FACULDADE DE TECNOLOGIA DE SÃO JOSÉ DOS CAMPOS

FATEC PROFESSOR Jessen Vidal

LEONARDO LOPES NUNES

DETECÇÃO DO USO DE MÁSCARAS USANDO MODELO BASEADO EM APRENDIZAGEM DE MÁQUINA

Orientador: Fabricio Galende Marques de Carvalho

São José dos Campos

2021

SUMÁRIO

[1 Introdução 4](#_Toc48569212)

[1.1 Definição do Problema 4](#_Toc48569213)

[1.2 Objetivo 4](#_Toc48569214)

[2 Desenvolvimento 4](#_Toc48569215)

[2.1 Coleta de Dados 4](#_Toc48569217)

[2.2 Conjunto de Dados 5](#_Toc48569216)

[2.3 Separação dos Dados 6](#_Toc48569217)

[2.4 Aumento de Amostras de imagens 6](#_Toc48569216)

[2.5 Treino do Modelo 7](#_Toc48569216)

[2.6 Epoch 7](#_Toc48569216)

[2.7 Cross Validation 8](#_Toc48569216)

[2.8 Overfitting 8](#_Toc48569216)

[2.9 Ferramentas 8](#_Toc48569216)

2.9.1 Python 8

2.9.2 Tensor Flow 9

2.9.3 Keras 9

2.9.4 OpenCV 9

[3 Desenvolvimento 10](#_Toc48569212)

[3.1 Requisitos de Sistema 10](#_Toc48569216)

[3.2 Instalação das Bibliotecas 10](#_Toc48569216)

[3.3 Importação das Bibliotecas 10](#_Toc48569216)

[3.4 Definição de Layers e Parâmetros 11](#_Toc48569216)

[3.5 Geração de novas Amostras 11](#_Toc48569216)

[3.6 Salvamento dos Parâmetros 12](#_Toc48569216)

[3.7 Treino do Modelo 12](#_Toc48569216)

[3.8 Exportação dos Modelos 12](#_Toc48569216)

[3.9 Importando Modelos 13](#_Toc48569216)

[3.10 Configurando Ouput 13](#_Toc48569216)

[3.11 Executando o Modelo 14](#_Toc48569216)

[4 Resultados 16](#_Toc48569212)

[4.1 Modelo 16](#_Toc48569216)

[4.2 Predições 17](#_Toc48569216)

[5 Considerações Finais 22](#_Toc48569212)

[5.1 Contribuições 22](#_Toc48569216)

[5.2 Dificuldades 22](#_Toc48569216)

[5.3 Trabalhos Futuros 22](#_Toc48569216)

# Introdução

Este Capítulo apresenta o problema a ser resolvido e o objetivo deste trabalho.

## Definição do Problema

Sob a pandemia de COVID-19 em andamento (causada pelo coronavírus SARS-CoV-2) a lei 140.19, de 2020, é muito clara, é obrigatório o uso de máscaras de proteção individual para a circulação em espaços públicos e privado acessíveis ao público, em vias públicas e transportes públicos.

A medida segue as comprovações levantadas pela OMS comprovou para a efetividade da máscara no combate ao contágio do COVID-19. Ficou a cargo da Vigilância Sanitária a verificação do uso de máscaras e o respeito as orientações com relação ao distanciamento social nos estabelecimentos comerciais, estão sujeitos a multas pessoas que estiverem sem máscara nos locais exigidos e os estabelecimentos que permitirem a presença de pessoas sem proteção e/ou situação irregular.

## Objetivo

A criação de um modelo de *Deep Learning* que consegue identificar quem está usando ou não as máscaras, com o propósito de auxiliar na fiscalização e alertar sobre uma pessoa transitando sem máscara. Um modelo que possa se integrar a outros sistemas e assim criando uma aplicação cada vez maior para ele.

# Fundamentação TéCNICA

Este Capítulo apresenta detalhes sobre o desenvolvimento do sistema para automatização de viagens de negócio.

