None

Deep Learning model which can identify if the person is wearing a mask or not, also detecting if people vilating social distancing norms.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_cs
v)
import cv2
from scipy.spatial import distance
# Input data files are available in the read-only "../input/" directo
ry
# For example, running this (by clicking run or pressing Shift+Enter)
 will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
%%html
<style>
.output_png img {
    display: block;
    margin-left: auto;
    margin-right: auto;
}
.rendered_html code {
  background-color: #d5eaff;
  margin: 0px 2px;
  font-family: monaco;
  border: solid 0.5px white;
  box-sizing: border-box;
}
/* change markdown code snipet style */
.rendered_html pre code {
  background-color: #f7f7f7;
  font-family: monaco;
}
/* change markdown code snipet wrapper style */
.rendered_html pre {
  margin: 1em 2em;
  padding: 0px;
  background-color: #f7f7f7;
  border-left: solid 5px #d5eaff;
  padding-left: 5px;
}
/* pull down the position of result output area */
.output {
  margin-top: 5px;
}
/* change output area style */
div.output_text {
  text-align: center;
```

```
  color: #000;
  line-height: 1.21429em;
#  border: solid .5px #2af8ff4d;
#  border-left: solid 3px #2af8ff4d;
}
/* decreace the minimam space size of left side of html area and outp
ut area */
.prompt {
  min-width: 13ex;
}
/* change font of code input area */
.CodeMirror-code {
    outline: none;
    font-family: monaco;
}
</style>
```

Using haar cascade to detect faces

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. We'll be using a Haar Cascade Model trained to detect faces in order to obtain the bounding box coordinates of faces in an image.

```
#loading haarcascade_frontalface_default.xml
face_model = cv2.CascadeClassifier('haarcascade_frontalface_default.x
ml')
```

```
import matplotlib.pyplot as plt
#trying it out on a sample image
img = cv2.imread('images/maksssksksss749.png')

img = cv2.cvtColor(img, cv2.IMREAD_GRAYSCALE)

faces = face_model.detectMultiScale(img,scaleFactor=1.05, minNeighbor
s=4) #returns a list of (x,y,w,h) tuples

out_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #colored output image

#plotting
for (x,y,w,h) in faces:
    cv2.rectangle(out_img,(x,y),(x+w,y+h),(0,0,255),1)
plt.figure(figsize=(12,12))
plt.imshow(out_img)
```

```
<matplotlib.image.AxesImage at 0x201fbcc5520>
```


0
50
100
150
200
250
300
350

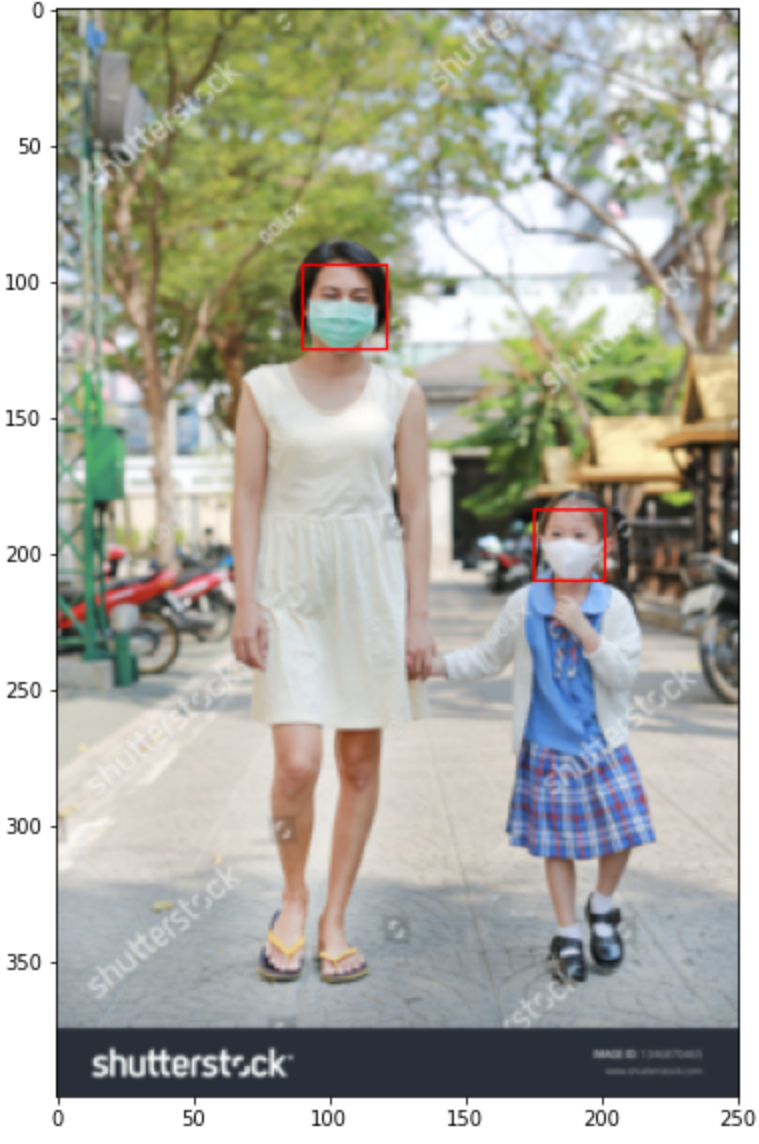
Detecting social distancing violations

This can be done by iterating over the coordinates of faces and calculating the distance for each possible pair, if the distance for a particular pair is less than MIN_DISTANCE then the bounding boxes for those faces are colored red. MIN_DISTANCE must be manually initialized in such a way that it corresponds to the minimum allowable distance in real life (ex. 6ft in India).

