Face Recognition in Videos

RAWAL KHIRODKAR - 130050014

SUPERVISOR: PROF. GANESH RAMAKRISHNAN

UNDERGRADUATE THESIS

INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

ABSTRACT

The project focuses on applying state-of-the-art face recognition methods to a specific setting of classroom surveillance footage. The goal is to monitor proceedings in the classroom such as quality of instruction imparted by teachers, student attendance, teacher identity verification. We look into state-of-the-art methods based on deep learning for tasks like face detection and face recognition which are highly customised to work for our particular kind of data-set. An *LSTM* based approach for people detection in frames is discussed which forms the baseline for face recognition module of this project. Two famous deep learning models *FaceNet* and *VGG-Face* are used as face recognition methods on our dataset.

ACKNOWLEDGEMENTS

I am greatly indebted to my supervisor Prof. Ganesh Ramakrishnan for his excellent guidance and support throughout the project. I would also like to express my gratitude towards Vishal Kaushal (PhD student, CSE IIT Bombay) and Raju (Final year MTech student, CSE IIT Bombay) for their support. My colleagues from IITB Video Analytics group helped me in data collection and have been very encouraging and supportive throughout the project. This was a great learning experience overall and I sincerely hope to carry on the work with this team in future.

Contents

1. Introduction

1.1 Overview

The objective of this project is to provide **Ministry of Rural Development (MoRD)** with a user-friendly video analytics platform to analyse the classroom surveillance footage collected from class rooms held under the rural development initiative. The kind of assessment which this system should provide falls under majorly two categories.

- **Quantitative Assessment**: This paradigm involves extracting straightforward quantitative information from the videos. Specifically, we aim to solve three problems falling under this problem.
  - Classroom Attendance Estimation: Estimates the number of students attending the classes.
  - Teaching Duration Estimation: Estimates the time for which teacher actually taught in the classroom.
  - Trainer Identity Verification: Verifies whether the person instructing the class is actually the teacher on role.
- **Qualitative Assessment**: This paradigm involves assessment of quality of instruction imparted during teaching. For now we aim to solve two problems belonging to this category.
  - Measuring interaction between teacher and students: Quantify the level of interaction between the teacher and the students during the class.
  - Student attentiveness profiling: Provides statistical data which correlates with the attentiveness shown by a student in the classroom.

Initial phase of our project majorly focuses on providing solutions to all problems falling under the Quantitative Assessment category. This project specifically provides a solution for "Trainer Identity Verification" and later extends the solution to "Teaching Duration estimation".

Figure 1.1: Camera Angle: Trainer Facing

Figure 1.2: Camera Angle: Trainer Facing

Figure 1.3: Camera Angle: Student Facing

Figure 1.4: Camera Angle: Student Facing

1.2 MoRD Data

The data-set consists of footage from cameras installed in classrooms in the training center of Ministry of Rural Development (MoRD). The teaching job at these training centers are outsourced to private organisation who are paid according to number of students attending the classes. This has given rise to malpractices like forging the attendance count, poor-quality of training to students. Thus, our system would act as an automated monitor to avoid such misuse of resources.

The data set majorly consists of three types of camera angle:

- **Trainer facing**: Figure 1.1 and Figure 1.2 are examples of such classrooms. This camera angle is desirable for Trainer Identity Verification.
- **Student facing**: Figure 1.3 and Figure 1.4 are examples of such classrooms. This camera angle is desirable for Student Attendance Estimation.

Clearly frontal face exposure is necessary for face recognition. This is the recommended camera setting which is used to test our module.

1.3 Challenges

- These videos typically contain a lot of faces in each frame, thus, making the task of face recognition challenging.
- Temporary occlusion of faces occur in active classroom.
- The trainer (very understandably) moves and turns in the classroom. Thus, hindering the facial feature capture by the cameras.
- The camera orientation is not always ideal.

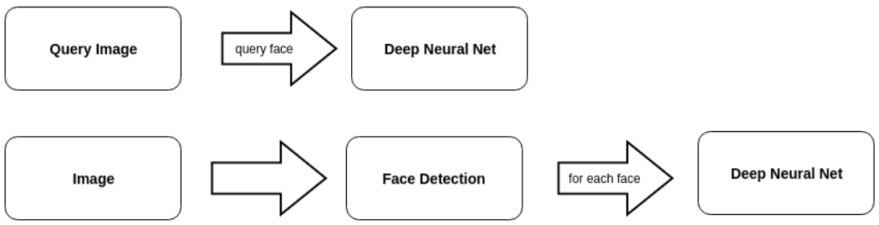

Figure 1.5: Algorithm

- The footages are pixelated due to use of low resolution cameras.

1.4 Approach

In this section we will formally discuss the problem and the approach adapted to solve it.

1.4.1 Input and Output

Let us first clearly specify input and output for this project.

- **Input**: Video and Query Image of the face of the trainer (or anything other person whose identity is to be verified).
- **Output**: Annotated video identifying the identity (if present) represented by the query image.

1.4.2 Algorithm

Deep Neural networks are used to embed an input face image to feature space of faces.

