```
pip install MTCNN
```

```
Collecting MTCNN
  Downloading https://files.pythonhosted.org/packages/67/43/abee91792797c609c1bf
     |████████████████████████████████| 2.3MB 2.7MB/s
Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.6/
Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (fr
Installing collected packages: MTCNN
Successfully installed MTCNN-0.1.0
```

+ Code + Text

```
#aight, so you need MTCNN for face detection, makes life easy don't wanna make anothe
#and for some reason you can't place a comment above the pip install, OCD dudes, get
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import cv2
import matplotlib.patches as patches
import tensorflow as tf
from keras.layers import Flatten, Dense, Conv2D, MaxPooling2D, Dropout
from keras.models import Sequential
from mtcnn.mtcnn import MTCNN
```

```
#babayoda googur drivuuuu
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
#rootdirectory - change ber user
root_dir = "/content/drive/My Drive/facemask/face-mask-detection-dataset/"
```

```
#just linking, nothing fun
images       = os.path.join(root_dir + "Medical mask/Medical mask/Medical Mask/images
annotations  = os.path.join(root_dir + "Medical mask/Medical mask/Medical Mask/annota
```

```
#read em csvs
train        = pd.read_csv( os.path.join(root_dir + "train.csv" ))
submission   = pd.read_csv( os.path.join(root_dir + "submission.csv" ))
```

```
#meh formalities
print(len(train))
```

```
print(len(train))
train.head()
```

```
15412
```

	name	x1	x2	y1	y2	classname
0	2756.png	69	126	294	392	face_with_mask
1	2756.png	505	10	723	283	face_with_mask
2	2756.png	75	252	264	390	mask_colorful
3	2756.png	521	136	711	277	mask_colorful
4	6098.jpg	360	85	728	653	face_no_mask

```
print(len(submission))
submission.head()
```

```
8142
```

	name	x1	x2	y1	y2	classname
0	1800.jpg	NaN	NaN	NaN	NaN	NaN
1	1800.jpg	NaN	NaN	NaN	NaN	NaN
2	1800.jpg	NaN	NaN	NaN	NaN	NaN
3	1799.jpg	NaN	NaN	NaN	NaN	NaN
4	1799.jpg	NaN	NaN	NaN	NaN	NaN

```
len(os.listdir(images))
```

```
6031
```

```
#random cuz yeah, sort em files
a=os.listdir(images)
b=os.listdir(annotations)
a.sort()
b.sort()
```

```
print(len(b),len(a))
```

```
4328 6031
```

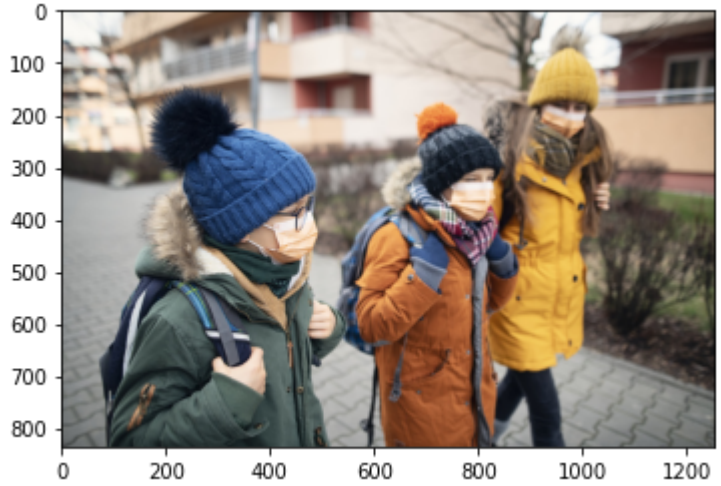
```
#train has everything that's from 1nice8 onwards
train_images=a[1698:]
test_images=a[:1698]
```

```
test_images[0]
```

```
'0001.jpg'
```

```
#meh
img=plt.imread(os.path.join(images,test_images[0]))
plt.imshow(img)
plt.show()
```

