Indian Institute of Technology Kanpur

Kalyanpur

Kanpur -208 016

Final Project Report on Machine Learning And Data Science

Topic:

CNNs for Face Detection and Recognition

Submitted By:

Name: V. Sai Nikhil (16MIS0257)

College: Vellore Institute of Technology – Vellore

E-mail: v.sainikhil2016@vitstudent.ac.in

Abstract of Project:

We have many face detection methods for detecting the face of human. It is used in many industries, classrooms, offices and other places for security reasons. Usually this surveillance will detect human faces. In our project we do both detection of human face and also recognize the persons face.

This face detection project tackle the main problem of working in real world environment since it has high rate of accuracy and run in real time system. We store the persons image in the dataset and use the system to train these images and detect and recognize the human face. Not only this detect the single face in the frame, but also multiple faces in the image frame.

Methodology:

In Implementation, we have developed 3 parts of the Real-time object detection and recognition,

Celebrity Face Recognition in real-time videos

- Input will be a video for a celebrity
- The network (CNN) is trained with this celebrity
- Output will be the video with bounding boxes and labels around the celebrity images.

COCO real-time Multiple object detection using YOLO

- Input will be an image or a video of common objects (COCO)
- YOLO is trained with the COCO dataset.
- Output will be the image or video with bounding boxes and labels around multiple objects detected.

Realtime Single object detection of input from webcam

- Input will be an image, or a video of an object
- Object will be captured using webcam by drawing bounding boxes
- The network YOLO would have been trained with that in the process.
- Output will be the image or video with confidence levels.

Celebrity Face Recognition Implementation

The celebrity Face Recognition is implemented using Python libraries Keras, Dlib and OpenCV.

Steps followed

The whole process for face recognition using Keras can be divided in four major steps:

1. Detect/ Identify faces in an image using Dlib and OpenCV
2. Convert image into grayscale and crop into 200X200 pixels
3. Design convolutional neural network using Keras
4. Train the model on the test model on testing dat

ConvNets 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.

Why Convolutional Neural Networks?

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification.

There are four main operations in the ConvNet:

1. Convolution
2. Non-Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

These operations are the basic building blocks of every Convolutional Neural Network.

Some of the important concepts are underlined,

What is an Image?

An Image is a matrix of pixel values. Essentially, every image can be represented as a matrix of pixel values.

What is a Channel?

Channel is a conventional term used to refer to a certain component of an image.

An image from a standard digital camera will have three channels – red, green and blue – one can imagine those as three 2d-matrices stacked over each other (one for each color), each having pixel values in the range 0 to 255.

Why we are using the grayscale image?

A grayscale image, on the other hand, has just one channel.

We will only consider grayscale images, so we will have a single 2d matrix representing an image. The value of each pixel in the matrix will range from 0 to 255 – zero indicating black and 255 indicating white.

Keras

Keras is a high-level library, used specially for building neural network models. It is written in Python and is compatible with both Python – 2.7 & 3.5.

Keras was specifically developed for fast execution of ideas. It has a simple and highly modular interface, which makes it easier to create even complex neural network models. This library abstracts low level libraries, namely Theano and TensorFlow so that, the user is free from "implementation details" of these libraries.

VGG-Face Model

We have used the VGG-Face model. Briefly, the VGG-Face model is the same Neural Net architecture as the VGG16 model used to identity 1000 classes of object in the ImageNet competition.

The VGG16 name simply states the model originated from the Visual Geometry Group and that it was 16 trainable layers.

The main difference between the VGG16-ImageNet and VGG-Face model is the set of calibrated weights as the training sets were different.

The model architecture is a linear sequence of layer transformations of the following types:

- Convolution + ReLU activations
- MaxPooling

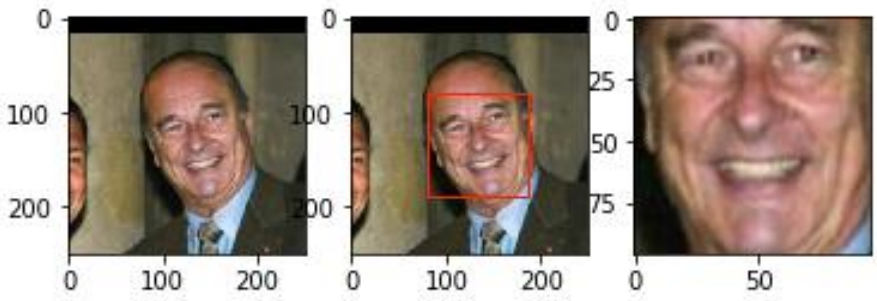
- SoftMax

Nn4.small2.v1 Model – Face Alignment

The nn4.small2.v1 model was trained with aligned face images, therefore, the face images from the custom dataset must be aligned too.