The algorithm is as follows:

- Extract an embedding from the query face image using the neural network
- For each frame in the video:
  - Detect faces in the frame.
  - For each detected face, extract an embedding from the neural network.
  - Thus we have mapped every face present in the frame along with the query image into a feature space.
  - Further, use nearest-neighbour search using any kind of similarity measure (we use Euclidean norm) between these faces and the query face.

Figure 1.5 represents the algorithm pictorially.

2. Face Detection

Face detection in videos is the first step towards our goal to achieve video summarisation. This boils down to doing object detection in an image where your object is a human face. Most common methods to do object detection are feature based where you look for regions in the image whose features closely resemble features of the object being detected.

We initially looked into widely used feature based techniques like Histogram of Oriented Gradient (HOG) and Haar-Cascades for face detection.

2.1 Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The method is credited to Navneet Dalal and Bill Triggs, researchers for the French National Institute for Research in Computer Science and Automation (INRIA) when they presented it in CVPR (2015).

2.1.1 Description

The method employs sliding window approach to detect faces in the image. Consider an image patch from the window, at each pixel in the image patch, the intensity gradient and its direction is computed. Now a histogram of gradients is built using gradient vectors at each pixel. The contribution of each gradient to the corresponding bin in the histogram is directly proportional to gradient's magnitude. After normalising the histogram, we get a feature vector extracted from the image patch whose dimensionality is equal to the number of bins in the histogram. Using these feature vectors, a binary classifier (most popularly Support Vector Machine) is trained to classify the image patch as containing a face or not.

2.1.2 Results

A pre-trained classifier from OpenCV was used to test the performance of HOG feature based face detection on our data set. Figure 2.2 shows the result on our dataset.

Figure 2.1: Frontal Face detection using HOG

The number of false positives are significant and all the detections are wrong. Clearly the results were not encouraging enough.

Following are the reasons behind the failure of HOG feature based face detector:

- The facial features are not clearly visible
- The training data on which the classifier is trained could be significantly different from our data.

Thus, training the model on our data set would a plausible way to fix the problem. But again this approach restricts the domain of videos (videos similar to training data) where the model is applicable. The inability of this approach to generalize to wide array of setting (intensity and pose variation) strengthened our decision to drop it.

2.2 Haar-Cascades

This machine learning based approach to do object detection uses a cascade function trained on a lot of positive and negative images. The method was proposed by Paul Viola and Michael Jones in their paper published in 2001. [2]

2.2.1 Description

Again, this method employs a sliding window approach where a binary classifier (trained and boosted) is used whether the face is present in the image patch represented by the window or not. The features (refer figure 2.2) used in the training of the classifier are called as the Haar features which are obtained by subtraction of sum of pixel values in rectangular patches in the image. The number of resulting features can be extremely large. Therefore, Adaptive Boosting (Adaboost) is used to select only relevant features.

A *weak classifier* is trained corresponding to each Haar feature and using adaptive boosting the final classifier (*strong classfier*) is a weighted sum of the selected (minimum error) weak classifier. To increase efficiency instead of applying all selected weak classifiers on a particular image patch at once, the classifiers are grouped together forming various stages. The initial stages consists of few weak classifiers whose each representing a very effective feature vector. The subsequent stages

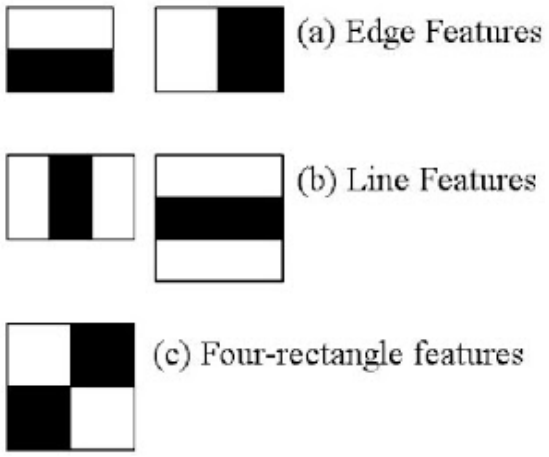

Figure 2.2: Haar features

Figure 2.3: Face detection using Haar-Cascades

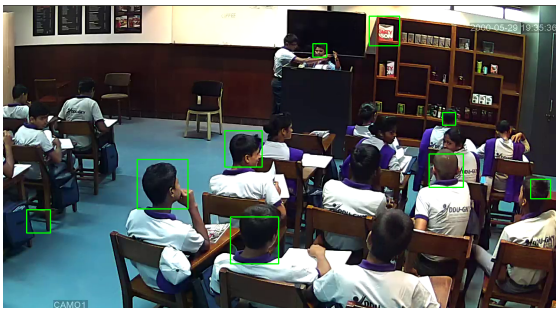

Figure 2.4: Head detection using Haar-Cascades

of the strong classifier are only applied when the image patch is not discarded by the initial stages. Thus the stages of the classifier are used in a cascaded manner.

2.2.2 Results

A pre-trained classifier from OpenCV was used to test the performance of Haar feature based face detection on our data set. Figure 2.3 shows the result on our dataset.The number of false positives were significantly less than HOG based detector. However, many faces (very understandably) were not detected due very less frontal face exposure.

