**Realtime Custom Object Detection and Recognition Project**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | |  |  | |  |
|  | | |  | | |  |
|  | | |  | | |  |
|  |  | |  |  | |  |
|  |  |  |  |  |  |  |
|  |  |  |

**Abstract**

In this paper, we discuss our project “Realtime Custom Object Detection and recognition”. In this project we have implemented Celebrity Face Recognition in images, COCO real-time Multiple object detection using YOLO and Realtime Single object detection of input from webcam and videos. The great progress of automatic face recognition has made large- scale face identification possible for many practical applications . This application is widely used when the images for the persons to be recognized are available beforehand.

**1 Introduction**

The goal of the project “Realtime Custom Object Detection & Recognition” is to develop a machine learning model which a) Detects the objects in Realtime scenarios. b) With an additional capability to recognize the missing persons that were somewhere lost (may be at airports, shopping complexes, food court etc.). c) Network will be re-trained to detect the objects in COCO dataset and the persons listed in our extended dataset (containing the labelled images of some persons) in Realtime.

The motivation behind this project were namely, a) Recognition and detection of objects from a video has been one of the blooming topics. b) Real time recognition and detection is bounded by many constraints like background, multiple objects, higher FPS etc. – “an interesting area to explore more!”. c) Often, humans recognize a multitude of objects in images with little effort that makes our effort useful. d) Using such powerful concepts, we will be able to find missing persons in public places like airports, hotels etc.

Even it’s one of the crucial concepts in the field of autonomous driving. e) A good future scopes.

The technical requirements of the project were, it implemented in Python using many widely used

libraries namely, OpenCV (for image manipulations

and detection), NumPy, imutils, argparse, time, Dlib

etc. The operation system used is Linux.

We have used the dataset COCO (common objects in context) along with a custom dataset having celebrity faces from Kaggle. The project used 2 pre-trained neural networks YOLO and nn4.small2.v1.

**2 Dataset Description**

We have two datasets, one from COCO (Common objects in context) and the other is the custom dataset having Celebrity faces from Kaggle.

Therefore, for Object detection of common objects like airplane, dog, cat, cars etc. we have gathered the data from MS COCO dataset (already provided to us).  The link to the dataset is [http://cocodataset.org/#download : link for downloading 2017 Train/Test/Val dataset](http://cocodataset.org/" \l "download : link for downloading 2017 Train/Test/Val dataset)

For Person Recognition (Person Finder) or Celebrity Recognition like recognizing a celebrity in Realtime, we gathered the data from Kaggle. The link to the dataset is <https://www.kaggle.com/dansbecker/5-celebrity-faces-dataset>. The dataset somewhat looks like shown in Figure 1 and 2 below.



**Figure 1 The COCO dataset (some images).**

**Figure 2 The Celebrity Faces Kaggle dataset(some handpicked images).**

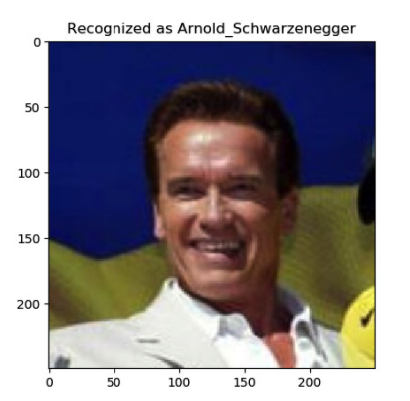
**3 Project Description**

We have followed the “Supervised Training” here for each of the project parts. That is the network is trained with labelled input having desired classes known at the training time and after training, new inputs are predicted that were not a part of the training dataset. We have developed three parts of the Realtime object and

Recognition namely,

**3.1.1** **Celebrity Face Recognition in Images**

Here the Input will be an image for a celebrity with which the network (CNN) is trained and output will be the image with bounding boxes and labels around the celebrity faces. The output will be somewhat like the one in the figure 3 shown below.



**Figure 3 The Celebrity Recognition output**

The celebrity faces from the Kaggle dataset are labelled using the YAT (YOLO Annotation Tool).

**3.1.2 COCO real-time multiple object detection using YOLO**

Here the Input will be a video of common objects (COCO) that the YOLO is trained with and output will be the video with bounding boxes and labels around the objects. The output will be somewhat like the one in the figure 4 shown below.



**Figure 4 The COCO multiple object detection output**

Here no preprocessing was done because the COCO dataset is already pre-processed.