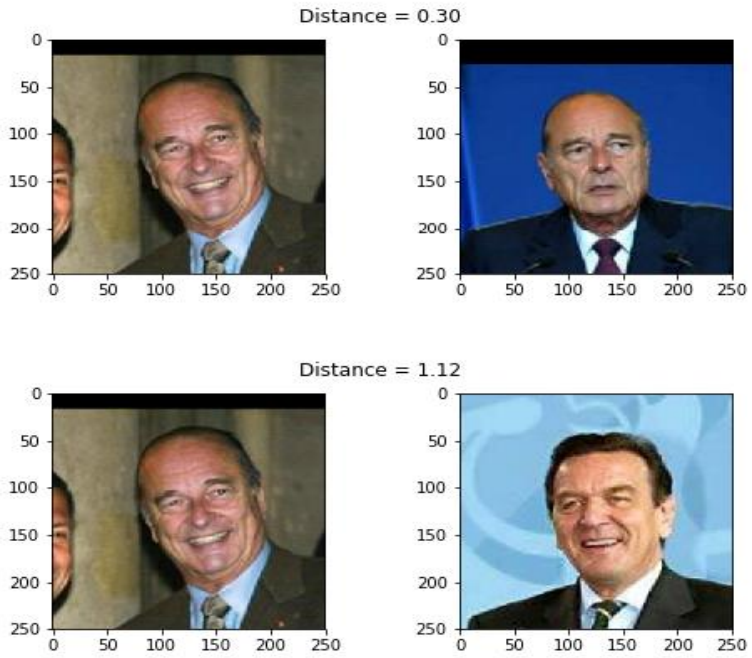
Here, we have used Dlib for face detection and OpenCV for image transformation and cropping to produce aligned 96x96 RGB face images.

That is,



Loss – Squared L2 Distance

We calculate the squared L2 distance and get the result as,

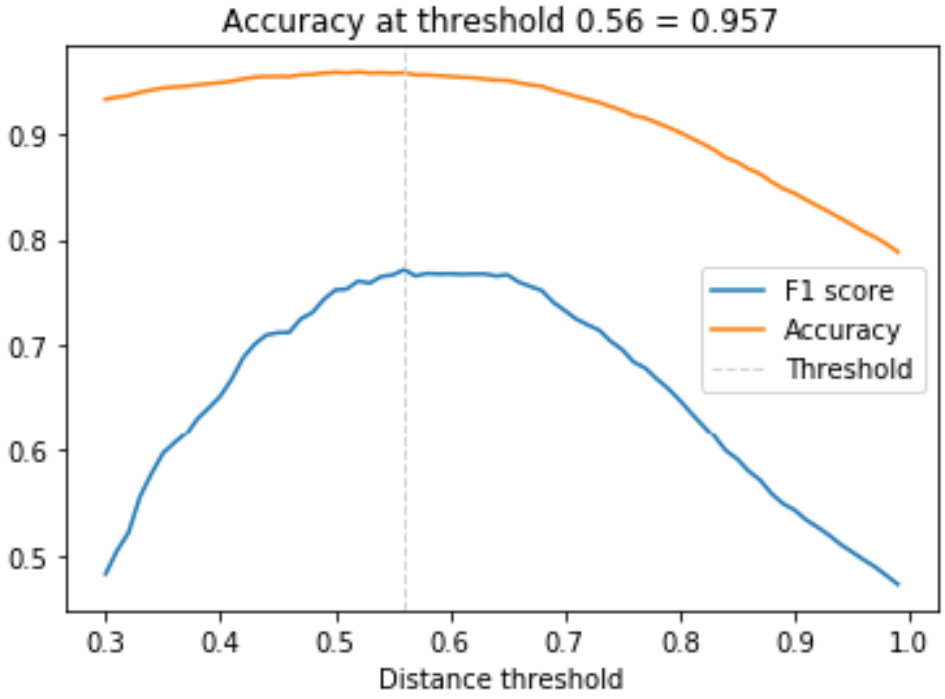


Here, the distance between the two images of Jacques Chirac is smaller than the distance between an image of Jacques Chirac and an image of Gerhard Schröder ($0.30 < 1.12$). But we still do not know what distance threshold τ is the best boundary for making a decision between same identity and different identity.

