UNIVERSIDADE FEDERAL DO ABC

GUSTAVO BRITO DA SILVA

GUSTAVO RODRIGUES PAES

VICTOR REIS FONTES DE OLIVEIRA

Desenvolvimento de um protótipo de Porta Automática controlada através de técnicas de robótica e visão computacional para reconhecimento da utilização correta de máscaras.

Santo André – SP

2021

GUSTAVO BRITO DA SILVA

GUSTAVO RODRIGUES PAES

VICTOR REIS FONTES DE OLIVEIRA

Desenvolvimento de um protótipo de Porta Automática controlada através de técnicas de robótica e visão computacional para reconhecimento da utilização correta de máscaras.

Trabalho apresentado como requisito parcial para a conclusão do curso de Engenharia de Instrumentação, Automação e Robótica da Universidade Federal do ABC.

Orientador: Prof. Dr. Alexandre Acácio de Andrade.

Coorientador: Prof. Dr. Luneque Del Rio de Souza e Silva Junior

Santo André – SP

2021

AGRADECIMENTOS

A nossos pais e mães que nos apoiaram durante toda a jornada da vida e nos forneceram as condições para que chegássemos aqui.

Aos nossos amigos, que proporcionaram tantos momentos de alegria durante todos estes anos.

A Universidade Federal do ABC e seus professores, que nos proporcionaram uma visão completamente nova do mundo, preparando-nos para os desafios da vida profissional.

Ao James, meu cachorro que esteve sempre ao meu lado.

RESUMO

O reconhecimento de imagens é um desafio realizado pelo homem a décadas, onde este processo está se tornando mais fácil a cada dia, isso devido a proporção e quantidade de dados que possuímos atualmente, além do desafio do reconhecimento das imagens temos o desafio da captura das imagens com qualidade por meio de câmeras que devem identificar o indivíduo e capturar a imagem adequada para a análise. Visando a segurança e controle nos locais decorrente a saúde pública e o combate ao COVID-19. O trabalho descreve a utilização de um sistema para captura de imagens e detecção da utilização de máscaras através de ferramentas de *Machine Learning*. Para realizar a análise das imagens capturadas foram utilizadas ferramentas de aprendizado de máquina, como Tensorflow e OpenCV, aliados com bibliotecas de imagens, tornando possível desenvolver um modelo para classificação de imagens de usuários que faziam uso ou não de proteção facial adequada. O modelo obteve uma boa performance com uma acurácia de cerca de 90%. A partir disso, o sistema gera uma probabilidade de classificação e envia os resultados para o *hardware*, que de acordo com o *output* fornecido abrirá ou não o mecanismo que controla a porta.

Palavras-chave: Classificador, Protótipo, Visão computacional, Redes Neurais artificiais, Máscara, Covid-19

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ARM - *Advanced RISC Machine (*Máquina Avançada RISC*)*

CPU - Central Process Unit (Unidade Central de Processamento)

GPIO - General Purpose Input Output (Entrada/Saída de Propósito Geral)

GPU - Graphics Processing Unit (Unidade de Processamento Gráfico)

HDMI - High-Definition Multimedia Interface (Interface Multimídia de Alta Definição)

HD - Hard disk (Disco Rígido)

LED - Light Emitting Diode (Diodo Emissor de Luz)

SD - Secure Digital Card (Cartão de Segurança Digital)

TPU - Tensor Processing Unit (Unidade de Processamento de Tensor)

USB - Universal Serial Bus (Barramento Serial Universal)

OMS - Organização Mundial da Saúde

OpenCV - *Open Source Computer Vision Library* (Biblioteca de Código Aberto para Visão Computacional)

SoC - System-on-a-chip (Sistema-em-um-chip)

EPI - Equipamento de Proteção Individual

Sumário

1. INTRODUÇÃO

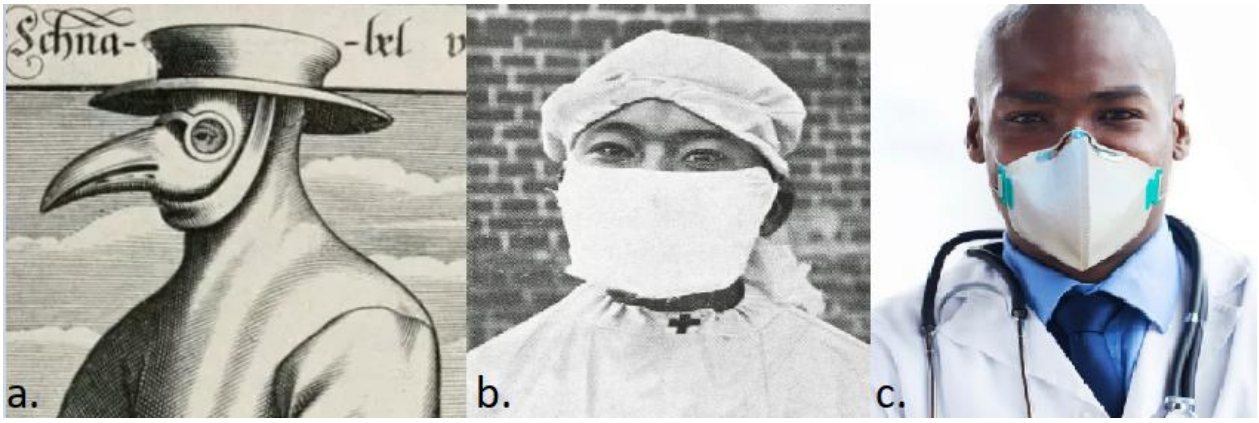
Devido a pandemia causada pelo vírus SARS-CoV-2, já foram registrados mais de 170 milhões de casos de coronavírus com mais de 3,5 milhões de mortes registradas no mundo até maio de 2020 (Worldometer, 2021). Segundo a OMS (Organização Mundial da Saúde, 2020) a transmissão do SARS-CoV-2 é feita principalmente por meio de gotículas respiratórias e por contato físico. A prática do uso de máscara já data milhares de anos para os mais variados fins, como por exemplo a utilização de máscaras feitas de pele de carneiro por Plínio, o Velho para proteção contra gases, a mais de dois mil anos atrás, ou também a clássica máscara de bico utilizado por médicos durante a peste negra sob o pretexto de purificar o ar contaminado, que era de onde se acreditava que se vinham as doenças (MUSSAP, 2019).

O uso médico de máscaras de proteção é mais recente, com a descoberta dos microrganismos por Louis Pasteur e a consequente conclusão de que microrganismos poderiam ser agentes infecciosos (BERCHE, 2012). Durante a Gripe Espanhola, em 1918, a máscara foi item de proteção essencial, sendo item fundamental para contenção da pandemia, desde então houve diversos avanços na tecnologia de construção de máscaras faciais visando a menor permeabilidade dos microrganismos, como as máscaras Pff2 desenvolvidas pela equipe do pesquisador Peter Tsai (TSAI; SCHREUDER-GIBSON; GIBSON, 2002). Olhando em um panorama mais recente, foi observado na pandemia da SARS (vírus da mesma família que o SARS-COV-2), em 2003, que máscaras faciais diminuem o contágio do vírus, sendo fortemente recomendado o uso pela população em ambientes públicos para evitar a sua transmissão (EIKENBERRY et al., 2020).

As máscaras funcionam como uma barreira física que impede a liberação de gotículas no ar que ocorrem durante a fala, tosse e espirro. Estudos apontam que as máscaras faciais, quando adaptadas adequadamente, interrompem efetivamente a dispersão das partículas expelidas por meio da tosse ou espirro, impedindo a transmissão de doenças respiratórias (PEREIRA-ÁVILA et al., 2020). Segundo experimentos realizados, máscaras de pano tiveram uma eficiência de até 77%, enquanto máscaras do tipo N95 possuíram uma eficiência de até 99% contra outros tipos

de doenças transmitidas por meio da tosse (EIKENBERRY et al., 2020). Alguns dos tipos de máscara utilizados durante a história encontram-se na Figura 1.

Figura 1: (a) Mascará comum no século XVI (b) Máscara utilizada durante a gripe espanhola, em 1918 (c) Máscara PFF2



Fonte: Wikimidia, 2021

Devido aos perigos e o risco de morte proporcionados aos humanos devido ao vírus SARS-COV-2, diversos lugares passaram a adotar medidas de restrição de acesso à pessoas que não estivessem atendendo as recomendações de proteção necessárias recomendadas pela OMS (WANG et al., 2020), onde destacam-se dentro das principais restrições à utilização de máscara e a medição da temperatura corpórea para possível detecção da febre, um dos principais sintomas do COVID-19, de modo a evitar uma possível contaminação. Muitos lugares adotaram estratégias de designar um trabalhador para realizar essas conferências e medições, o que garante uma maior segurança no ambiente, contudo o trabalhador responsável por esse controle fica em uma situação de risco que poderia facilmente ser evitada com a utilização de um sistema automatizado.

O presente trabalho é sobre o desenvolvimento de um protótipo de porta automática que utilizando técnicas de visão computacional, utilizando como *hardware* o Raspberry Pi 3B e como software Python e suas bibliotecas Tensorflow e OpenCV para realizar a identificação da utilização correta das máscaras de proteção antes da abertura automática da porta, aumentando a segurança e reduzindo a disseminação do vírus.

Uma das técnicas de neural network mais comumente utilizadas para reconhecimento facial é a de visão computacional, que é a utilização da rede para classificação de objetos em imagens, em tempo real ou não (VOULODIMOS et al., 2018).

1.1. OBJETIVOS

1.1.1. OBJETIVO GERAL

Desenvolver um protótipo de Porta Automática controlada por visão computacional e controlada através de técnicas de reconhecimento de imagens para verificação da correta utilização de máscaras.

1.1.2. OBJETIVOS ESPECÍFICOS

Construção de uma porta automática feita com Raspberry Pi 3B e uma câmera, utilizando técnicas de visão computacional para permitir o protótipo identificar a utilização correta da máscara e permitir o acesso ao indivíduo detectado.

Modelamento de rede neural com OpenCV e Tensorflow para reconhecimento das máscaras de proteção.

Realização da análise da relevância dessas tecnologias para o desenvolvimento de protótipos.

2. METODOLOGIA

- Realização de levantamento bibliográfico sobre o COVID-19, algoritmos de reconhecimento facial e eletrônica.
- Levantar dados e imagens, assim como sua devida preparação, para a construção de um *dataset* de treino.
- Construção de um modelo de reconhecimento facial utilizando redes neurais que classifique de maneira adequada a utilização de máscaras de proteção facial.
- Utilizar ferramentas para execução do modelo de reconhecimento de máscaras para atuar em tempo real.
- Desenvolver um sistema físico capaz de realizar a abertura de portas utilizando o modelo previamente treinado.
- Sugerir métodos e tecnologias que poderiam ser utilizadas para aperfeiçoar o sistema construído.

