| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

Surveillance System using Face Recognition with Tensor Flow Object Detection API

Vamsi Anamalamudi[1], Tabassum Khatheeja[2], Sumanth S[3], Farhana Kausar[4]

B.E. Student, Department of Computer Engineering, Atria Institute of Technology, Bengaluru, Karnataka, India[1]

B.E. Student, Department of Computer Engineering, Atria Institute of Technology, Bengaluru, Karnataka, India[2]

B.E. Student, Department of Computer Engineering, Atria Institute of Technology, Bengaluru, Karnataka, India[3]

Professor, Department of Computer Engineering, Atria Institute of Technology, Bengaluru, Karnataka, India[4]

ABSTRACT: With increase in population could lead to serious security consequences not only for a country but for the entire world. The crime rate in India has shown a steep rise in the past years. With growing crime rate it is necessary to bring in new advancement in detecting cases or solving them, in a crowded nation like India identifying the culprits is a tedious task, by introducing the camera surveillance technology in solving these crimes can be a major advancement in that field leading to a lower crime rate, this particular system installed in every street keeps track of all the activities happening in that particular region helping cops to identify the culprits, by combining face recognition we can easily keep track of that person's movement indicating his current location whenever his presence is detected in a particular camera , it compares the database and updates the current location of the culprit helping to solve crimes and gradually reducing the crime rates.

KEYWORDS: Face recognition, Video surveillance, TensorFlow object detection API.

I. INTRODUCTION

Over the past few years, the government and various organizations are understanding the importance of intelligent surveillance systems. Traditional methods had a lot of human dependence, which had certain inadequacies like cost, multi-screen monitoring etc. Intelligent video surveillance replaces traditional methods, which are more accurate in monitoring.

The goal of this paper is to recognise faces of criminals using TensorFlow object detection API with higher accuracy. This solution is proposed after satisfying results obtained from tests with rich databases in terms of pose, light and subjects. This API is an accurate machine learning API. We can use this API for multiple use cases like object detection, person recognition etc.

II. RELATED WORK

Face recognition algorithms can be used for identifying criminals or suspects. There are many algorithms for this purpose, but they differ in efficiency, requirements and processing time. Convolutional Neural Networks is considered most efficient among the available face recognition algorithms out there. The main idea is to get a deep neural network to produce a bunch of numbers that describe a face commonly known as face encodings. There are few trained models like dlib that can be used.

The TensorFlow's Object Detection API can be used to train an object detection classifier for multiple objects. There are many use cases of this TensorFlow Object Detection API. This API is mostly used for recognizing objects like car, gun, playing cards etc.

The object detection model is trained to identify the objects present in the given image or video stream and also specifies their positions within the image. For example, we can train a model with images that contain various pieces of flower with its label that specifies the class of flowers it represents (E.g. Lilies, Orchids, Roses) and specifying the location where each object appears in the image.

When we give an image to the trained model, it will return the list of objects it detects in it, the location of a bounding box that contains the object detected, and the score that indicates the confidence that detection was correct. So, we trained our model to recognize faces and the results turned out to be pretty good.

III. PROPOSED ALGORITHM

There are certain steps for setting up the training model to recognizing faces:

1. Setting up the environment for anaconda.

International Journal of Innovative Research in Computerand Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

2. Collect the images for training and label the faces in the images.
3. Generating the training data.
4. Training the model.
5. Run the training.

Step 1: Initially we need to install Anaconda, CUDA, and cuDNN. We need to check the CUDA and cuDNN versions compatible for your system from Tensorflow website. Download TensorFlow Object Detection API repository from GitHub and also download tensorflow object detection repository from their official github repository. Now lets download the Faster-RCNN-Inception-V2-COCO model from TensorFlow's model zoo. We use this model when the processor has good speed and is capable of producing good results, If not we have to use MobileNet-SSD-V1.To train a custom object detector, delete the following files accordingly.
Create a virtual environment by using the command conda create -n ENV_NAME pip python=3.5. Then, activate the environment. Then install tensorflow GPU version or CPU version. Install the other necessary packages like pillow,lxml, Cython, contextlib2, jupyter,matplotlib, pandas, opencv using pip command. Configure the paths respectively.

