Detecção de Máscaras em Faces Utilizando Aprendizado de Máquina

1st Ícaro Peretti Baseggio
Instituto Federal Catarinense
Videira, Brasil
icaroperetti50@gmail.com

***Resumo*—Com a utilização de técnicas do Machine Learning (aprendizado de máquina) e a linguagem de programação python, é possível desenvolver um algoritmo capaz de identificar se há ou não uma máscara no rosto de uma pessoa utilizando a biblioteca OpenCV, desenvolvida pela Intel para realizar o processamento das imagens, e, a biblioteca scikit-learn para realizar o treinamento do modelo de classificação a fim de obter o resultado da verificação para tal identificação.**

***Index Terms*—inteligência artificial, machine learning, detecção de faces, visão computacional**

I. Introdução

Durante os anos de 2020 e 2021 o mundo presenciou uma pandemia causada pelo vírus SARS-Cov-2 também conhecida como COVID-19, diversos países adotaram medidas preventivas como: o isolamento social, a higienização das mãos, objetos e alimentos e principalmente a utilização de máscara facial tanto em ambientes públicos fechados quanto em ambientes abertos afim de mitigar a propagação da doença. A partir deste ponto, pode-se considerar que um sistema de detecção de máscaras em rostos de grande utilidade, e vai além, são diversas as empresas onde o uso de máscaras é obrigatório, a exemplo as do ramo alimentício, essa empresas podem se beneficiar do sistema a partir de uma adaptação do modelo para realizar a captura de video em tempo real ao invés de imagens e aplicar em algum sistema de monitoramento.

Portanto o presente trabalho demonstra um sistema de detecção de máscaras faciais utilizando das técnicas do Machine Learning (aprendizado de máquina) tendo como ferramentas a linguagem de programação python, a biblioteca desenvolvida pela intel OpenCV que fornece diversos modelos de aprendizado de máquina prontos, e outras biblioteca como pandas que é amplamente utilizada na manipulação e analise de dados, numpy blibloteca que fornece diversas ferramentas matemáticas, scikit-learn que fornece diversos classificadores de aprendizado de máquina e matplotlib para realizar a exibição das imagens.

II. Referencial Teórico

A. Aprendizado de Máquina

O aprendizado de máquina é uma subárea da inteligência artificial que tem por objetivo o desenvolvimento de sistemas computacionais capaz de obter conhecimento de maneira autônoma [1].

B. Aprendizado Supervisionado

Os algoritmos de aprendizado de máquina que utilizam aprendizado supervisionado, de acordo com [1] buscam obter um bom classificador a partir de uma base de dados rotulada, ou seja, se há uma base de dados com múltiplas imagens de gatos e cachorros cada uma adequadamente classificada (rotulada) o objetivo será desenvolver um modelo capaz de identificar a partir de novas imagens não rotuladas fornecidas ao algoritmo se nelas há um gato ou um cachorro. Outra forma é realizar a extração de características de uma imagem e converte-las para um vetor de valores de atributos.

- **Atributos descrevem características ou aspectos de um exemplo podendo estes serem nominais (cores) ou contínuos (peso, números reais ou números inteiros)**

C. Visão Computacional

Para este projeto o aprendizado de máquina aplicado a cada imagem contida no dataste foi o de feature extraction (extração de características) que realiza a quantificação da imagem de entrada de acordo com o algoritmo de extração utilizado. Após realizada a extração destas características as mesmas são utilizadas para dar entrada no modelo de aprendizado de máquina [2].

D. K-nearest neighbors (KNN)

A técnica dos k-vizinhos mais próximos é uma técnica que considera a proximidade entre os dados para realizar as predições. Este algoritmo trabalha com a hipótese de que dados similares tendem a se concentrar em um espaço de dispersão. O algoritmo k-nearest-neighbor é normalmente baseado na distância euclidiana entre a amostra de teste e a amostra de treinamento [3].

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

Supondo que existam duas classes, sendo elas azul e vermelho e uma quantidade de vizinhos (k=3), quando um novo dado é adicionado sem rotulação, é realizada o calculo da distância do novo dado em relação aos dados já rotulados (classificados), ocorrendo uma espécie de "eleição"onde a classificação dos k próximos vizinhos vence [3].

Na Figura 1 é possível observar que existem duas amostras de classe vermelha e uma de classe azul, portanto a eleição fica em 2x1 a favor da classe vermelha, consequentemente o novo dado será classificado como vermelho, como pode-se observar na Figura 2.

