VISUAL RECOGNITION
PROJECT REPORT

FACE-MASK DETECTION

Kanumuru Jhanwwee	**Anjan Vikas**	**Neetha Reddy**	**Salman Patel**
IMT2018035	**IMT2018049**	**IMT2018050**	**IMT2018066**

Problem Statement:

"Build a vision based automated door entry system, to open the door only if a human wears a mask"

Approach:

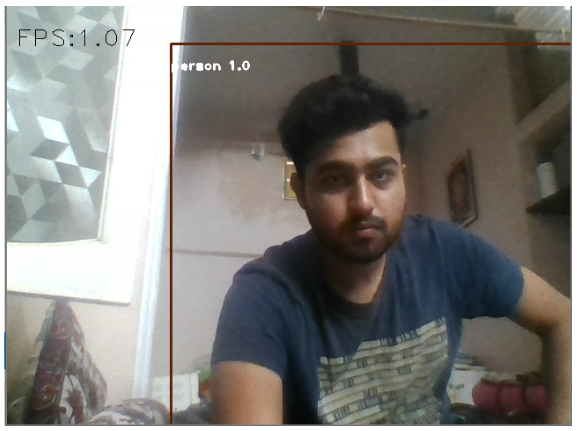
Given problem can be solved by dividing entire task into 3 sub-tasks as following:
(1) Human detection using YOLO in the video feed and then we extract that a portion where person/human is found
(2) Face Detection using Viola Jones, if a person gets detected in the step1.
(3)Mask Detection using CNN and MobileNetv2 on the face we extracted in step 2

Human Detection Using Yolo:

In images, You Only Look Once (YOLO) is an advanced approach to object detection. YOLO applies a single CNN to the entire image which further divides the image into grids. Prediction of bounding boxes and respective confidence score are calculated for each grid.
YOLO is a state-of-the-art, real-time object detection system. Yolo V3 can process up to 45 frames per second on GPU. Which is really very fast then the current exsign system. Yolo even has its variant, tiny Yolo, which can process 220 frames per second on GPU by compromise on accuracy.
We load the YoloV3 weights and the configuration file using the dnn module from OpenCV. We then pass the names of layers for which the output is to be

computed into the cv2.dnn module. We perform preprocessing on the data using blobFromImage and blobFromImags functions. The forward() function is then used to return information about all of the detected objects for all classes of objects set earlier and the class with the highest score is considered to be the predicted class. After making sure that there are no duplicate detections, each object has a bounding box drawn about it.

Human Detection



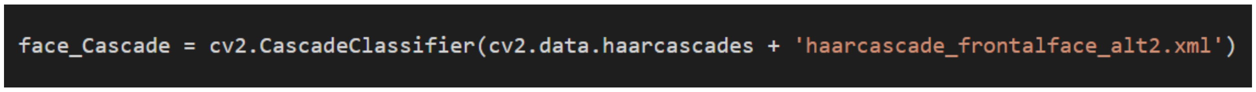
Face detection using VIOLA JONES:

The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.Viola Jones is Haar Classifier. Haar Cascade Classifiers are basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images

Haar-like Features: A Haar-like feature consists of dark regions and light regions. It produces a single value by taking the sum of the intensities of the light regions and subtracting that by the sum of the intensities of dark regions.

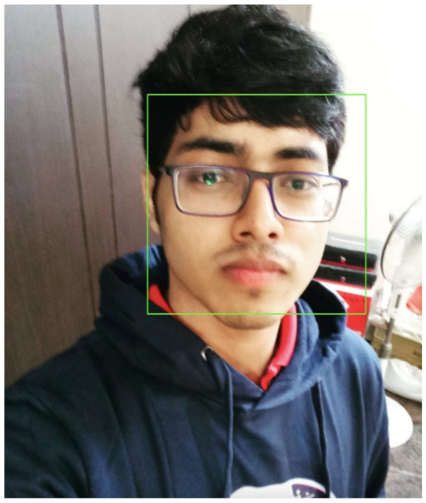
Integral Images: An Integral Image is an intermediate representation of an image where the value for location (*x*, y) on the integral image equals the sum of the pixels above and to the left.