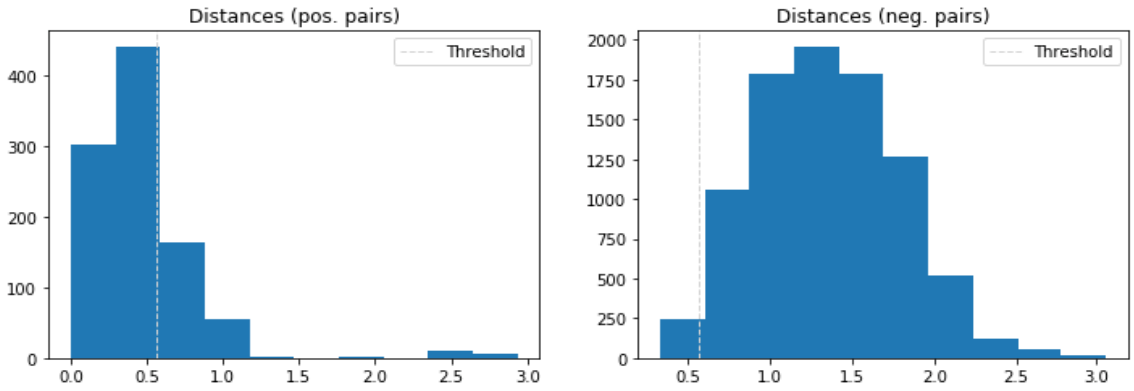
What should be the Distance Threshold?

To find the optimal value for τ, 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 face verification accuracy at $\tau = 0.56$ 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%).



Face Recognition – Measuring the Accuracy

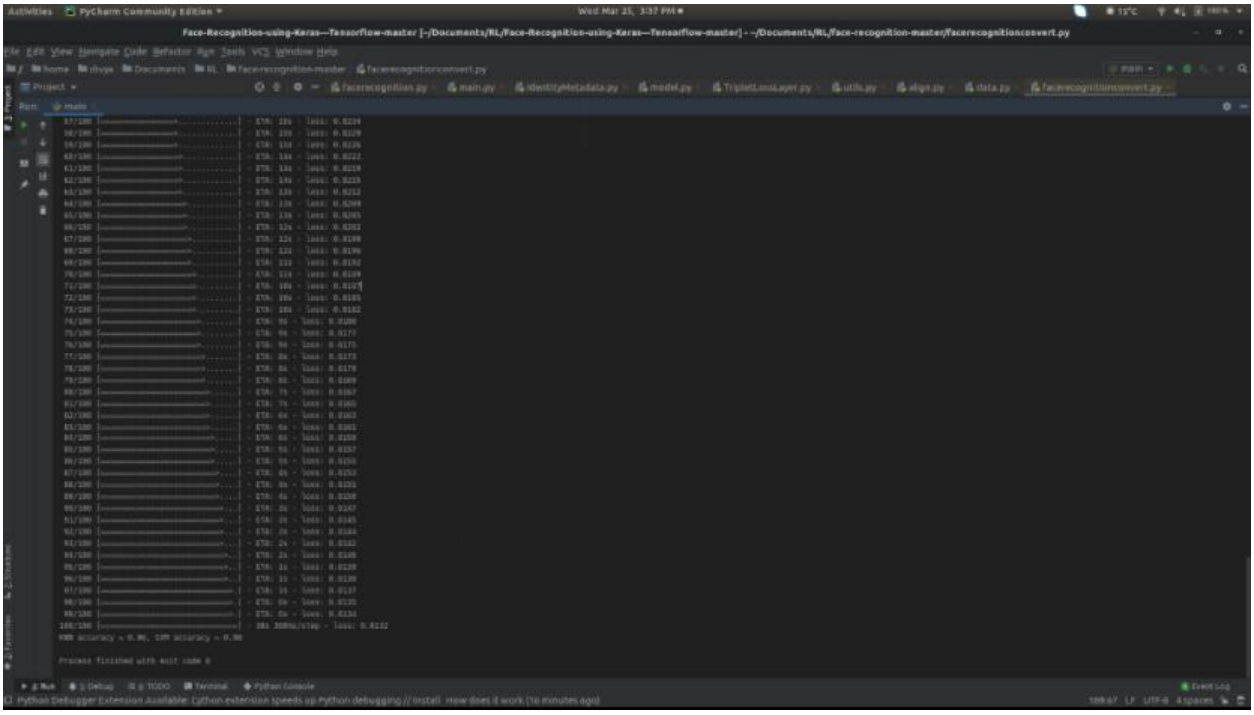
Now if we have an estimate of the distance threshold τ, face recognition is as simple as calculating the distances between an input embedding vector and all embedding vectors in a database.

