesso em: 22 mar. 2019, 06:00:29.

KRIZHEVSKY, Alex.; SUTSKEVER, Ilya; HINTON, Geoffrey. Imagenet classification with Deep Convolutional Neural Networks. In: **Neural Information Processing Systems**. [S.l.:s.n.], p. 1106–1114, 2012.

KOCH, Gregory; ZEMEL, Richard; SALAKHUTDINOV, Ruslan. Siamese Neural Networks for One-shot Image Recognition. In: **International Conference on Machine Learning**, Deep Learning Workshop, vol. 2, 2015.

LECUN, Yann; BENGIO, Yoshua. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, p.255-258, 1998.

LECUN, Yann.; BENGIO, Yoshua.; HINTON, Geoffrey. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LIU, Jingtuo et al. Targeting Ultimate Accuracy: Face Recognition via Deep Embedding. **arXiv preprint arXiv: 1506.07310**. 2015. Disponível em: <https://arxiv.org/abs/1506.07310>. Acesso em: 15 set. 2019, 07:03:26.

MANSANO, Alex Fernandes. **Aprendizado de Máquina Multivisão Aplicado à Análise de Correferência em um Sistema de Aprendizado Sem Fim**. Dissertação (Mestrado em Ciência da Computação) – Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Carlos, 2018.

MENEGOLA, Afonso. **Deep Learning in Melanoma Screening**. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2018.

MILBORROW S.; MORKEL J.; NICOLLS F. The MUCT Landmarked Face Database. In: **Pattern Recognition Association of South Africa**, 2010.

NAIR, Vinod; Geoffrey, HINTON. Rectified Linear Units Improve Restricted Boltzmann Machines. In: **International Conference on Machine Learning**. p. 807-814, 2010.

NUMPY DEVELOPERS. **NumPy User Guide**. 2019. Disponível em: <http://www.numpy.org/>. Acesso em: 16 mar. 2019, 07:19:22.

NVIDIA CORPORATION. **Nvidia Tesla K80**. 2019. Disponível em: <https://www.nvidia.com/pt-br/data-center/tesla-k80/>. Acesso em: 18 mar. 2019, 07:39:47.

NWANKPA, Chigozie et al. Activation Functions: Comparison of Trends in Practice and Research For Deep Learning. **arXiv preprint arXiv:1811.03378**. 2018. Disponível em: <https://arxiv.org/abs/1811.03378>. Acesso em: 07 mar. 2019, 13:01:22.

OPENCV TEAM. **OpenCV 2.4.13.7 documentation**. 2019. Disponível em: <https://opencv.org/>. Acesso em: Acesso em: 17 mar. 2019, 11:19:22.

OCTAVIANO, Adriana Arruda. **Deep learning é a tecnologia de aprendizado de máquina que mais cresce em todo o mundo**. 2019 Disponível em: <https://www.inova.unicamp.br/noticia/deep-learning-e-tecnologia-de-aprendizado-de-maquina-que-mais-cresce-em-todo-o-mundo>. Acesso em 11 mai. 2019, 07:22:11.

PHILLIPS, Jonathan. **Color FERET Database**. 2016. Disponível em: <https://www.nist.gov/itl/iad/image-group/color-feret-database >. Acesso em: 05 set. 2019, 09:39:47.

PHILLIPS, Jonathan et al. The FERET program. In: **Proceedings of Office of National Drug Control Policy, CTAC International Technology Symposium**, 1997, p. 8–11.

PRADO, Kelvin Salton do. **Comparação de técnicas de reconhecimento facial para identificação de presença em um ambiente real e semicontrolado**. Dissertação (Mestre em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo 2018.

PYTHON SOFTWARE FOUNDATION. **Python 3.7.3 documentation**. 2019. Disponível em: <https://docs.python.org/3>. Acesso em: 10 mar. 2019, 08:01:23.

RAJGOPAL, Venkat. **Understanding The Deep Learning Framework Behind Apple Face Id**. 2018. Disponível em: <https://medium.com/@venkatrajgopal/understanding-the-deep-learning-framework-behind-apple-face-id-6372612397ac>. Acesso em: 05 mai. 2019, 06:35:23.

