Person Identification and Tracking using Clustering, Face Matching and YOLOv3 for Surveillance

Akshat Rajkumar Jain
19095009, B.Tech
Department of Electronics Engineering,
Indian Institute of Technology (BHU)
Varanasi-221005
akshat.rajkumarjain.ece19@itbhu.ac.in

Akshat Sood
19095010, B.Tech
Department of Electronics Engineering,
Indian Institute of Technology (BHU)
Varanasi-221005
akshat.sood.ece19@itbhu.ac.in

Aman Kumar
19095011, B.Tech
Department of Electronics Engineering,
Indian Institute of Technology (BHU)
Varanasi-221005
aman.kumar.ece19@itbhu.ac.in

Aman Mishra
19095012, B.Tech
Department of Electronics Engineering,
Indian Institute of Technology (BHU)
Varanasi-221005
aman.mishra.ece19@itbhu.ac.in

Supervisor : Dr. Manoj Kumar Meshram
Professor
Department of Electronics Engineering,
Indian Institute of Technology (BHU)
Varanasi-221005
mkmeshram.ece@iitbhu.ac.in

ABSTRACT

Human Surveillance in the present times has become a cumbersome job to handle manually as it requires tracking of each person separately which sometimes requires a lot of human efforts and this manual way also sometimes unable to track a person doing unethical practice like burglary, theft at a public place.
With the advent in technologies especially in the field of deep learning and autonomous systems, these tasks can be performed by machines to reduce human labour and perform efficient recognition and tracking of each person. In our project we are using a blend of pretrained and self implemented methods to make a system which will account for speed/accuracy tradeoff and run on real time to tackle the challenges of person identification.

LITERATURE REVIEW

In research there had been many different methods proposed in application of person identification and Re-identification. Some problems encountered in research are:-

1. Ambient light conditions
2. Face representation in ambient situations is not much pronounced.
3. Speed vs Acc tradeoff

Different methods had been proposed in this field like:

1. Using Full Body motion
2. Using Multimodal Biometrics
3. Re-Identification Based Methods and so on...

All these methods have their own merits and demerits which you can find in the above links.

APPROACH

Person Identification along with tracking is a very tough job even for a machine to handle as it involve complex tasks like

1. Detecting individuals from Camera Frame
2. After Detecting performing Recognition of individual
3. Efficient search and Comparison of detected person from database
4. Tracking of recognised individual
5. Speed vs accuracy of proposed framework (pipeline).

Taking into Consideration each factor **we have proposed a yolo and facenet based pipeline along with clustering for efficient search**. Block diagram (Flowchart) of our approach is shown in Figure-1.

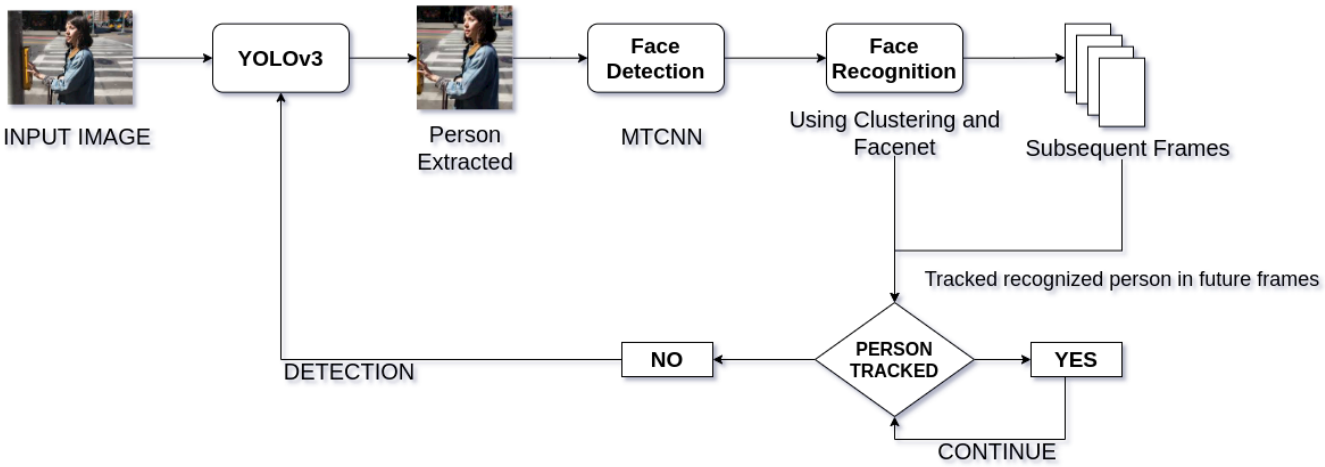


Figure 1 - Pipeline of Our Approach

1. Object Detection

What does one mean by detecting an object? When we see an object, we can exactly point where it is and determine what it is with ease. For computers though, the task is not so simple. This has been an active area of research for years and continues today.