```
MIN_DISTANCE = 130
```

```
if len(faces)>=2:
    label = [0 for i in range(len(faces))]
    for i in range(len(faces)-1):
        for j in range(i+1, len(faces)):
            dist = distance.euclidean(faces[i][:2],faces[j][:2])
            if dist<MIN_DISTANCE:
                label[i] = 1
                label[j] = 1
    new_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #colored output im
age
    for i in range(len(faces)):
        (x,y,w,h) = faces[i]
        if label[i]==1:
            cv2.rectangle(new_img,(x,y),(x+w,y+h),(255,0,0),1)
        else:
            cv2.rectangle(new_img,(x,y),(x+w,y+h),(0,255,0),1)
    plt.figure(figsize=(10,10))
    plt.imshow(new_img)

else:
    print("Caras detectadas < 2")
```


0
50
100
150
200
250
300
350
0
50
100
150
200
250
shutterstock

Red box shows violation of social distancing.

Using VGG19 for mask detection

```
from keras.applications.vgg19 import VGG19
from keras.applications.vgg19 import preprocess_input
from keras import Sequential
from keras.layers import Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
```

```
#Load train and test set
train_dir = 'Face Mask Dataset/Train'
test_dir = 'Face Mask Dataset/Test'
val_dir = 'Face Mask Dataset/Validation'
```

```
# Data augmentation

train_datagen = ImageDataGenerator(rescale=1.0/255, horizontal_flip=T
rue, zoom_range=0.2,shear_range=0.2)
train_generator = train_datagen.flow_from_directory(directory=train_d
ir,target_size=(128,128),class_mode='categorical',batch_size=32)

val_datagen = ImageDataGenerator(rescale=1.0/255)
val_generator = train_datagen.flow_from_directory(directory=val_dir,t
arget_size=(128,128),class_mode='categorical',batch_size=32)

test_datagen = ImageDataGenerator(rescale=1.0/255)
test_generator = train_datagen.flow_from_directory(directory=val_dir,
target_size=(128,128),class_mode='categorical',batch_size=32)
```

```
Found 10000 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
Found 800 images belonging to 2 classes.
```

Building VGG19 transfer learning model.

```
vgg19 = VGG19(weights='imagenet',include_top=False,input_shape=(128,1
28,3))

for layer in vgg19.layers:
    layer.trainable = False

model = Sequential()
model.add(vgg19)
model.add(Flatten())
model.add(Dense(2,activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Pa
ram #
=================================================================
vgg19 (Functional)           (None, 4, 4, 512)         20
024384
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0


_________________________________________________________________
dense (Dense)                (None, 2)                 16
386
=================================================================
Total params: 20,040,770
Trainable params: 16,386
Non-trainable params: 20,024,384
_________________________________________________________________
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metric
s ="accuracy")
```

```
history = model.fit_generator(generator=train_generator,
                              steps_per_epoch=len(train_generator)//3
2,
                              epochs=20,validation_data=val_generator
,
                              validation_steps=len(val_generator)//32
)
```

