UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE - UFRN
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO - DCA

LEANDRO DE SOUZA RODRIGUES

RECONHECIMENTO DO USO DE MÁSCARA - COVID19

Processamento Digital de Imagens

Natal, RN
12/ 2020

1. Introdução

Uma imagem pode ser definida como uma função bidimensional, f (x, y), em que x e y são coordenadas espaciais (plano), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem nesse ponto. No caso do processamento de imagens coloridas, a intensidade pode ser dividida em três níveis RGB[1].

A visão computacional tem como meta utilizar computadores para emular a visão humana, incluindo o aprendizado e a capacidade de fazer inferências e agir com base em informações visuais. Essa área representa um ramo da inteligência artificial (AI, de artificial intelligence) cujo objetivo é emular a inteligência humana. A área da AI ainda está em seus estágios iniciais de desenvolvimento e o progresso tem sido muito mais lento do que o originalmente previsto. A área da análise de imagens (também chamada de compreensão de imagens) está situada entre o processamento de imagens e a visão computacional[2].

Em 2020, ano que o vírus Covid19 se espalhou rapidamente e foi considerado uma pandemia. A OMS(Organização Mundial da Saúde) recomenda o uso de máscaras para diminuir a transmissão do vírus. A máscara deve tampar o nariz até o queixo, dando a proteção total.

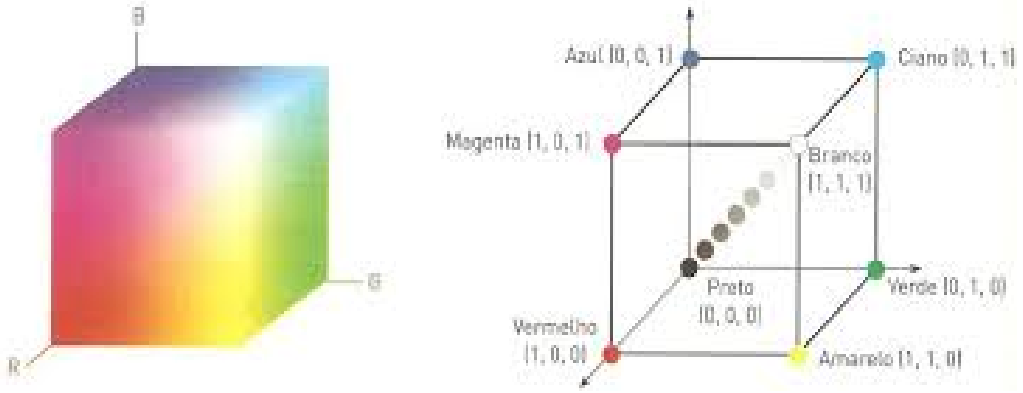
Neste contexto de visão computacional e importância do uso da máscara na população, este relatório desenvolve um detector de máscara em uso humano. Para isso, foi usado o tutorial "COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning", em que foi implementado o algoritmo adaptando o banco de dados e alguns parâmetros.

2. Análise Teórica

2.1 Modelo RGB

No modelo RGB, cada cor aparece em seus componentes espectrais primários de vermelho, verde e azul. Esse modelo se baseia em um sistema de coordenadas cartesianas. O subespaço de cores de interesse é o cubo, Figura 01, no qual os valores RGB primários estão em três vértices; as cores secundárias ciano, magenta e amarelo estão em outros três vértices; o preto está na origem; e o branco está no vértice mais distante da origem. Nesse modelo, a escala de cinza (pontos de valores RGB iguais) estende-se do preto até o branco ao longo do segmento de reta que une esses dois pontos. As diferentes cores nesse modelo são pontos no cubo ou dentro dele e são definidas por vetores que se estendem a partir da origem. Quando alimentadas em um monitor RGB, essas três imagens se combinam na tela para produzir uma imagem de cores compostas[1].

