Attendance System using Face Recognition

CS337 (Artificial Intelligence & Machine Learning)

Aaryan Gupta
Sarthak Mittal
Shikhar Mundra
Utkarsh Ranjan

November 24, 2022

Contents

1 Introduction

"Deep learning advancements in recent years have enabled widespread use of face recognition technology. This project tries to exploit deep learning models used for face recognition and introduces a simple framework for creating and using a custom face recognition attendance system."

This project is based on building an automated attendance system using face recognition. We make use of multiple models chained together to achieve this. Our framework consists of a pre-trained deep neural network Caffe model (ref. [6]) for face detection within an image, deployed using OpenCV DNN configuration (ref. [5]), followed by alignment using Haar Cascade Eye Detector (ref. [8]). This is followed by use of Inception-ResNet v1 architecture (ref. [4]) combined with Triplet Loss (ref. [10]). We conclude by checking classification using Softmax.

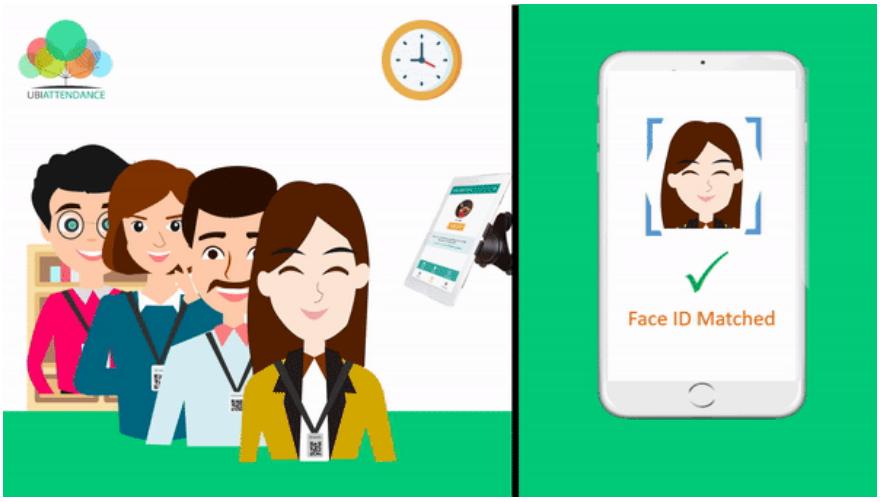
1.1 Overview

Face recognition can be divided into multiple steps, which can be seen in the pipeline below:

Face recognition pipeline

We perform face detection using DNN (Caffe Model), alignment using Cascade Classifier, feature extraction using Inception-ResNet and classification using Softmax Classifier. We thank Luka Dulcic at Ars Futura (ref. [1]) for his article.

1.2 Sample Input



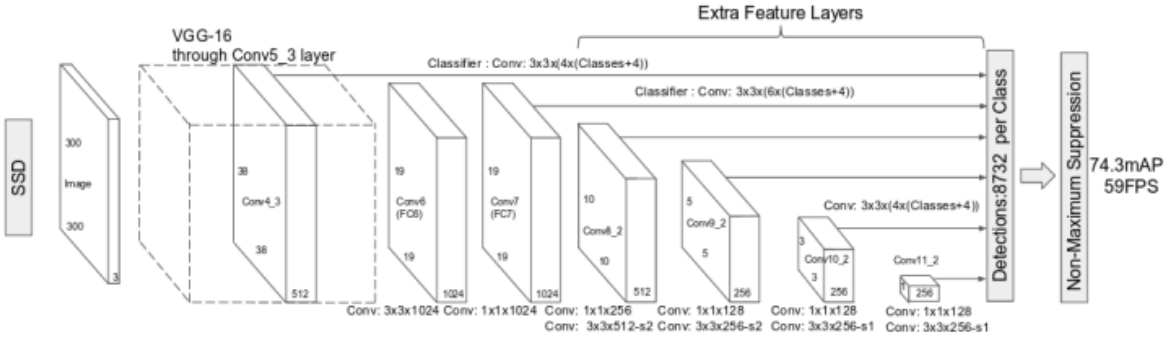
A sample marking of automated attendance

2 Model

We now analyse the steps in the pipeline sequentially. We thank Caffe (ref. [3]), the paper by Viola and Jones (ref. [9]), the article by Nana Kwame Kankam (ref. [4]), the triplet loss Wikipedia page (ref. [10]) and the blog by Luka Dulcic (ref. [1]).

2.1 Detection using DNN Caffe Model

Face detection is an early stage of a face recognition pipeline. It plays a pivotal role in pipelines. Herein, deep learning based approach handles it more accurate and faster than traditional methods. We make use of an external pre-trained ResNet model by OpenCV. The model structure is as shown below:



ResNet SSD

The network defines the entire model bottom-to-top from input data to loss. As data and derivatives flow through the network in the forward and backward passes, Caffe stores the information as blobs: the blob is the standard array and unified memory interface for the framework.