- 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%.

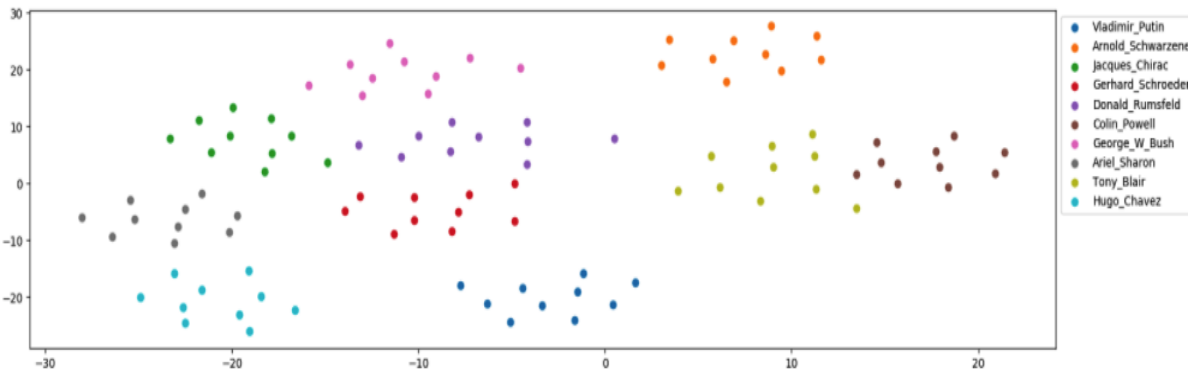
The accuracy for celebrity recognition using 1. K-NN 2. Support Vector Machines (SVM) is 96% and 98% respectively, as shown in the screenshot below



Dataset Visualization

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.



COCO Realtime Multiple Object Detection using YOLO

The steps followed for implementation:

- 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.