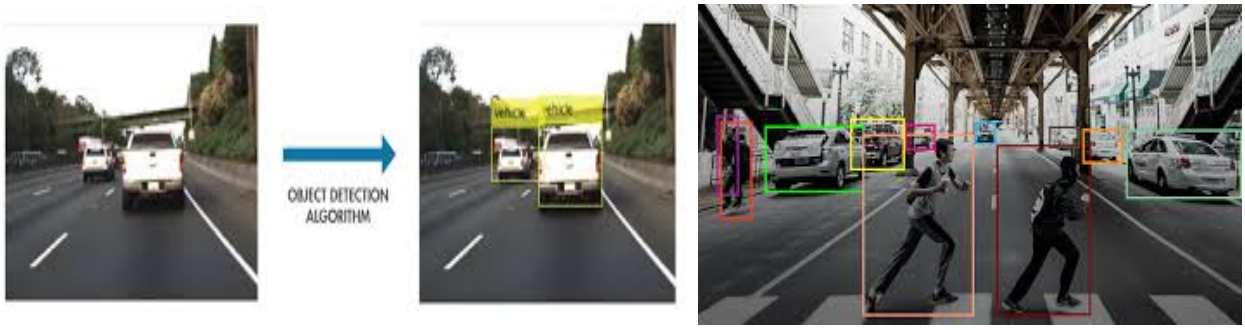


Figure 2 - Object detection

On the basis of Algorithms we used different detection techniques and selected best among them on basis of speed/Accuracy tradeoff:

1. SVM classification using histograms of oriented gradient (HOG) features
2. R-CNN Methods
3. YOLO Based Methods.

1.1 SVM classification using histograms of oriented gradient (HOG) features

A class of objects such as a vehicle vary so much in color. Structural cues like shape give a more robust representation. Gradients of specific directions capture some notion of shape. To allow for some variability in shape, we'll use features known as Histogram of Oriented Gradients (HOG).

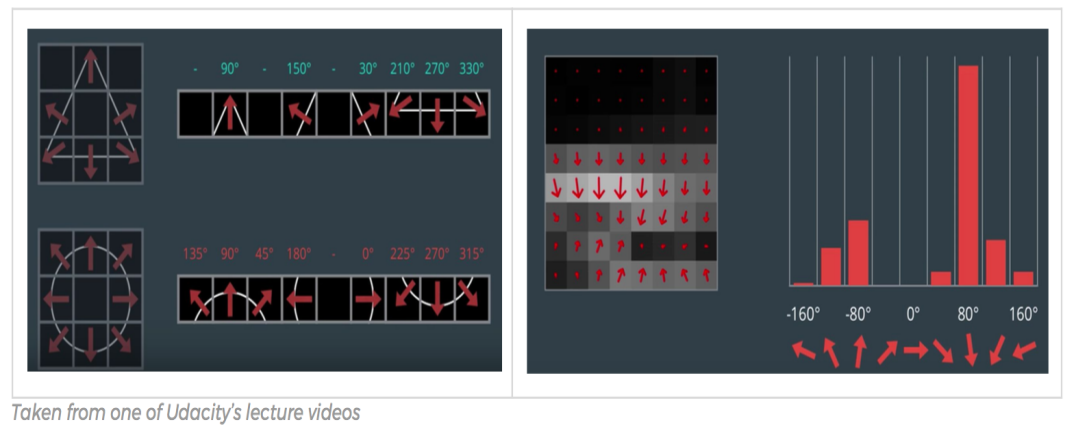


Figure 3 - HOG Features

The idea of HOG is instead of using each individual gradient direction of each individual pixel of an image, we group the pixels into small cells. For each cell, we compute all the gradient directions and group them into a

number of orientation bins. We sum up the gradient magnitude in each sample. So stronger gradients contribute more weight to their bins, and effects of small random orientations due to noise is reduced. This histogram gives us a picture of the dominant orientation of that cell. Doing this for all cells gives us a representation of the structure of the image. The HOG features keep the representation of an object distinct but also allow for some variations in shape.

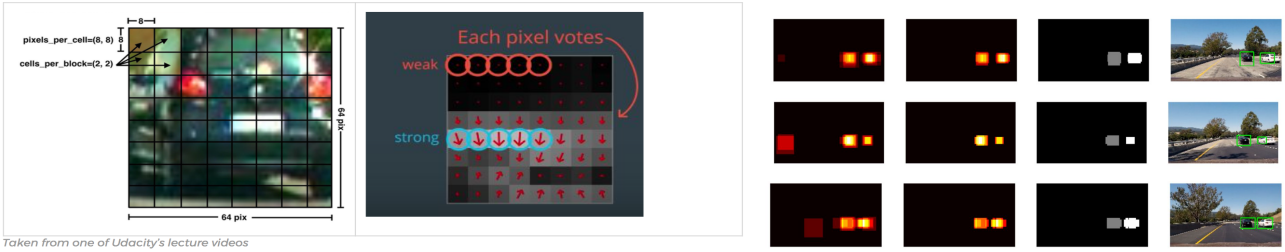


Figure 4 - HOG Based Detection explanation

1.2 RCNN Based Methods

RCNN is a two stage deep learning based pipeline for object detection. It uses a region proposal network(RPN) along with a classification network to perform detection.Two-Step Object Detection involves algorithms that first identify bounding boxes which may potentially contain objects and then classify each bounding separately.