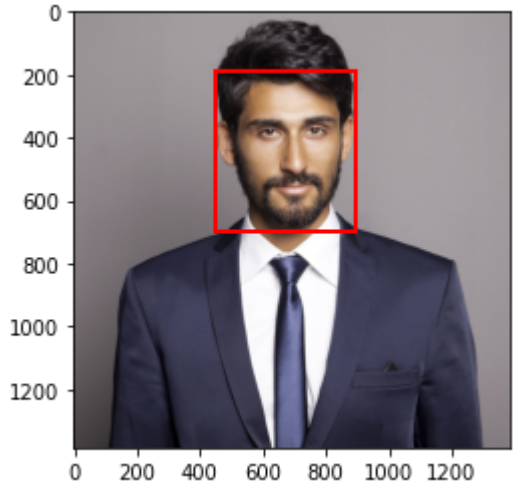

```
img=plt.imread(os.path.join(images,train_images[1]))
plt.imshow(img)
plt.show()
```




```
#setting the output class to yep, mask and yep, no mask, and then re-arrange based on
options=['face_with_mask','face_no_mask']
train= train[train['classname'].isin(options)]
train.sort_values('name',axis=0,inplace=True)


#got bored and left for a break i guess

#juking them trains into that bbox so we can see red rectangles around the faces
#apparently this makes viewability better or something, idk why i did this, not neede
```

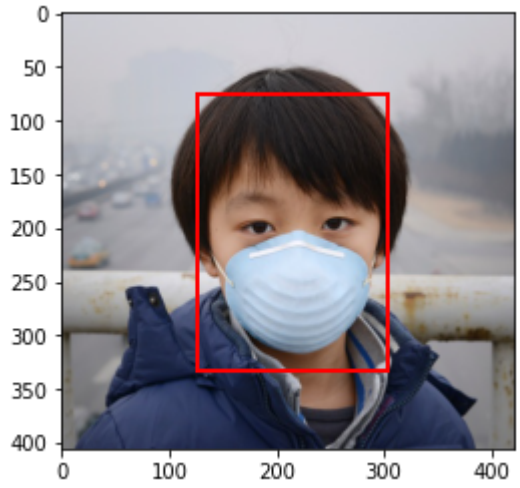
```
bbox=[]
for i in range(len(train)):
    arr=[]
    for j in train.iloc[i][["x1",'x2','y1','y2']]: #yup, the 4 corners of a BOX
        arr.append(j)
    bbox.append(arr)

train["bbox"]=bbox

def get_boxes(id):
    boxes=[]
    for i in train[train["name"]==str(id)]["bbox"]: #mapping the name and id for CONC
        boxes.append(i)
    return boxes

#Brint the boundaries
print(get_boxes(train_images[3]))

#load image into img
image=train_images[3]
img=plt.imread(os.path.join(images,image))

fig,ax = plt.subplots(1)
ax.imshow(img)
boxes=get_boxes(image)

#draw them lines from the set of boxes on the img
for box in boxes:
    rect = patches.Rectangle((box[0],box[1]),box[2]-box[0],box[3]-box[1],linewidth=2,
    ax.add_patch(rect)
plt.show()
```

```
 [[451, 186, 895, 697]]
```