## Coleta de Dados

Para efetuar o treino do modelo devemos primeiramente coletar os dados, para este modelo os dados coletados foram fotos de pessoas utilizando máscara ou de pessoas com um desenho de máscara em seus faces.

Como o conjunto de dados já veio preparado, não foi necessário nenhum tratamento especial para o dado.

## Conjunto de Dados

O conjunto de dados consiste em rostos de pessoas com e sem o uso de máscara separando as em categorias de “Com máscara” e “Sem máscara”.

Para o modelo ser treinado foi criado um conjunto de dados com 755 amostras de pessoas utilizando máscara e 754 pessoas sem máscara.

As amostras de imagens por si só são muito poucas para que o modelo tenha uma performance precisa, visto que modelos de aprendizado de máquina necessitam de um número muito grande de amostras para alcançar tal precisão.

A Figura 1 ilustra um exemplo de amostra de pessoa sem máscara:

**Figura 1** – Amostra de Dado “sem máscara”

Homem de camisa preta

Descrição gerada automaticamente

A Figura 2 ilustra um exemplo de amostra de pessoa utilizando máscara:

**Figura 2** – Amostra de Dado “com máscara”

Homem com camisa azul

Descrição gerada automaticamente

## Separação dos Dados

A separação dos dados se torna necessária para que não sejam utilizados dados já conhecidos pelo modelo no momento de teste, com isso é possível validar a qualidade do modelo.

Os dados são separados em:

* Dados de treinamento: usado para treinar o modelo.
* Dados de teste: usado para comprovar que aquele modelo realmente funciona. São dados ignorados no treinamento.

Do conjunto de dados de pessoas com máscara, do seu valor total de 755 foram separados 12% das amostras para realizar o teste.

Do conjunto de dados de pessoas sem máscara, do seu valor total de 754 foram separados 12% das amostras para realizar o teste.

## Aumento de Amostras de Imagens

O aumento de imagens ou mais comumente conhecido como “image augmentation” é uma técnica que expande o tamanho do conjunto de dados. Com ele é possível criar imagens transformadas de seu conjunto de dados original, assim o modelo terá mais recursos para efetuar seu treino mesmo com poucas amostras de dados.

Como citado anteriormente o conjunto de amostras coletadas nesse trabalho é muito pouco, por isso, para contornar a falta de amostra de dados foi utilizado essa técnica para gerar mais amostras e consequentemente ter uma precisão melhor do modelo.

## Treino do Modelo

A Figura 3 ilustra as etapas do treino do modelo, na primeira etapa, todas as amostras de treinamento (em azul à esquerda) são encaminhadas para o modelo, que gera as previsões (em azul à direita).

Na segunda etapa, as previsões são comparadas com os dados de teste - o que resulta no cálculo de um valor de perda.

O modelo pode ser subsequentemente otimizado através de um novo ciclo de treino direcionando o modelo para longe do erro, alterando seus pesos, até que ele finalmente encontre um valor de imprecisão para detectar uma face utilizando a máscara.

O processo então começa novamente. Presumivelmente, o modelo tem um desempenho melhor desta vez.

**Figura 3** – Fluxo do Treino do Modelo

Diagrama

Descrição gerada automaticamente

## Epoch

O número de epoch é um hiper parâmetro que define o número de vezes que o algoritmo de aprendizado irá percorrer em todo o conjunto de dados de treinamento.

Uma epoch significa que cada amostra no conjunto de dados de treinamento teve a oportunidade de atualizar os parâmetros do modelo interno, ou seja, definir o peso de alguns parâmetros que o modelo está construindo. Uma epoch é composta por um ou mais conjunto de dados, neste trabalho foi visto que 10 epochs já levariam o modelo a um melhor resultado.

Em outras palavras, a epoch faz com que durante o treino do modelo ele veja o que ele pode melhorar, definido um número de epochs ele irá executar essa mesma quantidade de treino sempre melhorando e olhando para os parâmetros que ele pode considerar em sua construção.