Less frontal face exposure is a very major problem with our dataset. As a result we decided to shift to Head Detection instead of a Face Detection majorly because head detectors would be robust enough to detect low resolution (blurred) faces as a "head" is rather a simple object to detect than a "face". We compared HOG and Haar based head detectors, and again Haar-cascades seemed to do better than HOG based detectors. Figure 2.4 shows result of Haar-Cascade based Head Detector.The number of false positives increased, but this is not a major issue as face recognition would reject objects not similar to faces using a nearest neighbour search.

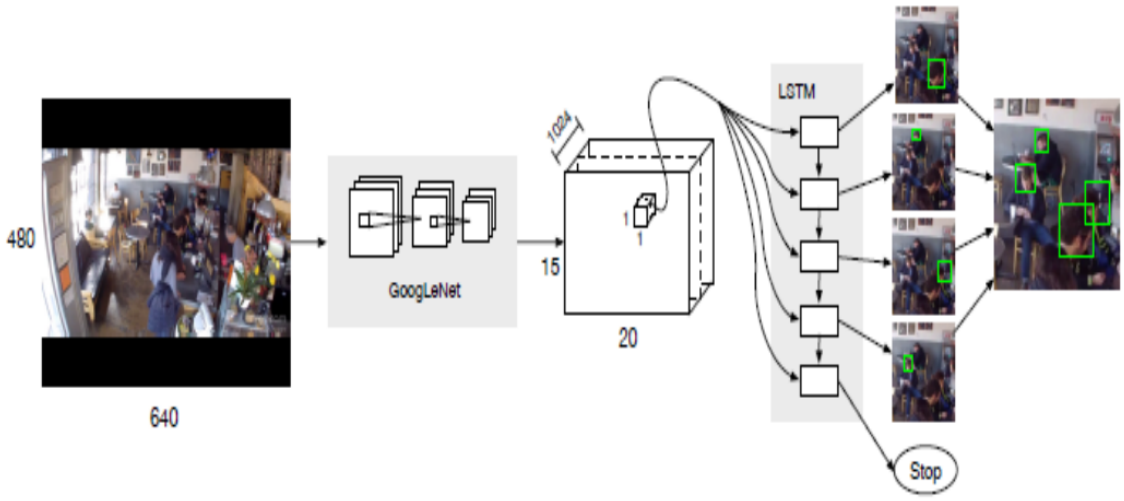

Figure 2.5: Long Short Term Memory network architecture

2.3 Long Short Term Memory Networks

Haar-cascades didn't prove to be effective on our dataset majorly because the videos are quite crowded and you have a lot of heads to detect. In search of an effective method which works in crowded scenes, we used the method proposed in "End-to-end people detection in crowded scenes" which uses Long Short Term Memory (LSTM) networks for detecting people.

2.3.1 Description

In this method, first the image is encoded into descriptors using a Convolutional Neural Network (CNN). This descriptors are further decoded into a bounding boxes corresponding to head detection in the image. A chain of LSTM units is used where each LSTM unit corresponds to one head detection (shown in figure 2.5). For more details about the model architecture and its result on our dataset please refer to my colleague Unmesh's undergradute thesis.

2.3.2 Results

Figure 2.6 and Figure 2.7 depict results of LSTM based head detector on our data-set. Clearly this approach performs better than our previous approaches. The model was able to detect nearly 60-70% heads in a frame. The number of detections can be increased using a lower confidence threshold for a valid detection. However, this can very likely result in increase in number of false positive. We did several experiments to tune this hyper-parameter (confidence threshold) to maintain a sweet spot between precision and recall. The project uses LSTM based people detection module to do face recognition.

Figure 2.6: Head detection using LSTM networks

Figure 2.7: Head detection using LSTM networks

3. Face Recognition

This section discusses about the the face recognition module which is also the last stage of the algorithm. Eigenface along with Principal Component Analysis (PCA) [**4**] is the standard approach used in computer vision for face recognition. However, recent approaches using Deep learning have performed better on benchmark datasets (eg. Labelled Faces in Wild [**5**], YouTube face database [**6**]) for face verification.Deep learning methods have also shown **Illumination and Pose invariance** which is a characteristic of our dataset.

We implemented two deep model based approaches (top ranked on LFW dataset) mentioned in the following papers on our dataset:

- "**FaceNet**: A Unified Embedding for Face Recognition and Clustering", Computer Vision and Pattern Recognition 2015 [**7**] - by Google
- "**VGG-Face**: Deep Face Recognition", British Machine Vision Conference, 2015 [**8**] - by Visual Geometry Group, Oxford

3.1 FaceNet

The method is based on learning an Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared *L2* distances in the embedding space directly correspond to face similarity:faces of the same person have small distances and faces of distinct people have large distances.