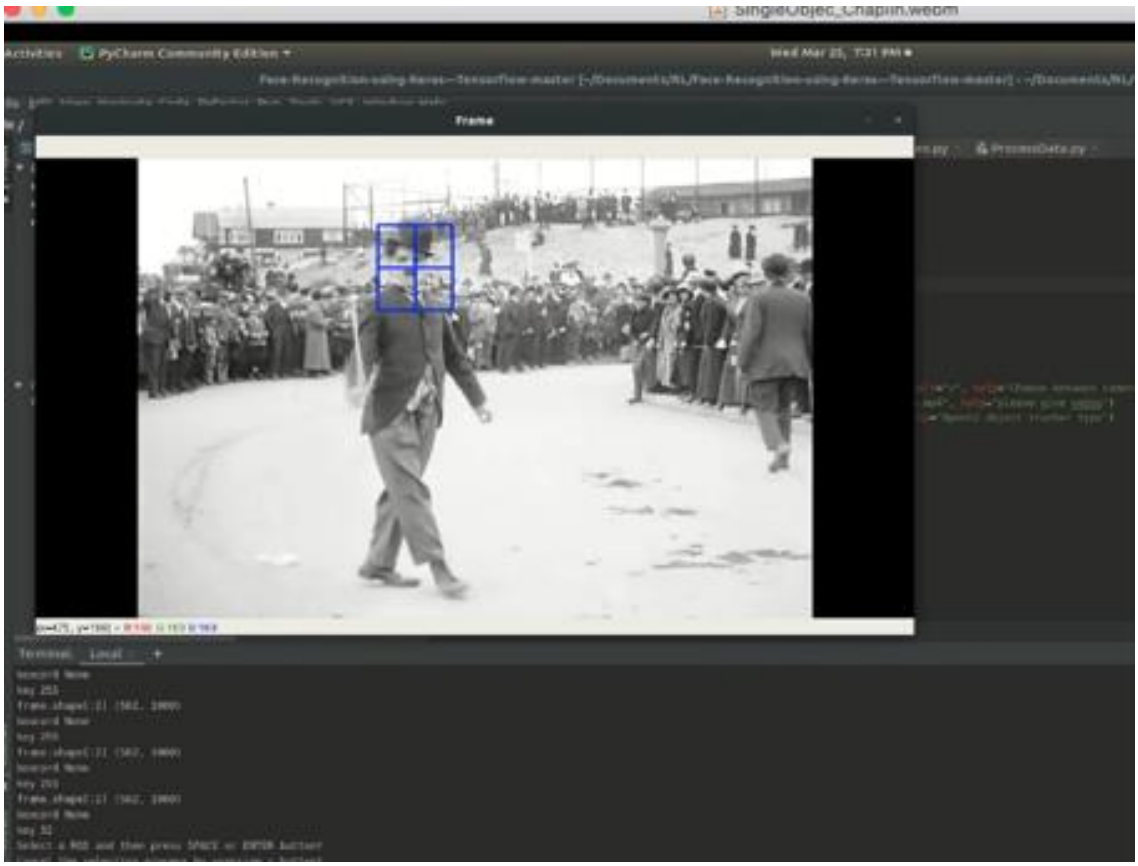
5 Realtime Single Object Detection

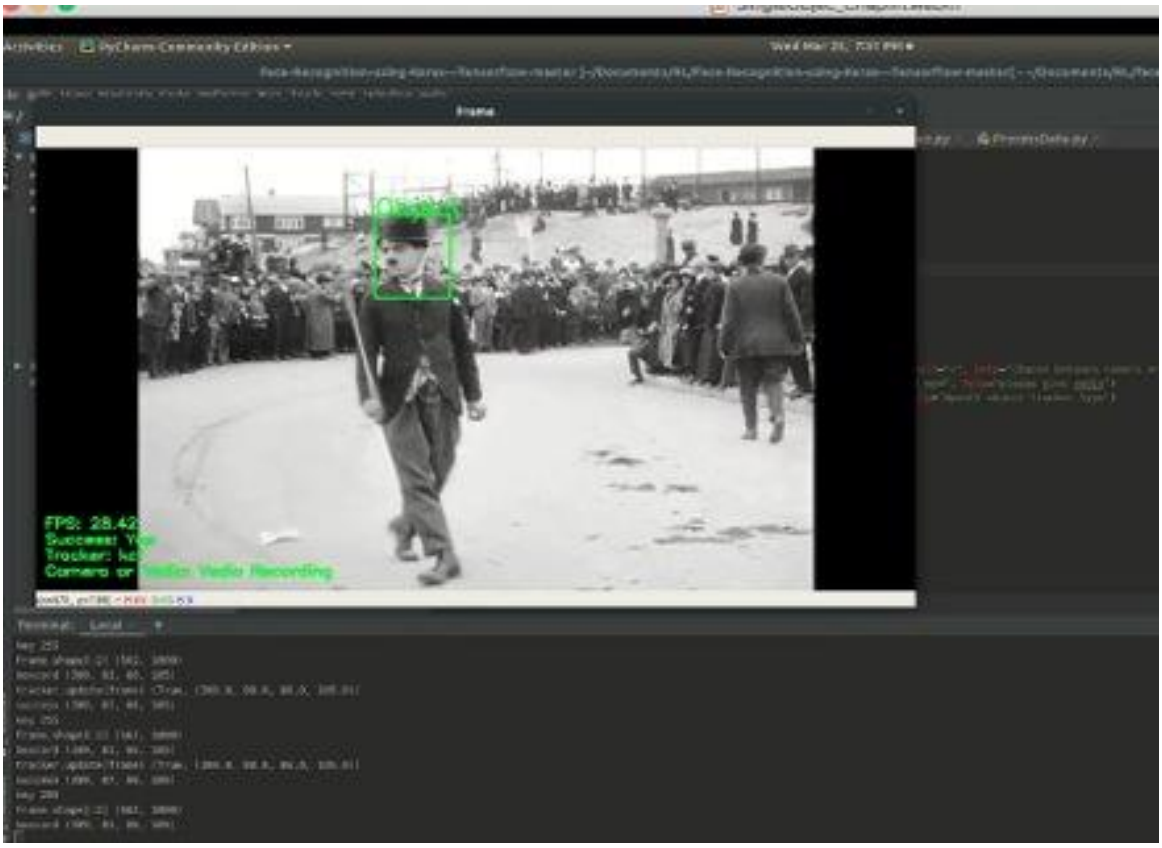
The steps followed for implementation:

- The input will be captured by drawing the bounding boxes around the object to detect in real-time captured by the webcam.
- The input will be the video or webcam (camera) whichever argument we supply at runtime.
- The videos for input will be taken from the /videos folder in the project directory.

We can also change the trackers by supplying the relevant tracker IDs as outlined below,

- csrt": cv2.TrackerCSRT_create
- kcf": cv2.TrackerKCF_create
- boosting": cv2.TrackerBoosting_create
- mil": cv2.TrackerMIL_create,
- tld": cv2.TrackerTLD_create
- medianflow": cv2.TrackerMedianFlow_create
- mosse": cv2.TrackerMOSSE_create




Frame
FPS: 28.42
Success: Yes
Tracker: kcf
Camera or Video Video Recording