Step 2: Collect the images for training. It is recommended to have at least 200 pictures overall to have a great accuracy. Now We need to label the desired faces in every picture, this can be done using labelImg software. Draw a box around each object in each image from \images\train directory.

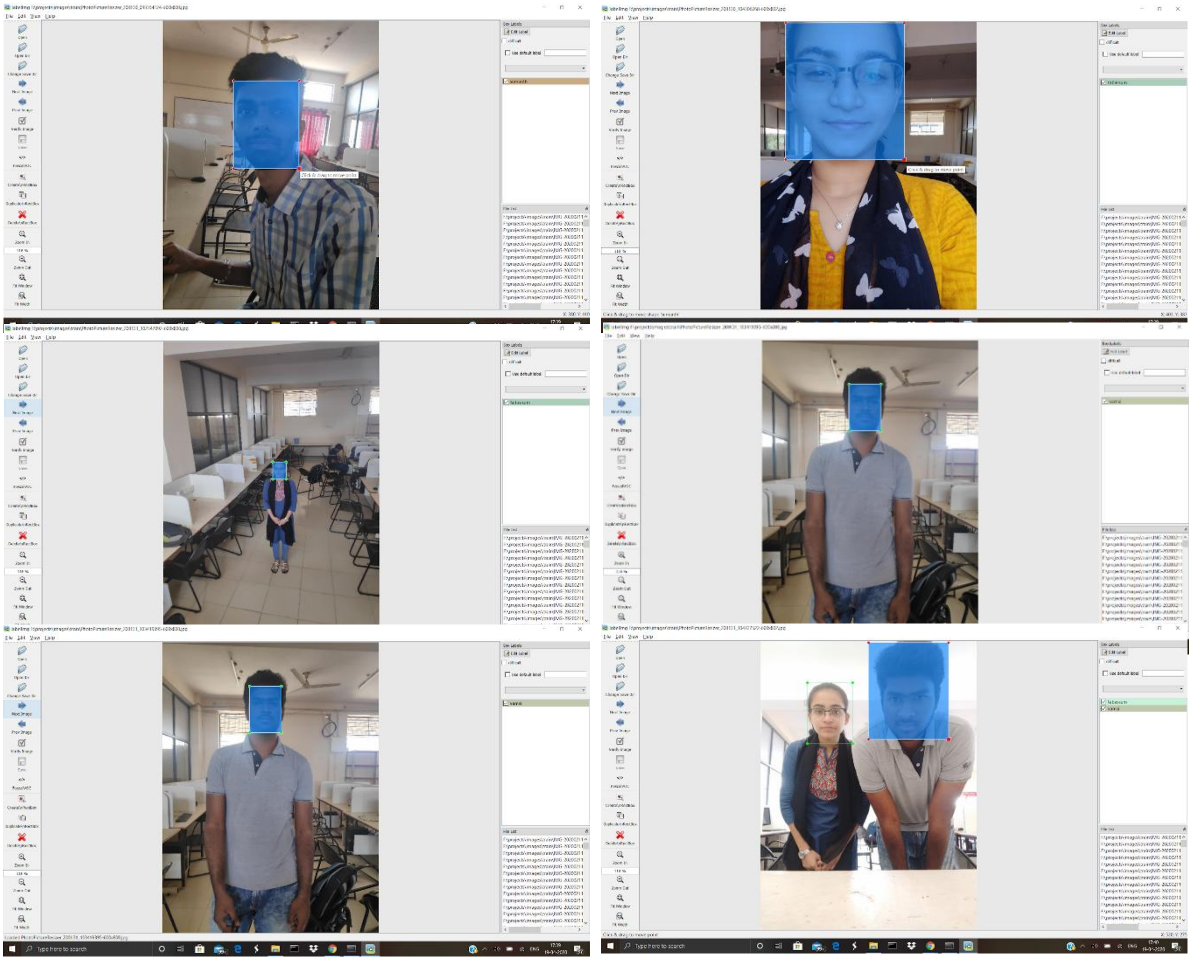

Fig.1. Labelling of faces in pictures

Repeat the process for all the images in the \images\test directory too. LabelImg saves the file in .xml file which contains the label data for each image.