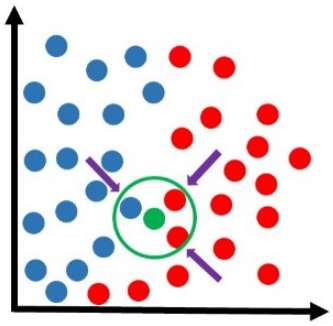

Figura 1. Fonte:[3]

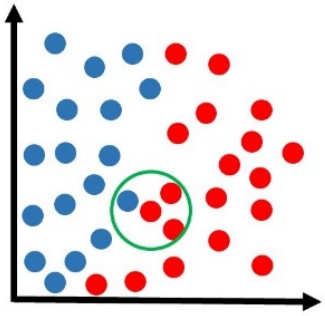

Figura 2. Gráfico após a eleição ocorrer. Fonte:[3]

E. Cascade Classifier

O Cascade Classifier trata-se de uma implementação do algoritmo conhecido como *Viola-Jones* e está contido na biblioteca OpenCV, um algoritmo desenvolvido para realizar a detecção de objetos em imagens, e também é efetivo para detecção de rostos. Este algoritmo é amplamente utilizado devido sua performance sendo capaz de realizar a detecção de objetos de uma maneira incrivelmente rápida [6]. O algoritmo *Viola-Jones* utiliza Features de Haar para detectar o objeto proposto, no caso do presente trabalho, realizar a detecção de rostos em imagens.

Na Figura 3 é possível observar como as features de haar são aplicadas no problema de detecção de faces, diferentes segmentos da face serão mapeados por meio de "máscaras"que buscam detectar padrões de luminosidade em cada segmento e também cada máscara será aplicada em direções diferentes. As features de Haar irão retornar um valor único calculado pela seguinte fórmula:

$$f(x) = \sum_{black}^{n} - \sum_{white}^{n}$$

Sendo o valor de f(x) a subtração dos pixels pretos menos os pixels brancos detectados pelas features de Haar[7].

Utilizando a biblioteca OpenCV e o algoritmo Cascade Classifier fornecido pela biblioteca, é possível aumentar a performance do algoritmo de *Viola-Jones* (e outros) utilizando *Adaptive Boosting*, o adaptive boosting busca construir um classificador forte utilizando um ou mais algoritmos de aprendizado de máquina que não possuem um desempenho elevado, tendo como intuito de que cada preditor realize a correção da classificação feita pelo seu predecessor [9].

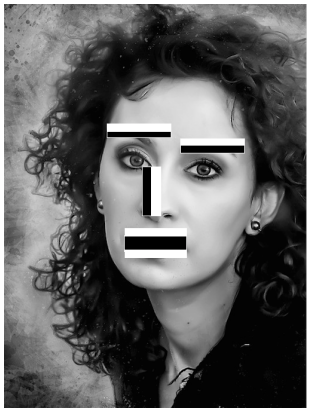

Figura 3. Features de Haar. Fonte:[7]

III. Desenvolvimento

Para realizar o desenvolvimento do presente trabalho a linguagem de programação escolhida foi Python, o ambiente de desenvolvimento foi construído pela plataforma Collaboratory fornecida gratuitamente pelo Google visto que já possui diversas ferramentas já instaladas. Para realizar o processamento das será utilizada a biblioteca OpenCV, para realizar o treinamento do modelo foi utilizada a biblioteca sklearn.

Previamente ao desenvolvimento do modelo que irá realizar a classificação das imagens, é necessário desenvolver um código que seja responsável por detectar faces em uma imagem. Para realizar a detecção da face em uma imagem foi necessário realizar um pré-processamento a fim de diminuir a quantidade de ruídos na imagem, a tática adotada foi realizar a conversão da imagem do padrão BGR para o padrão cinza RGB.

Como citado na seção do referencial teórico, o algoritmo selecionado para realizar a detecção das faces foi o Cascade Classifier, tratando-se de uma implementação do algoritmo de *Viola-Jones*. A biblioteca do OpenCV possui diversos arquivos xml que são utilizados para realizar o adaptive boosting do Cascade Classifier, durante o processo de desenvolvimento do algoritmo, diferentes arquivos xml para detecção de faces foram utilizados, sendo eles:

- `haarcascade_frontalface_default.xml`
- `haarcascade_frontalface_alt.xml`
- `haarcascade_frontalface_alt2.xml`
- `haarcascade_frontalface_alt_tree.xml`

Tendo o melhor desempenho sendo apresentado pelo `haarcascade_frontalface_alt2.xml`. Assim armazenado a localização do arquivos de features de Haar já disponibilizados pelo OpenCV em um caminho para que seja utilizado posteriormente. Após realizar todo provisionamento do modelo de cascade classifier, foi possível então realizar detecção de faces em imagens a partir do método *detectMultiScale*, passando a imagem convertida para escala de tons de cinza e então e destaca-las com um quadrado de cor amarela, como observado na Figura 4. Em seguida é aplicado um redimensionamento da imagem para que a face fique em maior destaque como observado na Figura 5. Cada face encontrada irá retornar os pontos noeixo x e y iniciais e finais da largura e altura.