The conventional blob dimensions for batches of image data are number (N) $\times$ channel (K) $\times$ height (H) $\times$ width (W). The layer comes next as the foundation of both model and computation. The net follows as the collection and connection of layers. The details of blob describe how information is stored and communicated in and across layers and nets.

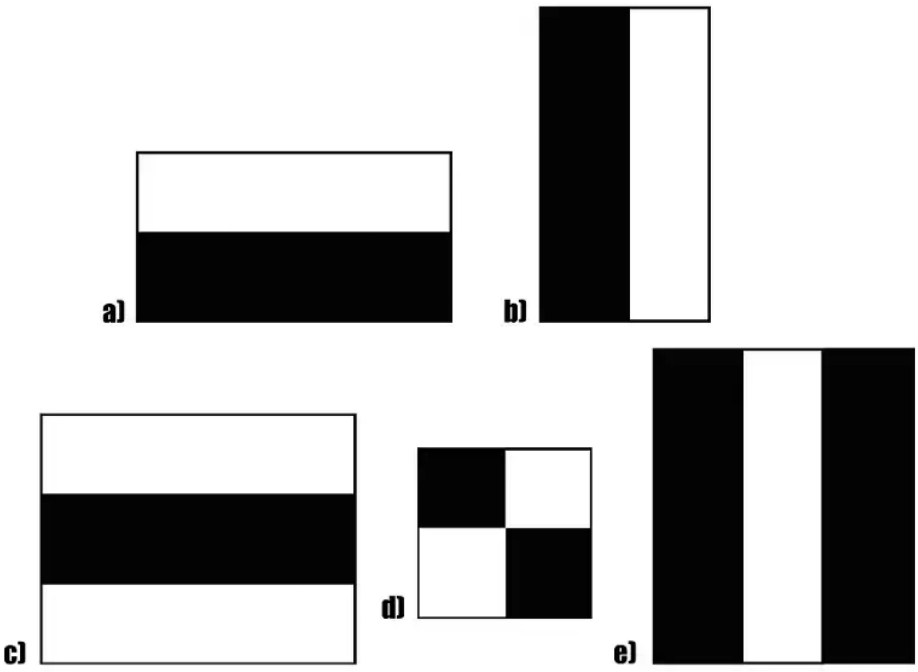
The layer is the essence of a model and the fundamental unit of computation. Layers convolve filters, pool, take inner products, apply non-linearities like rectified-linear and sigmoid and other element-wise transformations, normalize, load data, and compute losses like softmax and hinge.

2.2 Alignment using Haar Cascade Classifier

Haar cascade classifier, is a rapid machine learning object detection program that identifies objects in media.

The first is the introduction of a new image representation called the "Integral Image" which allows the features used by our detector to be computed very quickly. It is a method for combining increasingly more complex classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. It has a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. There are other haar features as well, which will detect edges in other directions and any other image structures.

To detect an edge anywhere in the image, the haar feature needs to traverse the whole image. The haar feature continuously traverses from the top left of the image to the bottom right to search for the particular feature. This is just a representation of the whole concept of the haar feature traversal.



A sample of Haar features used in the Original Research Paper published by Viola and Jones

To summarise, the method consists of primarily three steps:

- new image representation called an integral image that allows for very fast feature evaluation
- method for constructing a classifier by selecting a small number of important features using AdaBoost
- method for combining successively more complex classifiers in a cascade structure which dramatically increases the speed of the detector by focusing attention on promising regions of the image

2.2.1 Generating Features

The object detection model uses some simple features to classify instead of pixels. We use three kind of features:

1. two-rectangle feature, the difference between the sum of the pixels within two rectangular regions
2. three-rectangle feature, difference of sum in center rectangle and sum within two outside rectangles
3. four-rectangle feature, the difference between diagonal pairs of rectangles