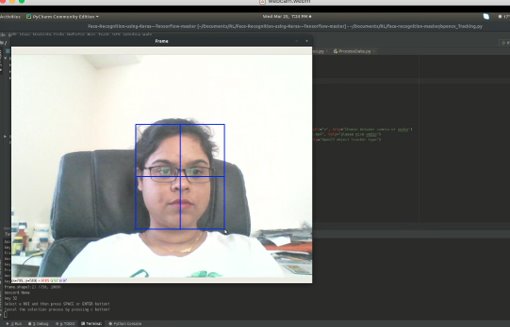
**3.1.3** **Realtime object detection of input from webcam/Video**

Here the Input will be the one from webcam, or a video of an object captured using webcam by drawing bounding boxes, since the network YOLO has been trained with that in the process therefore, output will be the video with confidence labels.

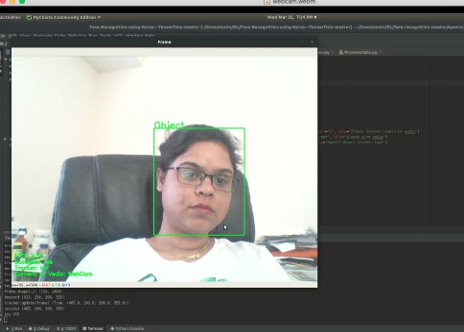
We have the following trackers that can be used here namely,

1. csrt": cv2.TrackerCSRT\_create​
2. kcf": cv2.TrackerKCF\_create​
3. boosting": cv2.TrackerBoosting\_create​
4. mil": cv2.TrackerMIL\_create,​
5. tld": cv2.TrackerTLD\_create​
6. medianflow": cv2.TrackerMedianFlow\_create​
7. mosse": cv2.TrackerMOSSE\_create

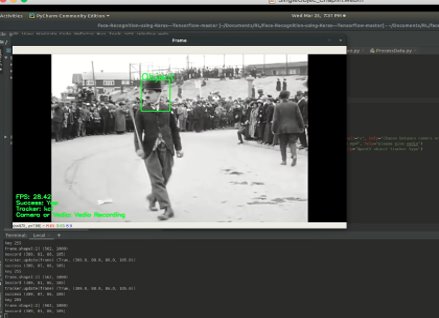
The output or series of steps followed will be somewhat like the one in the figure 5a,5b and 6 shown below.



**Figure 5a Capturing input through webcam & making bounding box over the feature to capture**



**Figure 5b the captured live object is detected and confidence intervals are shown**



**Figure 6 the captured live object is detected in video and confidence intervals are show**

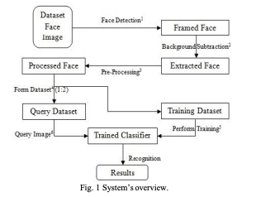
**3.2** **Main references used**

The first reference used for the multiple object detection for COCO dataset is from YOLO present at the link <https://pjreddie.com/media/files/papers/YOLOv3.pdf>We have not implemented or used the entire paper here, just some of the algorithms are used like the section “2.1 Bounding Box Prediction” - It divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. We choose YOLO because for real time object detection it is extremely fast and accurate. And there is an easy tradeoff between speed and accuracy simply by changing the size of the model without retraining.

The second reference used is from the COCO at the link: <https://arxiv.org/pdf/1405.0312>. We used this dataset because it was provided by the professor as an implementation pre-requisite and secondly, it’s one of the finest datasets used in the machine learning projects because of the following,

1. It’s one of the large-scale object detections, segmentation, and captioning dataset, having several features:
2. Object segmentation
3. Recognition in context
4. Super pixel stuff segmentation
5. 330K images (>200K labeled)
6. 1.5 million object instances
7. 80 object categories
8. 91 stuff categories
9. 5 captions per image
10. 250,000 people with key points

The last or the third reference used is from the Person finder application for Celebrity recognition, the link to the paper is: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwjXi_L1jbvnAhVGIqwKHTJlDNQQFjABegQIAhAB&url=https%3A%2F%2Farxiv.org%2Fpdf%2F1302.6379&usg=AOvVaw3T9G28b-P1ECXauRR-dskp>. From the paper, we used the process or sequence of steps that should be followed for extracting the face from an image, aligning the extracted face etc. Figure 7 shows the process followed.

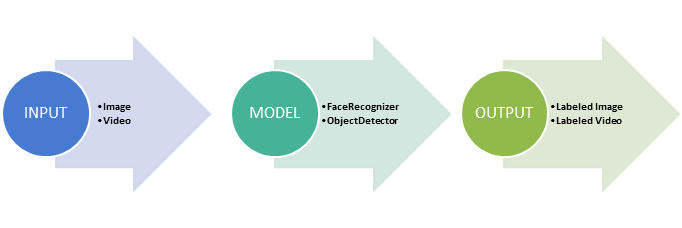


**Figure 7 Process flow for Face Detection and recognition**

**3.3** **Implementation Methods**

Supervised Training is executed, that is the training data is labeled using the YAT Annotation tool. The input to the model will be an image or a video not from the training dataset. The output will be the image or video having the bounding boxes with the labels around the corresponding object recognized as a part of this process.