```
                    Epoch 1/20
9/9 [==============================] - 118s 13s/step - lo
            ss: 0.4012 - accuracy: 0.8611
                    Epoch 2/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.2136 - accuracy: 0.9201
                    Epoch 3/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.1513 - accuracy: 0.9514
                    Epoch 4/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.1440 - accuracy: 0.9410
                    Epoch 5/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.1263 - accuracy: 0.9514
                    Epoch 6/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.0935 - accuracy: 0.9757
                    Epoch 7/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.0873 - accuracy: 0.9722
                    Epoch 8/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.0917 - accuracy: 0.9722
                    Epoch 9/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.0942 - accuracy: 0.9653
                   Epoch 10/20
9/9 [==============================] - 114s 13s/step - lo
            ss: 0.0594 - accuracy: 0.9896
                   Epoch 11/20
9/9 [==============================] - 126s 14s/step - lo
            ss: 0.0592 - accuracy: 0.9861
                   Epoch 12/20
```

```
9/9 [==============================] - 124s 14s/step - lo
             ss: 0.0627 - accuracy: 0.9826
                      Epoch 13/20
9/9 [==============================] - 124s 14s/step - lo
             ss: 0.1160 - accuracy: 0.9479
                      Epoch 14/20
9/9 [==============================] - 115s 13s/step - lo
             ss: 0.0960 - accuracy: 0.9722
                      Epoch 15/20
9/9 [==============================] - 127s 14s/step - lo
             ss: 0.1177 - accuracy: 0.9618
                      Epoch 16/20
9/9 [==============================] - 137s 15s/step - lo
             ss: 0.0897 - accuracy: 0.9688
                      Epoch 17/20
9/9 [==============================] - 136s 15s/step - lo
             ss: 0.0953 - accuracy: 0.9653
                      Epoch 18/20
9/9 [==============================] - 128s 14s/step - lo
             ss: 0.0864 - accuracy: 0.9618
                      Epoch 19/20
9/9 [==============================] - 132s 15s/step - lo
             ss: 0.0440 - accuracy: 0.9931
                      Epoch 20/20
9/9 [==============================] - 127s 14s/step - lo
             ss: 0.0677 - accuracy: 0.9792
```

```
model.evaluate(test_generator)
```

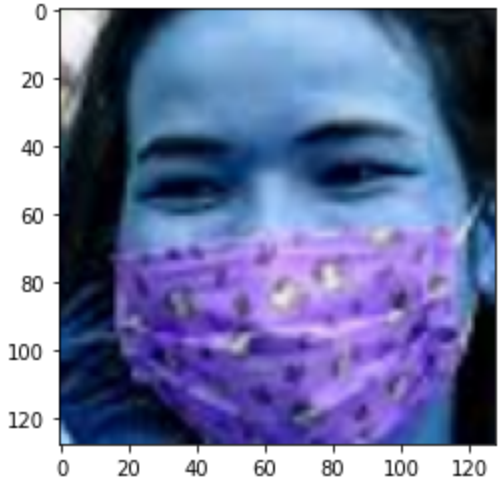
```
25/25 [==============================] - 343s 14s/step -
            loss: 0.0492 - accuracy: 0.9862

        [0.04917881637811661, 0.9862499833106995]
```

Our model achieved 98.6% accuracy on test data.

Testing the model on the test data

```
sample_mask_img = cv2.imread('Face Mask Dataset/Test/WithMask/1203.png')
sample_mask_img = cv2.resize(sample_mask_img,(128,128))
plt.imshow(sample_mask_img)
sample_mask_img = np.reshape(sample_mask_img,[1,128,128,3])
sample_mask_img = sample_mask_img/255.0
```




```
model.predict(sample_mask_img)
```

```
array([[0.9601026 , 0.19185427]], dtype=float32)
```

The model is able to classify if the person is wearing a mask or not.

Save the model.