3.1.1 Model Structure

The training of the network consists of a batch of image given as input to the convolutional network. The deep network embeds the images into a feature space (i.e each image is now represented by a vector of real numbers). This embedding is further *L2* normalized and the resultant vector is used to determine the loss over the training data. The loss used by FaceNet is called as **Triplet Loss** (discussed in next section) which is used to do back-propagation on the convolutional network. This whole methodology is pictorially represented by Figure 3.1

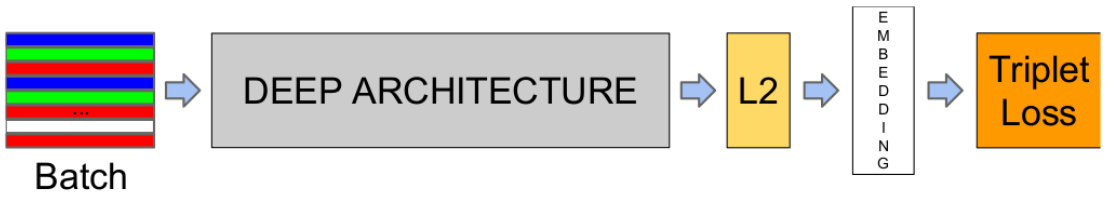

Figure 3.1: Pictorial Representation of FaceNet

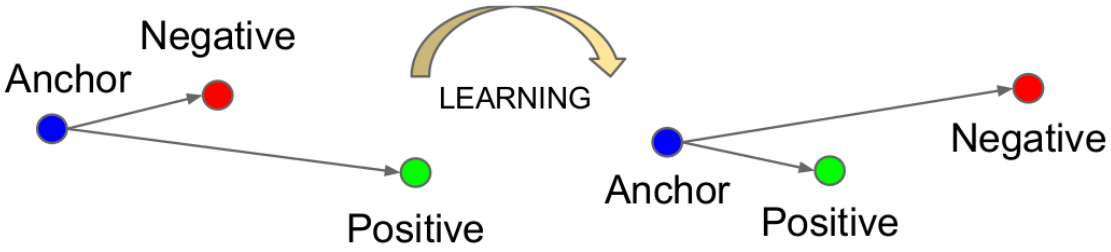

Figure 3.2: Minimize distance between *anchor* and *positive*, maximize distance between *anchor* and *negative*

3.1.2 Triplet Loss

The loss function used by FaceNet is one of key reasons why FaceNet performs so well. Firstly, the loss function directly focuses on improving the feature representation. This is not similar to the case when general deep models are trained as a multi-class classifier and the feature representation is extracted from one of the in-between layer of the network.

Following is the description of the Triplet Loss.

- Let x be an image, and its embedding in d-dimensional Euclidean space is represented by $f(x) \in \mathbb{R}^d$.
- Additionally, the embedding is contrained to live on the d-dimensional hypersphere, i.e $\|f(x)\|_2 = 1$.
- The loss function is based on a triplet. The i^{th} triplet is represented as $({x_i}^a, {x_i}^p, {x_i}^n)$ where ${x_i}^a$, ${x_i}^p$, ${x_i}^n$ are called as anchor, positive and negative image respectively.
- The loss function ensures that an image ${x_i}^a$ (*anchor*) of a specific person is closer to all other images ${x_i}^p$ (*positive*) of the same person than it is to any image ${x_i}^n$ (*negative*) of any other person. Figure 3.2 shows this notion pictorially.
- Thus we want,

$$\|{x_i}^a - {x_i}^p\|_2 + \alpha < \|{x_i}^a - {x_i}^n\|_2, \quad \forall \ ({x_i}^a, {x_i}^p, {x_i}^n) \in \tau$$

  where α is margin enforced between positive and negative pairs and τ is set of all possible triplets in training set with cardinality N.
- Therefore the loss over a batch of input images is defined as

$$Loss = \sum_i^N [\|f({x_i}^a) - f({x_i}^p)\|_2 - \|f({x_i}^a) - f({x_i}^n)\|_2 + \alpha] \ .$$

- Not all possible triplets are used for training as many would be easily satisfied and would thus result in slower convergence (as they would still be passed through the network).
- α is set to 0.2 which is found after cross validation.

3.1.3 Triplet Selection

As mentioned earlier only selected triplets are actually used in the training procedure. We further look into the exact way to choose triplets from the training data which are used to compute the triplet loss. Let us first define two essential terms *Hard Positive* and *Hard Negative*.

- **Hard Positive** $({x_i}^p)$: Given an anchor ${x_i}^a$,

$$ {x_i}^p = argmax_{{x_i}^p} \|f({x_i}^a) - f({x_i}^p)\|_2 $$

- **Hard Negative** $({x_i}^n)$: Given an anchor ${x_i}^a$,

$$ {x_i}^n = argmin_{{x_i}^n} \|f({x_i}^a) - f({x_i}^n)\|_2 $$

Therefore **(anchor image, hard positive, hard negative)** thus constitutes a challenging triplet and is used for training.

It is unfeasible to compute *argmax* and *argmin* across the whole training data after each back-propagation step.Thus triplets are generated from a large mini batch of size 1800 images.To obtain a meaningful representation of anchor-positive distances, it needs to be ensured that a minimal number of images of any one identity is present in each mini-batch. Implementation uses 40 faces per identity per mini-batch.