2.1. Classificação de Imagens

A classificação de imagens é um tema que está em constante evolução devido a diversas aplicações que podem ser desenvolvidas através do processo de classificação, a utilização da classificação é vista principalmente na área de segurança onde está enquadrado o projeto apresentado como na área de estudo científico.

A detecção de faces em uma imagem consiste em localizar em uma imagem o rosto das pessoas que estão presentes na foto, além da captura e detecção das faces também temos uma extração de características onde é possível fazer a captura da direção dos olhos, objetos utilizados como óculos, touca e até mesmo máscara onde está baseado o desenvolvimento do projeto.

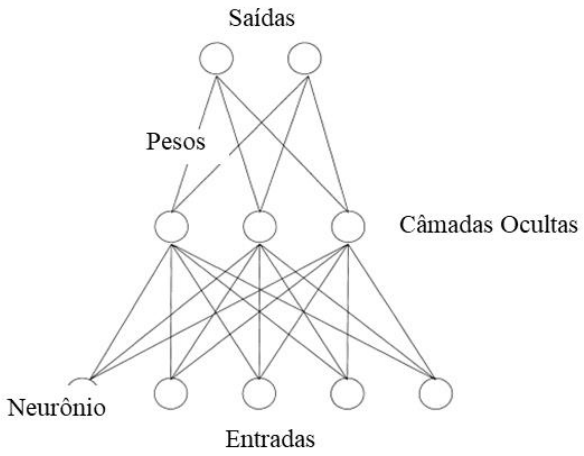
Classificação de imagens apesar de ser algo comum e muito utilizado apresenta uma série de problemas e dificuldades atreladas no processo, como a perda de foco, mudança na iluminação, tamanhos diferentes e qualidades diferentes das fotos. Todos estes fatores podem acabar atrapalhando a realização da classificação das imagens. Mudanças na Iluminação são capazes de confundir os algoritmos de reconhecimento a

ponto de considerarem uma imagem com iluminação invertida por exemplo como uma outra pessoa (ZHAO et al 2003).

2.2. Redes Neurais

Uma rede neural artificial é um método de *deep learning* que consiste em uma série de *inputs,* que constituem uma rede de neurônios de entrada, seguida de algumas camadas ocultas e, por fim, uma ou mais camadas de saída (WANG, 2003), conforme observa-se na Figura 2, estas redes são treinadas com a utilização de exemplos de entrada e saídas já conhecidas, de modo a serem criadas associações entre as características da entrada e do output do sistema, constituindo o peso dos neurônios e de suas sinapses (ABRAHAM, 2005), além de utilizar algoritmos de *back-propagation* (KARNIN, 1990).

Figura 2: Arquitetura de uma rede neural



Fonte: Wang, 2003

Conforme Wang (2003) apresenta, cada conexão está associada a um determinado peso e cada saída, h_i, do neurônio i é dada pela Equação 1.

$$h_i = \sigma(\sum_{j=1}^{N} V_{ij} x_j + T_i^{hid}) \qquad \text{Eq. 1.}$$

Onde σ é a função de transferência, N o número de neurônios, V_{ij}são os pesos, $x_j$os neurônios de entrada e T_i^{hid} o limiar dos neurônios ocultos (*hidden layers*)

3. CONFIGURAÇÕES EXPERIMENTAIS

3.1. OpenCV

O OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de código aberto usado para visão computacional e aprendizado de máquina. Foi criado para padronizar a infraestrutura de aplicativos de visão computacional. A biblioteca tem mais de 2500 algoritmos, nos quais, podem ser usados para detectar e reconhecer rostos, classificar gestos em vídeos, rastrear movimentos, rastrear e identificar objetos, produzir nuvens de pontos 3D de câmeras estéreo, entre outras utilidades.

O OpenCV tem mais de 47 mil usuários e cerca de 18 milhões de downloads. Grandes empresas como Google, Microsoft, IBM empregam a biblioteca em diversos tipos de projetos como: ajudar robôs a navegar e pegar objetos, detecção de acidentes de afogamento em piscinas na Europa e inspeção de rótulos em produtos em fábricas ao redor do mundo.

Possui implementação em C++, Python, Java e MATLAB e suporta Windows, Linux, Mac OS e Android (OpenCV, 2021).

3.2. Tensorflow

O Tensorflow é um sistema para aprendizado de máquinas que opera principalmente em larga escala, sendo utilizado principalmente para a criação de redes neurais artificiais profundas (*Deep Learning*), para os mais diversos fins, como visão computacional, robótica, processamento de imagens, séries temporais, reconhecimento de voz etc. (ABADI, 2016). Devido à sua arquitetura, é possível utilizar o poder computacional de CPU, GPU ou TPU de maneira facilmente escalável, seja em aplicações pessoais, de servidores ou até mesmo *mobile* com utilização de APIs (JIA *et al.*, 2017).

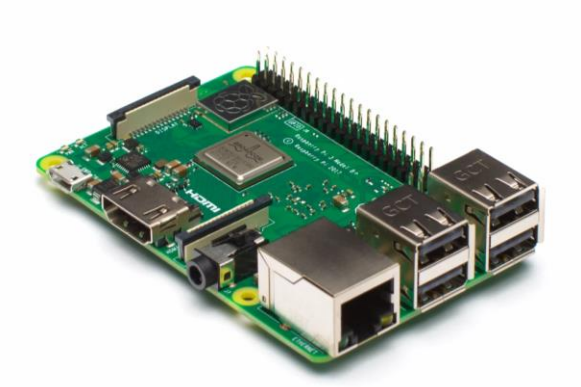
Nascendo como um projeto *open-source* nos laboratórios da Google, inicialmente composto por um time advindo de outro projeto de *machine learning* da empresa, o Sibyl, uma nova revolução começou a surgir nos campos de algoritmos, o *Deep Learning*, o que levou a equipe a iniciar o desenvolvimento de um novo *framework*. Passado o tempo,

em 2017, a primeira versão do Tensorflow foi lançada, oferecendo suporte a uma ampla variedade de aplicações e treinamento de *Deep Learning* (Tensorflow, 2021). Em 2019, o Tensorflow 2.0 foi oficialmente lançado, trazendo melhorias no processamento e novas funcionalidades, sendo a principal delas o Tensorflow eager, que altera o esquema para diferenciação automática do gráfico computacional.

3.3. Raspberry Pi

O Raspberry Pi é um computador com o custo reduzido e de forma física compacta, por conta do seu baixo custo e pelo seu porte físico ele é utilizado para diversos projetos acadêmicos e industriais, ele é produzido pela Raspberry Pi Foundation. O objetivo inicial dos criadores era utilizar o Raspberry como ferramenta de ensino devido ao seu baixo custo, porém por conta do seu tamanho e o valor acessível trouxe a difusão desse hardware mundialmente (UPTON; HALFACREE, 2016, p. 3-9). Os processadores do Raspberry utilizam arquitetura ARM, arquitetura que utiliza um conjunto reduzido de instruções, utilizando menos transistores e tendo um consumo baixo de energia (UPTON; HALFACREE, 2016, p. 3-9). Além da memória ARM o Raspberry também apresenta conexões USB, HDMI, RCA e internet. Existem diversos modelos que são distinguidos pelas suas especificações de hardware. Um exemplo deste microcomputador pode ser observado na Figura 3.

Figura 3: Raspberry Pi 3B



Fonte: Raspberry Pi, 2021

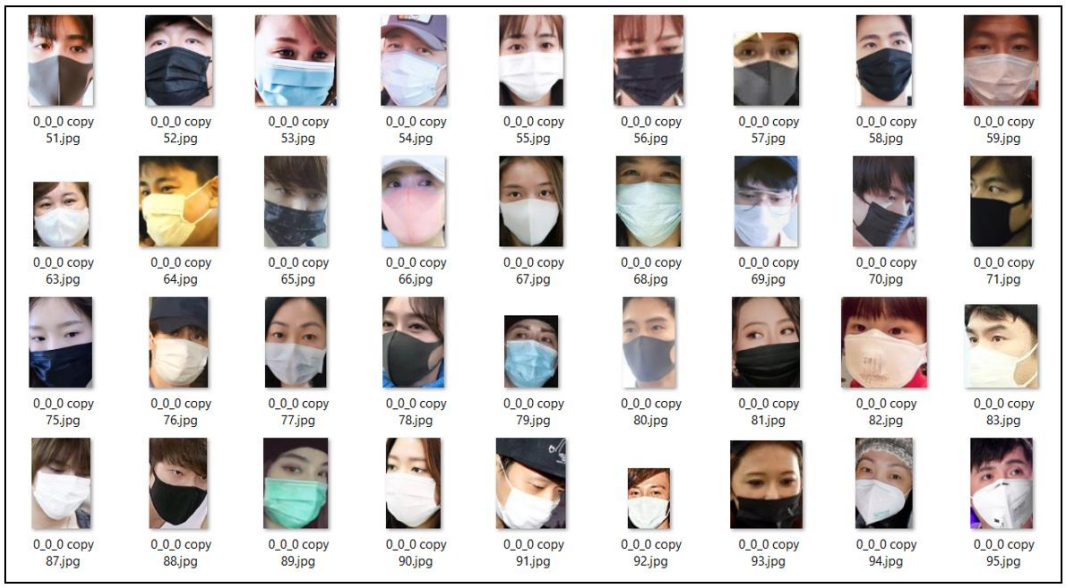
Destaca-se também como uma das principais vantagens do Raspberry a enorme comunidade (De Andrade, 2018) que se criou em torno dos dispositivos, com milhões de pessoas em todo mundo compartilhando seus projetos, códigos e configurações para o microcomputador, que vão desde utilizações simples, como um sistema que controla uma câmera de vigilância, até sistemas complexos, como para rastrear satélites pelo globo.

3.4. Dataset

O primeiro desafio foi encontrar um *dataset* que suprisse todas as necessidades idealizadas no projeto, um *dataset* que contém fotos de pessoas com máscara, sem máscara e usando máscara de forma errada. O *dataset* utilizado foi o MaskedFace-Net que possui um volume de 40GB de imagens, imagens nas quais possuem pessoas com máscara e com a utilização incorreta das máscaras. Também foi utilizado o *dataset* Flickr-Faces-HQ para selecionar algumas imagens de pessoas que não usavam máscaras, este *dataset* contém 2,76 TB de imagens com rostos de pessoas do mundo inteiro, além dos *datasets* citados também realizamos a inserção de imagens encontradas na internet com a utilização de máscaras de todas as cores e modelos com objetivo de aumentar a acurácia do modelo e não o treinar com um tipo de máscara específica, já que o nosso objetivo é abranger todo o público independente das características físicas das máscaras.