We define a quantity named the integral image (ii) as

$$ii(x, y) = \sum_{x', y'} i(x', y')$$

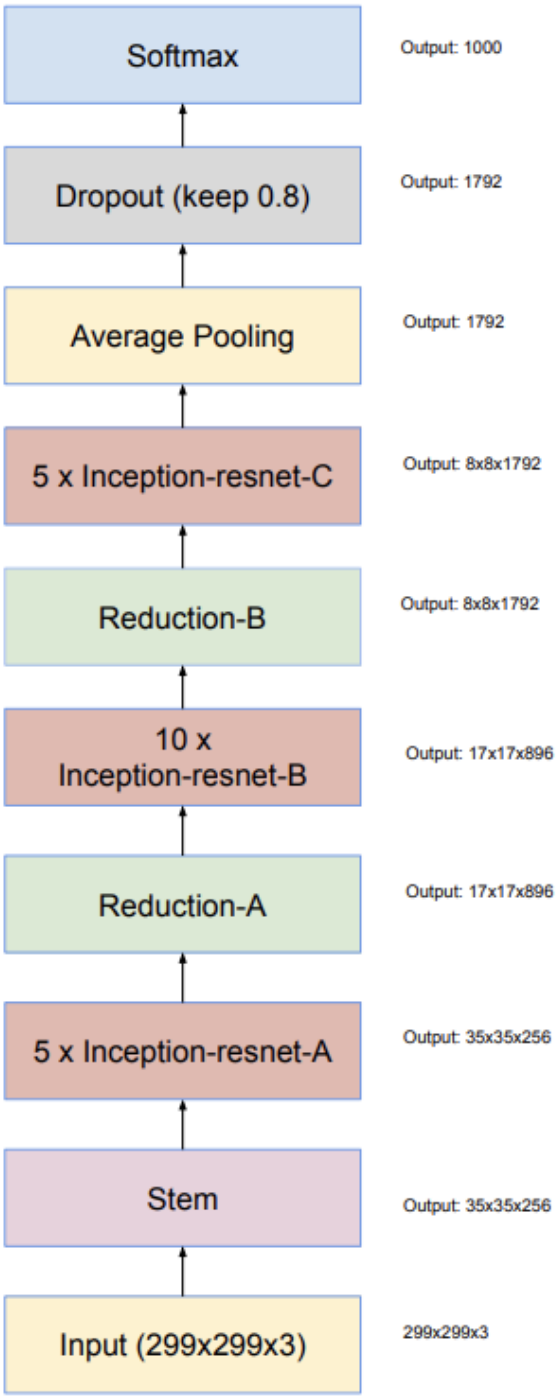
where i represents the pixel function. We use the integral image to calculate the features.

2.2.2 Learning Feature Classification (AdaBoost)

Given the test set, the positives and the negatives, we use the AdaBoost learning algorithm to classify images. These images are classifies based on the features calculated above. However, calculating all features is very inefficient and we approximate that only a few of these features are needed to classify.

2.3 Feature Extraction using Inception-ResNet

The Inception-ResNet network is a hybrid network inspired both by inception and the performance of resnet. The key functionality of the Inception-ResNet is that the output of the inception module is added to the input (i.e.data from previous layer).



Schematic for Inception-ResNet v1

2.3.1 Generating Embeddings

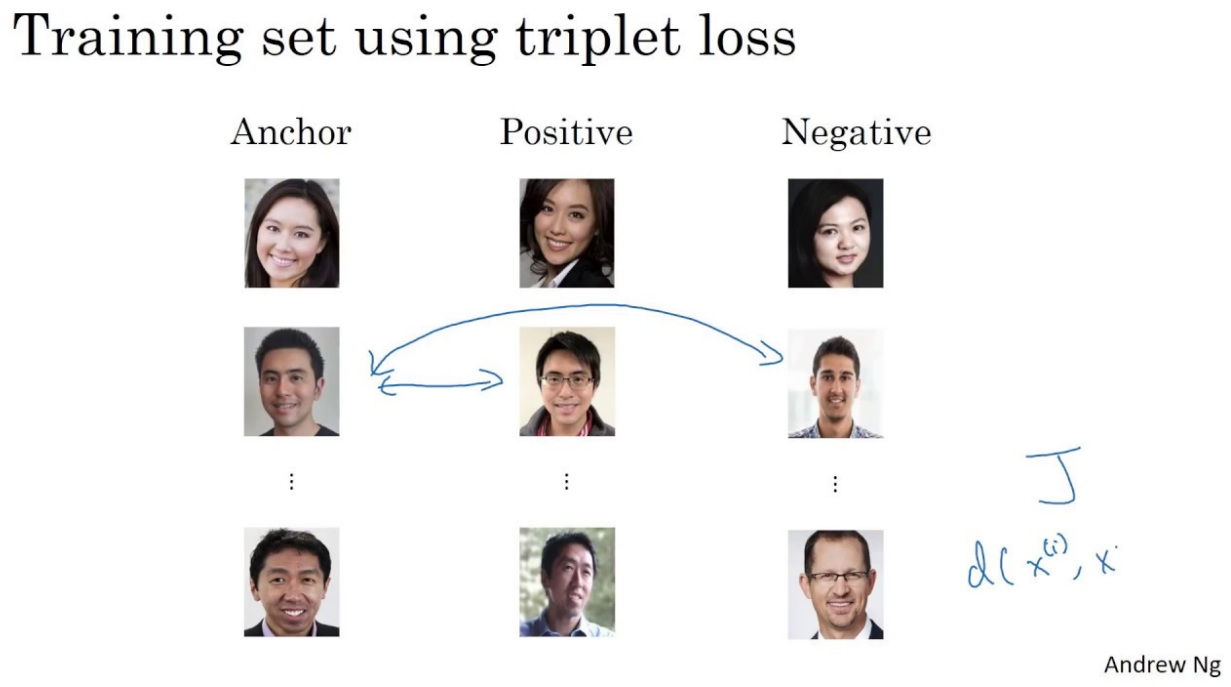
Just like how FaceNet (ref. [7]) takes an image of the person's face as input and outputs a vector of 128 numbers which represent the most important features of a face (called an embedding), Inception-ResNet also creates embeddings. All the important information from an image is embedded into this vector. Ideally, embeddings of similar faces are also similar.