3.1.4 Architecture

The CNN (Convolutional Neural Network) used in implementation uses rectified linear units as the non-linear activation. Overall the model has 11 Convolutional Layers and 3 Fully connected layers results in an overall depth of 14 layers. The model is inspired from the model used by Zeiler & Fergus [**9**].

The model uses *1 x 1 x d* as suggested in [**10**] between the convolutional layers. It has a total of 140 million parameters and requires around 1.6 billion Floating Point Operations pe Second (FLOPS) per image. Table 3.1 depicts the exact architecture of the CNN used in FaceNet.

Input/Output size = *rows* x *cols* x *#filters*

kernel = *rows* x *cols*, stride

3.1.5 Implementation Details

The CNN is trained using Stochastic Gradient Descent (SGD) with standard backprop and AdaGrad on a training set of 200 million images containing 8 million identities. The learning rate is decreased with time and has a starting value of 0.05. The model was initialised from random and trained on a CPU cluster for 2000 hours.

Usual approach in deep learning based methods for face recognition use a classification layer trained over a set of known face identities and then take an intermediate bottle-neck layer as a representation which is used to generalize recognition beyond the set of identities used in training.

Layer	Size-In	Size-Out	Kernel	Parameters
conv1	220x220x3	110x110x64	7x7x3,2	9K
pool1	110x110x64	55x55x64	3x3x64,2	0
conv2a	55x55x64	55x55x64	1x1x64,1	4K
conv2	55x55x64	55x55x192	3x3x64,1	111K
pool2	55x55x192	28x28x192	3x3x192,2	0
conv3a	28x28x192	28x28x192	1x1x192,1	37K
conv3	28x28x192	28x28x384	3x3x192,1	664K
pool3	28x28x384	14x14x384	3x3x384,2	0
conv4a	14x14x384	14x14x384	1x1x384,1	148K
conv4	14x14x384	14x14x256	3x3x384,1	885K
conv5a	14x14x256	14x14x256	1x1x256,1	66K
conv5	14x14x256	14x14x256	3x3x256,1	590K
conv6a	14x14x256	14x14x256	1x1x256,1	66K
conv6	14x14x256	14x14x256	3x3x256,1	590K
pool4	14x14x256	7x7x256	3x3x256,2	0
fc1	7x7x256	1x32x128	ReLU	103M
fc2	1x32x128	1x32x128	ReLU	34M
fc3	1x32x128	1x1x128		524K
L2	1x1x128	1x1x128		0

Table 3.1: Deep Convolutional Network Architecture

For example, DeepFace (Facebook) [**11**] is trained using a multi-class classifier on 4.4 million images with 4,030 identities (classes). The feature representation from the model in case of DeepFace is extracted from second last layer (layer before the softmax layer).

There are two major downsides to this approach:

- **Indirectness**: You are learning the embedding indirectly by learning a multi-class classifier. In this case, one has to hope that bottleneck representation will generalize well to new faces. Thus we severely restrict the scope of the faces were the model performs well.
- **Inefficiency**: Bottleneck layer representation size per face is usually very large (thousands in dimension eg: DeepFace generates a 4096 embedding of an input face). The bottleneck representations are large in dimension majorly because the model heavily depends on the output of this layer. Thus a very discriminative model would need a very big feature vector to actually capture all of the information from the input. This gives rise of memory issues and increases the number of floating point operations during training. Thus, turning out to be very inefficient.

FaceNet avoids both of this difficulties by directly training its output to be a compact 128 dimensional embedding using a triplet based loss function.

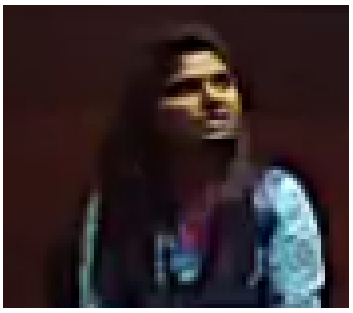

Figure 3.3: Query Image

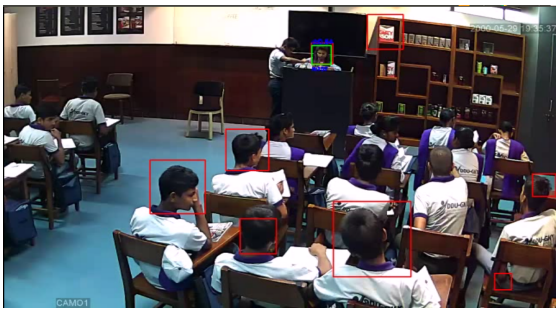

Figure 3.4: Correct Recognition, ($d^2 = 0.44$)

3.1.6 Results

The query image of the trainer was taken itself from the video due to lack of database of images on the trainer. However, to ensure pose discrimination the query was cropped from a different instance of the video (where the trainer sits) and was analysed on different setting of the video.

