Capstone Project

Machine Learning Engineer Nanodegree

Syed Shakeeb Assil

May 16,2019

Definition

This project is about taking attendance using your face as a identity with the help of Convolutional Neural Network instead of the biometric system.

In this project, I created a Attendance Face-Recognition application in Python using Keras and OpenCV where the user have to place the face in front of the camera using VGG Face as a classifier for face recognition and attendance will be added accordingly.

Problem Statement

The goal of the project is to identify the individual identity and based on the identity the attendance will be given.

The task involved are:

1. Generate your own Dataset and create a balance Dataset containing equal number of data.
2. Preprocesses the dataset.
3. Train the Convolutional Neural Network model with the particular dataset.
4. Test the model and check the accuracy.
5. Predict the identity by the new image with the trained model.
6. Add the attendance with the name of the user.

The model is expected to identify the face of the user and add the attendance with the name of the user.

Metrics

Accuracy

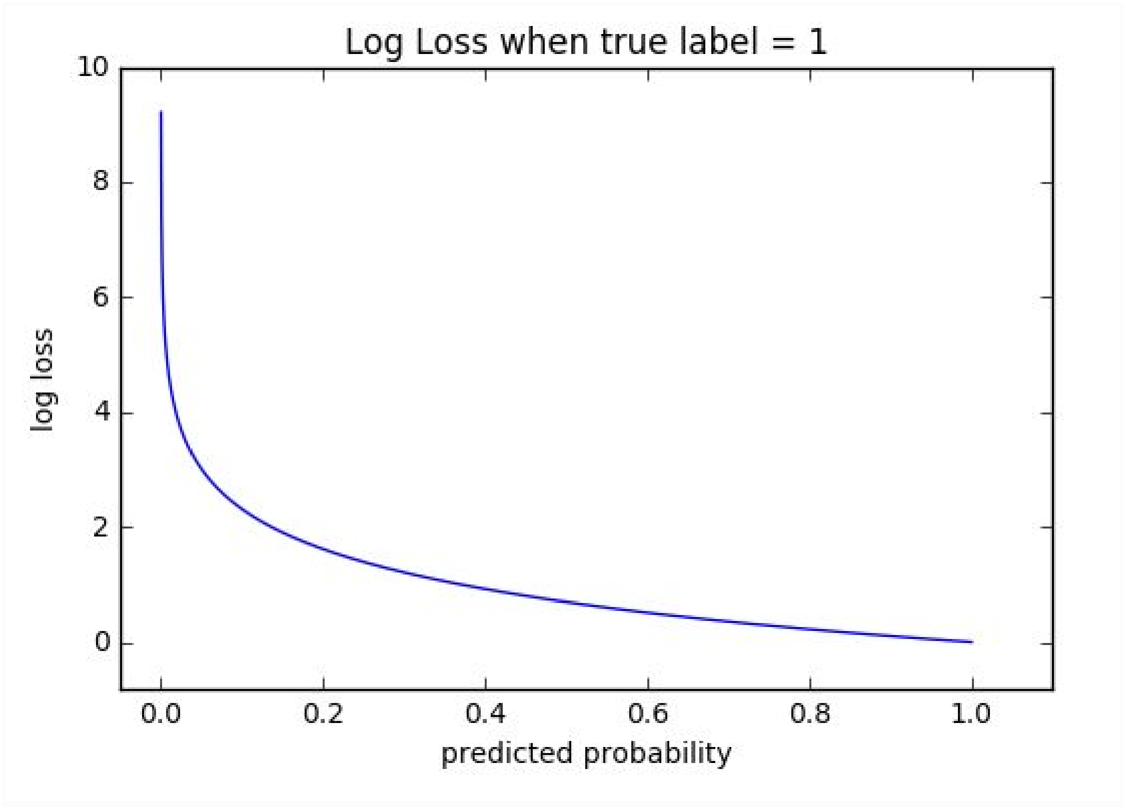
Accuracy is the metric mainly used in classification problems where it can calculate the how well the model have predicted the output the correct label with validation data or the testing data.

$$accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

In this project, we will calculating the accuracy of the training model when training and also the accuracy of how well the model have evaluated during testing.

Cross Entropy Loss

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.



Lower the value of entropy, the better the model have learned or vice versa.
Classification Delay

Classification Delay is the delay of the model to identity the face of the individuality as this metrics is very important to determine the time taken by the model to identity the face of the individuality in real-time image.