FIGURA 1 - Modelo RGB

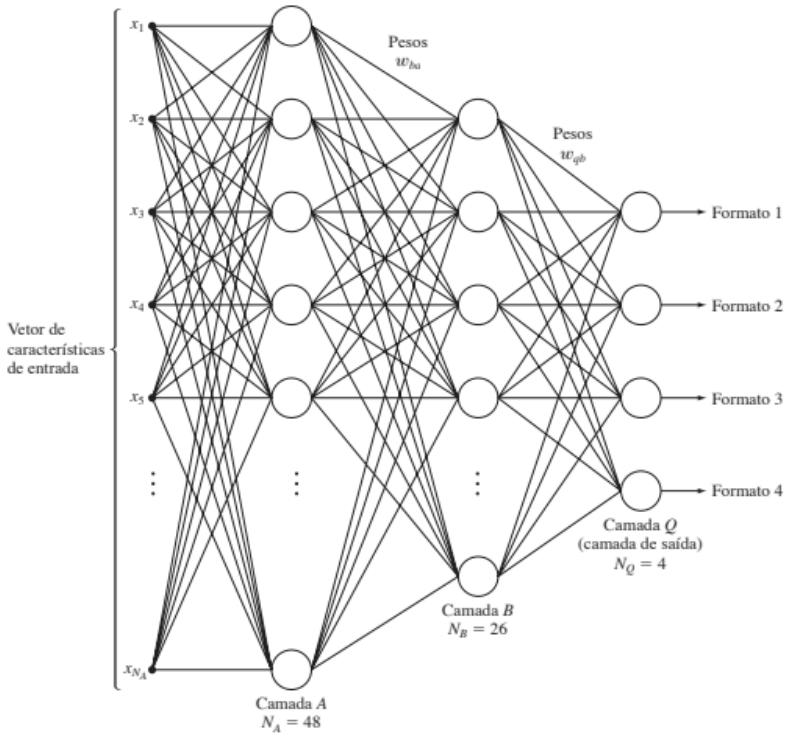


2.2 Aprendizado Profundo

A aprendizagem profunda, do inglês Deep Learning é um ramo de aprendizado de máquina baseado em um conjunto de algoritmos que tentam modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento, compostas de várias transformações lineares e não lineares. Aprendizado profundo é então, um tipo de aprendizado de máquina no qual os computadores formam grandes redes neurais artificiais, semelhantes às encontradas no cérebro humano[2].

Com a deep learning, grandes redes neurais artificiais são alimentadas por algoritmos de aprendizado e quantidades crescentes de dados, melhorando continuamente sua capacidade de “pensar” e “aprender”, conforme processam mais e mais dados. “Profundo” por conta das muitas camadas que a rede neural acumula com o tempo, e o desempenho melhora quanto mais profunda a rede for. Hoje, grande parte do aprendizado profundo acontece sob a supervisão humana, mas o objetivo é criar redes neurais que consigam treinar a si mesmas e “aprender” de modo independente. A figura 02 mostra um exemplo artificial de camadas de neurônios se interligando.

FIGURA 02- Exemplo de aprendizado profundo



3. Metodologia

A metodologia demonstrada neste relatório foca na parte de processamento de imagens e apenas comenta o que se refere ao aprendizado profundo do detector de máscara. Todos os códigos e imagens podem ser visto no Anexo A. Foi usado a linguagem Python para a programação com as seguintes bibliotecas: Keras/TensorFlow e OpenCV. Os dados de entradas foram divididos em duas classes: com máscara e sem máscara , e também podem ser vistos no Anexo A. Um ponto importante de se destacar é que as máscaras utilizadas indevidamente foram consideradas na classe “sem máscara”[2].

As funções principais da parte de processamento digital são as expostas na figura 03 E 04. A “loaf_img” carrega a imagem no formato requisitado pela programação python, a “img_to_array” transforma a imagem em conjunto de vetor no formato de 3 canais RGB, o “preprocess_input” destina-se a adequar sua imagem ao formato que o modelo requer. A função “ImageDataGenerato” realiza várias transformações aleatórias na imagem, como rotação, translação. Isso ajuda o banco de dados a se tornar mais flexível. Esse vetor gerado com as intensidades RGB já é suficiente para que o modelo de aprendizagem profunda possa entender os padrões da imagem, as transformações realizadas na imagem ajuda ao modelo se tornar mais robusto[3].

FIGURA 03- Processamento inicial da imagem

```
image = load_img(imagePath, target_size=(64, 64))
image = img_to_array(image)
image = preprocess_input(image)
```

FIGURA 04 - Transformações na imagem

```
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

Na parte do aprendizado profundo, a parte principal está na figura 05. Pode ser visto como o código foi treinado, utilizou-se de uma camada inicial de 32 neurônios e última de 2, com a ativação softmax, no qual possibilita que a saída esteja no intervalo [0, 1].

FIGURA 05 - Modelo de treinamento

```
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(1, 1))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(32, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

Na parte de exibição da imagem de teste(Figura 06), demonstrando se está com a máscara, foram usadas funções como intuito de localizar o rosto, extrair regiões de interresse(ROI), e identificar pontos com nariz, olhos etc. Depois disso, é verificado se o modelo previu qual rótulo e escreve acima do rosto identificado tal rótulo. Nos Resultados, é demonstrado em uma imagem tais operações[4].