Observations:

- Figure 3.4, Figure 3.5, Figure 3.7 shows the instance where the model was correctly able to recognize the trainer in the frame (marked in green bounding box). Other faces detected by LSTM face detector are shown in red bouding box. d^2 here depicts the Euclidean distance squared between the embedding of the query image and its closest matching face in the frame.
- Figure 3.6 shows the case when the Head detector was not able to capture the trainer's face which led to a wrong recognition.
- Figure 3.8 shows the case when the model wrongly predicted the trainer even after trainer's face was detected by the LSTM model. This happened due to sudden change in posture in the video when the trainer was teaching.
- An interesting observation here is that for every correct recognition the d^2 value is significantly less than when there was a failed recognition.
- This fact can be used to avoid failed detection like in figure 3.8 by introducing a hyperparameter which represents the minimum d^2 value necessary to deem the detection as similar to the query image. (d^2 equal to 0.5 clearly works in this example).
- By experiments, it was observed that this hyperparameter depends on the background, intensity of the image i.e the overall background setting of the image. Thus an absolute hyperparameter suitable for one video may not give good results on other videos.

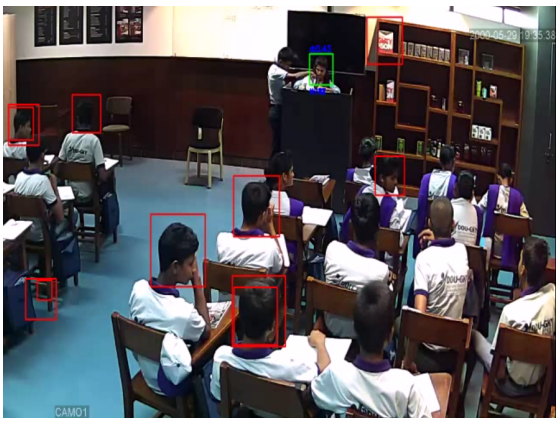

Figure 3.5: Correct Recognition, ($d^2 = 0.43$)

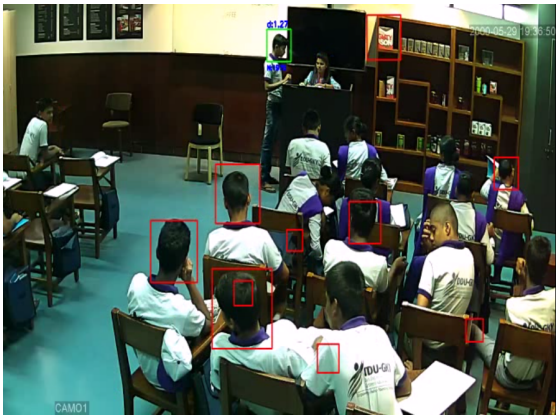

Figure 3.6: Missing Head, failed recognition, ($d^2 = 1.27$)

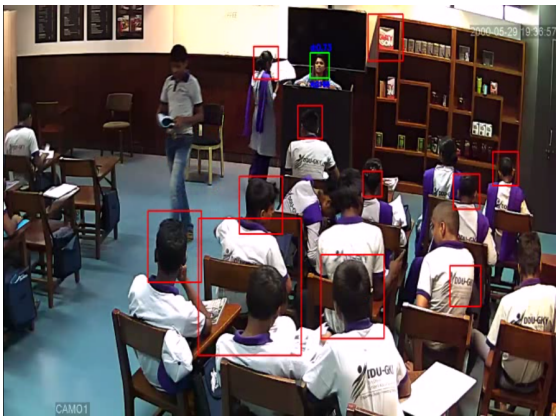

Figure 3.7: Correct Recognition, ($d^2 = 0.33$)

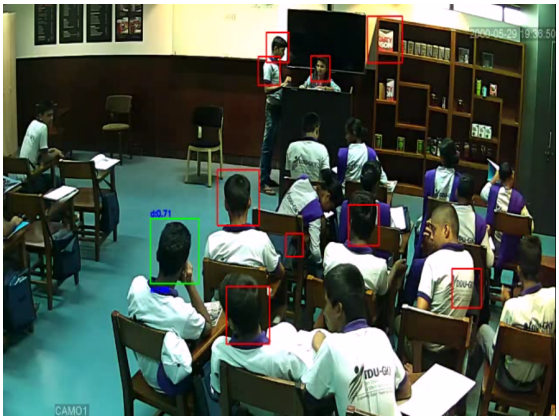

Figure 3.8: Failed recognition despite of head detection, ($d^2 = 0.71$)

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Figure 3.9: VGG Face Architecture

3.2 VGG-Face

After great results from FaceNet on our dataset we decided to try out deep network by Visual Geometry Group, Oxford which beats FaceNet on the benchmark dataset of YouTube Faces [**6**].

Here are the quantitative results from [**8**]:

- Labelled Faced in the Wild (LFW) [**5**]:
  - FaceNet: **99.63%**
  - VGG-Face: **98.95%**
- YouTube Face Database (YTF) [**6**]:
  - FaceNet: **95.1%**
  - VGG-Face: **97.3%**

The interesting fact however is that the training data used to train VGG Face is significantly smaller than the one used for training FaceNet. Here is a quick comparison:

- FaceNet Training Data: 8 million identities, 200 million images
- VGG Face Training Data: 2,662 identities, 2.6 million images

3.2.1 Architecture

Please refer to figure 3.9 for the exact architecture details. The Convolutional Network comprises of 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and Max Pooling. The first 8 blocks are Convolutional and last 3 blocks are Fully Connected (FC). The use of softmax layer is for the multi-class classification problem. All convolutional layers are followed by a rectification layer (ReLU). The final embedding is extracted from the second last fully connected layer. Thus, giving us a vector of 4096 dimension. Overall we have 13 convolutional layers and 3 fully connected layers.