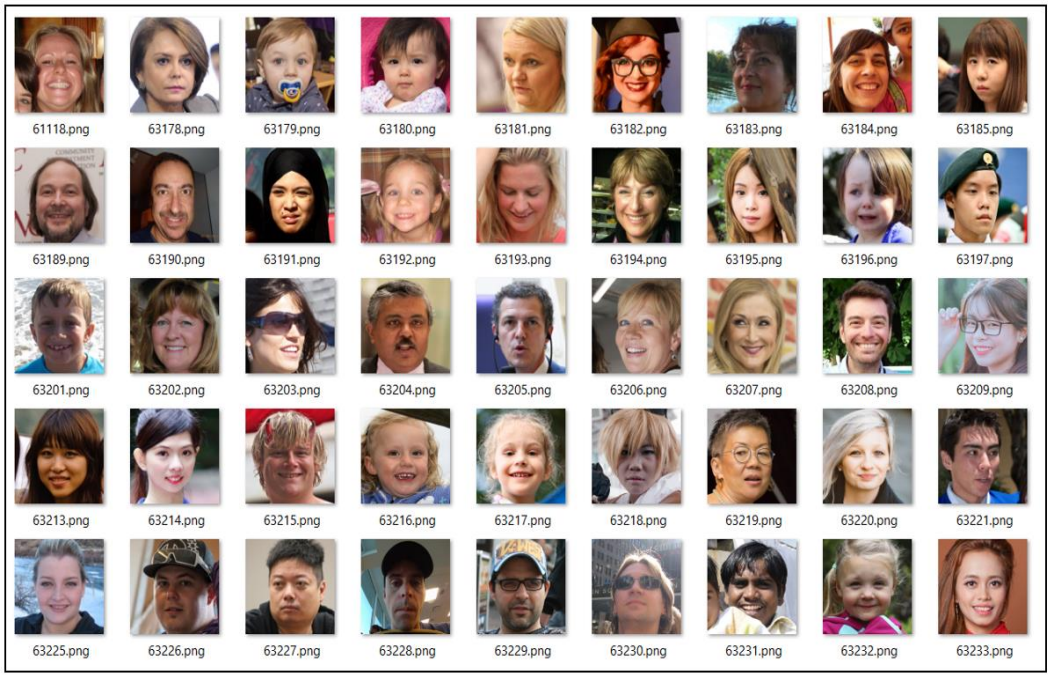
As figuras abaixo são alguns exemplos das imagens utilizadas no desenvolvimento do sistema inteligente. Foram escolhidas imagens com a maior variedade de cores e tipos de máscaras possíveis, assim como mostrado nas Figuras 4, 5 e 6, onde foi escolhido imagens de pessoas de diferentes etnias.

Figura 4: Exemplo de imagens com pessoas usando máscaras.



Fonte: LARXEL, 2021.

Figura 5: Exemplo de imagens de pessoas sem máscara.



Fonte: NVLABS, 2021.

Figura 6: Exemplo de imagens de pessoas usando máscara de forma incorreta.

Fonte: CABANI, 2021.

Dessa forma, com essas três características apresentadas, foi possível criar um modelo, baseado em um algoritmo de rede neural, que fosse capaz de classificar tanto as pessoas que estivessem usando máscara corretamente quanto as pessoas que estivessem sem máscara ou usando-a de forma incorreta. Onde, a partir dessa classificação, o sistema abrirá a porta ou não.

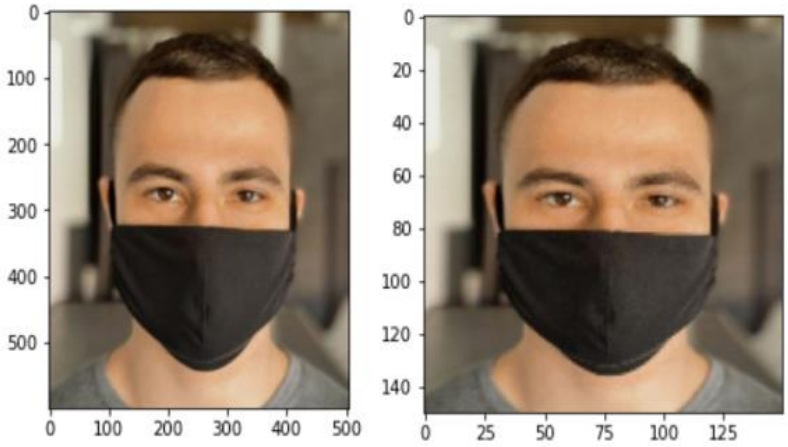
Um detalhe importante a citar sobre redes neurais, é que é possível realizar o treinamento com utilização de figuras que passaram por edição de imagem, não precisando necessariamente utilizar imagens reais para o treinamento, uma das grandes vantagens de utilizar esse método é conseguir gerar um grande *dataset* de maneira mais simples e que consegue treinar o modelo para atuar na prática.

4. IMPLEMENTAÇÃO

4.1. Pré-processamento das imagens

Após a seleção do *dataset*, o próximo passo foi a preparação das imagens, para tratar um grande volume de dados de uma maneira que mantivesse a variabilidade das imagens, o desafio foi a melhor forma de tratar esse grande volume de dados, e como solução inicial para o tratamento foi realizado uma diminuição da resolução das imagens para diminuir o tempo de processamento e para uma melhor eficiência na construção do projeto foi definido uma resolução de 150 x 150 pixels, como mostra a Figura 7.

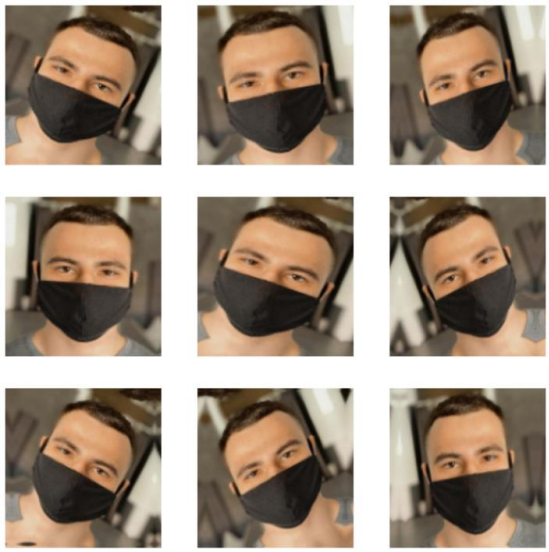
Figura 7: Resultado da alteração da resolução.



Fonte: Elaborado pelos autores, 2021.

Em seguida, foi utilizado uma técnica de *data augmentation* para aplicar efeitos aleatórios de rotação, zoom, extensão vertical/horizontal e *flip* nas imagens, aumentando a variedade de imagens usadas para treinamento do modelo. Essa técnica facilita o modelo a reconhecer as máscaras faciais, mesmo que a pessoa não se posicione de forma alinhada à câmera de detecção. A Figura 8 ilustra os possíveis efeitos:

Figura 8: Efeitos da aplicação do algoritmo de *data augmentation.*



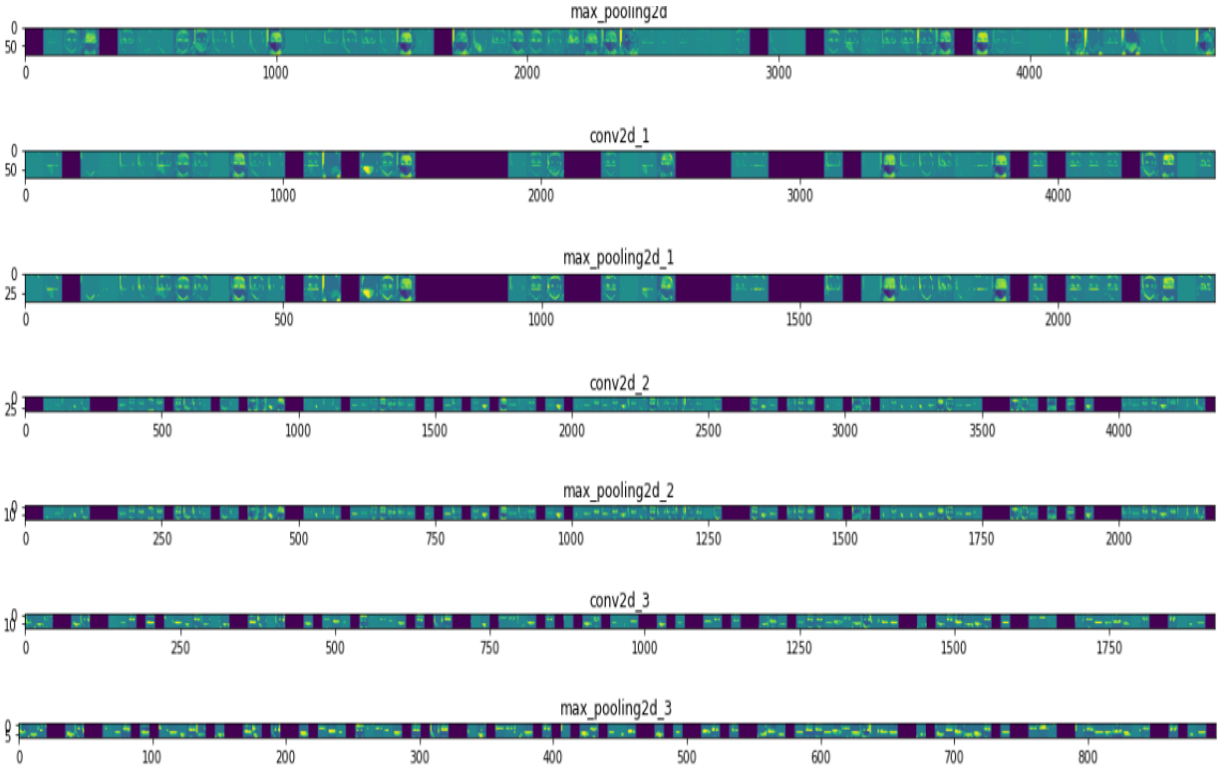
Fonte: Elaborado pelos autores, 2021.