Analysis

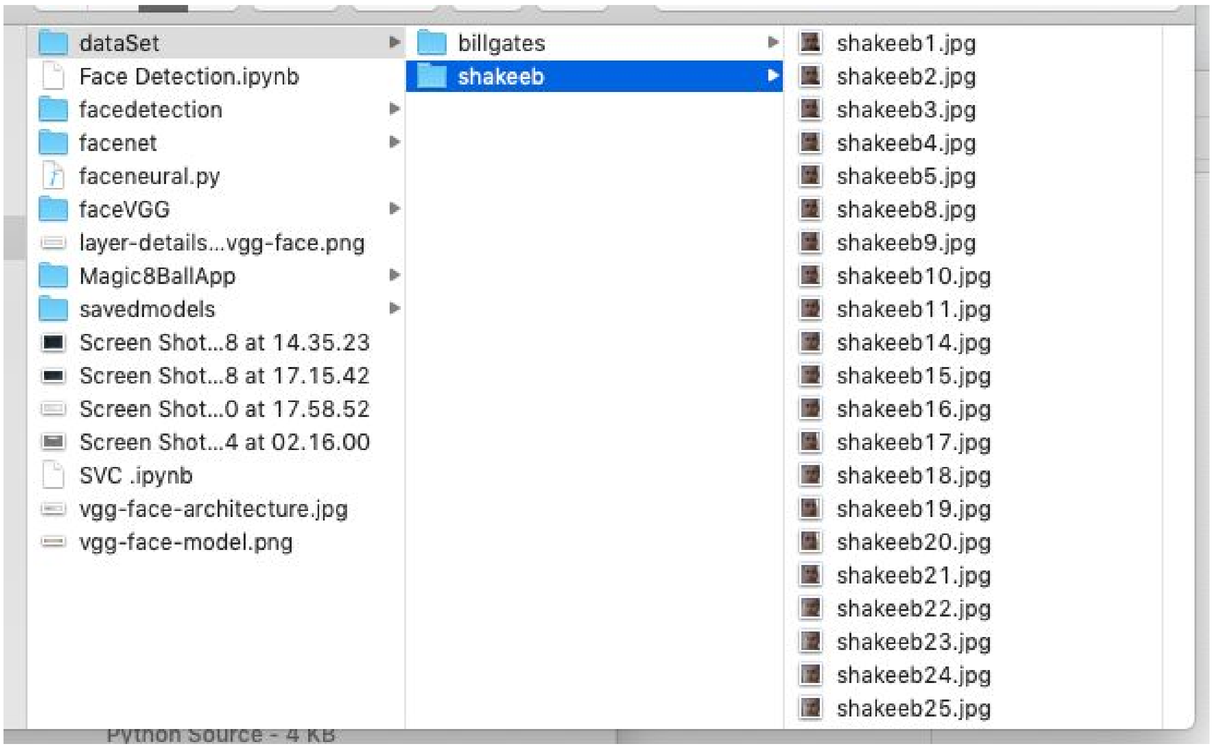
Data Exploration

In this project, I have generated my own dataset where I have captured my own photo in the webcam and also downloaded few of the images of the famous personality from Google Images as in this case I have download few of the images of Bill Gates.

Attendance using Facial Recognition requires your own photo to recognise the face of the user and then later trained to the model after preprocessing.

The dataset is separated into every user file name where each file contains at least 25 images of the user with different angles and poses of the face.

The images clicked from the webcam and downloaded from the Google Images are .jpg extension file.


dataSet
Face Detection.ipynb
facedetection
facenet
faceneural.py
faceVGG
layer-details...vgg-face.png
Magic8BallApp
savedmodels
Screen Shot...8 at 14.35.23
Screen Shot...8 at 17.15.42
Screen Shot...0 at 17.58.52
Screen Shot...4 at 02.16.00
SVC .ipynb
vgg-face-architecture.jpg
vgg-face-model.png

billgates
shakeeb

shakeeb1.jpg
shakeeb2.jpg
shakeeb3.jpg
shakeeb4.jpg
shakeeb5.jpg
shakeeb8.jpg
shakeeb9.jpg
shakeeb10.jpg
shakeeb11.jpg
shakeeb14.jpg
shakeeb15.jpg
shakeeb16.jpg
shakeeb17.jpg
shakeeb18.jpg
shakeeb19.jpg
shakeeb20.jpg
shakeeb21.jpg
shakeeb22.jpg
shakeeb23.jpg
shakeeb24.jpg
shakeeb25.jpg

Python Source - 4 KB

Visualization

The dataset contains around 25 images of each user which are separated from each file and the image are downloaded in .jpg file.

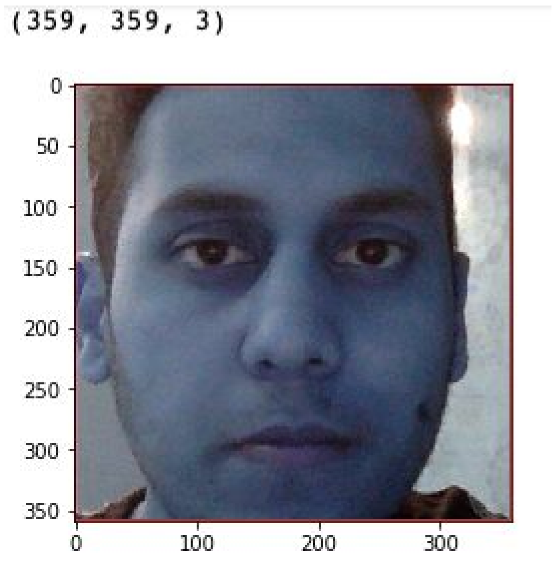
Images contains 3-dimensional array which can be either generated from the webcam or downloaded from Google Images.

The image generated by the webcam is having a low resolution which is having a height of 359 and the width of 359 with the depth of 3.