## Cross Validation

Cross Validation é uma técnica muito utilizada para avaliação de desempenho de modelos de aprendizado de máquina. O Cross Validation consiste em particionar os dados em conjuntos, onde um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo. A utilização do Cross Validation tem altas chances de detectar se o seu modelo está sofrendo overfitting.

No trabalho apresentado o Cross Validation ocorre no mesmo momento em que o treino do modelo está sendo feito, assim ao fim de cada ciclo do treino do modelo já é possível comprovar sua acurácia e saber se a forma como o treino do modelo está sendo executada é válida.

## Overfitting

Overfitting é uma característica que o modelo pode atingir quando o modelo aprende demais sobre os dados. Neste caso, o modelo mostra-se adequado apenas para os dados de treino, como se o modelo tivesse apenas decorado os dados de treino e não fosse capaz de generalizar para outros dados nunca vistos antes. Quando isso acontece, os dados de treino apresentam resultados excelentes, enquanto a performance do modelo cai drasticamente com os dados de teste.

## Ferramentas

As ferramentas são utilizadas para auxiliar a construção do modelo, abaixo segue um detalhamento das funções de cada uma utilizada no processo de criação do modelo

### Python

Um dos motivos da popularidade do Python é o seu forte apoio na área de inteligência artificial (IA), de acordo com o IEEE. O Python também oferece um bom número de bibliotecas e pacotes que os programadores podem usar para que não construam determinado código do zero.

O grande número de bibliotecas e pacotes disponíveis para o Python contêm código para certas funções básicas, para que os programadores não precisem escrevê-las do zero. Especificamente, o estudo do IEEE e vários especialistas apontaram para a biblioteca *Keras* e o *TensorFlow*, por meio dos quais os programadores que usam IA podem trabalhar com redes neurais.

### Tensorflow

O TensorFlow proporciona flexibilidade e controle com recursos como a API funcional Keras e a API de subclassificação de modelos para a criação de topologias complexas.

O TensorFlow é uma das bibliotecas principais para que nesse trabalho seja executado o treino e a geração do modelo, pois ele irá auxiliar o Keras nesse processo.

### Keras

O Keras auxilia na criação de camadas neurais, funções de custo, otimizadores, esquemas de inicialização, funções de ativação e esquemas de regularização são todos módulos independentes que podem ser combinados para criar modelos.

Com isso, o Keras será o responsável por criar, treinar e executar o nosso modelo com o auxílio do Tensorflow.

### Open CV

OpenCV (Open Source Computer Vision Library) é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto. O OpenCV foi construído para fornecer uma infraestrutura comum para aplicativos de visão computacional e para acelerar o uso da percepção da máquina em produtos comerciais.

A biblioteca possui mais de 2.500 algoritmos otimizados, que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina clássicos e de última geração. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmera, rastrear objetos em movimento etc.

Nesse trabalho é utilizado os recursos de identificação da face do Open CV para agilizar o processo dessa identificação do modelo.

# Desevolvimento

Este Capítulo apresenta detalhes sobre o desenvolvimento do modelo para a identificação de uso de máscara.

## Requisitos de sistema

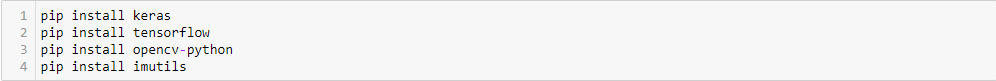
Para que o projeto consiga ser executado, é preciso que alguns requisitos tanto no sistema como no hardware:

* Câmera integrada ao computador (WebCam)
* Python 3
* Pip

## Instalação das Bibliotecas

O primeiro passo como ilustrado pela Figura 4 é instalar através do gerenciador de bibliotecas pip as bibliotecas que serão necessárias para que o modelo consiga ser criado:

**Figura 4** – Instalação das Bibliotecas



## Importação das bibliotecas

Feito a instalação das bibliotecas nós iremos importá-las em nosso código, isso é feito para que consigamos utilizar as funções e utilitários das bibliotecas em nosso código.