Step 3: We need to convert the .xml files to .csv files for further process. Model repository which was downloaded above already has a script which does it for you. We need to just run the xml_to_csv.py script.

Then we need to generate TFRecords by using the generate_tfrecord.py in the object_detection folder.
We need to modify the labelMap.pbtxt according to the objects you want the model to recognise.

Step 4: Before starting the training process we need to modify the train.py according to the file setup. Then start training. It might take a while based on your system speed.

Step 5: We can run the model on an image / video / live video stream using the scripts in the object_detection_image.py / object_detection_video.py / object_detection_webcam.py respectively.

IV. PSEUDO CODE

Step 1: Anaconda virtual environment can be created using command:

```
conda create -n ENV_NAME python=x.x anaconda
```

To activate the environment:

```
activate ENV_NAME
```

Installation of necessary packages can be done using:

```
conda install -c anaconda protobuf
conda install pillow
conda install lxml
conda install Cython
conda install contextlib2
conda install jupyter
conda install matplotlib
conda install pandas
conda install opencv-python
```

Step 2: Collect the images for training. I used 181 pictures to train my face recognizer. Each image should be less than 200KB each, and their resolution shouldn't exceed 720x1280. After you have all images save 20% of them in \test and 80% in \train directories of \object_detection\images.

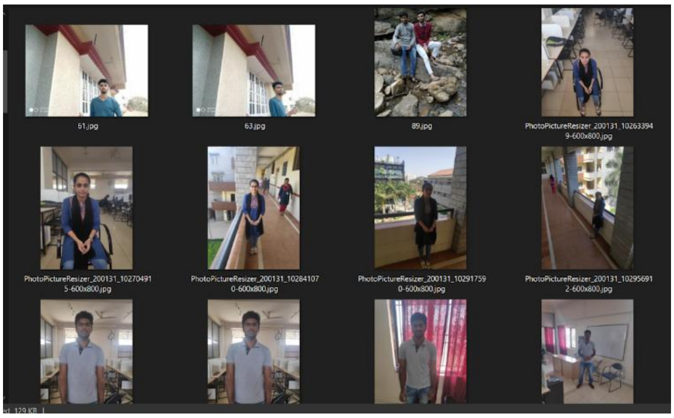

Fig.2.train folder before labelling

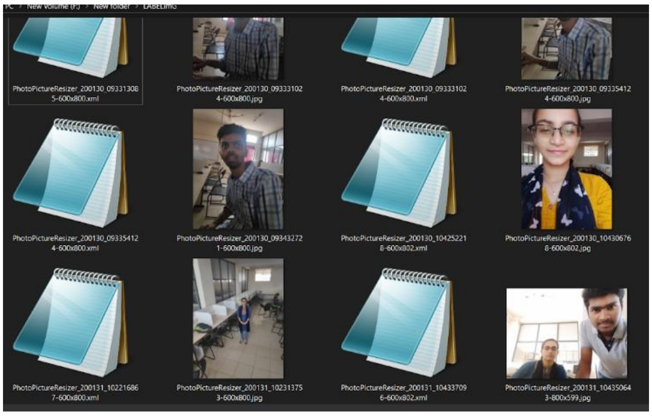

Fig.3.train folder after labelling

Step 3: After running the xml_to_csv.py script, it'll create two files test_labels.csv and train_labels.csv in the images folder.