2.3.2 Learning Embedding Classification (Triplet Loss)

$$\sum_{i=1}^{N} \left[(f_i^a - f_i^p)^2 - (f_i^a - f_i^n)^2 + N \right]$$

Triplet loss formula

Triplet loss is a loss function for machine learning algorithms where a reference input (called anchor) is compared to a matching input (called positive) and a non-matching input (called negative). Triplet loss works directly on embedded distance, embedding in a way that a pair of samples with same labels are smaller in distance than those with different labels.



Training with triplet loss

We pose the problem as a similarity learning problem instead of a classification problem. Here the network is trained to output a distance which is small if the image belongs to a known person and large if the image belongs to an unknown person. However, if we want to output the closest images to a given image, we would like to learn a ranking and not just a similarity.

2.4 MTCNN

Multi-task Cascaded Convolutional Networks (MTCNN) are a library implementation of the paper by Zhang. It is used to to detect faces in an image and returns the rectangle around a face. It consists of three stages:

1. P-Net (Proposal Network) - This is a fully convolutional network and is used to obtain the candidate window boxes and their bounded box regression vectors.
2. R-Net (Refine Network) - This takes in the candidate windows from P-Net and reduces the number of candidates and calibrate. This is a CNN and not an FCN as it uses a dense layer at the end and outputs whether a candidate is a face or not and the bounded box for the face.
3. O-Net (Output Network) - Similar to R-Net but outputs facial features like nose, eyes, and mouth.

MTCNN tries to achieve three tasks:

1. Face Classification - a binary classification problem that uses cross entropy loss.
2. Bounding Box Regression - a regression problem that uses the euclidean distance between output and ground truth as the loss function.
3. Facial Landmark localisation - same as 2 but for facial features like eyes, nose, mouth.

2.5 Face Classification using Softmax

Softmax classifier is used as a final step to classify a person based on a face embedding. Softmax was a logical choice since the entire stack is neural networks based, but if the face embeddings themselves are good, other classifiers such as KNN, SVM, Random Forest, etc. should also perform well at this step (ref. [1]).

3 Conclusion

The combined model functions like an attendance system, and is able to classify images of each person in the validation data into cluster of images of the same person as per the training data. The data that we used, a subset of the LFW Face Dataset (ref. [2]), can be found here in the project repository. We pre-processed the input images (using Face Detection and Eye Alignment) and then trained the Softmax Classification model on the training data along with the Inception-ResNet model (using Triplet Loss). We achieved around **98**% accuracy in training, and $\mathbf{76-79}\%$ accuracy in validation.

References

[1] L. Dulcic. Face recognition with facenet and mtcnn, 2019. URL `https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/`.

[2] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[3] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[4] N. K. Kankam. Understanding inception-resnet v1. URL `https://iq.opengenus.org/inception-resnet-v1/`.

[5] OpenCV. OpenCV DNN Face Detector deployment definition, 2019. URL `https://github.com/opencv/opencv/tree/3.4/samples/dnn/face_detector`.

[6] A. Rybnikov. OpenCV DNN Face Detector caffe model, 2017. URL `https://github.com/opencv/opencv_3rdparty/tree/dnn_samples_face_detector_20170830`.

[7] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015. URL `http://arxiv.org/abs/1503.03832`.

[8] O. D. Suarez. OpenCV Haar Cascade eye detector, 2010. URL `https://github.com/kipr/opencv/tree/master/data/haarcascades`.

[9] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001. ISBN 0-7695-1272-0. doi: 10.1109/CVPR.2001.990517.

[10] Wikipedia contributors. Triplet loss — Wikipedia, the free encyclopedia, 2022. URL `https://en.wikipedia.org/w/index.php?title=Triplet_loss&oldid=1096016823`. [Online; accessed 24-November-2022].