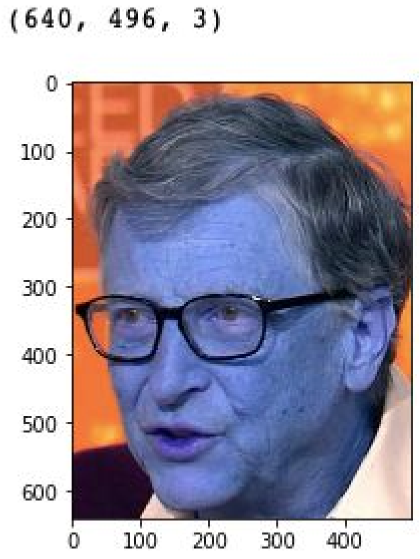
The image downloaded from Google Images is having a high resolution of height of 640 and the width of 496 with the depth of 3.

As you can observe that Images from the webcam have lower resolution than the images downloaded from Google Images.

We can reduce the size of the images to (224,224,3) to every image in preprocessing section before sending for training in the model.



```
(359, 359, 3)
```



```
(640, 496, 3)
```

<u>Algorithms and Techniques</u>

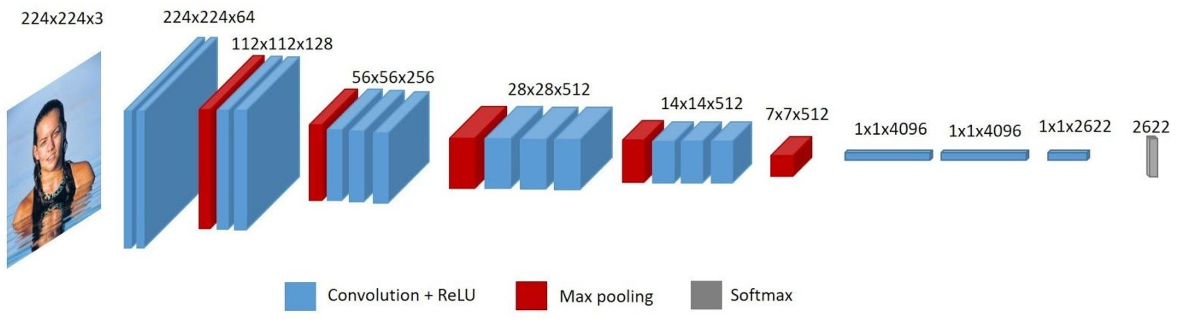
The classifier used here is VGG-Face Neural Network Model which is the implementation of VGG16 Model which was originally developed by the Visual Geometry Group. The model is mostly used in Transfer-Learning without creating the model architecture from scratch and specifying the fully connected layers.

VGG-Face Model is mainly used for the Face Recognition which was developed proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”.

The VGG-Face model is the implementation of Convolutional Neural Network, Zero Padding and Max Pooling which contains around 16 Trainable Layers.

VGG-Face Architecture:

- VGG-Face Architecture contains the model architecture is a linear sequence of layer transformations of the following types :
  - Convolution + ReLU activations
  - Max-Pooling
  - Softmax



The model uses fixed amount of learning-rate, epochs, image-size, momentum, the activation functions and the number of layers.
This model requires heavy CPU processing the run the model and the training of the model can be done fastly with the help of GPU processing but I have tried to train the model under CPU so that the system have minimum configurations will be able to run smoothly.

After the training, the weights of the model is saved and then faces can be classified accordingly.

Benchmark

Benchmark for this project are SVM and Convolutional Neural Network with a single layer where I have observed that:

❖ SVM(Support Vector Machine)

        - Support Vector Machine is a very good algorithm for classifying the faces in the images by using algorithms such as eigenfaces and PCA.
        - The Images used here is if 2-dimension and images with three-dimensions cannot be used.
        - The accuracy and f1-score is not very good using this classifier for 2-dimensional image.
        - The model takes a longer time to train and predict the new image.
- Convolutional Neural Network(Single Layer)
    - Convolutional Neural Network requires a lot of layers to train the image in order to get a good accuracy to classify the images.
    - Using a Single Layer can cause underfitting with a minimal difference between the training accuracy and testing accuracy and the loss is nearly to minus value.
    - Using Dropout will also not help to avoid the model from the overfitting.
    - The model achieve the accuracy of 100% but does not classify the face.

Methodology

Data Preprocessing

Before every model needs to be trained, the dataset need to be pre-processed to meet the requirement of the model and the computer understands the data.

Data Preprocessing is done with the help of OpenCV and keras.

Data Preprocessing is as follows:-

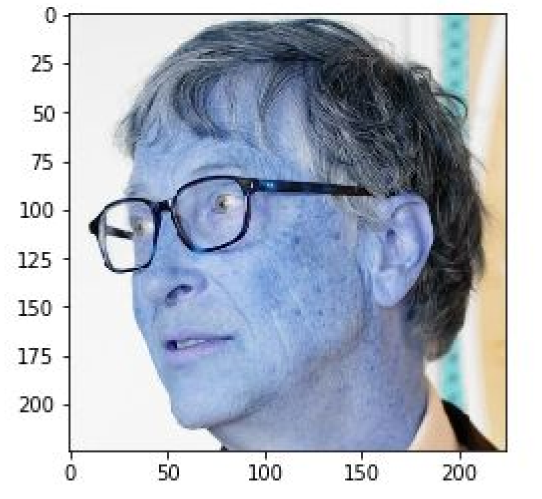
- Image is clicked from the webcam using OpenCV and then stored into the specific file.
- Image is then cropped to face for training and testing to avoid noise from the Image by detecting the face in the Image.
- Image is resized into 224*224*3 as VGG-Face model takes only the image size of (224,224,3)
- Image is converted to array and the array size of depth of 3.
- Image is divided into training and testing datasets.
- Image is converted into the data type of float by dividing with 255.