As importações devem ser feitas como na Figura 5 elas serão responsáveis por fornecerem utilitários que facilitarão a criação do modelo e a definição de parâmetros para o treino e exportação do modelo.

**Figura 5** – Importação das Bibliotecas

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

## Definição de layers e parâmetros

O modelo precisa saber qual formato de entrada ele deve esperar. Por esse motivo, a primeira camada em um modelo Sequencial precisa receber informações sobre sua forma de entrada.

A Figura 6 ilustra como devem ser feitas as definições das entradas do modelo, nele é definido que serão utilizadas camadas de convolução para o modelo compreender e identificar as imagens.

Uma Rede Neural Convolucional (ConvNet) é um algoritmo de *deep learning* que pode captar uma imagem de entrada e atribuir importâncias como pesos e vieses a vários aspectos e objetos da imagem e ser capaz de diferenciar umas das outras.

**Figura 6** – Definição de Layers

Texto

Descrição gerada automaticamente

## Geração de novas amostras

Os parâmetros e camadas do modelo são definidos conforme mostra a Figura 7, as imagens são coletadas do diretório de treino e teste para utilizá-las no método de aumento de amostras de imagens.

**Figura 7** – Geração de Amostras de Dados

Texto

Descrição gerada automaticamente

## Salvamento dos parâmetros

A Figura 8 ilustra o uso da função *ModelCheckpoint* que é utilizada em conjunto com o treinamento do modelo para salvar um modelo e seus pesos como um checkpoint, para que o modelo ou pesos possam ser carregados posteriormente para continuar o treinamento do estado salvo.

Esta configuração irá ajudar o modelo a ter uma progressão a cada epoch rodada.

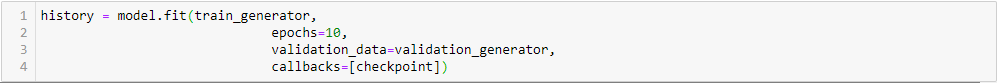
**Figura 8** – Função de Salvamento de Parâmetros



## Treinamento

O modelo executa o seu treino conforme ilustra a Figura 9 em cima de todos os parâmetros definidos, executando 10 epochs que foram definidas gerando um valor de acurácia e de imprecisão do modelo, ao ser executado o treino os logs serão mostrados como é ilustrado na Figura 10.

**Figura 9** – Execução do Treino



**Figura 10** – Logs da execução do treino

Texto

Descrição gerada automaticamente com confiança média

## Exportação dos modelos

Todos os modelos gerados nas epochs são salvos para serem utilizados ou consultados, por padrão são salvos na raiz do projeto como mostra a Figura 11 e em caso de um novo treinamento eles serão sobrescritos no diretório.

**Figura 11** – Modelos salvos no sistema de arquivos

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

## Importando os modelos

Para importar os modelos criados e treinados na etapa anterior, foi criado um arquivo .py separado do arquivo train.py chamado de test.py

Neste arquivo é necessário importar algumas bibliotecas conforme a Figura 12 para que o modelo seja carregado, sendo elas:

**Figura 12** – Importação de Bibliotecas

Texto

Descrição gerada automaticamente com confiança média

O modelo então pode ser importado utilizando a função *load\_model()* ilustrada na Figura 13 da biblioteca Keras e armazenando-o em uma variável. O caminho do arquivo onde o modelo está salvo é o valor a ser passado para essa função.

**Figura 13** – Carregamento do Modelo



## Configurando output

O modelo irá identificar os rostos que estiverem utilizando máscaras e aos que não estiverem, ele irá apresentar uma mensagem indicando a ausência de máscara.

A Figura 14 ilustra a definição de dois dicionários do Python, o dicionário *labels\_dict* irá armazenar as mensagens ao usuário e o dicionário *color\_dict* irá armazenar as cores verde e vermelha, sendo a verde para indicar o uso correto da máscara e o vermelho ao identificar a ausência da máscara.