RASPBERRY PI FOUNDATION. **Products**. 2019. Disponível em: <https://www.raspberrypi.org/products/>. Acesso em: 22 mar. 2019, 07:04:27.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**. American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUDER, Sebastian. An Overview of Gradient Descent Optimization Algorithms. **arXiv preprint arXiv:1609.04747**. 2016. Disponível em: <https://arxiv.org/abs/1609.04747>. Acesso em: 08 mar. 2019, 11:01:23.

KOCH, Gregory; ZEMEL, Richard; SALAKHUTDINOV, Ruslan. Siamese Neural Networks for One-shot Image Recognition. In: **International Conference on Machine Learning**, Deep Learning Workshop, vol. 2, 2015.

SAMARIA F.; HARTER A. Parameterizations of a stochastic model for human face identification. **In: 2nd IEEE Workshop on Applications of Computer Vision**, p.138-142, 1994.

SCHMIDHUBER, Juergen. Deep learning in neural networks: An overview. **Neural Networks Elsevier**, v. 61, p. 85–117, 2015.

SCHMIDT, Blake. **China's Mass Surveillance More Sophisticated Than Thought**. 2019. Disponível em: < https://finance.yahoo.com/news/alibaba-backed-face-scans-show-210100466.html>. Acesso em: 06 mai. 2019, 09:44:17.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: A Unified Embedding for Face Recognition and Clustering. **arXiv preprint arXiv:1503.03832**. 2015. Disponível em: <https://arxiv.org/abs/1503.03832>. Acesso em: 02 mai. 2019, 06:44:13.

SOSA, Blanca Rosa Maquera. **Conceitos Básicos para o Processamento Digital de Imagens**. Apostila da disciplina Processamento Digital de Sinais do curso de Engenharia Elétrica da Universidade de Passo Fundo, 2018.

STROUSTRUP, Bjarne. **Programming**: Principles and Practice Using C++. 2.ed. New Jersey: Addison-Wesley Professional, 2014.

SUTTON, Richard; BARTO, Andrew. **Reinforcement Learning:** An Introduction. 2.ed. Cambridge: MIT Press, 2018.

TABACOF, Pedro. **Exploring Adversarial Images in Deep Neural Networks**. Dissertação (Mestrado em Engenharia de Computação) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2017.

TAIGMAN, Yaniv et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. **IEEE Conference on Computer Vision and Pattern Recognition**, p. 1701-1708, 2014.

TENSORFLOW GUIDE. **Writing layers and models with TensorFlow Keras**. 2019. Disponível em: <https://www.tensorflow.org/alpha/guide/keras/custom_layers_and_models>. Acesso em: 21 mai. 2019, 11:44:22.

UCSD COMPUTER VISION. **Extended Yale Face Database B**. 2019. Disponível em: <http://vision.ucsd.edu/content/extended-yale-face-database-b-b>. Acesso em: 06 set. 2019, 10:31:21.

UNIVERSITY OF MASSACHUSETTS. **Label Faces in the Wild Home**. 2018. Disponível em: <http://vis-www.cs.umass.edu/lfw/>. Acesso em: 21 set. 2019, 08:30:22.

VANDERPLAS, Jake. **Python data science handbook**: essential tools for working with data. Sebastopol: O'Reilly Media, 2017.

VIOLA, Paul; JONES Michael. Rapid Object Detection using a Boosted Cascade of Simple Feature. In: **IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, vol. 1, p. 511-518, 2001.

WEST, Jesse Davis. **History of Facial Recognition**. 2017. Disponível em: <https://www.facefirst.com/blog/brief-history-of-face-recognition-software>. Acesso em: 12 mai. 2019, 06:30:23.

XU, Yi et al. Virtual U: Defeating Face Liveness Detection by Building Virtual Models from Your Public Photos. In: **USENIX Security Symposium**, 2016.

ZEILER, Matthew. ADADELTA: An Adaptive Learning Rate Method. **arXiv preprint arXiv: 1212.5701**. 2012. Disponível em: <https://arxiv.org/abs/1212.5701>. Acesso em: 04 set. 2019, 07:01:23.