After Data Preprocessing the Image can be used for training the Neural Network but in this project I will not be dividing the datasets as the weights of the model have been saved that can be directly used for predicting.

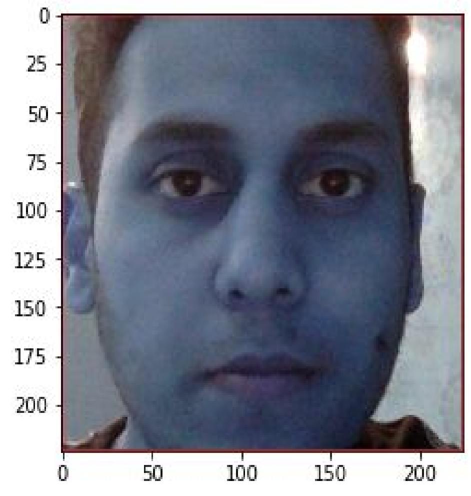
When using the web-camera for real-life face recognition data preprocessing is also required for every image the web-camera recognizes the face using OpenCV and Keras.

- Face in the image is being recognized which further crops the image ignoring the background.
- Image is resized into (224,224,3) so that the model can predict with the array shape.
- Pixel values is then divided by 127.5 which gives the mean pixel values.
- Image is sent for prediction to the VGG-Face model.

```
(224, 224, 3)
```



```
(224, 224, 3)
```



Implementation

Implementation of the project can be done after data preprocessing can be done with the help of VGG-Face Model.

It can also be done with the help of transfer learning by adding fully connected Layers using activation Softmax in VGG16 which VGG-Face have implemented.

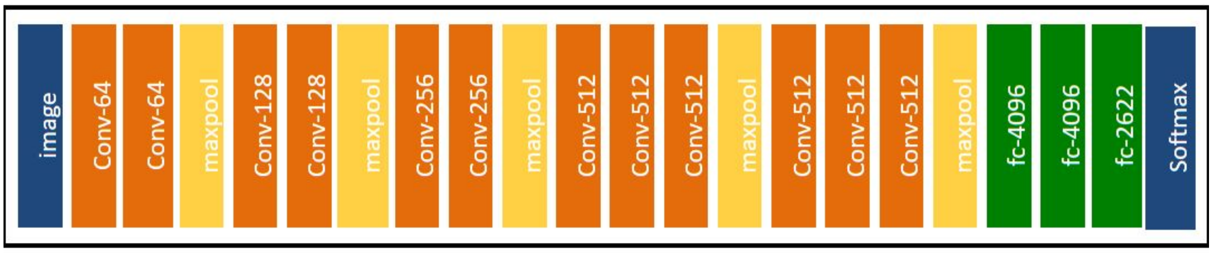
The model can be done in two methods:

- The model can be trained using your own datasets.
- Using predefined weights of the models that is already trained.

The model can either be trained from scratch by saving your own weights which the weights can be later loaded or use the predefined model of weights where the model have been trained with 14 million images of Facenet Dataset.

As the training can be time-taking and requires a lot of CPU and GPU preprocessing due to 16 training layers having 4096 channels.

Define the Network Architecture of VGG-Face that should be defined in the Sequential Manner which further the layers in can be added in the Sequential Manner as defined below:



The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center).

The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers. Max-pooling is performed over a 2×2 pixel window, with stride 2.

In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected layers follow a stack of convolutional layers:

- The first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class).
- The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

The overall structure of the model can be checked using Keras using the method summary() which determines the shape of the image it can take and the type of layer used in the model for visualizing the image for both training and testing.

Training of the model using fit() method by loading training and testing datasets where we can calculate the accuracy of how well the model have evaluated and loss to calculate the performance of the model.

Depending upon the accuracy of the training and testing model we can evaluate that the model have trained properly or not by checking overfitting or underfitting.

Save the weights of the model and then use the weights to predict the image using the pre-trained model.

Instead of training the Model from scratch, we can also use predefined weights of the model which the model have already been trained with 14 million of faces of images with the accuracy achieved of 92.7%.
In this project, I have used the predefined weights of the model that is loaded to the model and then the model is trained by creating a Dictionary where the name of the user is mapped the matrix vector of the image which is later compared to the predicted matrix vector of the real-time image from webcam that the model predicts.

If the model recognized the face of the user by predicting it with a good accuracy after preprocessing the real-time image from the webcam, attendance will be added in csv file of the name which the model predicted and time to know when the attendance was taken.

Refinement

As I am defining the predefined weights of the model which was already trained with 14 million images where the accuracy achieved was 92.7%.

But later training the model with the predefined weights is quite a difficult task where the dataset of generated by the web camera needs to be trained and the model can be trained by predicting the image of the dataset and mapping in the Dictionary with name of the face to the matrix vector of the image which is preprocessed.

The model have achieved the accuracy of predicting of 80% as more dataset is required to train the model and to predict the face with higher accuracy.

