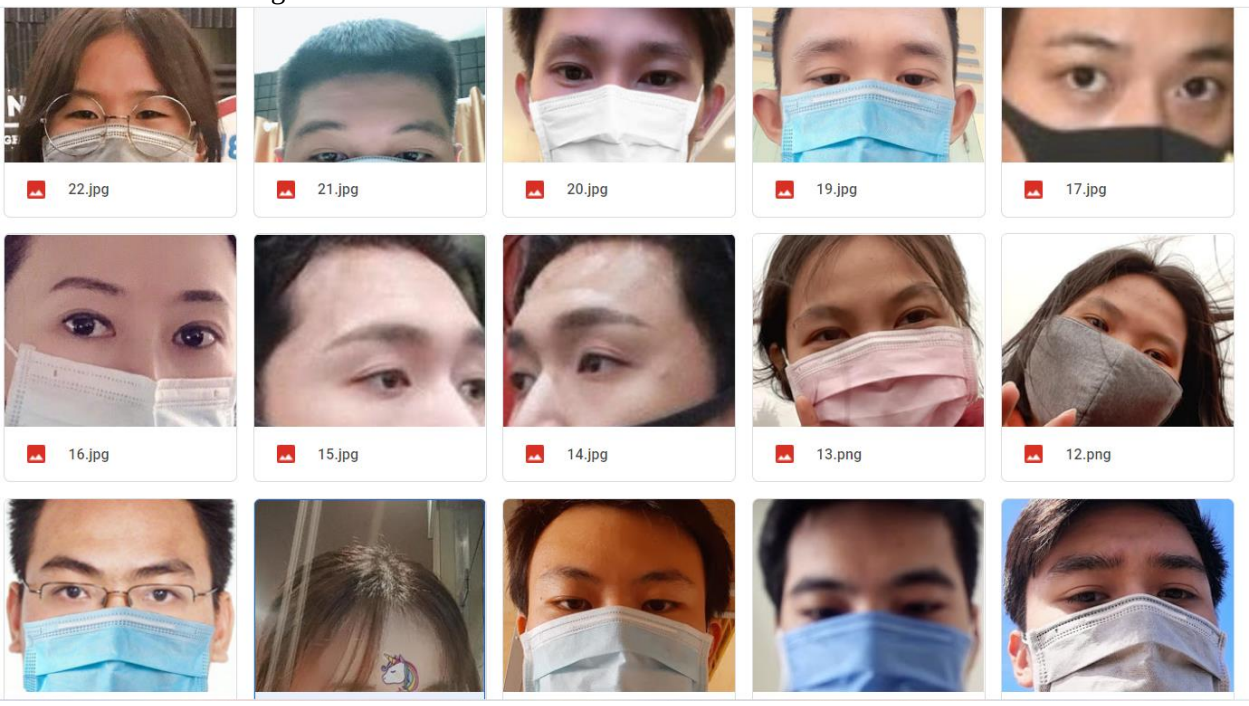
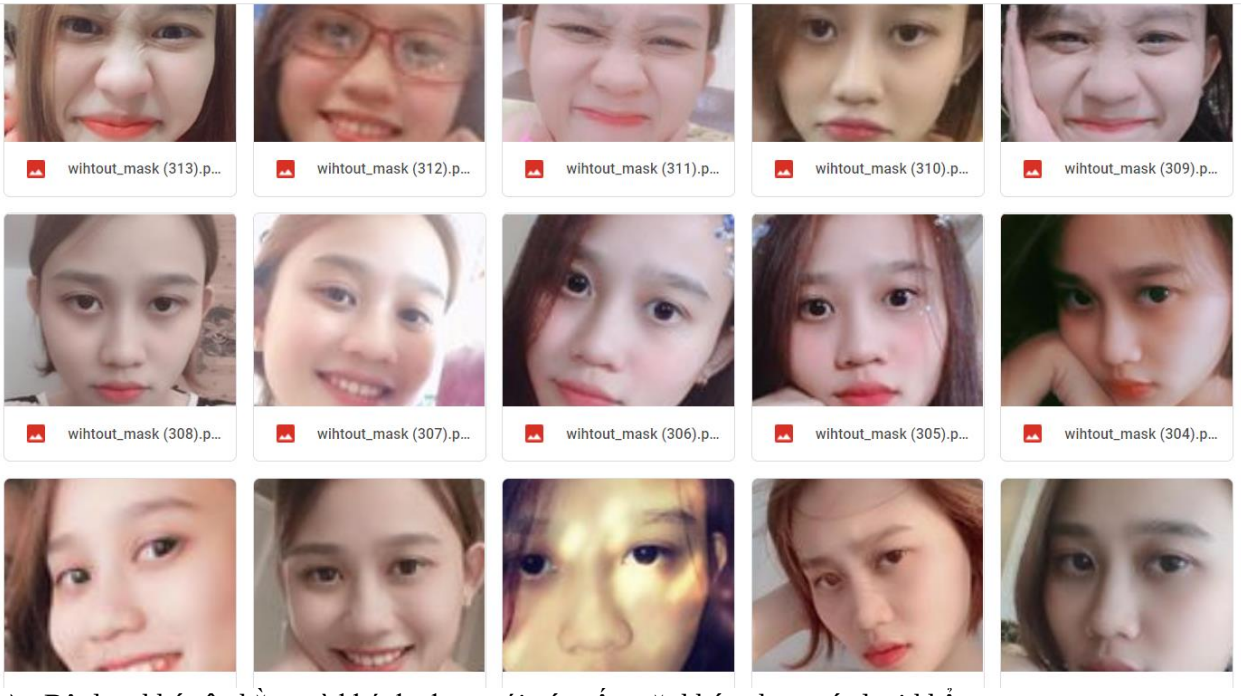
BÁO CÁO ĐỒ ÁN CUỐI KỲ