4.2. Modelo de Classificação

A construção do modelo de classificação foi realizada inicialmente pela escolha do *software*, mais especificamente das bibliotecas, isso ocorreu pelo fato de que o Tensorflow é uma biblioteca *open-source* criada pela Google e que apresenta uma maneira simples de implementar uma rede neural, fazendo o processamento de dados em larga escala de maneira rápida devido a sua otimização com tensores (Tensorflow, 2021), o que foi necessário dado ao tamanho do *dataset* utilizado, além da utilização do Tensorflow também utilizamos a biblioteca Keras, que possibilitou o treinamento da rede neural para a classificação do uso da máscara. O treinamento da rede neural teve a função do reconhecimento do uso correto da máscara de proteção e do seu uso incorreto ou a não utilização da máscara de proteção. A rede utilizou o *dataset* informado para treinamento.

A rede neural foi configurada com quatro camadas de convolução bidimensional, depois foi adicionado à camada *flatten* para vetorizar a imagem e, por fim, os dados passam pelas camadas *dense* para que a saída resulte em um vetor com três valores referentes a predição das categorias desejadas. A Figura 9 mostra, para cada camada da rede neural, como o algoritmo ressalta características das imagens para detectar o objeto de interesse e classificá-lo conforme especificado.

Figura 9: Exemplo de uma imagem passando pela rede neural.



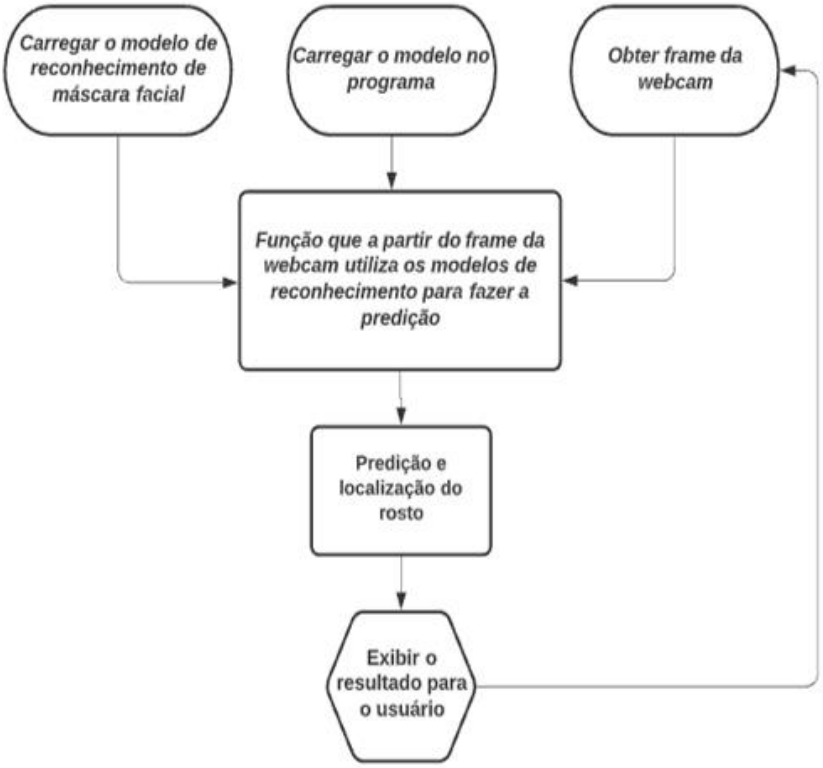
Fonte: Elaborado pelos autores, 2021.

Ao classificar uma imagem, o modelo então traz um vetor contendo a probabilidade de cada um dos possíveis *outputs*, se baseando nos pesos das camadas, cabendo então ao programa ponderar qual é a saída correta.

4.3. Detecção em Tempo Real

Após o treinamento do modelo de reconhecimento de máscara facial, foi feito um programa para integrar o modelo com a câmera e com os motores, funcionando em tempo real, a partir das imagens obtidas da câmera. Para isso, foi necessário utilizar um modelo de visão computacional já treinado, usado para reconhecer rostos de pessoas em imagens. Isso facilita o reconhecimento das máscaras, pois o ambiente no fundo da imagem pode ser cortado e o modelo pode focar em detectar apenas a área do rosto da pessoa. Foi utilizado o modelo CAFFE_DNN (BALU, 2021). O Fluxograma 1 mostra o esquemático.

Fluxograma 1 - Etapas do desenvolvimento do programa



Fonte: Elaborado pelos autores, 2021.

Primeiramente, foi utilizado a biblioteca *Imutils* que disponibiliza funções que podem ser implementadas em Python para configurar a câmera conectada ao sistema. A partir do frame obtido da câmera, a imagem passa pelo modelo de reconhecimento de rostos para identificar a localização do rosto da pessoa. Depois, o programa localiza 4 pontos que formam um retângulo em volta do rosto. Com essas coordenadas, o *frame* é cortado nessa localização, formando a imagem que o modelo de reconhecimento de máscara facial classifica. Antes de ser classificada a imagem passa por um processamento, onde a resolução é alterada para 150 x 150 px e a imagem é vetorizada para que o modelo consiga ler.

Com o resultado da classificação, isto é, se na imagem capturada pela câmera a pessoa está usando máscara de forma certa, de forma errada ou está sem máscara, e com as coordenadas da localização do rosto, o programa plota na interface do usuário o frame com uma “caixa” em torno o rosto da pessoa e com o resultado da classificação.

A cor da “caixa” foi definida conforme a classificação com a intenção de apresentar um resultado mais intuitivo e de fácil compreensão. A Tabela 1 mostra as cores utilizadas:

Tabela 1: Esquemas de cores para status do uso da máscara

Classificação	Cor
Máscara Correta	Verde
Máscara Errada	Amarelo
Sem Máscara	Vermelho

Fonte: Elaborado pelos autores, 2021.

4.4. Protótipo físico

A escolha do hardware ser o microcomputador Raspberry Pi 3B ocorreu por ser um “system-*on*-*a*-*chip*” (SoC), isto é, ele é um computador completo. A configuração escolhida utiliza o sistema operacional Debian, baseado em Linux, e que suporta uma vasta gama de linguagens de programação. A placa possui portas GPIO, entrada Micro SD que é utilizada como um HD, processador de quatro núcleos com 1,2 GHz, entrada de alimentação, entradas HDMI, conector para o módulo de câmera, onde será realizada a conexão com a câmera que identifica os objetos em tempo real, e entrada para cabo de rede e saídas USB.

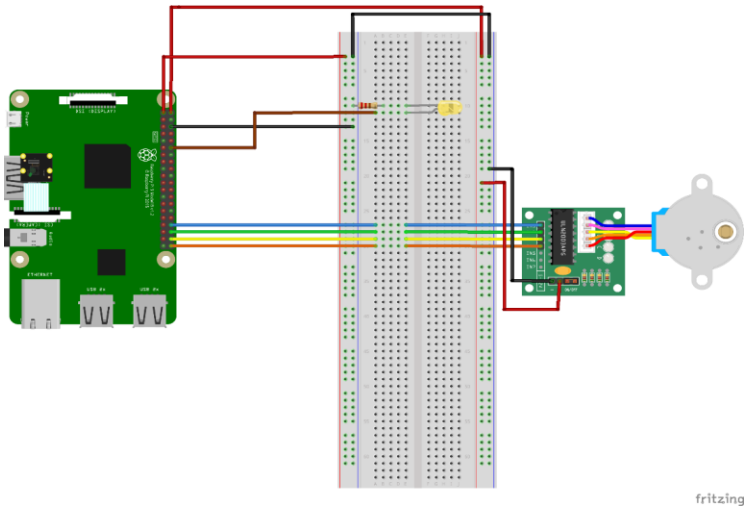
Além do microcomputador principal, alguns outros componentes foram necessários para a montagem do setup experimental, como motores de passo e uma câmera. A relação completa dos componentes utilizados e suas especificações:

- Raspberry
    - Modelo: Pi 3B
    - Memória RAM: 1GB
    - GPIO: estendido de 40 pinos
    - USB: 4 portas
    - Porta da câmera: CSI
    - Porta de Armazenamento: Micro SD

- Motor de Passo
    - Modelo: 28BYJ-48
    - Tensão de operação: 5V
    - Frequência de operação: 100Hz
    - Torque: 600-1200 gf.cm
- Driver para motor de passo
    - Modelo: ULN2003
- Câmera
    - Resolução: 5 Megapixel
    - Contagem de pixels: 2592x1944
    - Abertura:1/4 5m
    - Conexão: Flat
- Cartão de memória
    - Modelo: SanDisk SDSQUNS-032G-GN3MA Ultra
    - Espaço: 32GB
    - Velocidade de Leitura: 80MB/s
- LED
    - Tensão: 2-2.2V
    - Corrente: 20-25mA
    - Cor: Amarelo
- Protoboard
    - Quantidade de pinos: 400
- Resistor
    - Resistência: 330 Ω
- Jumpers

Com base nestes equipamentos, foi desenvolvido um esquema do circuito proposto, construído através do *software* Fritzing, conforme monstrado na Figura 10.

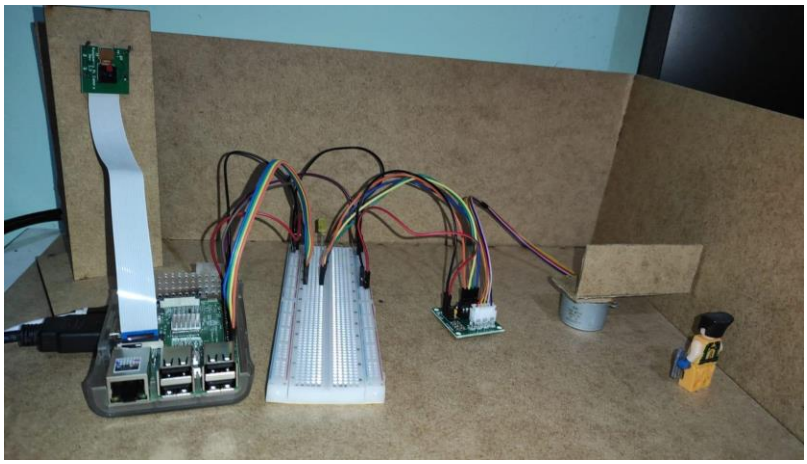
Figura 10: Diagrama do circuito proposto



Fonte: Elaborado pelos autores, 2021.

Com base na simulação, o próximo passo foi construir fisicamente o circuito proposto, o resultado do arranjo é demonstrado na Figura 11.

Figura 11: Sistema de portas controladas por visão computacional



Fonte: Elaborado pelos autores, 2021.