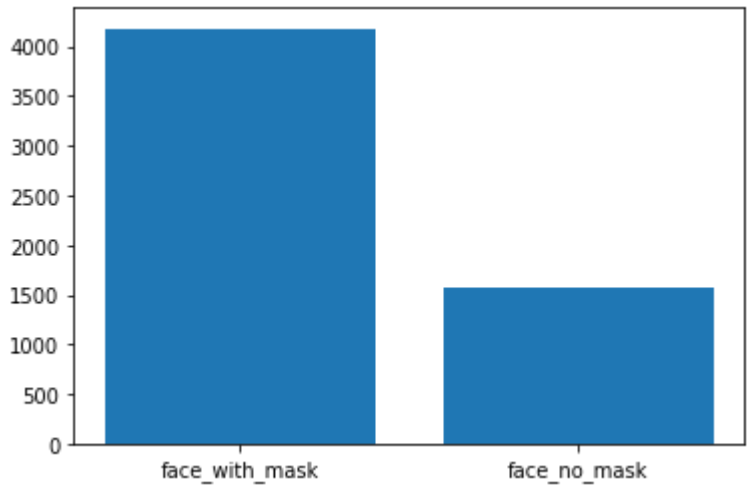

```
#yeah, yeah same thing nothing fun here

image=train_images[5]
```

```
img=plt.imread(os.path.join(images,image))

fig,ax = plt.subplots(1)
ax.imshow(img)
boxes=get_boxes(image)
for box in boxes:
    rect = patches.Rectangle((box[0],box[1]),box[2]-box[0],box[3]-box[1],linewidth=2,
    ax.add_patch(rect)
plt.show()
```




```
plt.bar(['face_with_mask','face_no_mask'],train.classname.value_counts())
```

```
<BarContainer object of 2 artists>
```




```
#resizing images takes like 10 min (690.83s nice10), smh

#writing some poetery when this is going on i guess
'''
yoooda yoooda doo
yooda doo
yoooda da doo da dooo yaa

yoooda doooda doo
```

```
yooda doo
'''

img_size=50
data=[]

#path setting for directory
path= root_dir + 'Medical mask/Medical mask/Medical Mask/images/'
def create_data():
       for i in range(len(train)):
            arr=[]
            for j in train.iloc[i]:
                  arr.append(j)
            #unrolling? the jpegs into respective (H*W*3) arrays cuz color
            img_array=cv2.imread(os.path.join(images,arr[0]),cv2.IMREAD_GRAYSCALE)
            #ok it's grayscale, that didn't age well (H*W*1) dims
            #yeet out the stuff that isn't the face
            crop_image = img_array[arr[2]:arr[4],arr[1]:arr[3]]
            new_img_array=cv2.resize(crop_image,(img_size,img_size))

            #slap the new images into this new ummmm thingy
            data.append([new_img_array,arr[5]])

            if i % 1000 == 0:
                print(str(i) + " loaded")
create_data()
```

```
0 loaded
1000 loaded
2000 loaded
3000 loaded
4000 loaded
5000 loaded
```

```
#yes, YES, YESSS!!!! only the face of BEARD MAN in Blain BLUE CUZ LOLS
data[0][0]
plt.imshow(data[0][0])
```

```
<matplotlib.image.AxesImage at 0x7fc7427c2f28>
```



```
#standard 'meh' stuff for linking em labels with images
x=[]
y=[]
for features, labels in data:
    x.append(features)
    y.append(labels)
from sklearn.preprocessing import LabelEncoder
lbl=LabelEncoder()
y=lbl.fit_transform(y)
```

```
#more standard boring stuff, reshaping the images and normalizing to make umm somethi
x=np.array(x).reshape(-1,50,50,1)
x=tf.keras.utils.normalize(x,axis=1)
from keras.utils import to_categorical
y = to_categorical(y)


#printing the shapes, cuz was bored
print("Shape of X: " + str(x[0].shape) )
print("Shape of Y: " + str(y[0].shape) )
```