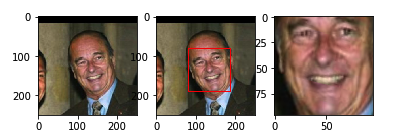
We can see as pictured in the figure 8 below.



**Figure 8 Process flow**

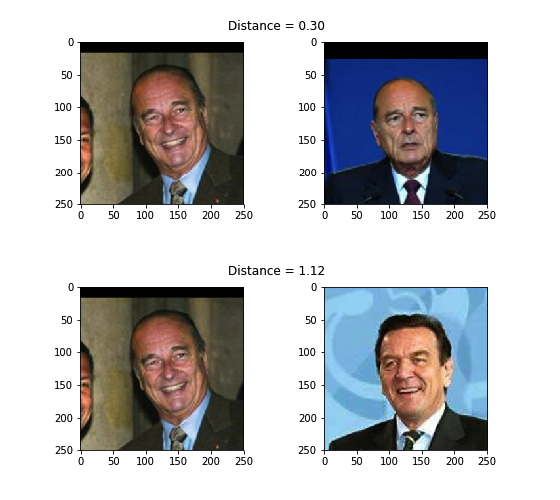
The whole process for face recognition using Keras can be divided in four major steps: a)Detect/ Identify faces in an RGB image using Dlib and OpenCV. b) Crop RGB image into 200X200 pixels. c)Design convolutional neural network using Keras. d) Train the model on the test model on testing data .

We have used ConvNets in the project, which derive their name from the “convolution” operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. It preserves the spatial relationship between pixels by learning image features using small squares of input data. The nn4.small2.v1 model used was trained with aligned face images, therefore, the face images from the custom dataset must be aligned too. We aligned the face images using Dlib for object detection and OpenCV for image transformation and cropped to produce aligned 96x96 RGB images. After the alignment of faces, embedding vectors were created and used as a database for classifying the unknown inputs later during classifications. The alignment process is shown in Figure 9 below.



**Figure 9 Face alignment for celebrity Recognition**

As similarity metric, we used L2 Squared Distance with a good enough threshold set to 0.56. The results for the process are shown in figure 10.



**Figure 10 similarity metric L2 squared distance**

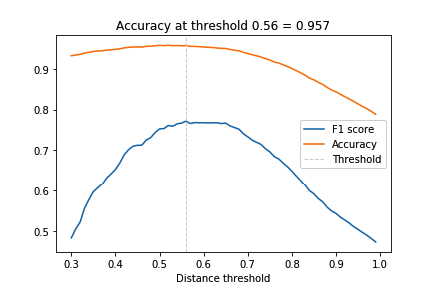
For the COCO multiple object detection since the COCO dataset is already pre-processed, therefore no need for pre-processing. The dataset images or videos will be given as input to YOLO network that has been trained on COCO dataset. The videos for input will be taken from the /videos folder in the project directory. The output video with the bounding boxes with labels will be stored in the folder /output in the project directory.

The research paper for YOLO was used as an extension to the third part of our project.

**3.4** **Evaluation or Performance**

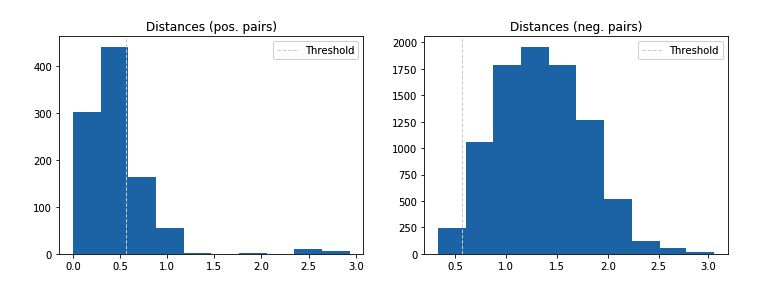
To find the optimal value for τ (distance threshold), the face verification performance needs to be evaluated on a range of distance threshold values. At a given threshold, all possible embedding vector pairs are classified as either *same identity* or *different identity* and compared to the ground truth. Since we’re dealing with skewed classes (much more negative pairs than positive pairs), we use the ***F1 score as evaluation metric instead of accuracy***.