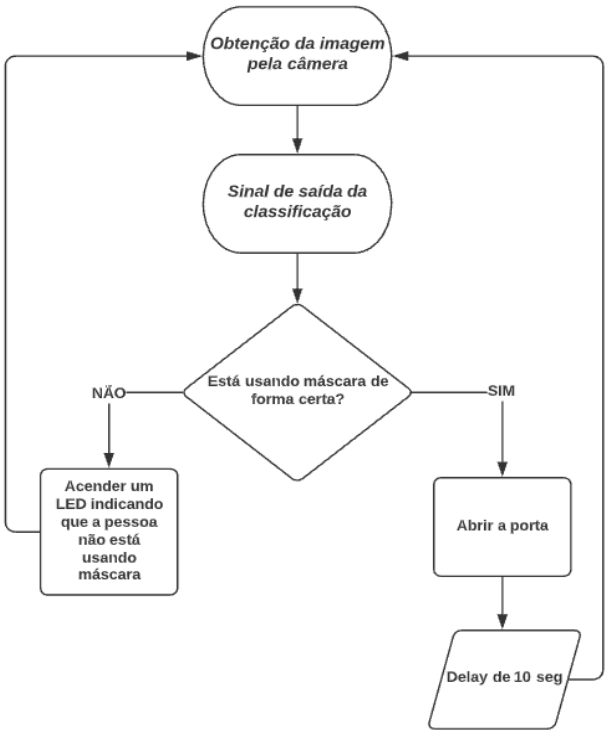
Passando assim a inicialização do sistema do Raspberry, para isto, o sistema operacional escolhido foi o Raspberry Pi OS, um sistema baseado na distribuição Debian do Linux otimizado para o presente *hardware*.

4.5. Integração entre *hardware* e sistema de reconhecimento facial

A integração entre o sistema de reconhecimento facial e o Raspberry Pi acontece em alguns diferentes momentos, o primeiro se tratando da instalação de um interpretador para a linguagem Python, o qual foi escolhido o interpretador padrão da distribuição do Python 3.7. Outro ponto importante de atenção foi devido a arquitetura do *hardware*, que conta com um processador ARM, de modo que a distribuição padrão do Tensorflow não apresentou o funcionamento adequado dentro do sistema, logo tornou-se necessário a utilização de uma distribuição própria do *framework* para a arquitetura utilizada.

Com isto devidamente preparado, o sistema apresenta o comportamento demonstrado no Fluxograma 2.

Fluxograma 2: Sistema de controle do protótipo



Fonte: Elaborado pelos autores, 2021.

O controle do motor de passo, assim como do LED, foi realizado por meio de classes criadas em Python, utilizando como base a biblioteca gpiozero. Para a câmera, foi utilizado uma biblioteca própria, a picamera. Todo o código encontra-se disponível em repositório Github (PAESGUS, 2021).

5. RESULTADOS E DISCUSSÃO

5.1. Treinamento do modelo

Com a compilação do modelo, foi possível obter os resultados do treinamento, assim como as métricas da performance do modelo. A Tabela 2 mostra a quantidade de parâmetros usados no treinamento em cada etapa, resultando em um total de cerca de 3,5 milhões de parâmetros. Além disso, o modelo foi treinado durante 25 épocas, isto é, os dados passaram pelo algoritmo da rede neural 25 vezes para obter uma acurácia cada vez melhor.

Tabela 2: Quantidade de parâmetros calculados por cada camada

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 74, 74, 64)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dropout (Dropout)	(None, 6272)	0
dense (Dense)	(None, 512)	3211776
dense_1 (Dense)	(None, 3)	1539

Total params: 3,473,475
Trainable params: 3,473,475
Non-trainable params: 0

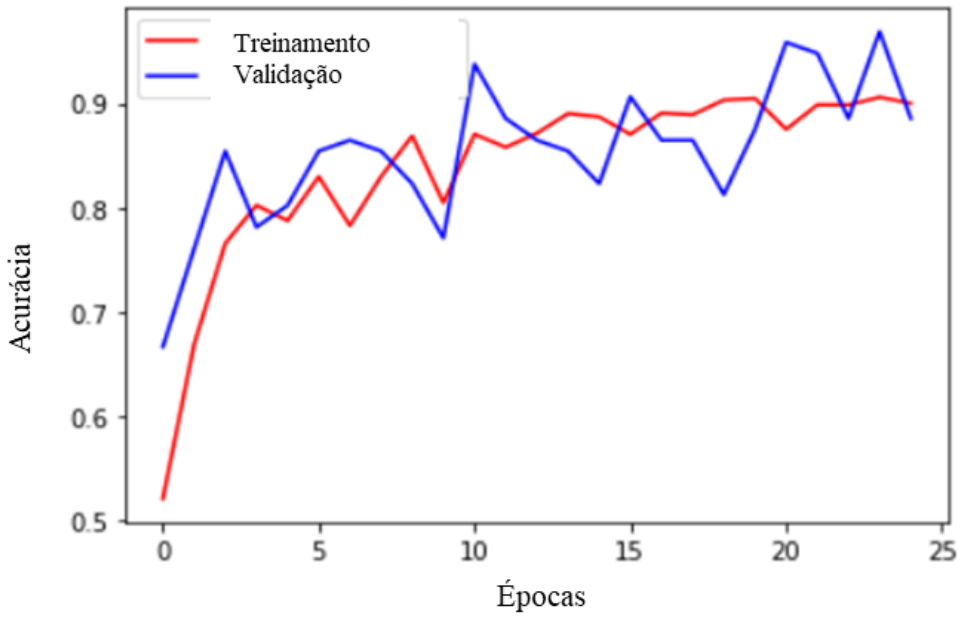
,

Fonte: Elaborado pelos autores, 2021.

É possível verificar, na Figura 12, o gráfico da acurácia de treinamento e de validação em cada época do treinamento, onde a acurácia de treinamento ficou em torno de 90%, enquanto a acurácia de teste, também chegou em cerca de 90%, como pode

ser visto, ambas medidas obtiveram valores próximos em cada época, indicando que o modelo possivelmente não sofreu *overfitting*. Sendo *overfitting* um comportamento indesejado de algoritmo de aprendizado de máquina, ocorrendo quando o desempenho do modelo é bom apenas para os dados usados no treinamento, mas quando são utilizados novos dados, diferentes dos dados de treino, o modelo não apresentaria bons resultados, ou seja, o modelo não conseguiria classificar os dados corretamente.

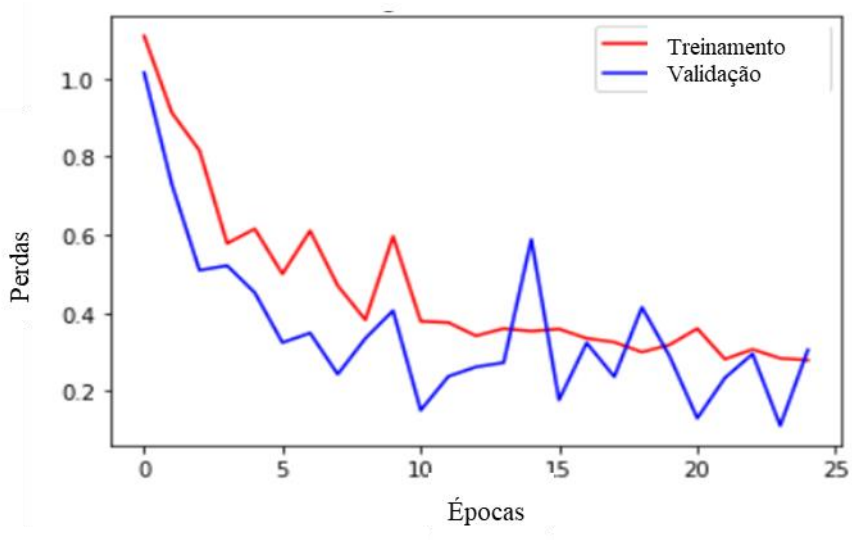
Figura 12: Acurácia de treinamento e de validação.



Fonte: Elaborado pelos autores, 2021.

Já a Figura 13 mostra os gráficos dos erros cometidos pelo modelo durante o treinamento. Pode ser notado que durante as épocas teve um grande decréscimo de perdas, indicando uma melhora na performance do modelo ao decorrer das épocas.

Figura 13: Erros de treinamento e de validação.



Fonte: Elaborado pelos autores, 2021.

Após a conclusão do treinamento, a execução do modelo foi feita através da câmera de captura do Raspberry. A Figura 14 mostra o resultado do programa rodando em tempo real. Foi possível notar que as três classificações foram preditas com uma porcentagem maior que 90%, isto é, o modelo classificou a três imagens nas categorias corretas com uma certeza maior que 90%. Também foi possível perceber que a detecção possuía um resultado melhor quando o ambiente estava bem iluminado e a câmera fixada em algum suporte, evitando que a imagem ficasse tremida. Também é notável que o modelo apresenta um melhor funcionamento com máscaras mais claras, o que pode ser explicado em virtude do *dataset* utilizado, que possui uma gama maior de pessoas com máscaras claras.

Figura 14: Resultado do sistema de detecção

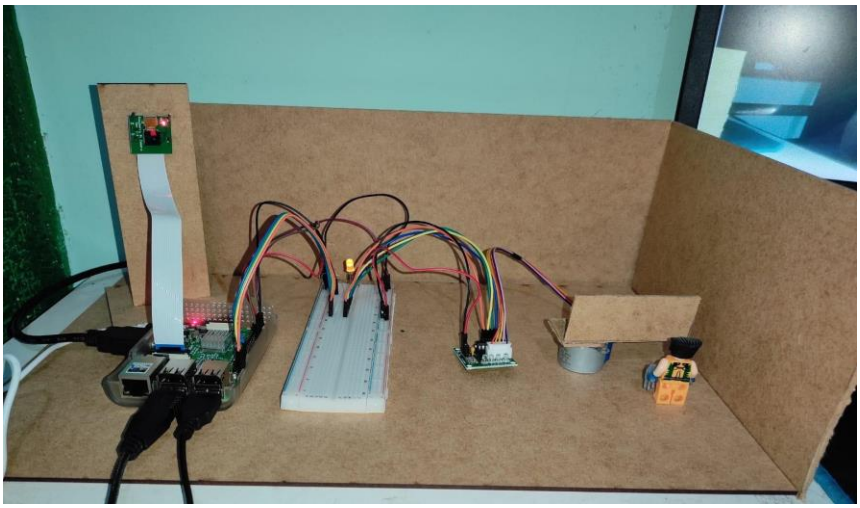
Fonte: Elaborado pelos autores, 2021.