To generate TFRecords we need to modify it as follows and run generate_tfrecord.py

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

```python
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'sharan':
        return 1
    elif row_label == 'suraj':
        return 2
    elif row_label == 'sumith':
        return 3
    elif row_label == 'vamsi':
        return 4
    elif row_label == 'tabassum':
        return 5
    elif row_label == 'sumanth':
        return 6
    else:
        None
```

Fig.4. Modification of generate_tfrecord.py

We need to modify the labelMap.pbtxt according to faces you want the model to recognize.

```
item {
  id: 1
  name: 'sharan'
}

item {
  id: 2
  name: 'suraj'
}

item {
  id: 3
  name: 'sumith'
}

item {
  id: 4
  name: 'vamsi'
}

item {
  id: 5
  name: 'tabassum'
}

item {
  id: 6
  name: 'sumanth'
}

```

Fig.5. Modification of labelMap.pbtxt

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

Step 4: Training the model will take a few hours based on your system performance.

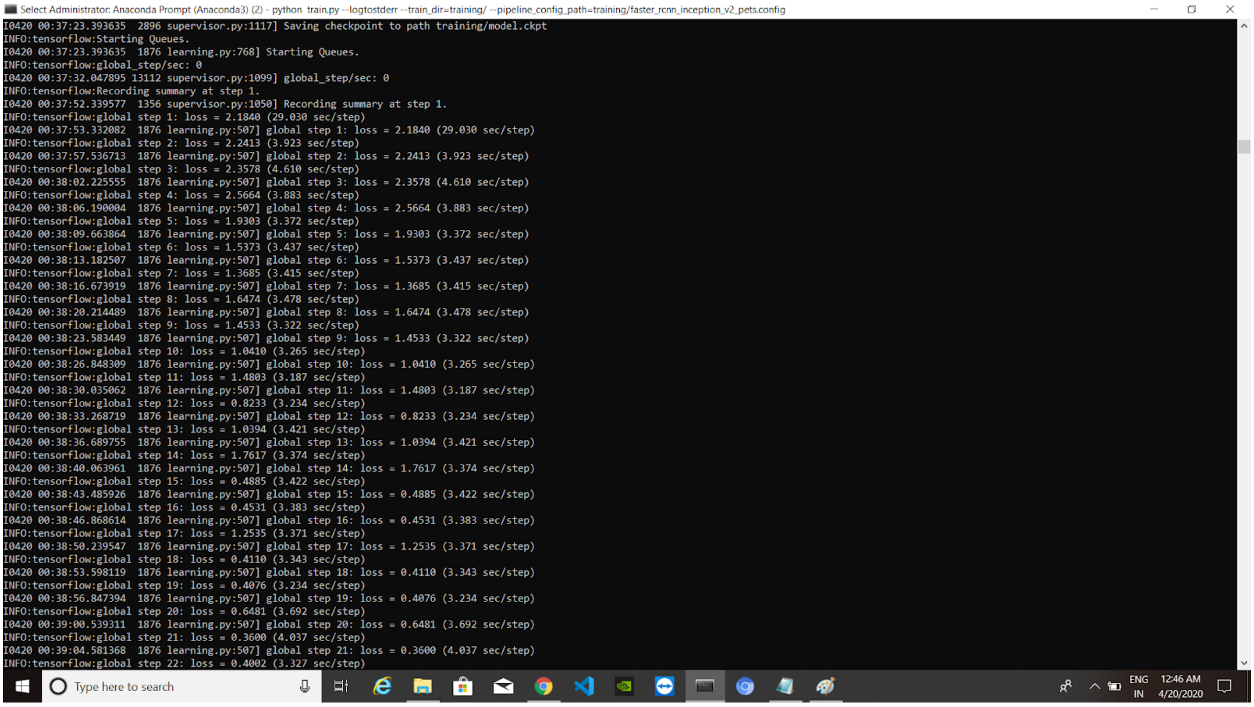

Fig.6. Training of model

As your training the model, you can view the training graph by using the command:

>>>tensorboard --logdir=training

We have to stop the training when the loss reaches less than 0.05. After stopping the training we have to export the inference graph by using the command:

>>> python export_inference_graph.py
--input_typeimage_tensor --pipeline_config_path training/faster_rcnn_inception_v2_pets.config
--trained_checkpoint_prefix training/model.ckpt-XXXX
--output_directoryinference_graph

Step 5: We can use the newly trained classifier:
Modify the parameters in the object_detection_webcam.py according to the number of faces and run the file using command:

>>> python object_detection_webcam.py

In addition to these steps we can print the confusion matrix by using the command:

>>> python3 confusion_matrix.py
--detections_record=F:/projects/tensorflow1/models/research/object_detection/detections.record
--label_map=F:/projects/tensorflow1/models/research/object_detection/training/labelmap.pbtxt
--output_path=F:/projects/tensorflow1/models/research/object_detection/confusion_matrix.csv
And also the graphical representation of the real time face recognised in the camera.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

V. SIMULATION RESULTS

OUTPUT OF TESTING THE MODEL FOR INPUT IMAGE,

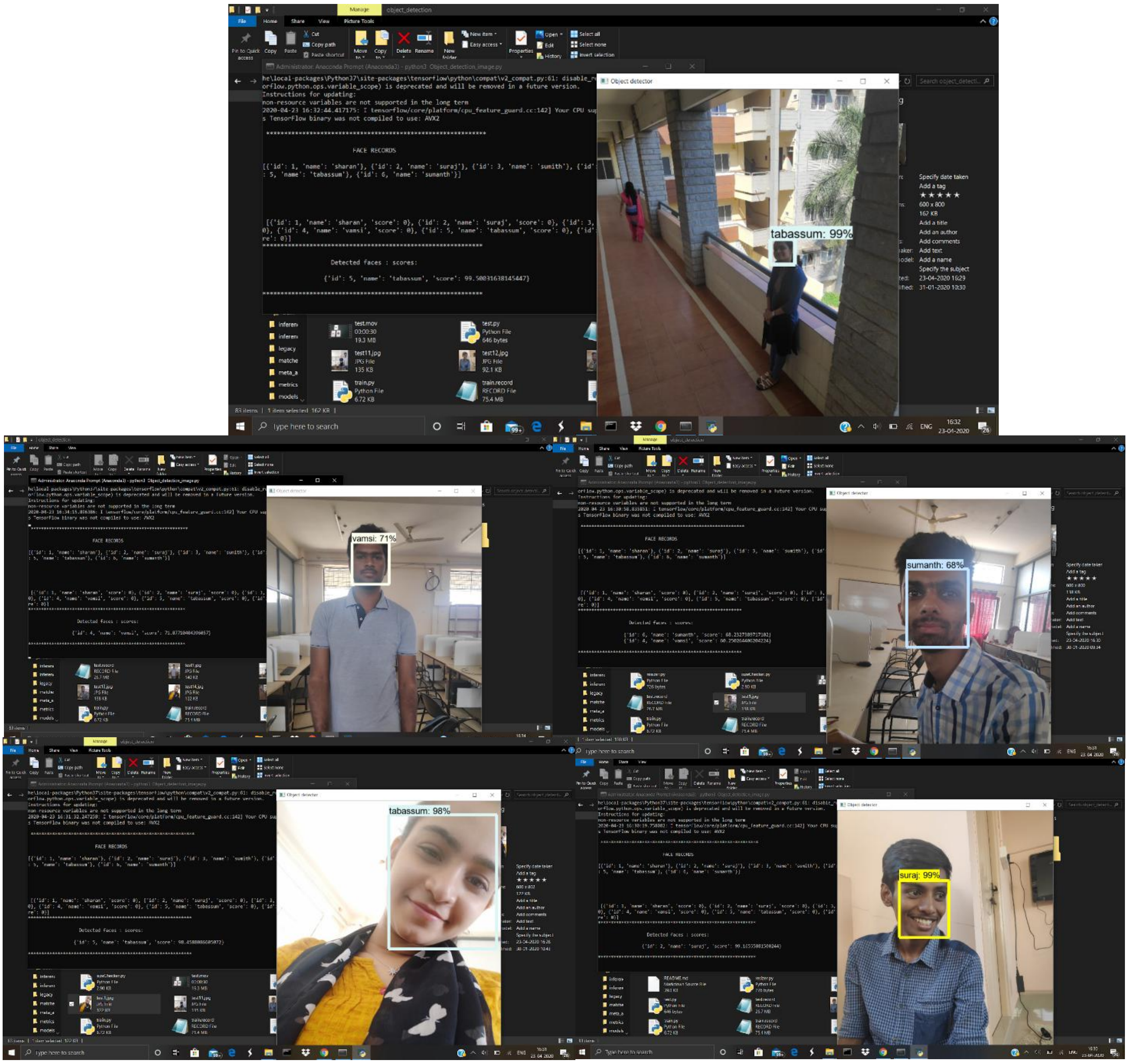

Fig.7.Output of image input

Output of testing the model using live webcam feed,

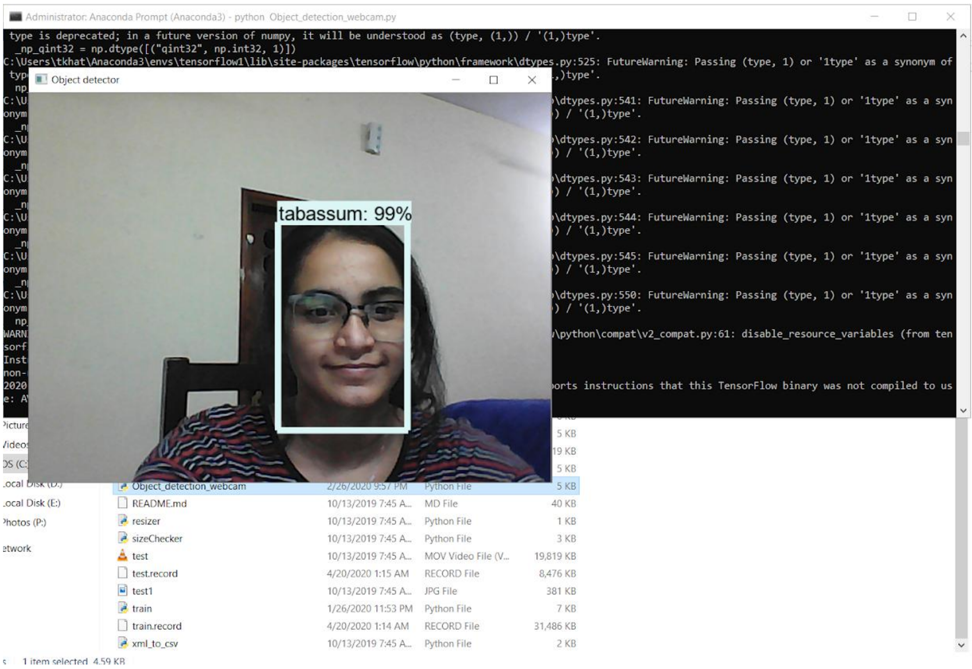

Fig.8.Output of live webcam feed

We can obtain the confusion matrix of the model as well by using this command

>>>python3 confusion_matrix.py
--detections_record=F:/projects/tensorflow1/models/research/object_detection/detections.record
--label_map=F:/projects/tensorflow1/models/research/object_detection/training/labelmap.pbtxt
--output_path=F:/projects/tensorflow1/models/research/object_detection/confusion_matrix.csv