1. Mô tả bài toán
   a. Bài toán : Nhận diện ảnh khuôn mặt một người đeo khẩu trang
   b. Lí do chọn bài toán :
   + COVID-19 , bệnh viêm đường hô hấp cấp do chủng mới của virus corona được phát hiện lần đầu tiên tại thành phố Vũ Hán , tỉnh Hồ Bắc , Trung Quốc vào tháng 12 / 2019.
   + Đến nay, có 215 quốc gia / vùng lãnh thổ trên toàn câu ghi nhân trường hợp mắc . Trong đó nước Mỹ là nước có người nhiễm nhiều nhất.
   + Hiện tại sau 99 ngày Việt Nam không có ca nhiễm trong cộng đồng thì dịch đang dần trở lại.
   + Cách để phòng dịch tốt nhất được chính phủ và WHO khuyến cáo là đeo khẩu trang khi ra đường hay tiếp xúc với người khác.
   ⇨ Chính vì thế nhóm đã lên ý tưởng xây dựng một mô hình tự động nhận diện xem mọi người có đeo khẩu trang hay không.
   ⇨ Bài toán này là bài toán cơ sở , khi kết hợp với bài toán nhận diện khuôn mặt người sẽ tạo thành một bài toán thực tế có thể áp dụng trên diện rộng.
   c. Mô tả:
   Input : hình ảnh khuôn mặt một người
   Ouput :
   + 1 : người này có đeo khẩu trang
   + 0 : người này không đeo khẩu trang
2. Mô tả về bộ dữ liệu
   - Các ảnh trong bộ data được thu thập từ các hình ảnh tự chụp , xin ảnh người quen từ các ảnh chụp trước và một số ảnh từ google
   - Ảnh sau khi được thu thập sẽ dùng tool CasadeClassifer của OpenCV cắt mặt ra hoặc tự cắt đối với các mặt tool không nhận được mặt
   - Sau khi cắt mặt từ bộ ảnh thu được :
   + 201 ảnh có khẩu trang



+ 284 ảnh không có khẩu trang



⇨ Bộ data khá cân bằng và khá đa dạng với các gốc mặt khác nhau, các loại khẩu trang khác nhau.

- Tiền xử lí dữ liệu:

+ Ảnh sau khi được load lên sẽ được resize về kích thước 32x32
+ Sau đó ảnh từ dạng mảng 3 chiều sẽ được làm phẳng thành 1 Vector

```
# load hình ảnh
image = cv2.imread(imagePath)

# preprocess ảnh
image = cv2.resize(image,(32,32)).flatten()
```

+ Cuối cùng các Vector ảnh là label của nó được chuyển sang dạng numpy

```
# chuyển label và data sang dạng mảng
labels = np.array(labels)
data = np.array(data)
```