Em seguida as imagens de um dataset foram organizadas em um diretório localizado no meu google drive, tornando possível a leitura dos arquivos por meio do google collaboratory. Em uma

pasta foram carregadas outras duas pastas uma chamada with-masks possuindo 724 imagens de pessoas utilizando máscara e uma chamada no-mask com cerca 273 imagens, devido a este desbalanceamento do dataset tendo em vista que existem mais imagens de pessoas usando máscara foi aplicada a técnica de balanceamento de dados chamada smote que é provida pela biblioteca imblearn. Após o carregamento destas imagens, com o auxilio da biblioteca pandas um dataframe é gerado a partir da leitura dos dados contidos dentro de cada pasta e então este dataframe é convertido para csv contendo a localização do arquivo, a rotulação sendo 1 para está utilizando mascara e 0 para não está utilizando máscara como observado na Figura 6.

Em seguida para cada arquivo do dataset é realizada a leitura de cada imagem contida na coluna *FILE* do dataset é realizada pelo meio do método imread da biblioteca OpenCV e então a mesma é convertida para escala de cinza e logo em seguida para um array, array este que será armazenado em uma nova coluna do dataset chamada *IMAGE*.

O conjunto de características (X) será a coluna do dataset chamada *IMAGE* e a classe de identificação será a WEARING_MASK (y). Após estas definições, será aplicada a técnica de balanceamento de dados citada previamente em seguida serão separados os conjuntos de treino e de teste, selecionando um tamanho de 80% para treinamento, pois utilizando 1/3 do dataset para treino os resultados foram relativamente ruins e o random_state possuindo o valor 7 .

Em seguida um método estatístico chamado PCA, que realiza a extração das melhores características foi aplicado em cada imagem contida no conjunto de treinamento e extraídos 30 componentes.

Uma vez realizada a extração das características, foi feito o treinamento do algoritmo de KNN. O KNN irá realizar a distância do novo elemento recebido em relação ao restante dos K elementos já presentes no conjunto, a partir do momento em que ele se aproxime de uma quantidade K de elementos com determinada classificação, ele passa a ser classificado com a mesma classe.

O método de GridSearch também foi utilizado, o GridSearch recebe por parâmetro o algoritmo de KNN e hiper parâmetros, sendo estes um objeto contendo o número de vizinhos, uma lista com opções de pesos e uma lista de opções de cálculos de métricas. A partir disto o GridSearchCV irá se responsabilizar por escolher os melhores parâmetros para realizar o treinamento do modelo.

A. Classificando novas imagens

Após a realização do treinamento do modelo, para realizar os testes um dicionário contendo os valores com máscara e sem mascara tendo seus valores respectivos como 1 e 0. Imagens buscada na internet e imagens pessoais foram processadas pelo modelo PCA.No primeiro teste realizando o carregamento de uma imagem de diversas pessoas sem máscara o modelo apresentou um resultado de 100%, pois a imagem possui 4 pessoas sem máscara, os quatro rostos foram detectados e classificados como sem mascara como mostra a Figura 6, no segundo teste o resultado esperado também foi obtido, uma imagem de uma pessoa do lado esquerdo sem máscara e do lado direito com máscara o modelo foi capaz de classificar corretamente como pode ser visualizado na Figura 7. Já no terceiro teste uma imagem contendo 3 pessoas, sendo duas sem a utilização de máscara e uma com a utilização de máscara o algoritmo classificou corretamente porém não reconheceu a face da pessoa que estava utilizando máscara.

IV. Conclusão

O presente artigo tem como fundamento de seu estudo a aplicação do Machine Learning (aprendizado de maquina) para detecção de máscaras em faces humanas, e, por meio de testes