5.2. Arranjo Experimental

O protótipo completo, unindo o modelo de reconhecimento facial e a operação do hardware apresentou um funcionamento adequado, alinhado com o que foi proposto previamente, isto é, na situação em que uma pessoa aparecia em frente a câmera, o software analisa a imagem gerada, verifica qual dos estados (com máscara, máscara incorreta ou sem máscara) apresenta a maior probabilidade e com isto gera uma ação para o hardware. Na situação em que uma pessoa apareceu com a máscara sendo utilizada de maneira correta, o programa ativou, de maneira assíncrona, o módulo que realizou o controle da porta, abrindo-a durante dez segundos e logo após, fechando. Já nas situações em que uma pessoa apareceu sem utilizar a máscara, ou utilizando-a de maneira incorreta, o sistema gerou um aviso, que no caso do sistema proposto foi através do acendimento de um LED amarelo, contudo a detecção no caso de máscara incorreta apresentou uma certa demora para realização da detecção quando em comparação com os outros dois estados.

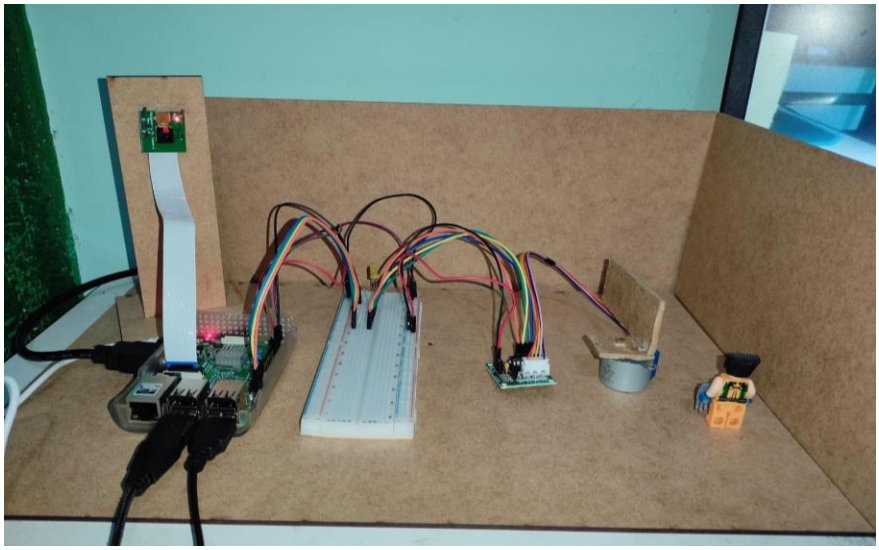
como pode ser observado nas Figuras 15 e 16, que demonstram o comportamento do sistema quando uma pessoa está com máscara e quando o acessório não está presente ou está sendo utilizado de forma incorreta.

Figura 15: Sistema de porta controladas por visão computacional – Pessoa com máscara incorreta



Fonte: Elaborado pelos autores, 2021.

Figura 16: Sistema de portas controladas por visão computacional – Pessoa com máscara



Fonte: Elaborado pelos autores, 2021.

Durante os experimentos e testes realizados, notou-se que durante a utilização do modelo no Raspberry, além dos desafios enfrentados pela iluminação do ambiente e dataset escolhido citados anteriormente, outro ponto a se destacar é referente ao sistema de detecção em tempo real, que apresentou um vídeo com menos *frames* durante a utilização no Raspberry, isto ocorrendo devido ao poder de processamento do microcomputador não ser tão robusto, contudo, isto poderia ser contornado com a utilização de um módulo externo de processamento, como uma TPU conectado via USB.

5.3. Testes Situacionais

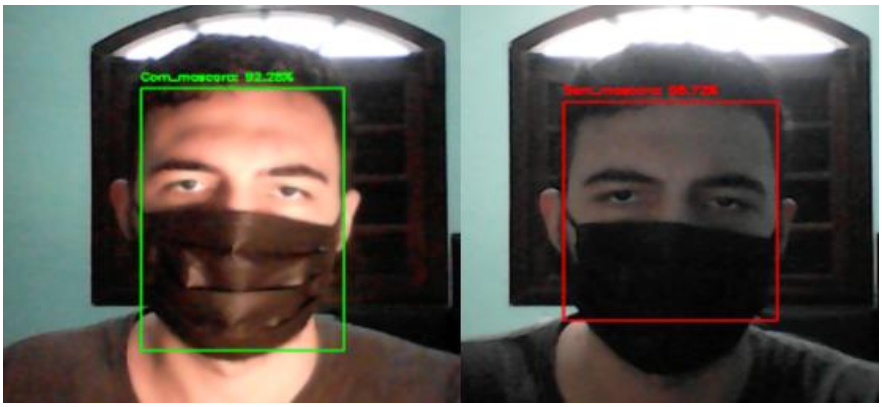
Para averiguar a validade do modelo criado nas mais diversas situações foram elaborados certos testes para validar como o algoritmo de reconhecimento facial se sai nas mais diversas situações.

5.3.1. Luminosidade

No teste inicial foi realizado o estudo de como a variação de luz incidindo sob o rosto afeta a inferência do algoritmo de reconhecimento de máscara na face. Para tal, foram realizadas em média cerca de 67 medições em cada situação, utilizando uma

máscara preta, onde as medições foram realizadas a 65 centímetros de distância da câmera, em um ambiente com uma luz LED de 9W como base e com a utilização de uma luminária EXBON ILUM-R06W5 de 5W de potência e no máximo 800 lúmens. Um exemplo das classificações pode ser observado na Figura 17.

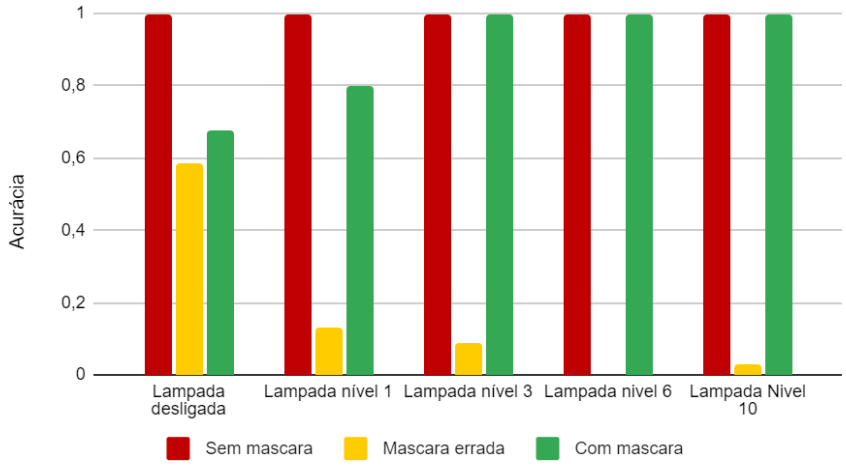
Figura 17: Teste situacional envolvendo luminosidade. A esquerda, luminária na potência máxima. A direita, apenas a luz ambiente



Fonte: Elaborado pelos autores, 2021.

Realizou-se medições em cinco distintas situações, sendo elas: Lâmpada desligada; Lâmpada em seu primeiro nível; Lâmpada em seu terceiro nível; Lâmpada em seu sexto nível e; Lâmpada em seu décimo e máximo nível. Os resultados podem ser encontrados na Figura 18.

Figura 18: Resultados do teste com luminosidade



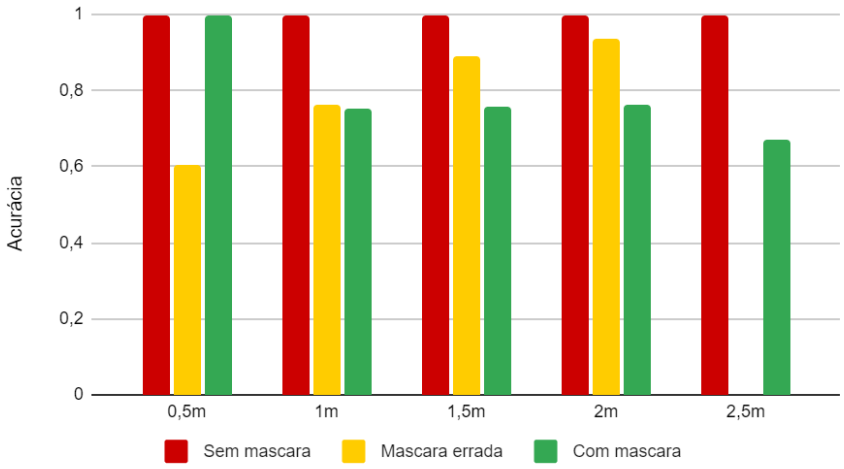
Fonte: Elaborado pelos autores, 2021.

Com base nos resultados encontrados, concluiu-se que para as classificações ‘Sem máscara’ e ‘Com máscara’ o sistema apresenta uma melhora considerável conforme o nível de luminosidade aumenta, de modo a classificar corretamente 100% dos casos destas duas situações, contudo, o mesmo não pode ser dito sobre a situação ‘Máscara errada’, que apresentou uma queda considerável em sua acurácia conforme o aumento da luminosidade, sendo erroneamente classificado como sendo a situação ‘Com máscara’, fato este que pode ter ocorrido devido a menor presença desta situação nas imagens de treino, o que pode ter enviesado o modelo.

5.3.2. Distância

Esta situação aborda a influência da distância na detecção e classificação correta do rosto, para tal realizou-se um ensaio contemplando cinco situações distintas, indo de 50 centímetros até 2 metros e meio, com 50 centímetros de diferença entre elas. Para realização de tais testes, utilizou-se uma máscara preta. Os resultados podem ser observados na Figura 19.

Figura 19: Resultados do teste da distância



Fonte: Elaborado pelos autores, 2021.

A partir dos resultados obtidos, pode-se inferir que o sistema possui uma queda de acurácia na situação ‘Com máscara’ conforme o indivíduo se afasta da detecção do sistema, porém a situação ‘Máscara errada’ apresenta um ganho de acurácia crescente

até os dois metros. Não é possível afirmar qual das situações apresentadas é a melhor, pois depende muito do ambiente na qual está inserida, isto é, em uma situação hospitalar, onde o uso incorreto de uma máscara pode ocasionar sérias consequências, a situação de 2 metros seria a ideal, pois apresenta uma grande acurácia nos casos de pessoas com ‘Máscara errada’ e ‘Sem máscara’, o que impediria a entrada das pessoas no ambiente.

5.3.3. Rotação

Esta situação aborda a influência da rotação na detecção e classificação correta do rosto, para tal realizou-se um ensaio contemplando de 0 graus a 180 graus. Para realização de tais testes, utilizou-se uma máscara branca. A representação do teste realizado está contida na Figura 20, assim como um exemplo real na Figura 21.