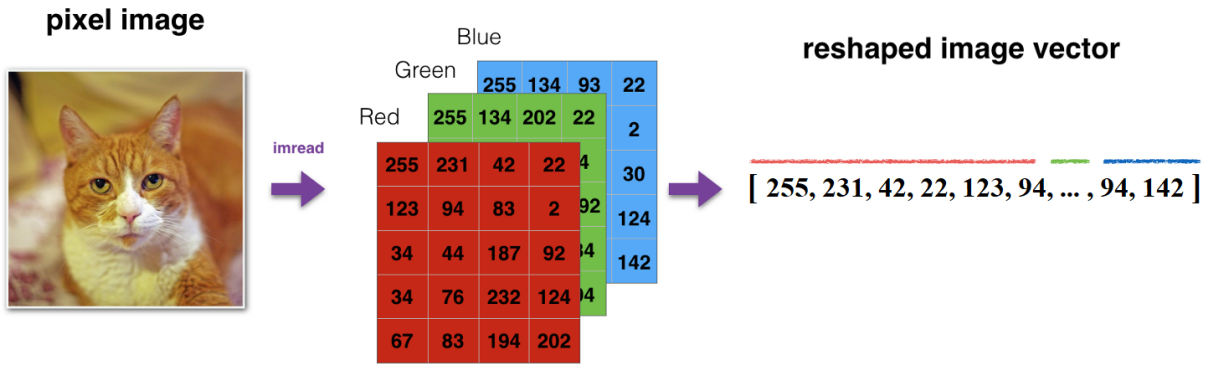
- Tiến hành chia dữ liệu thành 2 phần : 75% dùng để train và 20% dùng để test

```
# chia dữ liệu thành 75% để train và 25% để test
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.25, stratify=labels, random_state=1)
```

3. Mô tả về đặc trưng
   - Biến đổi từ 1 bức ảnh có mảng 3 chiều thành 1 vector
   - Từ một ảnh có kí thước 32x32 thì sẽ chuyển thành 1 vector có 32x32x3 phần tử vì một ảnh có 3 kênh màu Red , Green , Blue.
   - Việc này được thực hiện thông qua hàm flatten của thư viện numpy

```
# preprocess ảnh
image = cv2.resize(image,(32,32)).flatten()
```

- Hàm flatten sẽ lấy hàng ngang của từng cột sau đó ghép nó thành 1 hàng ngang duy nhất



4. Mô tả về thuật toán
   - Do label có dạng binary : 0 và 1 vì thế chọn các thuật toán Binary Classification
   - Tiến hành training và đánh giá model thông qua hàm classication_report
   - Chọn thuật toán có accuracy tốt nhất làm model để predict các ảnh thực tế
   - Các thuật toán được sử dụng :
     - Decision Tree

```
# Model Decision Tree
Model_DT = DecisionTreeClassifier()
Model_DT.fit(trainX,trainY)
print(classification_report(testY,Model_DT.predict(testX)))
```

     - Random Forest

```
# Model Random Forest
Model_RF = RandomForestClassifier()
Model_RF.fit(trainX,trainY)
print(classification_report(testY,Model_RF.predict(testX)))
```

     - Logicic Regression

```
# Model Logictic Regression
model_Logic = LogisticRegression()
model_Logic.fit(trainX,trainY)
print(classification_report(testY, model_Logic.predict(testX)))
```

5. Đánh giá kết quả, kết luận
   - Decision Tree:

	precision	recall	f1-score	support
0	0.90	0.89	0.89	71
1	0.85	0.86	0.85	51
accuracy			0.88	122
macro avg	0.87	0.88	0.87	122
weighted avg	0.88	0.88	0.88	122

   - Random Forest:

	precision	recall	f1-score	support
0	0.96	0.94	0.95	71
1	0.92	0.94	0.93	51
accuracy			0.94	122
macro avg	0.94	0.94	0.94	122
weighted avg	0.94	0.94	0.94	122

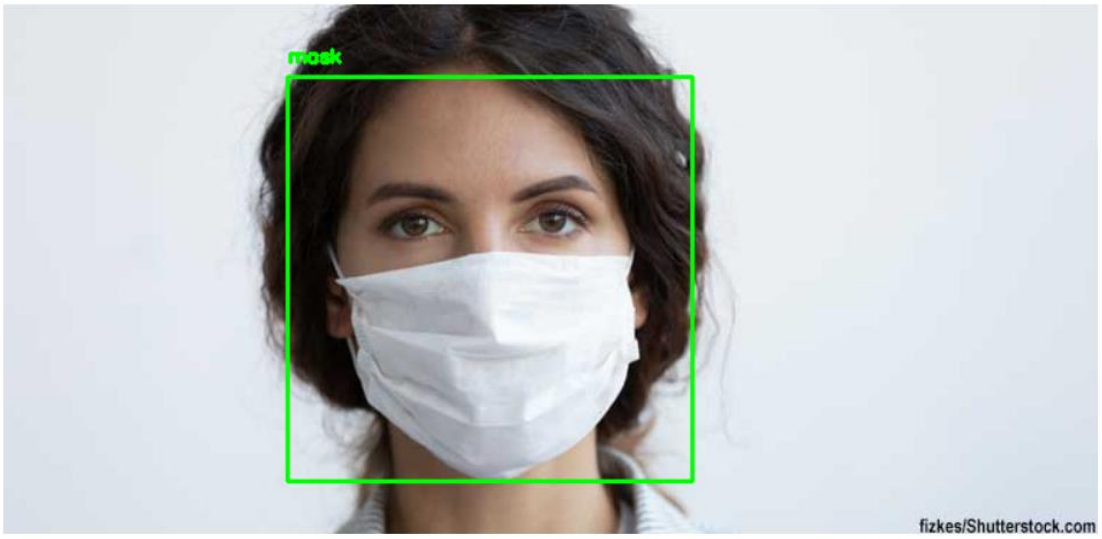
   - Logictic Regression:

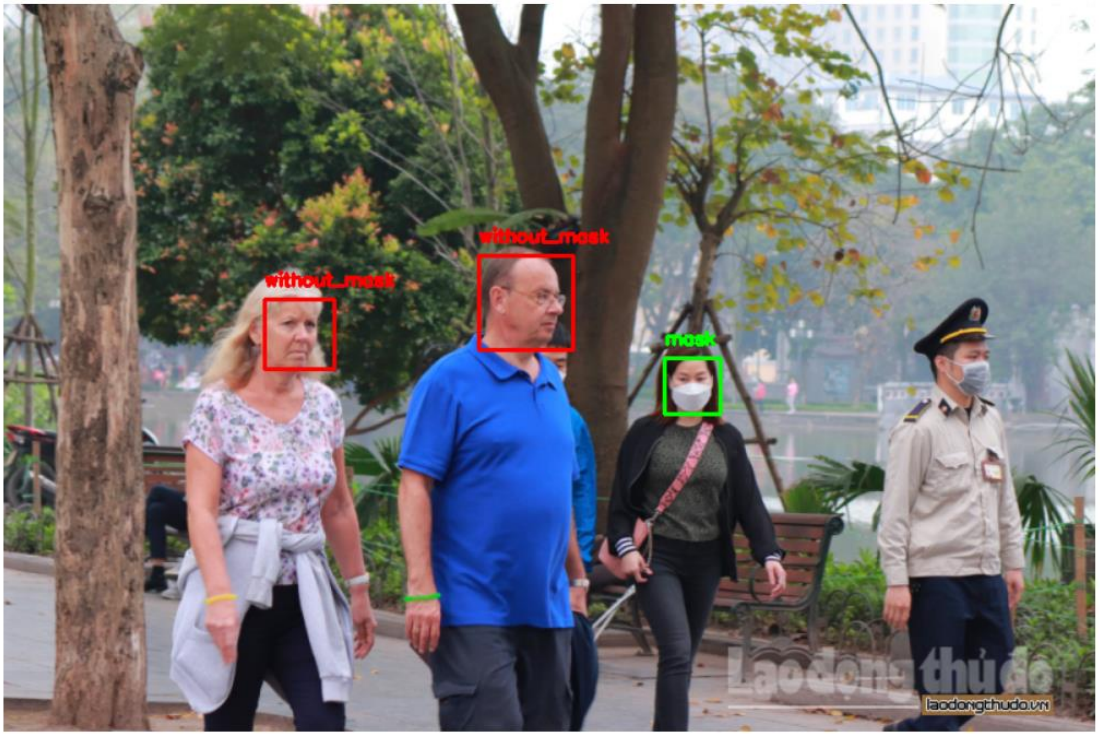
	precision	recall	f1-score	support
0	0.99	0.99	0.99	71
1	0.98	0.98	0.98	51
accuracy			0.98	122
macro avg	0.98	0.98	0.98	122
weighted avg	0.98	0.98	0.98	122

   - Model có độ chính xác rất tốt trên tập dataset : Logictic Regression với 98%
     ⇨ Chọn Logictic Regression để làm model chính dự đoán cho demo
   - Đối với các ảnh thực tế model nhận diện cũng khá tốt

```
# Load hình
img = cv2.imread("/content/example (5).jpg")

# Dự đoán
Predict_FaceMask(img)
```





- Tuy nhiên , để đánh giá được một bức ảnh nhiều người model phải phụ thuộc vào tool nhận diện khuôn mặt của openCV

----------Tool Không nhận diện được khuôn mặt!!!----------

```
# Load hình
face = cv2.imread("/content/example (1).jpg")

cv2_imshow(cv2.resize(face,(224,224)))

face = cv2.resize(face,(32,32)).flatten()
face = np.array(face).reshape(1,-1)

result = Model_LR.predict(face)

if (result == 0 ):
  print("------------------------------Wihtout_Mask------------------------------")
else:
  print("------------------------------Mask------------------------------")

```

------------------------------Mask------------------------------

6. Link tham khảo
   https://towardsdatascience.com/classifying-cat-pics-with-a-logistic-regression-model-e35dfb9159bb
   https://numpy.org/doc/stable/reference/generated/numpy.ndarray.flatten.html
   https://medium.com/@alkeshab/face-detection-using-opencv-in-google-colaboratory-a7529a2bb921