FIGURA 06 - Plotagem do rótulo no face da imagem de teste.

```
face = image[startY:endY, startX:endX]
face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
face = cv2.resize(face, (64, 64))
face = img_to_array(face)
face = preprocess_input(face)
face = np.expand_dims(face, axis=0)

# pass the face through the model to determine if the face
# has a mask or not
(mask, withoutMask) = model.predict(face)[0]

# determine the class label and color we'll use to draw
# the bounding box and text
label = "Com mascara" if mask > withoutMask else "Sem mascara"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
```

4. Resultados

A rede foi testada com 64 imagens com máscara e 64 sem máscara, com total de 128. Como resultados do reconhecimento de máscara, obteve a precisão de 80% com máscara e 69% sem máscara. A figura 07 mostra como o sistema de treinamento vai diminuindo o erro ou "loss" a medida que irá aumentar as rotas de treinamento, ou "epoch". Percebe que a 17 é melhor Epoch, pois tem menor erro. No total, foram utilizados 20.

FIGURA 07- Erro ao longo das Epoch's

A imagem usada no teste 1, figura 08, obteve 99,96% de certeza, um bom resultado; o modelo conseguiu prever que eu estava de máscara. As imagens que possuem menos acurácia são as imagens ditas com sem máscara em que a máscara está sendo usada de forma irregular, e possivelmente, isso abaixo a acurácia do modelo. A Figura 09 mostra um caso que a ele previu certo.