Instead of generating more dataset which can also lead to overfitting we can use a function called Cosine Similarity which can find the similarity of the training matrix-vector face image and real-time matrix-vector face image which basically calculates the distance between images.

The images must be converted to the array of the matrix vector in order to do cosine similarity as cosine similarity uses dot product of the matrix vector of both the images where the training image is transposed.

```
def findCosineDistance(vector_1, vector_2):
 a = np.matmul(np.transpose(vector_1), vector_2)

 b = np.matmul(np.transpose(vector_1), vector_1)
 c = np.matmul(np.transpose(vector_2), vector_2)

 return 1 - (a / (np.sqrt(b) * np.sqrt(c)))
```

Using Cosine Similarity the accuracy of the model of both training and predicting the image have been increased and also it does not require more image in the dataset to train the model with predefined weights of the model.

Results

Model Evaluation and Validation

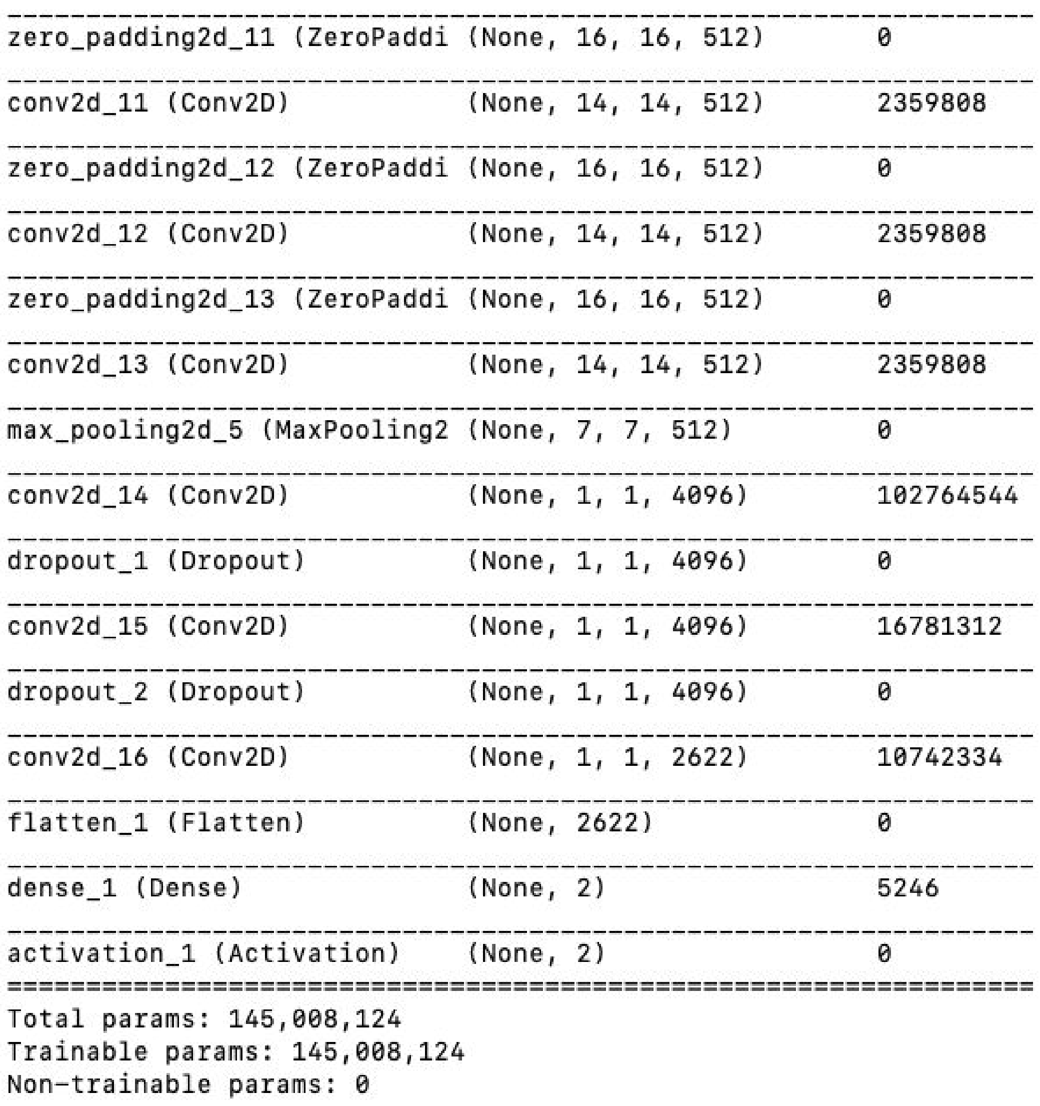
The model I have used here is VGG-Face which comes with predefined layers and input size of the image which was developed by K. Simonyan and A. Zisserman from University of Oxford and it is best performed for Face-Recognition with minimum system configurations of the machine.