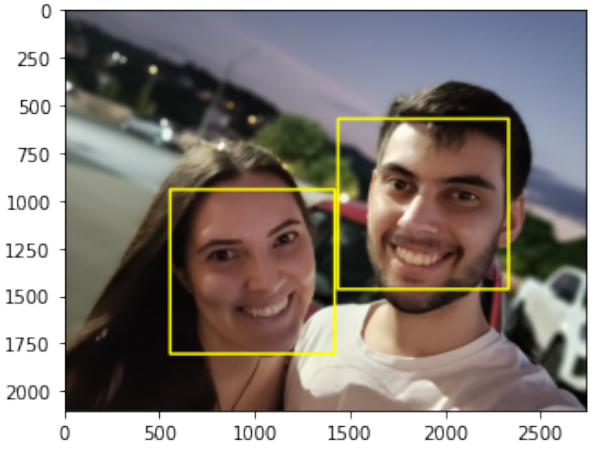

Figura 4. Destaque de rosto detectado. Fonte:Autor

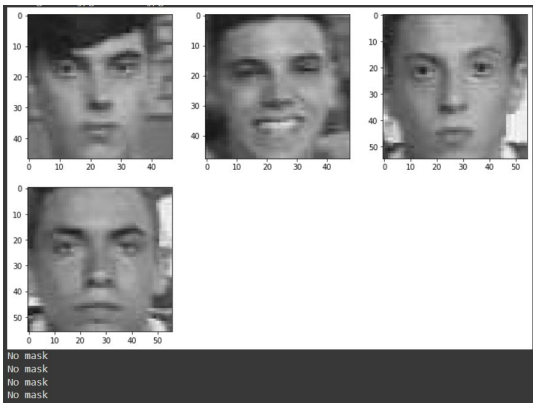

Figura 5. Primeiro teste de detecção. Fonte:Autor

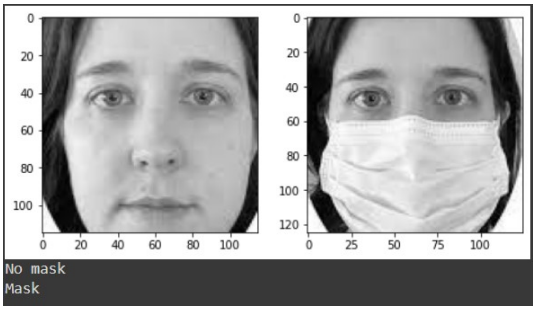

Figura 6. Segundo teste de detecção. Fonte:Autor

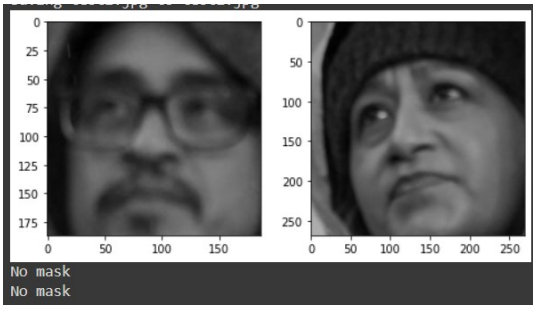

Figura 7. Terceiro teste de detecção. Fonte:Autor

realizados com cerca de 997 imagens de pessoas provou-se que o algoritmo tem uma acurácia de 86.62% sendo capaz de de identificar corretamente pessoas utilizando ou não máscaras em seus rostos. Porém é de grande valia informar que algumas falhas onde rostos não foram detectados, ou foram detectados rostos em locais onde não haviam, podendo ter como causa a falta de rostos compatíveis no conjunto de dados ou questões como qualidade e luminosidade da imagem. Para um futuro estudo este modelo poderia ser aplicado para realizar a detecção de faces em tempo real por videomonitoramento podendo também ser estendido a outros equipamentos, como equipamentos de proteção individual que são tidos como exigência pelo ministério do trabalho.

REFERÊNCIAS

[1] MONARD, M.; BARANAUSKAS, J. Capítulo 4 Conceitos sobre Aprendizado de Máquina. [s.l: s.n.]. Disponível em: https://bit.ly/3cbb0JR Acesso em: 11 jul. 2022.

[2] TIBAU, MARCELO. Visão Computacional Na Era Do Deep Learning. www.updateordie.com/2021/05/14/visao-computacional-na-era-do-deep-learning/. Acceso em: 11 jul. 2022.

[3] PETERSON, L. K-nearest neighbor. Scholarpedia, v. 4, n. 2, p. 1883, 2009 Acesso em: 11 jul. 2022.

[4] Como funciona o KNN (K-nearest neighbors). Disponível em: https://didatica.tech/o-que-e-e-como-funciona-o-algoritmo-knn. Acesso em: 13 jul. 2022.

[5] Sandeco. KNN e os K vizinhos mais próximos, 10 jan. 2019. Disponível em: https://www.youtube.com/watch?v=gJK4fmCvcWY. Acesso em: 12 jul. 2022.

[6] Computerphile.Detecting Faces (Viola Jones Algorithm) - Computerphile, 2019. Disponível em: https://www.youtube.com/watch?v=uEJ71VlUmMQ. Acesso em: 9 jul. 2022.

[7] ADAKANE, D. What are Haar Features used in Face Detection ? Disponível em: https://medium.com/analytics-vidhya/what-is-haar-features-used-in-face-detection-a7e531c8332b. Acesso em: 9 jul. 2022.

[8] SANTOS, T. Detecção de faces através do algoritmo de Viola-Jones. Disponível em: https://www.lcg.ufrj.br/marroquim/courses/cos756/trabalhos/2011/tulio-ligneul/tulio-ligneul-report.pdf. Acesso em: 10 jul 2022.

[9] AZAMBUJA, P. AdaBoost (Adaptive Boosting). Disponível em: https://pedroazambuja.medium.com/adaboost-adaptive-boosting-dbbec150fced. Acesso em: 14 jul. 2022.