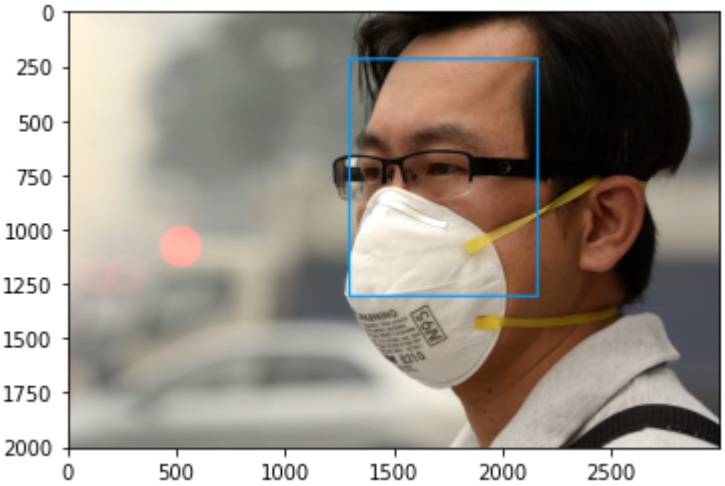
```
Shape of X: (50, 50, 1)
Shape of Y: (2,)
```

```
#wasn't frisky so i made a simple CNN, also loading google servers slow cuz
#also somewhere here i fricked my conda as was bored

from keras.layers import LSTM
model=Sequential()
model.add(Conv2D(100,(3,3),input_shape=x.shape[1:],activation='relu',strides=2))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(2, activation='softmax'))



#really getting bored, conda is a bit too messed up
#training takes like 7 minutes (414.71s)

#basic training junk
opt = tf.keras.optimizers.Adam(lr=1e-3)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x,y,epochs=30,batch_size=5)
```

```
Epoch 1/30
1150/1150 [==============================] - 15s 13ms/step - loss: 0.5543 - accu
Epoch 2/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.4767 - accu
Epoch 3/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.4237 - accu
Epoch 4/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.3822 - accu
Epoch 5/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.3490 - accu
Epoch 6/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.3256 - accu
Epoch 7/30
1150/1150 [==============================] - 14s 13ms/step - loss: 0.2929 - accu
Epoch 8/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.2684 - accu
Epoch 9/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.2434 - accu
Epoch 10/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.2269 - accu
Epoch 11/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.2053 - accu
Epoch 12/30
1150/1150 [==============================] - 15s 13ms/step - loss: 0.1894 - accu
Epoch 13/30
1150/1150 [==============================] - 15s 13ms/step - loss: 0.1706 - accu
Epoch 14/30
1150/1150 [==============================] - 16s 14ms/step - loss: 0.1460 - accu
Epoch 15/30
1150/1150 [==============================] - 15s 13ms/step - loss: 0.1300 - accu
Epoch 16/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.1243 - accu
Epoch 17/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.1075 - accu
Epoch 18/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.1032 - accu
Epoch 19/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.1018 - accu
Epoch 20/30
1150/1150 [==============================] - 14s 13ms/step - loss: 0.0896 - accu
Epoch 21/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0834 - accu
Epoch 22/30
1150/1150 [==============================] - 15s 13ms/step - loss: 0.0722 - accu
Epoch 23/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0633 - accu
Epoch 24/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0674 - accu
Epoch 25/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0600 - accu
Epoch 26/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0636 - accu
Epoch 27/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0593 - accu
Epoch 28/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0459 - accu
Epoch 29/30
```

```
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0557 - accu
Epoch 30/30
1150/1150 [==============================] - 14s 12ms/step - loss: 0.0429 - accu
<tensorflow.python.keras.callbacks.History at 0x7fc73b0787f0>
```

```
#yup, boring stuff here. face detector using the MTCNN thingy that we installed in th

detector=MTCNN()
img=plt.imread(os.path.join(images,test_images[0]))
face=detector.detect_faces(img)

'''
draw the boxes, draw the boxes
on all the images, on all the images
at this time i got bored, and i went to walk. i think so
'''

for face in face:
        bounding_box=face['box']
        x=cv2.rectangle(img,
              (bounding_box[0], bounding_box[1]),
              (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
              (0,155,255),
              10)
        plt.imshow(x)


img=plt.imread(os.path.join(images,test_images[2]))
face=detector.detect_faces(img)
for face in face:
        bounding_box=face['box']
        x=cv2.rectangle(img,
              (bounding_box[0], bounding_box[1]),
              (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
              (0,155,255),
              10)
        plt.imshow(x)
```