Total Depth = 16 layers.

Following notation is followed in figure 3.9:

- support = *dimension of kernel*
- filt dim = *depth of kernel*
- num filts = *output depth from the layer*

3.2.2 Training

Training follows the usual approach of bootstrapping the network initially using a Multi-class classifier and then fine tuning the embeddings directly using a loss function which directly incorporates the embedding in the loss. Bootstrapping stage is extremely necessary if you don't have a lot of training data. Muli-class classification stage directs the model in the "correct direction" quickly as learning a classifier is a relatively simpler task than learning a very discriminative embedding directly. This was not the case with FaceNet as it had the privilege of a lot of training data.

The training for VGG Face is done in two stages:

- Stage 1: Learning a Face Classifier
- Stage 2: Learning a Face embedding using a triplet loss.

3.2.3 Stage 1: Learning a Face Classifier

Initially the network is bootstrapped by considering the problem of recognising N = 2,662 unique individuals over 2.6 million images (Multi-class classification). The need of this step (as mentioned earlier) is due to the small scale of training data set. Loss function is the standard cross-entropy loss function.

Loss = -$\log(\mathbf{p}_k)$, where k is the index of true label and p_k is the softmax probability.

3.2.4 Stage 2: Learning a Face embedding using a triplet loss

During this stage of training, the network is further tuned using the triplet loss (Refer *Section 3.1.2*). The feature vector used in the loss function is the output of the second last fully connected (bottleneck) layer of 4096 dimensions. Triplet Selection (Refer *Section 3.1.3*) over a mini-batch is done as before. Thus, we have the loss function as follows:

$$L = \sum_{i}^{N} [\|f(x_i{}^a) - f(x_i{}^p)\|_2 - \|f(x_i{}^a) - f(x_i{}^n)\|_2 + \alpha]$$

Note: The notation is similar to the one followed in *Section 3.1*

3.2.5 Implementation Details

- The input to the network is a face image of 224x224 with average face image (computed from the training set) subtracted - this is critical for the stability of optimisation algorithm.
- Data augmentation was done by flipping the image left to right with 50% probability.
- However, no color channel augmentation is done to preserve rawness of the training data.
- 50% Dropout applied after the first two FC layers.
- 2D alignment on test images showed increase in performance. However, doing 2D alignment on training data did not provide any additional boost.
- The weights of the filters in the CNN are initialised by random sampling from a Gaussian distribution with zero mean and 10^{-2} standard deviation.
- Biases were initialised to zero.
- For learning the embeddings using triplet loss, the network is frozen except the last fully-connected layer which implements the discriminative projection.
- Stage 2 of training is executed for 10 epochs using Stochastic Gradient Descent with a fixed learning rate of 0.25.

Figure 3.10: Query Image

Figure 3.11: Correct Recognition, (d^2 = 5560.25)

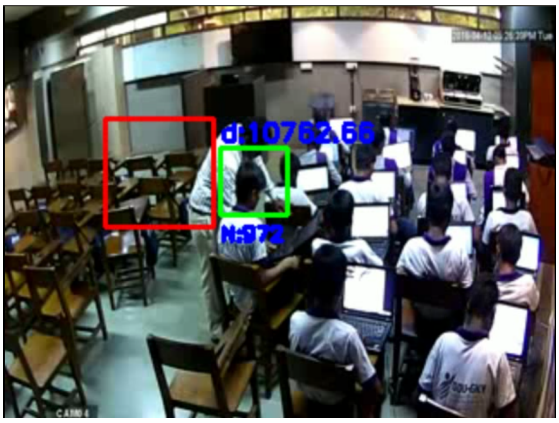

Figure 3.12: Missed head detection, (d^2 = 10762.25)

Figure 3.13: Missed head detection, ($d^2 = 9218$)

3.2.6 Results

The query image of the trainer was taken itself from the video due to lack of database of images on the trainer. For the sake of variety, we analyse two different videos here.

Observations:

- Figure 3.10 and figure 3.14 are the two query images.
- Figure 3.11 shows the instance where the model was correctly able to recognize the trainer in the frame (marked in green bounding box).
- Figure 3.12, figure 3.15 shows the case when the Head detector was not able to capture the trainer's face which led to a wrong recognition.
- Figure 3.16 shows the case when the model wrongly predicted the trainer even after trainer's face was detected by the LSTM model. The overall performance of VGG was considerably poor in comparison to FaceNet on our dataset.
- Again for every correct recognition the d^2 value is significantly less than when there was a failed recognition. See figure 3.11, figure 3.12 and figure 3.13

Therefore, as a step towards building a complete product for MoRD we decided to use FaceNet for now as our face recognition module.