Output of confusion matrix:

```
Confusion Matrix:
[[ 59.   0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.  69.   0.   0.   0.   0.   0.   0.   0.   0.]
 [  0.   0.  44.   0.   0.   0.   0.   0.   0.   2.]
 [  0.   0.   0.  72.   0.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.  89.   0.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   7.   0.   0.   0.   0.]
 [  0.   0.   0.   0.   0.   0. 728.   1.   0.  33.]
 [  0.   0.   0.   0.   0.   0.   0. 871.   0.   9.]
 [  0.   0.   0.   0.   0.   0.   0.   5.   0.   0.]
 [  2.   3.   0.   0.   5.   0. 234. 108.   0.   0.]]

confusion_matrix.py:116: RuntimeWarning: invalid value encountered in double_scalars
  precision = float(confusion_matrix[id, id] / total_predicted)
   category  precision_@0.5IOU  recall_@0.5IOU
0    sharan           0.967213        1.000000
1     suraj           0.958333        1.000000
2    sumith           1.000000        0.956522
3     vamsi           1.000000        1.000000
4  tabassum           0.946809        1.000000
5   sumanth           1.000000        1.000000
6      fire           0.756757        0.955381
7       gun           0.884264        0.989773
8     knife                NaN        0.000000
```

Fig.9.Confusion matrix

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

Graphical view of the output in bar graph,

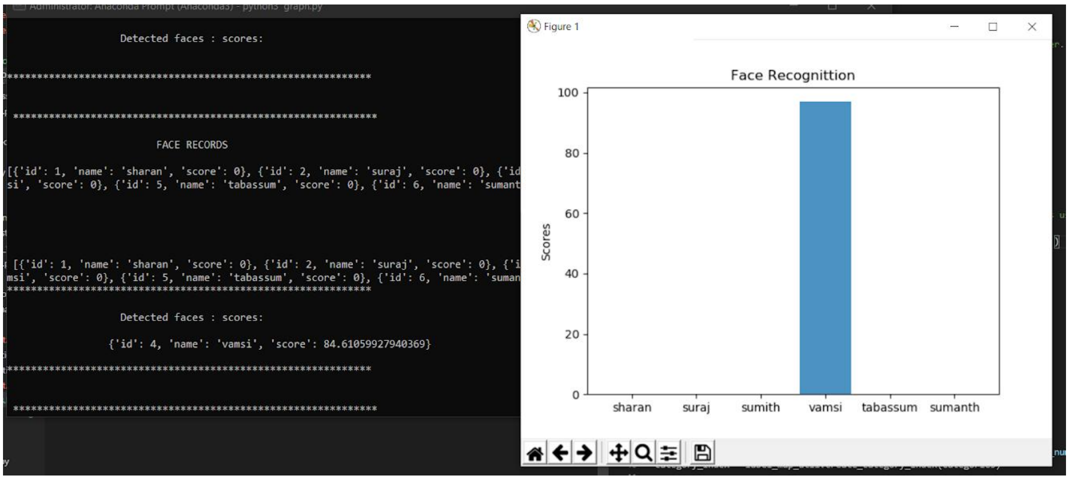

Fig.10.Graphical output

VI. Conclusion and Future Work

The facial recognition algorithms can be embedded into existing CCTV networks, to find missing persons or tracking suspected criminals. Parents or guardians can provide local authorities with photos of their children and police can match them with missing persons databases and thus can find missing children. This can play a key role in identifying suspects/criminals at airports, railway stations, bus stops, city centers etc. Image database investigations as well as searching in the Facebook social networking web site.

References

1. EdjeElectronics, "TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10","github", 22 june 2019.
2. SodhA, "TensorFlow Object Detection API tutorial — Training and Evaluating Custom Object Detector","mc.ai",26. March 2018.
3. Tensorflow, "Tensorflow Object Detection API", "github", Nov 13th, 2019.
4. Tensorflow, "Tensorflow detection model zoo","github", Nov 14, 2019
5. tzutalin,"labelImg","github" ,Jan 30 2020
6. Santiago L. Valdarrama, "Confusion Matrix in Object Detection with TensorFlow,"ShiftedUp",October 10, 2018.