The results are shown in the figure 11.



**Figure 11 Selected distance threshold**

The face verification accuracy at τ = 0.56 is 95.7% which is not bad given a baseline of 89% for a classifier that always predicts *different identity* but since nn4.small2.v1 is a relatively small model it is still less than what can be achieved by state-of-the-art models (> 99%). The result is visualized in figure 12.



**Figure 12 Value pairs and the selected threshold**

Now if we have an estimate of the distance threshold τ, face recognition is as simple as calculating the distances between an input embedding vector and all embedding vectors in a database. The input is assigned the label (i.e. identity) of the database entry with the smallest distance if it is less than τ or label unknown otherwise. It also supports one-shot learning, as adding only a single entry of a new identity might be sufficient to recognize new examples of that identity.

But we have used a more robust approach for labelling

the input using KNN classification with a Euclidean distance metric and a linear support vector machine (SVM) trained with the database entries and used to classify i.e. identify new inputs.

Note:- For training these classifiers we use 50% of the dataset, for evaluation the other 50%.

 The accuracy for celebrity recognition using 1. K-NN 2. Support Vector Machines (SVM) is 96% and 98% respectively.

To embed the dataset into 2D space for displaying identity clusters, t-distributed Stochastic Neighbor Embedding (t-SNE) is applied to the 128-dimensional embedding vectors. Except from a few outliers, identity clusters are well separated.

**3.5** **Contributions**

The contributions to the project involve,

1. Implementation of K-NN and SVM for Accuracy over classifiers.
2. Using L2 Squared distance metric as the similarity measure.
3. Performed experiments for deciding over a single distance threshold value.
4. Increased the accuracy from 85% to 95.6% for celebrity recognition by using a pre-trained network nn4.small2.v1.
5. Compared the performance of the classifiers using K-NN and SVM.

**4 Project Analysis**

*“What did I do well?”:* The great progress of automatic face recognition has made large-scale face identification possible for many practical applications . This application is widely used when the images for the persons to be recognized are available beforehand, and an accurate recognizer is needed for a large and relatively fixed group of people. For example, most of the face recognition application is used for search engine , recognition for public figure in media industry, and video streaming companies for movie character annotation.

*“What could I have done better?“:* We could implement entertainment app that identifies celebrity lookalikes or an app that identifies celebrities as part of automated footage tagging. The celebrity recognition is designed to be exclusively used in cases where you expect there may be a known celebrity in an image or a video. The celebrity recognition API would return the closest found matches, along with a similarity score.

*“What is left for future work?”:* We can merge all the sections of our project into one and implement a more diversed face recognition system. The celebrity recognition system that integrates face-based recognition and voice-based verification modules at the decision level. Our face-based celebrity recognition system can recognize hundreds of thousands of celebrity faces at this point by exploiting the tremendous depth of the internet with the consistency learning framework. However, unimodal recognition based solely on faces is hampered when the image quality is poor and/or when facial details are blurred due to motion or occlusion. The proposed method is a logical extension of our existing face recognition system to exploit the biometric characteristics of the voice modality.

**5** **Conclusions**

Our Celebrity face recognition project was successfully completed but still can be much extended to Realtime videos or increasing the dataset or celebrity faces in the classifier. It will not be difficult to develop further artistic and technical ideas and make steps in the further development of the project celebrity recognition, showing possibilities, drawing attention to the project, etc. This will also be possible since it can profit from all kinds of research on multi-user and multi- agent environments, distributed artificial intelligence and virtual humans.

It will however be clear that a full realization of the Celebrity recognition project, in particular the real, many celebrity faces supported and all of its contents, requires finances that go far beyond the possibilities of a research group. We hope that this paper will at least help to draw attention to this project and will receive feedback that can be useful to bring us closer to our goals.

**References**

[Person Finder]

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=2ahUKEwjXi\_L1jbvnAhVGIqwKHTJlDNQQFjABegQIAhAB&url=https%3A%2F%2Farxiv.org%2Fpdf%2F1302.6379&usg=AOvVaw3T9G28b-P1ECXauRR-dskp

[https://github.com/canxkoz/Multiple-Face-Recognition]

[https://medium.com/@manivannan\_data/how-to-train-multiple-objects-in-yolov2-using-your-own-dataset-2b4fee898f17]

[https://github.com/experiencor/keras-yolo2/blob/master/Yolo%20Step-by-Step.ipynb]

[https://www.tensorflow.org/tutorials/images/cnn]

[YOLO]

https://pjreddie.com/media/files/papers/YOLOv3.pdf

[COCO] https://arxiv.org/pdf/1405.0312