Figura 20: Rotação da cabeça

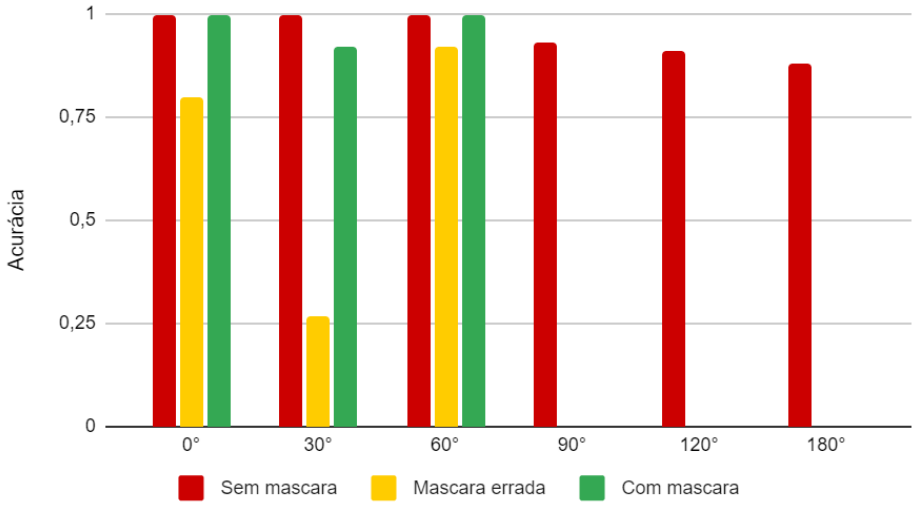
Fonte: Shutterstock, 2021.

Figura 21: Teste situacional envolvendo rotação do rosto

Fonte: Elaborado pelos autores, 2021.

Os resultados podem ser observados na Figura 22.

Figura 22: Resultados do teste da rotação da cabeça



Fonte: Elaborado pelos autores, 2021.

A partir dos resultados obtidos, pode-se inferir que o sistema possui uma queda de acurácia na situação ‘Com máscara’ e ‘Máscara Errada’ conforme o indivíduo rotaciona a sua face para situações a partir de 90°, isso ocorre devido a perca de identificação da face conforme o aumento do grau de rotação, isto se deve a maneira que o modelo foi treinado, onde apesar da utilização do *Data Augmentation*, o sistema ainda possuía predominantemente situações com o rosto centrado. Podemos observar que a melhor acurácia está presente quando temos nenhuma ou uma pequena rotação na face do indivíduo, observando uma melhor eficiência do sistema para faces que estão com um baixo grau de rotação quando relacionado com a câmera.

5.3.4. Inclinação Vertical

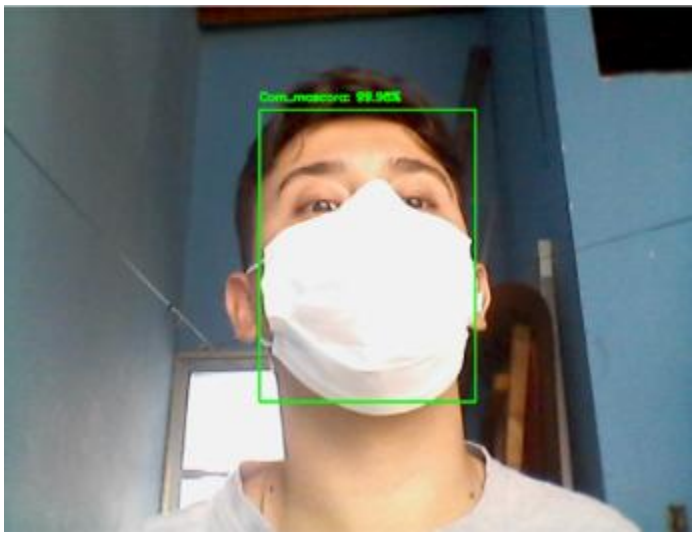
Esta situação aborda a influência da inclinação vertical para detecção e classificação correta do rosto, para tal realizou-se um ensaio contemplando de -80 graus a 80 graus. Para realização de tais testes, utilizou-se uma máscara branca. A

representação do teste realizado está contida na Figura 23, assim como um exemplo real na Figura 24.

Figura 23: Inclinação vertical

Fonte: Shutterstock, 2021.

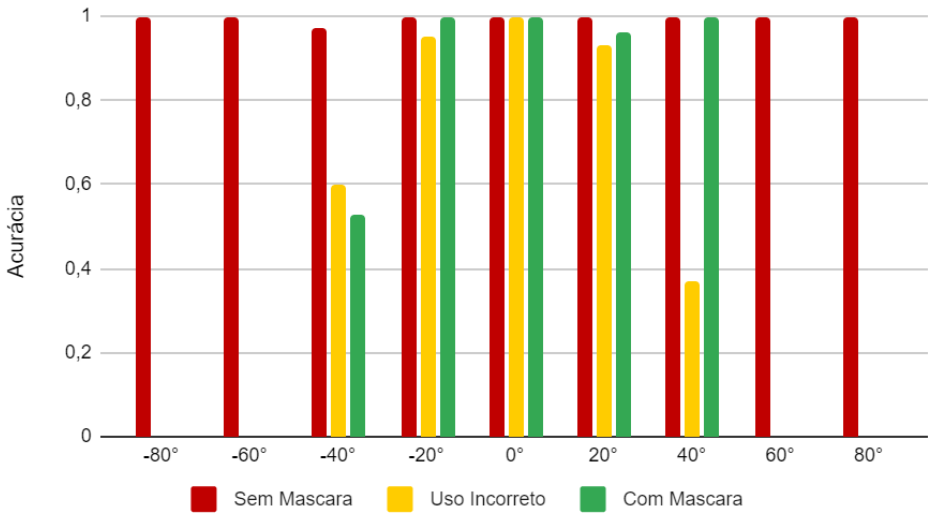
Figura 24: Teste situacional envolvendo inclinação vertical do rosto



Fonte: Elaborado pelos autores, 2021.

Os resultados podem ser observados na Figura 25.

Figura 25: Resultados do teste de inclinação



Fonte: Elaborado pelos autores, 2021.

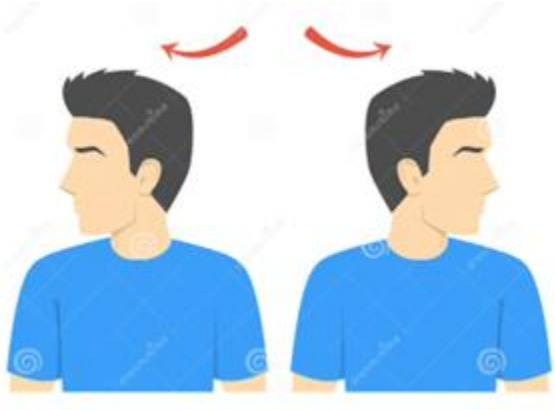
A partir dos resultados obtidos, pode-se inferir que o sistema possui uma queda de acurácia na situação ‘Com máscara’ conforme o indivíduo inclina o seu rosto, isso é observado, devido a perca de identificação da face realizada pelo programa, temos que a partir dos 60 graus todos os valores ficam configurados para sem máscara, a situação ‘Máscara errada’ apresenta uma perda de acurácia a partir de 60 graus possui o mesmo comportamento do teste ‘Com máscara’. Podemos observar que a melhor acurácia está presente quando não temos variação vertical das faces dos indivíduos, em 0 graus, observando uma boa eficiência do sistema para faces que estão com um baixo grau de variação vertical e perda total em sistemas com variações maiores que 60 graus.

5.3.5. Rotação no eixo vertical

Nesta situação foram realizados testes baseados na posição do rosto em relação a câmera. A câmera foi fixada na altura dos olhos e a 50 cm do rosto, com isso o usuário rotaciona a cabeça no eixo vertical, conforme mostra a Figura 26, realizando as medições para 30°, 60° e 90° (perfil), tanto para a esquerda quanto para a direita. Realizou-se 55

medições para cada caso. Sendo medições, a captura e a classificação feita pelo programa das imagens obtidas.

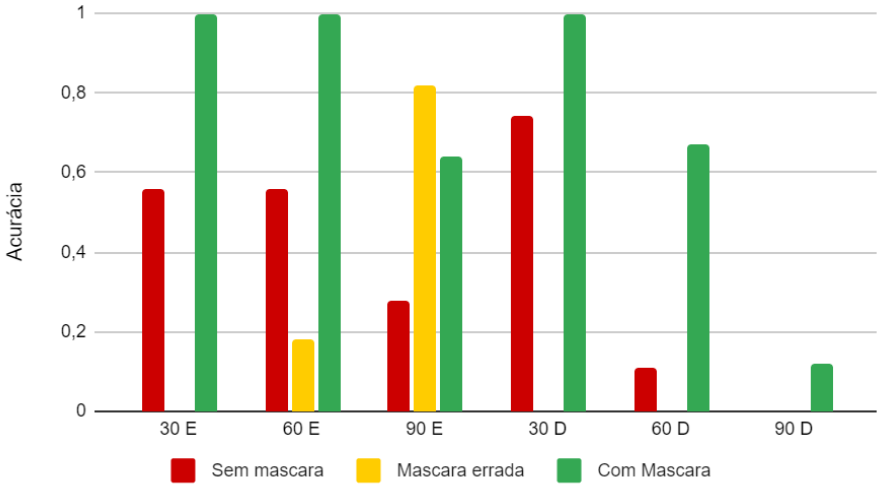
Figura 26: Rotação da cabeça no eixo vertical



Fonte: Shutterstock, 2021.

A Figura 27 mostra os resultados obtidos. Nota-se que para as situações em que se estava usando a máscara de forma correta o sistema conseguiu 100% de acurácia com o rosto virado para ambos os lados em 30°, isto é, o algoritmo foi capaz de reconhecer a máscara corretamente. Já para a situação sem máscara, a maioria dos casos teve um acerto menor que 60%, indicando que o programa não foi treinado suficientemente para prever essas situações. E a acurácia foi pior para os casos em que se estava usando a máscara errada, o que era previsto, pois a quantidade de dados encontrados para tal categoria foi bem menor.

Figura 27: Acurácia baseada na rotação da cabeça no eixo vertical; E = Esquerda; D = Direita.



Fonte: Elaborado pelos autores, 2021.