**Figura 14** – Estilização da Interface do Rosto



A Figura 15 exemplifica a configuração da câmera que será utilizada para coletar em tempo real as faces e fazer a predição referente a utilização da máscara, ao indicar a câmera 0 o sistema entende que é a câmera padrão do computador.

**Figura 15** – Utilização da Câmera



A biblioteca Open CV possui vários utilitários e um deles é um arquivo XML que irá ajudar o modelo a identificar as faces através de uma configuração inclusa no XML. A Figura 16 ilustra o método de carregamento do arquivo XML.

O arquivo XML é instalado na máquina assim que a biblioteca também é instalada.

**Figura 16** – Utilizando pré-configuração do OpenCV



## Executando o modelo

Para o modelo ser executado as configurações conforme a Figura 17 precisam ser passadas, de início se inicia uma estrutura de repetição que irá executar sem parar até que o usuário pressione a tecla indicada para interromper a execução do modelo.

A câmera então é ligada e as imagens por ela coletada são redimensionadas para que o modelo siga sempre um padrão de imagem coletada, com o classificador do Open CV é identificado as faces, o modelo tem capacidade de identificar múltiplas faces e por isso ele itera sobre cada rosto analisando sua imagem e ajustando as suas dimensões para o modelo e por fim realizando o predict.

Ao realizar o predict o modelo retorna os valores e junto ao Open CV ele exibe na tela o retângulo para cada face, indicando se a pessoa está ou não utilizando a máscara.

**Figura 17** – Executando Predição do Modelo

Texto

Descrição gerada automaticamente

A Figura 18 ilustra o código responsável pela interrupção do uso da câmera, em caso da tecla ESC ser pressionada a estrutura de repetição que está constantemente lendo os dados da câmera será interrompida, saindo da estrutura de repetição e finalizando a execução do modelo e câmera.

**Figura 18** – Interrupção da execução do modelo

Texto

Descrição gerada automaticamente

Para garantir que tudo foi interrompido, foi adicionado o código demonstrado na Figura 19 que finaliza a execução da câmera e do modelo por completo.

**Figura 19** – Interrupção da câmera junto ao modelo



# Resultados

Nesta seção será realizada uma análise crítica dos resultados obtidos, comparando com os esperados e os visualizados na Fundamentação Técnica.

## Modelo

Ao longo do trabalho foram treinados diversos modelos com diferentes hiper parâmetros, devido à falta de dados foi utilizado da técnica de *Data Augmentation* para gerar mais amostras de dados para o modelo, entretanto mesmo com a utilização da técnica o modelo não teve grandes ganhos em sua acurácia e demonstrou um problema referente a falta de amostras.

A Figura 20 apresenta os valores de perca e acurácia do modelo a cada epoch, devido a essa análise foi entendido que o valor de 10 epochs era suficiente.

**Figura 20** – Análise de acurácia e perda do modelo

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

O gráfico mostra que conforme a acurácia do modelo foi crescendo sua imprecisão foi reduzindo proporcionalmente.

A Figura 21 apresenta como executar mais epochs não traria nenhum ganho importante para o modelo, para ilustrar isso o gráfico mostra o modelo realizando o ciclo de 100 epochs e tendo quase os mesmos resultados.

**Figura 21** – Análise de treino excessivo do modelo

Gráfico, Histograma

Descrição gerada automaticamente

Diminuir o número de epochs faz com que o modelo seja treinado mais rapidamente e utilizando menos recurso computacional evitando que ele também desenvolva o Overfitting.

O modelo em si mostrou resultados interessantes em suas predições em casos simples.

## Predições

As predições realizadas pelo modelo se mostraram muito frágeis quando expostas a um fator muito diferente das imagens de treino, ainda sim foi possível conquistar resultados interessantes em casos simples, a seguir será apresentado uma série de resultados tanto positivos quanto negativos para evidenciar como as predições feitas pelo modelo resultaram.