The Cascade Classifier: A Cascade Classifier is a multi-stage classifier that can perform detection quickly and accurately. Each stage consists of a strong classifier produced by the AdaBoost Algorithm.



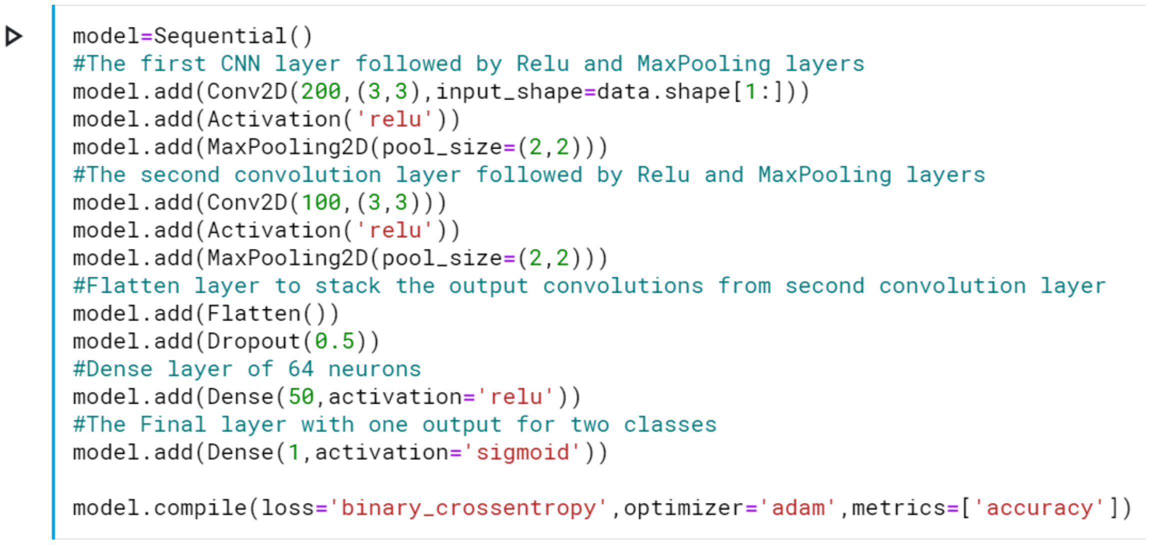
```
face_Cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_alt2.xml')
```

Face detection



Mask detection:

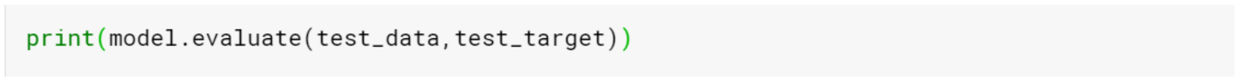
- **CNN**
  In this method, we are using convolutional neural networks to create a face mask detector including python, keras, tensorflow, opencv. The convolutional neural network contains 2 convolution layers followed by Relu and max-pooling layers, flatten layer to stack the output convolutions from the second convolution layer, dense layer of 64 neurons and the Final layer with one output for two classes. The model is compiled with binary cross entropy as loss and 'adam' as optimizer.



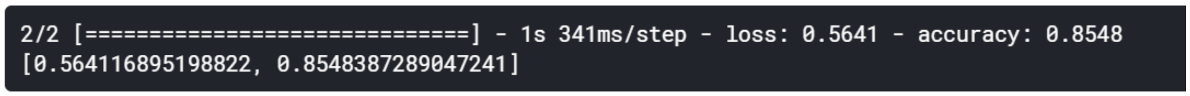
```
model=Sequential()
#The first CNN layer followed by Relu and MaxPooling layers
model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The second convolution layer followed by Relu and MaxPooling layers
model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#Flatten layer to stack the output convolutions from second convolution layer
model.add(Flatten())
model.add(Dropout(0.5))
#Dense layer of 64 neurons
model.add(Dense(50,activation='relu'))
#The Final layer with one output for two classes
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

This model is used to train the dataset of images with mask and without from kaggle along with validation set. We tried training by tweaking the hyperparameter and the best accuracy on the test set is 85%.