5.3.6. Cores de Máscaras

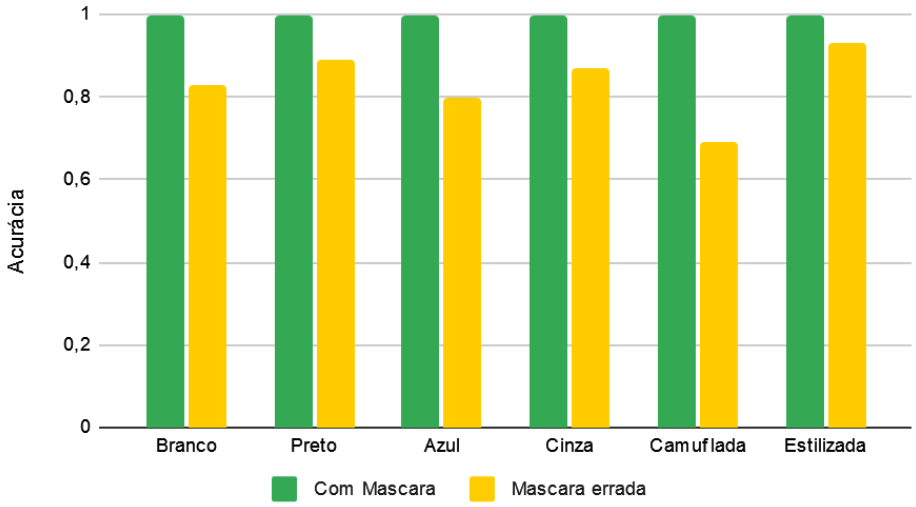
Para esse teste foram utilizadas seis máscaras de cores diferentes, como indica a Figura 28. Esse experimento foi realizado com o intuito de verificar a acurácia do programa com base na cor da máscara, pois ao implementar o sistema em uma situação real, os usuários podem estar usando máscaras de diferentes tipos. As medições foram feitas com a câmera na altura dos olhos a 50 cm de distância do rosto, além disso, o rosto estava centralizado e perpendicular à câmera. Foram realizadas um total de 55 medições para cada tipo de máscara.

Figura 28: Tipos de máscaras: 1-branca, 2-preta, 3-azul, 4-cinza, 5-camuflada, 6-estilizada

Fonte: Elaborado pelos autores, 2021.

A Figura 29 mostra os resultados obtidos. Ao analisar os gráficos, foi possível perceber que para todos os tipos de máscaras, foram obtidas acurácias de 100% para o uso correto da máscara, mostrando que o algoritmo está reconhecendo e categorizando a máscara pela posição dela no rosto e não só pela cor, pois as imagens de treinamento eram em sua maioria brancas e pretas. Quanto às situações em que as máscaras estavam sendo usadas de forma errada, isto é, com o nariz para fora, foi obtida uma acurácia média de 83%, indicando um bom desempenho do programa, ao considerar que havia poucos dados disponíveis para treinar a IA nessa categoria.

Figura 29: Acurácia baseada nas cores das máscaras.



Fonte: Elaborado pelos autores, 2021.

6. CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de um protótipo de porta automática controlada por visão computacional, onde seu controle ocorre através de técnicas de robótica para reconhecimento da utilização correta de máscaras.

O protótipo verificou que a utilização de redes neurais criadas, utilizando Tensorflow, para o reconhecimento e classificação de imagens que apresentam o uso correto ou incorreto da máscara de proteção, visando permitir a entrada apenas das pessoas que utilizam a máscara de forma correta. Dentre os fatores na qual influenciaram no reconhecimento das imagens podemos citar o tamanho da imagem, ângulo da face com relação à câmera e luminosidade do ambiente, os quais foram corrigidos no pré-processamento para que fosse obtido um valor mais coerente na análise.

Como mostra os testes situacionais, as cores das máscaras não possuem grande interferência no desempenho do programa, mantendo uma acurácia média de 83%, chegando na conclusão que a IA identifica as máscaras, também, pelo seu formato e posição no rosto. Quanto à luminosidade, foi possível concluir que, uma irradiância de luz maior melhora a acurácia das categorias com máscara e sem máscara, por outro lado, com a máscara errada não houve diferença significativa. Já em relação a distância, a categoria com máscara teve uma pequena perda de desempenho, além disso, no caso “máscara errada” não houve reconhecimento para distâncias maiores que 2 metros. Por fim, todos os testes de rotações e inclinações, mostram que o programa não é capaz de fazer um reconhecimento confiável em circunstâncias em que o rosto não estava alinhado a câmera, concluindo-se que a IA possui um bom desempenho para situações em que o rosto estava centralizado e perpendicular com a câmera, como era previsto, pois os dados de treinamento disponíveis não eram diversificados e não abrangiam todas essas situações.

Os resultados apresentados nos cenários experimentais indicaram uma eficiência significativa do modelo na detecção da utilização de máscara com uma acurácia superior a 90% no treinamento e nos testes, indicando que não houve *overfitting,* sendo esperado

que a utilização não-supervisionado mantenha com a mesma qualidade de classificação. Com a simulação no Raspberry observou-se que, apesar de funcionar corretamente, houve uma queda considerável nos *frames*, tornando o modelo mais lento do que o esperado, sendo necessário para um aumento de velocidade um hardware com um poder de processamento maior ou uma otimização no modelo, assim será possível manter uma alta taxa de *frames* de imagem por segundo nas análises práticas.

A importância deste trabalho é que todos os locais necessitam da restrição da entrada para pessoas que utilizam a máscara de forma incorreta, visando a segurança de todos os presentes, podendo as redes neurais artificiais serem ótimas opções para o reconhecimento e classificação da utilização das máscaras de proteção, com a possibilidade de expansão também para outros setores, como a utilização para check-in em aeroportos, verificação da utilização correta de EPIs em obras e diversas outras. É preciso continuar os estudos dessas tecnologias de reconhecimento para um aumento de velocidade de processamento e acurácia do modelo.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, Martín. Tensorflow: learning functions at scale. **Proceedings Of The 21St Acm Sigplan International Conference On Functional Programming**, [S.L.], v. 1, n. 1, p. 1-1, 4 set. 2016. ACM.

ABRAHAM, Ajith. **Artificial Neural Networks**. 2005. 8 f. Tese (Doutorado) - Stillwater, Oklahoma State University, Oklahoma, 2005.

BALU, Gopinath. **CAFFE_DNN at master**.Computer Vision. Disponível em:https://github.com/gopinath-balu/computer_vision/tree/master/CAFFE_DNN.Acesso em: 25 de maio de 2021

BERCHE, P.. Louis Pasteur, from crystals of life to vaccination. **Clinical Microbiology And Infection**, [S.L.], v. 18, p. 1-6, out. 2012. Elsevier BV.

CABANI. **MaskedFace-Net**. Disponível em: <https://github.com/cabani/MaskedFace-Net> Acessado em 23 de abril de 2021.

DE ANDRADE, Alexandre Acácio et al. **Low cost solution for home brewing and small brewing business using Raspberry Pi. In: Interdisciplinary Conference on Innovation**, Desgin, Entrepreneurship, And Sustainable Systems. Springer, Cham, 2018. p. 136-145.

EIKENBERRY, Esteffen; et al. **To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic.** Infectious Disease Modelling. [S.L.], v. 5, p. 293-308, 2020. ScienceDirect.

JIA, Chengfan et al. Improving the Performance of Distributed Tensorflow with RDMA. **International Journal Of Parallel Programming**, [S.L.], v. 46, n. 4, p. 674-685, 27 set. 2017. Springer Science and Business Media LLC.

KARNIN, E.D.. A simple procedure for pruning back-propagation trained neural networks. **Ieee Transactions On Neural Networks**, [S.L.], v. 1, n. 2, p. 239-242, jun. 1990. Institute of Electrical and Electronics Engineers (IEEE).

KIATRONICS. **28BYJ-48 Datasheet (PDF) - List of Unclassifed Manufacturers.**Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1132391/ETC1/28BYJ-48.html. Acesso em: 18 nov. 2021.

LARXEL. **Face Mask Detection**. Disponível em: https://www.kaggle.com/andrewmvd/face-mask-detection. Acesso em: 15 de maio de 2021.

MUSSAP, C. J. The Plague Doctor of Venice. ***Internal Medicine Journal**, v 49(5), p 671–676, maio 2019.*

NVLABS. **ffhq-dataset**. Disponível em: <https://github.com/NVlabs/ffhq-dataset> Acessado em 23 de abril de 2021.

OMS. **Coronavirus disease (COVID-19) advice for the public**. Disponível em: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public>. Acesso em: 06 de maio de 2021.

OPENCV. **OpenCV – 4.5.3**. Disponível em: https://opencv.org/. Acesso em: 10 de maio de 2021.

PAESGUS. **TG-UFABC**. Disponível em: https://github.com/paesgus/TG_UFABC Acessado em 24 de julho de 2021.

PEREIRA, Ávila. et al. **Factors associated with the use and reuse of face masks among Brazilian individuals during the COVID-19 pandemic**. Revista Latino-Americana de Enfermagem, São Paulo, v. 28, n. 1, p. 1-5, 22 abr. 2020. FapUNIFESP (SciELO).

ST. **E-ULN2003A Datasheet** (PDF) - STMicroelectronics.Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/641020/STMICROELECTRONICS/E-ULN2003A.html. Acesso em: 18 nov. 2021.

TENSORFLOW. **Tensor Flow Core - Overview.** Disponível em: <https://www.tensorflow.org/overview>. Acesso em: 29 de maio de 2021.

TSAI, Peter et al. Different electrostatic methods for making electret filters. **Journal Of Electrostatics**, [S.L.], v. 54, n. 3-4, p. 333-341, mar. 2002. Elsevier BV.

UPTON, Eben; HALFACREE, Gareth. **Raspberry Pi User Guide**. 4. ed. Chichester: Wiley, 2016. 315 p. ISBN 978-1-119-26436-1.

VOULODIMOS, Athanasios et al. Eftychios. Deep Learning for Computer Vision: a brief review. **Computational Intelligence And Neuroscience**, [S.L.], v. 2018, p. 1-13, 2018. Hindawi Limited.

WANG, Jiao *et al*. Mask use during COVID-19: a risk adjusted strategy. **Environmental Pollution**, [S.L.], v. 266, p. 115099, nov. 2020. Elsevier BV.

WANG, Sun-Chong. Artificial Neural Network. **Interdisciplinary Computing In Java Programming**, [S.L.], p. 81-100, 2003. Springer US.

WANG, Tim T. *et al*. Proceedings of the OMS COVID-19 Response Conference. **Journal Of Oral And Maxillofacial Surgery**, [S.L.], v. 78, n. 8, p. 1268-1274, ago. 2020. Elsevier BV.

WORLDOMETER. **COVID-19 CORONAVIRUS PANDEMIC.** Disponível em: https://www.worldometers.info/coronavirus/. Acesso em: 29 de maio de 2021.

ZHAO, W. *et al*. Face recognition. **Acm Computing Surveys**, [S.L.], v. 35, n. 4, p. 399-458, dez. 2003. Association for Computing Machinery (ACM).