A Figura 22 apresenta a predição com a máscara sendo utilizada corretamente:

**Figura 22** – Predição do modelo identificando a máscara

Uma imagem contendo pessoa, no interior, verde, homem

Descrição gerada automaticamente

A Figura 23 apresenta a predição com a máscara sendo utilizada incorretamente:

**Figura 23** – Predição do modelo ao não encontrar a máscara

Pessoa posando para foto

Descrição gerada automaticamente com confiança média

A Figura 24 apresenta a predição com a máscara cobrindo apenas parcialmente o rosto:

**Figura 24** – Predição do modelo não utilizando máscara corretamente

Homem com fone de ouvido

Descrição gerada automaticamente com confiança baixa

A Figura 25 apresenta a predição com a máscara contendo duas pessoas onde uma está utilizando corretamente e outra está sem máscara:

**Figura 25** – Predições distintas do Modelo

Tela de celular com foto de homem

Descrição gerada automaticamente com confiança baixa

A Figura 26 apresenta a predição com a máscara contendo duas pessoas onde uma está utilizando corretamente e outra está apenas tampando a face coma mão, evidenciando um problema no modelo:

**Figura 26** – Predição errada do Modelo

Tela de celular com foto de homem

Descrição gerada automaticamente com confiança baixa

A Figura 27 demonstra a predição com a máscara contendo duas pessoas utilizando corretamente a máscara:

**Figura 27** – Predição do modelo em dois rostos

Tela de celular com publicação numa rede social

Descrição gerada automaticamente com confiança baixa

A Figura 28 demonstra a predição com a máscara contendo duas pessoas utilizando máscara diferentes de forma diferente:

**Figura 28** – Predição do modelo em dois rostos

Uma imagem contendo no interior, pessoa, criança, verde

Descrição gerada automaticamente

# Considerações finais

Nessa seção será listado as contribuições do trabalho, experiências e dificuldades dos autores no decorrer do trabalho.

## Contribuições

O projeto contribui para identificar quem está usando ou não as máscaras, auxiliando na fiscalização e identificando caso exista uma pessoa transitando sem máscara, as contribuições desse projeto podem impactar positivamente a segurança a saúde da população.

## Dificuldades

A quantia de dados é muito para treinar o modelo, isso dificultou com que fosse possível criar um modelo melhor que aceitasse mais cenários e situações e paralelamente tivesse um bom resultado, foi utilizado de técnicas para aumentar a amostra de dados, entretanto não foi o suficiente para ajustar esse problema e o modelo ainda ficou com problemas em identificar algumas situações.

Encontrar os dados foram um problema visto que por depender de imagens muito especificas existiam poucas amostras pela internet.

## Trabalhos futuros

Este trabalho não encerra as contribuições na problemática de identificação de utilização da máscara, mas abre oportunidade para os seguintes trabalhos futuros:

- Otimização do modelo com mais dados

- Integração a sistemas comerciais

- Sistemas mais complexos acionados pelos resultados do modelo

- Controle local de pessoas transitando sem máscara

# REFERÊNCIAS

Analytics Vidhya. **Image Augmentation on the fly using Keras ImageDataGenerator.** Disponível em https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/Acesso em: 11/08/2020

Medium. **Cross Validation: Avaliando seu modelo de Machine Learning.** Disponível em https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78 Acesso em: 20/11/2020

Abracd. **Overfitting e underfitting em Machine Learning.** Disponível em https://abracd.org/overfitting-e-underfitting-em-machine-learning/Acesso em: 20/11/2020

Machinecurve. **How to use K-fold Cross Validation with TensorFlow 2 and Keras.** Disponível em https://www.machinecurve.com/index.php/2020/02/18/how-to-use-k-fold-cross-validation-with-keras/ Acesso em: 22/11/2020

Machinelearning Mastery. **Difference Between a Batch and an Epoch in a Neural Network.** Disponível em https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/Acesso em: 22/11/2020

Aliger. **As Redes Neurais Convolucionais no Deep Learning.** Disponível em https://www.aliger.com.br/blog/as-redes-neuronais-convolutivas-no-deep-learning/Acesso em: 27/11/2020