The structure of VGG-Face model is as follows:

- ➔ The input to cov1 layer is of fixed size 224 x 224 RGB image.
- ➔ The image is passed through a stack of convolutional (conv.) layers.
- ➔ The filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center).
- ➔ In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity).
- ➔ The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. Layers.
- ➔ Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.
- ➔ Three Fully-Connected layers follow a stack of convolutional layers:
    - ◆ The first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class).
    - ◆ The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

The summary of the VGG-Face of the model is described below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
zero_padding2d_1 (ZeroPaddin (None, 226, 226, 3)       0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 224, 224, 64)      1792      
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 226, 226, 64)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 64)      36928     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0         
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 114, 114, 64)      0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 128)     73856     
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 114, 114, 128)     0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 112, 112, 128)     147584    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 128)       0         
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 58, 58, 128)       0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 256)       295168    
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 58, 58, 256)       0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 256)       590080    
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 58, 58, 256)       0         
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 56, 56, 256)       590080    
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 256)       0         
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 30, 30, 256)       0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 512)       1180160   
_________________________________________________________________
zero_padding2d_9 (ZeroPaddin (None, 30, 30, 512)       0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 512)       2359808   
_________________________________________________________________
zero_padding2d_10 (ZeroPaddi (None, 30, 30, 512)       0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 512)       2359808   
```