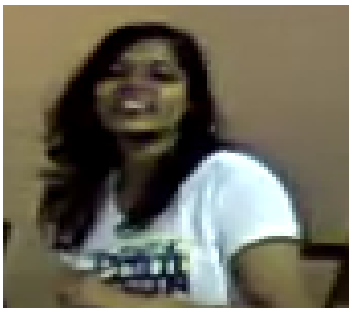

Figure 3.14: Query Image

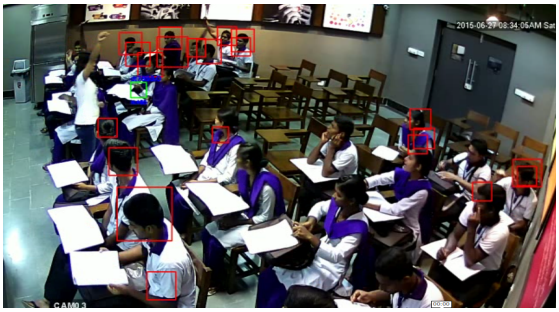

Figure 3.15: Failed face detection due to occlusion, ($d^2 = 5560.25$)

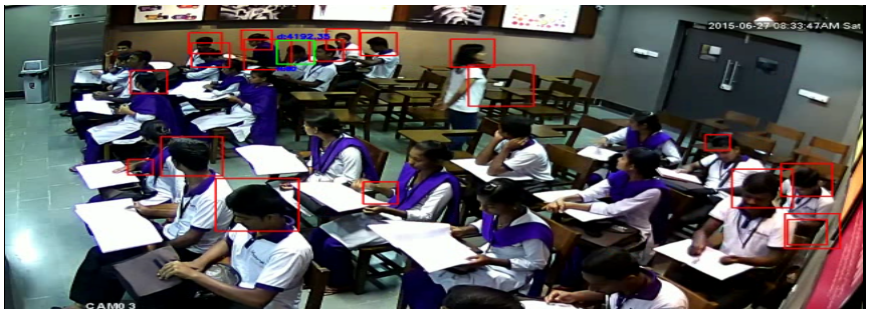

Figure 3.16: Failed face detection , ($d^2 = 4192.35$)

4. Further Work

We looked at two deep learning model each enjoying a different advantage architecture wise namely FaceNet and VGG-Face. FaceNet seems to perform significantly well on our dataset. Most of the wrong face recognitions were due to the fact that the trainer's face was not detected by the Face Detection module. We plan to train the LSTM Face detection model on our data set for better results which will also result in significant improvement with face recognition.

The hyperparameter (minimum distance threshold) introduced in *Section 3.1.6* depends on the environment setting. Therefore a plausible method to tackle this problem would be to do **Transactive Learning** [**12**] which essentially learns these parameters online also tuning the deep model on the fly.

The face recognition modules are susceptible to give wrong recognitions in case when the there is a sudden change in posture of the trainer. Also, use of convolutional network forward propagation for each detected face in a frame is computationally expensive. As we are looking at videos as input and there is not much change across two consecutive images we have decided to look into tracking techniques like **Correlation Tracker** [**15**] to do tracking of correct face recognitions for a certain number of frames and then reset the tracks again with doing forward propagation using the deep networks.

Bibliography

[1] HoG Human Detection
https://arxiv.org/pdf/1312.6229.pdf

[2] Rapid Object Detection using a Boosted Cascade of Simple Features
https://www.cs.cmu.edu/ efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[3] End-to-end people detection in crowded scenes
https://arxiv.org/pdf/1506.04878v3.pdf

[4] Eigenfaces for Recognition
http://www.face-rec.org/algorithms/PCA/jcn.pdf

[5] Labelled Faces in the Wild (LFW)
http://vis-www.cs.umass.edu/lfw/

[6] YouTube Face Database (YTF)
http://www.cs.tau.ac.il/ wolf/ytfaces/

[7] FaceNet: A Unified Embedding for Face Recognition and Clustering (CVPR 2015)
https://arxiv.org/pdf/1503.03832v3.pdf

[8] Deep Face Recognition - Oxford VGG (British Machine Vision Conference, 2015)
http://www.robots.ox.ac.uk/ vgg/publications/2015/Parkhi15/parkhi15.pdf

[9] Visualizing and Understanding Convolutional Networks (CVPR 2013)
http://www.cs.nyu.edu/ fergus/papers/zeilerECCV2014.pdf

[10] Network In Network (CVPR 2014)
https://arxiv.org/pdf/1312.4400v3.pdf

[11] DeepFace: Closing the Gap to Human-Level Performance in Face Verification
https://research.facebook.com/publications/deepface-closing-the-gap-to-human-level-perfor

[12] Transactive Learning
http://pubsonline.informs.org/doi/pdf/10.1287/orsc.1050.0143/

[13] OpenFace
https://cmusatyalab.github.io/openface/

[14] Visual Geometry Group, Oxford
http://www.robots.ox.ac.uk/ vgg/

[15] Correlation Tracker
http://www.bmva.org/bmvc/2014/files/paper038.pdf