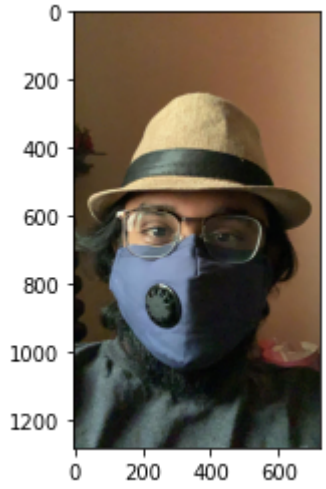


```
#shit's been running for 40 min now, bruh wtf
#so apparently you can see 2 ep of MG and it's perfectly done, hmmmmm

detector=MTCNN()
test_df=[]
#now we saucing out the test stuff, oooh fun :kappa:
#find those faces and box em all up
for image in test_images:
    img=plt.imread(os.path.join(images,image))
    faces=detector.detect_faces(img)
    test=[]
    for face in faces:
        bounding_box=face['box']
        test.append([image,bounding_box])
    test_df.append(test)

test=[]

#append in yup, mask or yup, no mask
for i in test_df:
    if len(i)>0:
        if len(i)==1:
            test.append(i[0])
        else:
            for j in i:
                test.append(j)

sub=[]
rest_image=[]

#meh, we could have messed up on attempt one, you guys
#so we roll them again the way snoop does, kekw

for i in test:
    sub.append(i[0])
for image in test_images:
    if image not in sub:
        rest_image.append(image)

detector=MTCNN()
test_df_=[]

#box em up
for image in rest_image:
    img=cv2.imread(os.path.join(images,image))
    faces=detector.detect_faces(img)
    test_=[]
    for face in faces:
        bounding_box=face['box']
        test_.append([image,bounding_box])
    test_df_.append(test_)
```

```
for i in test_df_:
    if len(i)>0:
        if len(i)==1:
            test.append(i[0])
        else:
            for j in i:
                test.append(j)


#all wrong calls are listed (here it's the ones with the negative value on face locat
negative=[]
for i in test:
    for j in i[1]:
        if j<0:
            negative.append(i)



#hard coding those cuz we already have them from the source, RIGHT, hmmmm!!!!

test_data=[]
def create_test_data():
            for j in test:
                if j not in negative:
                    img=cv2.imread(os.path.join(images,j[0]),cv2.IMREAD_GRAYSCALE)
                    img=img[j[1][1]:j[1][1]+j[1][3],j[1][0]:j[1][0]+j[1][2]]
                    new_img=cv2.resize(img,(50,50))
                    new_img=new_img.reshape(-1,50,50,1)
                    predict=model.predict(new_img)
                    print(predict)
                    test_data.append([j,predict])

create_test_data()


#filling in a csv the image id, the output, and the face boxing
image=[]
classname=[]
for i,j in test_data:
    classname.append(np.argmax(j))
    image.append(i)
df=pd.DataFrame(columns=['image','classname'])
df['image']=image
df['classname']=classname
df['classname']=lbl.inverse_transform(df['classname'])
image=[]
x1=[]
x2=[]
y1=[]
y2=[]
for i in df['image']:
```

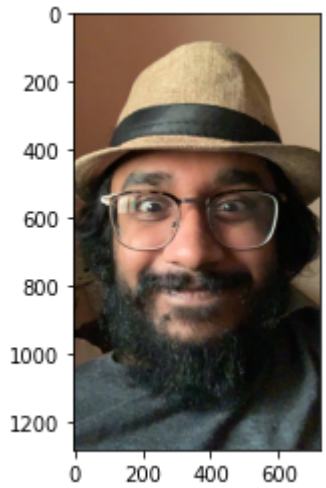
```
        image.append(i[0])
        x1.append(i[1][0])
        x2.append(i[1][1])
        y1.append(i[1][2])
        y2.append(i[1][3])
    df['name']=image
    df['x1']=x1
    df['x2']=x2
    df['y1']=y1
    df['y2']=y2
    df.drop(['image'],axis=1,inplace=True)


    df.sort_values('name',axis=0,inplace=True,ascending=False)
    df.to_csv(root_dir + 'res.csv')


    #beyond this is the graveyard and area51 test zone.
    #welcome (-:









    test_real_images       = os.path.join(root_dir + "Medical mask/Medical mask/Medical M


    os.listdir(test_real_images)

         ['person_no_mask.jpeg',
          'person_with_mask.jpeg',
          'shank_no.jpg',
          'shank_mask.jpg']


    img=plt.imread(os.path.join(test_real_images,'person_with_mask.jpeg'))
    plt.imshow(img)
    plt.show()

    detector=MTCNN()
    img=plt.imread(os.path.join(test_real_images,'person_with_mask.jpeg'))
    face=detector.detect_faces(img)
    for face in face:
            bounding_box=face['box']
            x=cv2.rectangle(img,
                  (bounding_box[0], bounding_box[1]),
                  (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
                  (0,155,255),
                  10)
```

```
        plt.imshow(x)
```

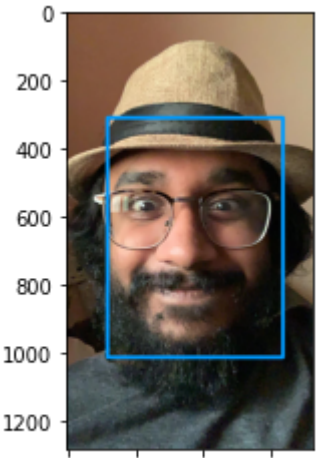



```
    WARNING:tensorflow:5 out of the last 14 calls to <function Model.make_predict_fu
```

```
img=plt.imread(os.path.join(test_real_images,'person_no_mask.jpeg'))
plt.imshow(img)
plt.show()
```