```
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
zero_padding2d_13 (ZeroPaddi (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 1, 1, 4096)        102764544
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 1, 4096)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 1, 1, 4096)        16781312
_________________________________________________________________
dropout_2 (Dropout)          (None, 1, 1, 4096)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 1, 1, 2622)        10742334
_________________________________________________________________
flatten_1 (Flatten)          (None, 2622)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 5246
_________________________________________________________________
activation_1 (Activation)    (None, 2)                 0
=================================================================
Total params: 145,008,124
Trainable params: 145,008,124
Non-trainable params: 0
```

The complete description of the final model of the project is as follows:-

- ➔ The model have already been trained which we can use the predefined weights of the model to predict the images.
- ➔ After preprocessing the images in the dataset to image array and adding the extra dimensions of axis=0.
- ➔ The model is being trained by rather predicting the image with the predefined weights of the model and converting it into matrix vector which maps to the name of the face using Dictionary.
- ➔ The model is then tested with the accuracy by predicting the faces in the real-time images from the webcam after cropping the image using cosine-similarity.
- ➔ The attendance is then added to csv file with name the model predicted and the time attendance was taken.

To view the robustness of the final model, a test was conducted using real-time images from the webcam which generates the low-pixel values:

- ❖ The model was classify the identity of the face in the image at different situations such as when the eyes is closed, the face is turned to either right or left by a small angle, when the expression of the face is changed.
- ❖ The model was also able to classify the identity of the face when the person is far away from the camera which OpenCV is able to detect the face properly.
- ❖ The model cannot classify when the model is too near to the camera as the OpenCV is not able to detect the face in the camera.
- ❖ The model is not able to identify the identity of the face when it is very dark in the background or when the camera is filled with water or foggy.

Justification

The model used here is VGG-Face which is developed by University of Oxford where the model is built under VGG-16 model.

The model is able to classify the face in a camera which produces good-quality of the image but in case if we use web camera of the Laptop which produces a low-pixel images the model is only able to classify the images if the background environment have the good amount of source of light.

The model have some classification delay when the model predicts the real-time image from webcam for about 4 seconds under CPU Processing that classifies the correct identity of the face.

Comparing with Support Vector Machine and CNN using Single Layer, the accuracy of both of benchmarks have better results which is higher than 90% where observed accuracy of the Support Vector Machine can lead to overfitting and CNN using Single Layer can lead to Underfitting which it does not classify the image properly.

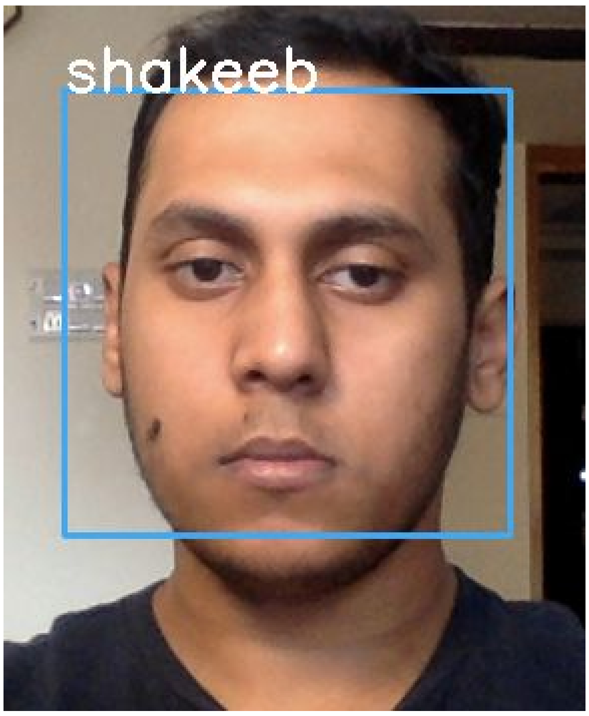
In summary, using this model can constraint to limited domains under CPU processing and requires a very good amount of light to detect the face and classify the image and thus a better hardware with more processing power such as GPU+CPU processing should be required to detect the face at any background and to reduce the classification delay.

Conclusion

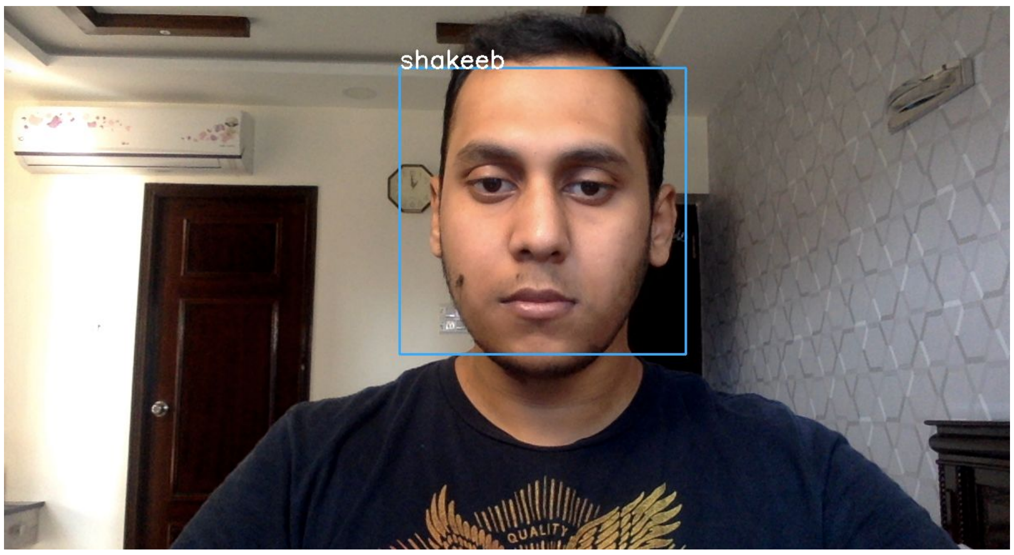
Free Form Visualization

Below are the real-world images are from the laptop web-camera.

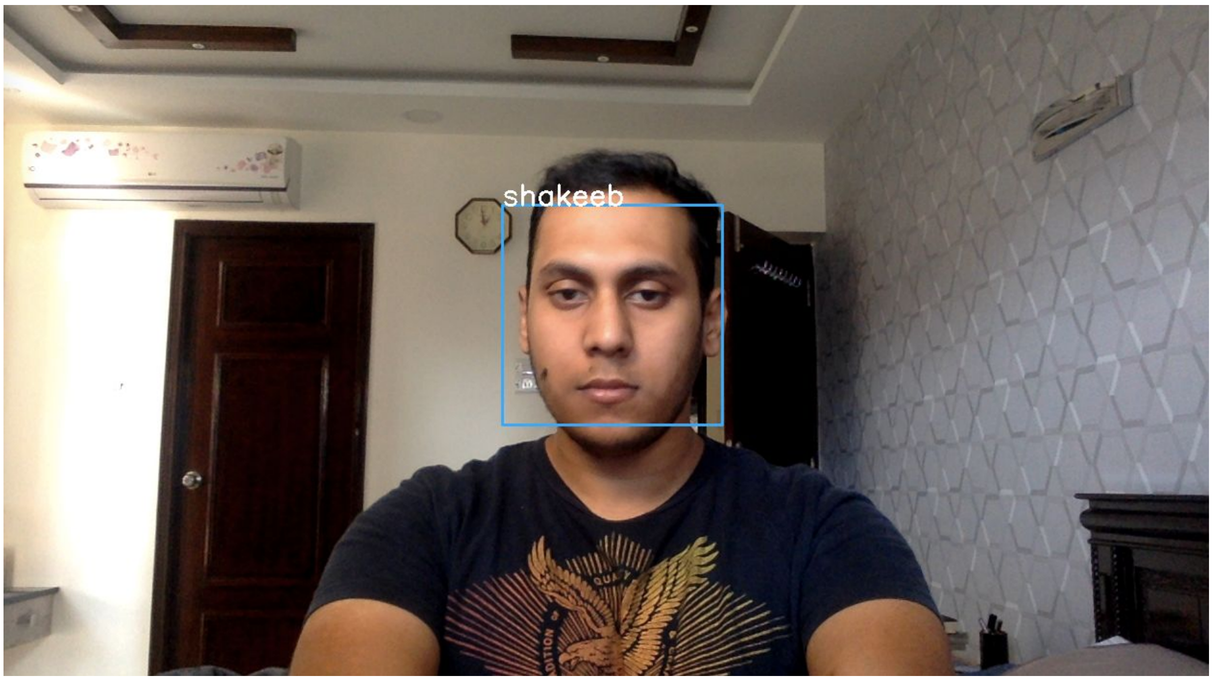
Detecting the face and classify the identity of the face.



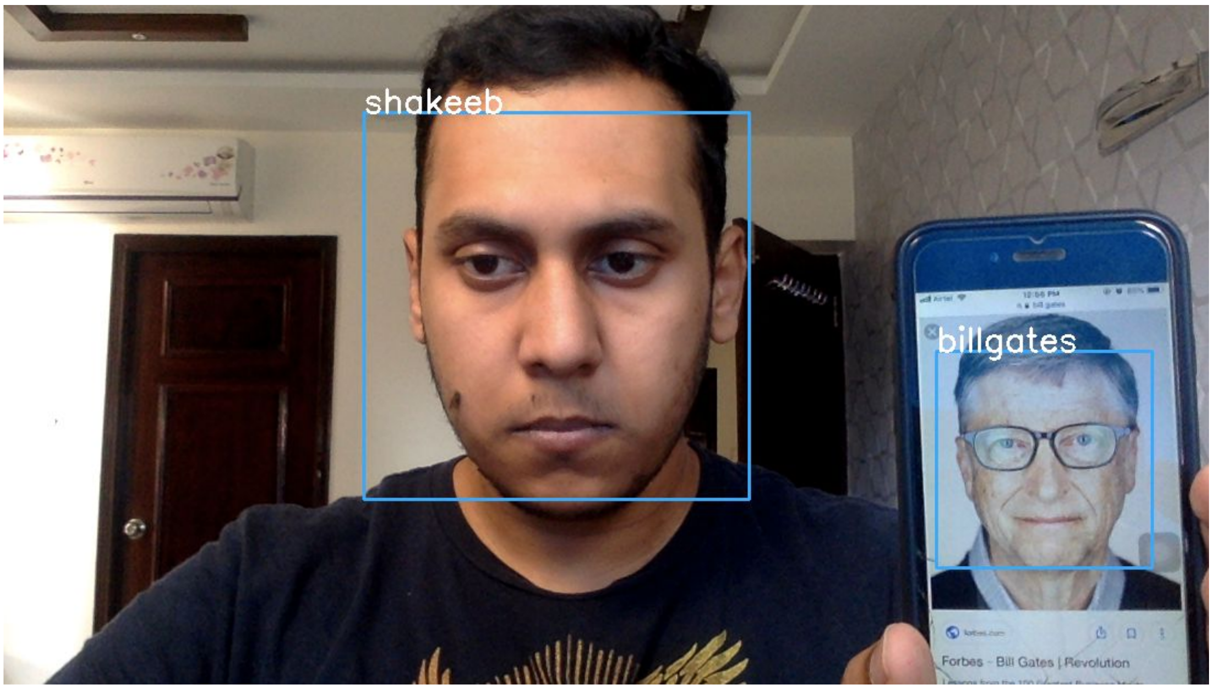
Detecting the face at the nearby distance and classifying it.



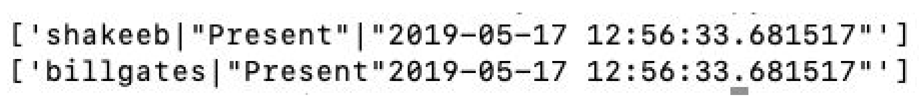
Detecting the image at a far-away distance.



The model detecting two faces and classifying the two faces.



After classification of the faces, the attendance is being added in the attendance sheet specifying the identity of the face and the time of the attendance.