```
print(model.evaluate(test_data,test_target))
```



```
2/2 [==============================] - 1s 341ms/step - loss: 0.5641 - accuracy: 0.8548
[0.564116895198822, 0.8548387289047241]
```

- **MobileNetv2**
  Mobilenet is a lightweight architecture which makes use of transfer learning to support classification, detection, etc keeping embedded devices in mind. It is also very low maintenance and therefore performs reasonably well with high speed. MobileNetv2 is an extension of Mobilenet architecture which includes linear bottlenecks between the layers and shortcut connections between the bottlenecks, making it use two times fewer operations, 30 percent fewer parameters and 30-40 percent faster all the while achieving higher accuracy. We loaded the ModelNetV2 network using the weight of a pretrained ImageNet model to achieve better accuracy. We then proceed to remove the last FC layer and then add pooling, flattening, dense(with “relu”

as activation) and dropout.

```
# Loading MobileNetV2 network using weights of pre-trained imagenet to achieve b
etter accuracy
# We are removing the FC layers
baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor = Input(shape=(224, 224, 3)))

# Head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
#Pooling layer
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False
print("[INFO] compiling model...")
```

This model gives us a high accuracy.

```
Epoch 20/20
15/15 [==============================] - 17s 1s/step - loss: 0.0258 - accuracy:
0.9974 - val_loss: 0.0118 - val_accuracy: 1.0000
```

Pipeline for Mask Detection on Webcam or Video:

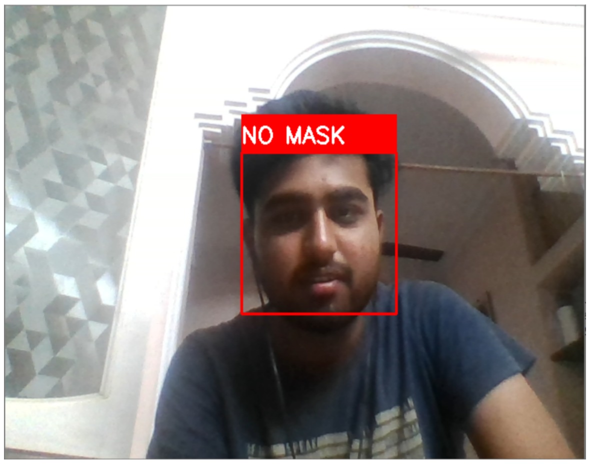
If someone is present in front of web camera then we have taken one by one all the frame of the video, now that frame goes under human detection YOLO(You Only Look Once) model and return extracted image of all the person/human detected in the video with its coordinate of bounding box of human where (x,y) represents starting position of bounding box and (w,h) width and height of the bounding box.

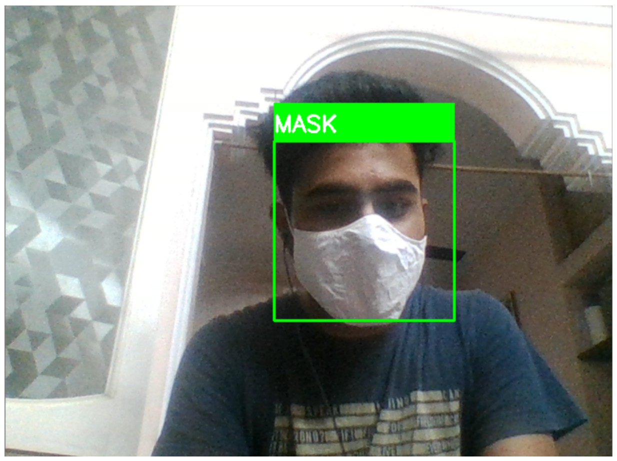
Now we have multiple or single person images extracted from a frame, now on each of these extracted images we will find the face of the person and extract only the face part.

After extracting face we will feed this face image into our model that we have trained earlier for mask detection. Now this model will give a decision whether a person has worn the mask or not. Based on this decision we will create a bounding box around that person and write “mask” text if he has worn the mask otherwise write “no mask”.

If a person has a mask then we will open the doors otherwise we will not open the doors.

Results:





References:

1. https://www.mygreatlearning.com/blog/viola-jones-algorithm/
2. https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/
3. https://towardsdatascience.com/object-detection-using-yolov3-and-opencv-19ee0792a420

4. https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299
5. https://www.youtube.com/watch?v=Ax6P93r32KU