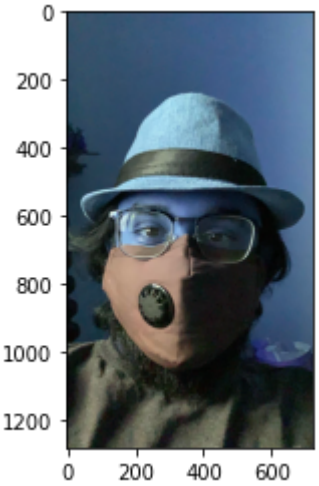



```
detector=MTCNN()
img=plt.imread(os.path.join(test_real_images,'person_no_mask.jpeg'))
face=detector.detect_faces(img)
for face in face:
        bounding_box=face['box']
        x=cv2.rectangle(img,
              (bounding_box[0], bounding_box[1]),
              (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
              (0,155,255),
              10)
        plt.imshow(x)
```




```
img_num = "person_with_mask"

img=cv2.imread(os.path.join(test_real_images,img_num+'.jpeg'))
plt.imshow(img)
new_img=cv2.resize(img,(50,50))
new_img=new_img.reshape(-1,50,50,1)
predict=model.predict(new_img)
print(predict)

img=cv2.imread(os.path.join(test_real_images,img_num+'.jpeg'),cv2.IMREAD_GRAYSCALE)
new_img=cv2.resize(img,(50,50))
new_img=new_img.reshape(-1,50,50,1)
predict=model.predict(new_img)
print(predict)

if predict[0][0] == 1:
    print("no ")
print("mask")
```

```
[[0. 1.]
 [1. 0.]
 [0. 1.]]
[[1. 0.]]
no
mask
```



```
#saving the model for export to some other country, yep unlike us these babes can mov
model.save('saves')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
INFO:tensorflow:Assets written to: saves/assets
```

```
#for some reason i was tryng to export and import weights, why do that for oil when y
model.load_weights(root_dir + "")
```

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fc74743bfd0
```

```
#10*nice KB file that contains all the stuff to run the prog

import numpy as np
import tensorflow as tf
import os
import cv2
from tensorflow import keras

root_dir = "/content/drive/My Drive/facemask/face-mask-detection-dataset/"
model = keras.models.load_model("saves")

test_real_images       = root_dir + "Medical mask/Medical mask/Medical Mask/test-imag
img_num = "person_with_mask"

img=cv2.imread(os.path.join(test_real_images,img_num+'.jpeg'))
#plt.imshow(img)
new_img=cv2.resize(img,(50,50))
new_img=new_img.reshape(-1,50,50,1)
predict=model.predict(new_img)
print(predict)

img=cv2.imread(os.path.join(test_real_images,img_num+'.jpeg'),cv2.IMREAD_GRAYSCALE)
new_img=cv2.resize(img,(50,50))
new_img=new_img.reshape(-1,50,50,1)
predict=model.predict(new_img)
print(predict)

if predict[0][0] == 1:
    print("no ")
print("mask")
```

```
[[0. 1.]
 [0. 1.]
 [0. 1.]]
```

```
path = "steve"
a = os.path.join("hello")
a
```

```
'hello'
```

```
#starting to think this should be in the start of the program, meh who cares
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_220 (Conv2D)          (None, 24, 24, 100)       1000      
_________________________________________________________________
max_pooling2d_112 (MaxPoolin (None, 12, 12, 100)       0         
_________________________________________________________________
conv2d_221 (Conv2D)          (None, 10, 10, 64)        57664     
_________________________________________________________________
max_pooling2d_113 (MaxPoolin (None, 5, 5, 64)          0         
_________________________________________________________________
flatten_38 (Flatten)         (None, 1600)              0         
_________________________________________________________________
dense_130 (Dense)            (None, 50)                80050     
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0         
_________________________________________________________________
dense_131 (Dense)            (None, 2)                 102       
=================================================================
Total params: 138,816
Trainable params: 138,816
Non-trainable params: 0
_________________________________________________________________
```