```
['shakeeb|"Present"|"2019-05-17 12:56:33.681517"']
['billgates|"Present"2019-05-17 12:56:33.681517"']
```

Reflection

The project can be summarized in two methods as follows:-

I Method

- ❖ Generates an image dataset folder with the name of the user.
- ❖ The dataset of each user images is then preprocessed using OpenCV and keras.
- ❖ The dataset is divided into training and testing dataset if doing training manually.
- ❖ The Benchmark was created for the VGG-Face model using Keras.
- ❖ The Architecture of VGG-Face is initialized by adding the specific layers with the fixed learning rate, image shape, and other parameters.
- ❖ Dictionary is created with target name to the matrix vector of the image.
- ❖ The model is then trained and weights of the model is created and loaded.
- ❖ The model is then tested by loading the weights and check the accuracy.
- ❖ The real-time image is then preprocessed and then the model predicts the face in the image using Cosine Similarity which checks the similarity.
- ❖ Attendance is added with name of the face and the time the attendance was taken.

II method

- ➔ Generates an image dataset folder with the name of the user.
- ➔ The dataset of each user images is then preprocessed using OpenCV and keras.
- ➔ The Benchmark was created for the VGG-Face model using Keras.
- ➔ The Architecture of VGG-Face is initialized by adding the specific layers with the fixed learning rate, image shape, and other parameters.
- ➔ The model uses the predefined weights of the model which was earlier trained.
- ➔ Dictionary is created with target name to the matrix vector of the image by predicting the preprocessed image that also trains the model.
- ➔ The model is then tested with the new images with the predefined weights by checking the accuracy.
- ➔ The real-time image is then preprocessed and then the model predicts the face in the image using Cosine Similarity which checks the similarity.
- ➔ Attendance is added with name of the face and the time the attendance was taken.

I found this algorithm to be much better than other algorithms as this algorithm does not require a lot of training by using the predefined weights available in the Internet.

However, I found 1st method to be very difficulty and also very much interesting as we got to know the architecture of the VGG-Face model and training of the model requires a lot of CPU Processing as my laptop does not Graphics Card as it took a lot of time to train the model after preprocessing the data.

But with the predefined weights of the model it made it very easy to actually use the trained model which does not much of the time and also do not take much of the CPU processing which the model can easily predict with the help of cosine similarity.

It was very difficult for me to do face recognition in simple CNN with single layer or multiple layers where the model was either underfitting or overfitting and was not able to classify the faces properly.

Improvement

To achieve better results and optimal user experience this model can perform better when the system have higher processing power such CPU and GPU combined together to achieve higher result and to reduce the classification delay.

The user experience can slightly improved by having the datasets of more people with folder name of the user and containing around 40 images with different angles of the faces and different face expressions of the user.