```
#model.save('masknet.h5')
from keras.models import load_model
model = load_model('masknet.h5')
```

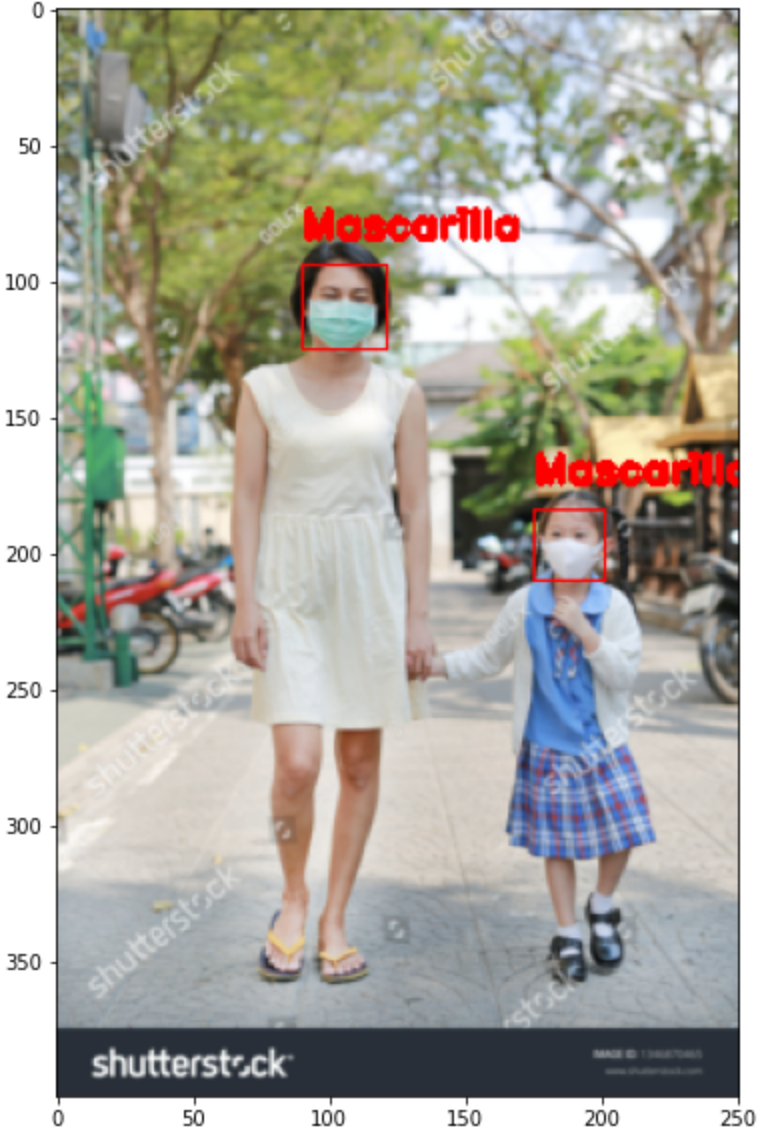
Integrating with haar cascade

We now take crops of the faces detected in the image and use the model trained in the above section to determine whether the individual faces have a mask or not.

```
mask_label = {0:'Mascarilla',1:'Multa'}
dist_label = {0:(0,255,0),1:(255,0,0)}
```

```
if len(faces)>=2:
    label = [0 for i in range(len(faces))]
    for i in range(len(faces)-1):
        for j in range(i+1, len(faces)):
            dist = distance.euclidean(faces[i][:2],faces[j][:2])
            if dist<MIN_DISTANCE:
                label[i] = 1
                label[j] = 1
    new_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #colored output im
age
    for i in range(len(faces)):
        (x,y,w,h) = faces[i]
        crop = new_img[y:y+h,x:x+w]
        crop = cv2.resize(crop,(128,128))
        crop = np.reshape(crop,[1,128,128,3])/255.0
        mask_result = model.predict(crop)
        cv2.putText(new_img,mask_label[mask_result.argmax()],(x, y-10
),cv2.FONT_HERSHEY_DUPLEX,0.5,dist_label[label[i]],2)
        cv2.rectangle(new_img,(x,y),(x+w,y+h),dist_label[label[i]],1)
    plt.figure(figsize=(10,10))
    plt.imshow(new_img)

else:
    print("No. of faces detected is less than 2")
```


Mascarilla
Mascarilla
shutterstock
0
50
100
150
200
250
300
350
0
50
100
150
200
250

Red boxes shows violation of social distancing.

```
#quita input y output

#!jupyter nbconvert mascarilla.ipynb --no-input --to html --TemplateE
xporter.exclude_code_cell=True

#quita input

!jupyter nbconvert mascarilla.ipynb --to html_with_lenvs --template f
ull --no-prompt
#--no-input

#quita input

!jupyter nbconvert mascarilla.ipynb --to pdf --no-prompt
#--no-input
```

```
[NbConvertApp] Converting notebook mascarilla.ipynb to ht
                    ml_with_lenvs
 [NbConvertApp] Writing 1864746 bytes to mascarilla.html
```

```
#WEBPDF
#!jupyter nbconvert mascarilla.ipynb --no-input --to webpdf --templat
e classic --allow-chromium-download
```

```

```

```

```