FIGURA 08- Imagem de teste 01 com previsão correta

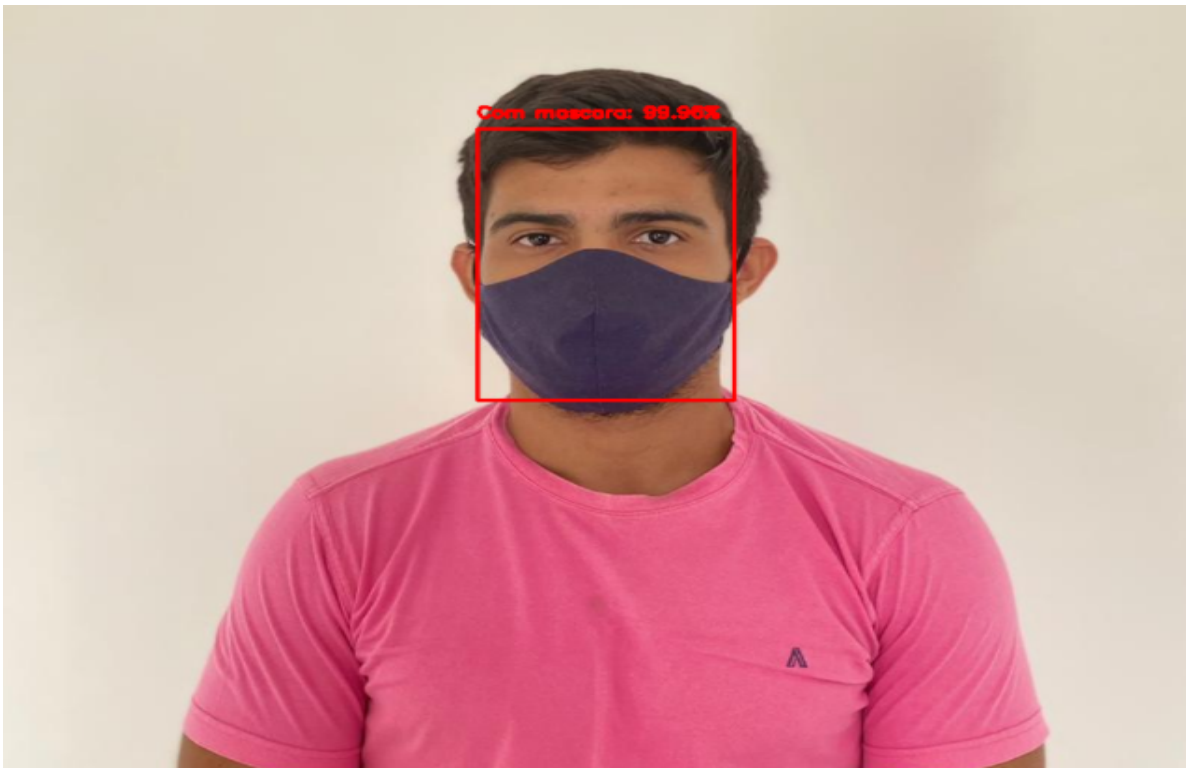
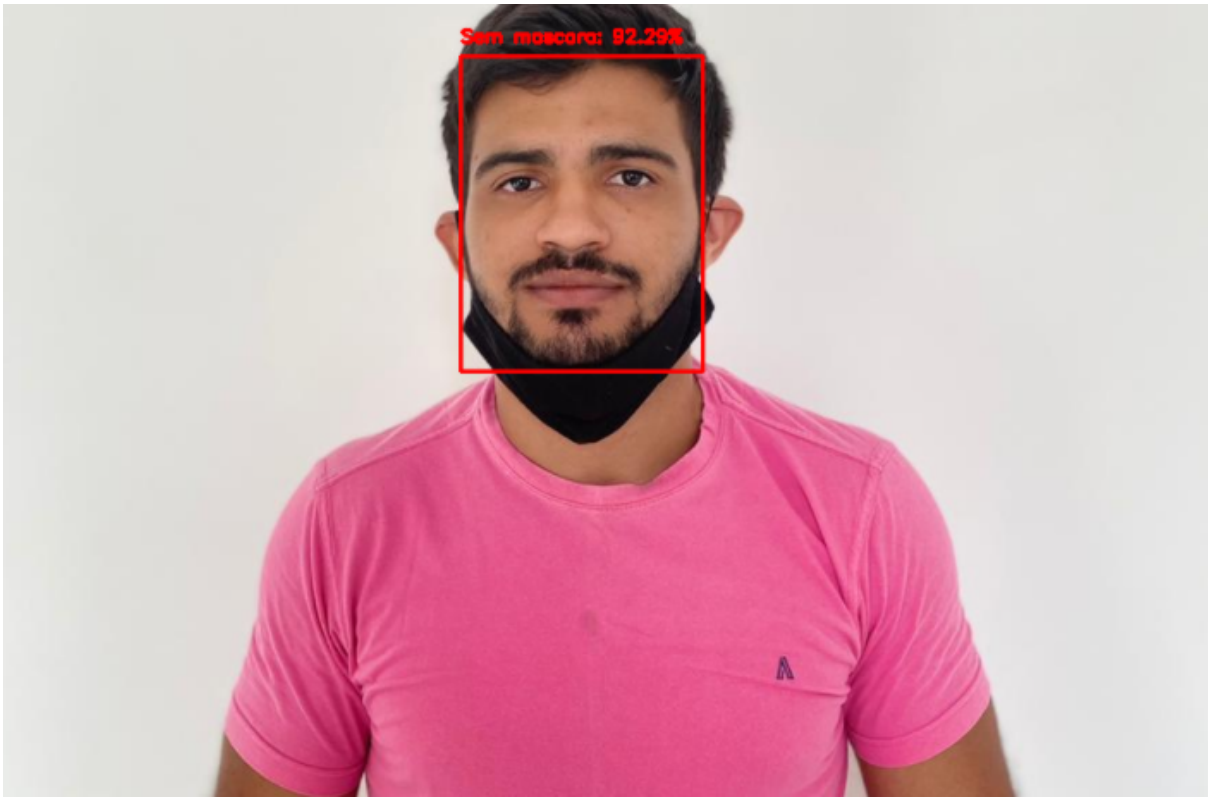


FIGURA 09- Imagem de teste 02 com previsão correta



5. Conclusões

O reconhecedor de máscara pode ser aplicado em shopping, condomínios etc para que a entrada das pessoas siga a legislação municipal. No modelo deste relatório, foram usadas poucas imagens e adicionado o caso de estar com a máscara no rosto, mas de forma irregular classificada como “sem máscara”, e isso possivelmente diminui a acurácia do modelo. Por mais que teve um resultado bom, poderia ser melhorado adicionando mais imagens e configurando melhor os parâmetros.

Outra coisa que poderia ser feita em caso de dados ruídos, má iluminação etc era utilizar convolução e outras técnicas de processamento de imagens. Como as imagens tinham boa qualidade, não foi preciso.

6. Referências

[1] GONZALES, WOODS. Processamento Digital de Imagens, 3º Ed. Editora Pearson, 2010.
[2] ROSEBROK, Adrian. COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning, 2020. Disponível em: <https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/>
[3] KERAS. Keras API reference. Disponível em : <https://keras.io/api/>
[4] OPENCV. Opencv modules. Disponível em : <https://docs.opencv.org/master/>

7. Anexo A

link do projeto : https://drive.google.com/drive/folders/1iKooWoRHIS5Ee_R9a9wOt7VRyLDYhY4f?usp=sharing