The first step requires a *Region Proposal Network,* providing a number of regions which are then passed to common DL based classification architectures. From the hierarchical grouping algorithm in RCNNs (which are extremely slow) to using CNNs and ROI pooling in Fast RCNNs and anchors in Faster RCNNs (thus speeding up the pipeline and training end-to-end), a lot of different methods and variations have been provided to these region proposal networks (RPNs).

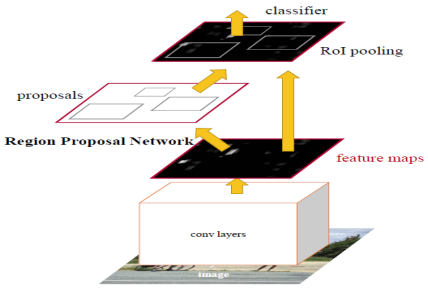


Figure 5 - RCNN Pipeline

These two stage detectors are accurate but are very slow to run in real time (~ around 12-15 fps on a GPU) which limits their usage in real time.

1.3 YOLO Based Methods

YOLO is a single step detector which classifies objects and predicts bounding boxes in a single pass. There are different versions of yolo developed in recent years. The most recently used one are yolov3 and yolov4.

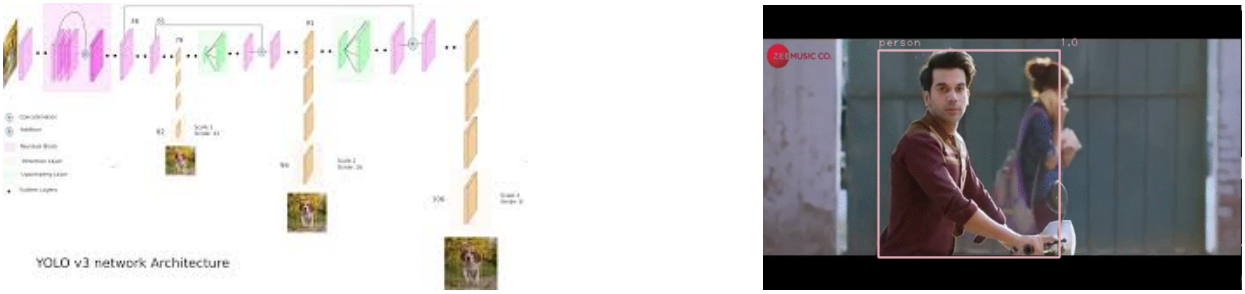


Figure 6 - YOLOv3 Architecture and Detection Example

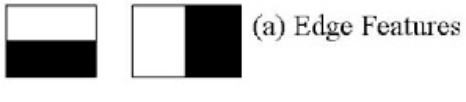
YOLOv3 is very fast, flexible and can run in real time (~30-40 fps) and this third version is quite accurate in detection. Based on the Speed-Accuracy tradeoff of all the methods we decided to use YOLOv3 for person detection as we need real time performance for surveillance with considerable accuracy.

2. Face Detection

Broadly speaking, Face detection involves the detection of faces in an image as well as localisation of it by determining the bounding box coordinates. A face cannot be recognised if it is not detected first. Facial data show a high degree of variability, and it is difficult to detect one using handcrafted filters. So machine learning methods need to be applied to detect various patterns.
We experimented with two methods for this task.

2.1 Face Detection using Haar Cascades

Paper: Rapid Object Detection using a Boosted Cascade of Simple Features (2001)
https://ieeexplore.ieee.org/document/990517
Haar-cascade (Wilson & Fernandez, 2006) is a method invented by Viola and Jones (Viola & Jones, 2001), which trains machine learning for detecting objects in a picture. In this context, it can be used to detect faces. The name of this method is composed of two important words, Haar and Cascade. Haar belongs to Haar-like features, which is a weak classifier and will be used for face recognition. A weak classifier is a classifier that is only slightly better than a random prediction.


(a) Edge Features


(b) Line Features

A Haar-like feature is a rectangle that is split into two, three or four rectangles. They are just like the convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle. Given that the base resolution of the detector is 24x24, the exhaustive set of rectangle features is quite large, over 16,000. This large no of features can be computed very easily using a technique called Integral Image. The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y).
Even though each feature can be computed very efficiently, computing the complete set is computationally expensive. The paper discusses the selection of useful features is done by a technique called Adaboost. Apply every feature on all the training images. For each feature, it finds the best threshold which will classify the faces positive and negative. Then select the features with a minimum error rate, which means they are the features that best classifies the face and non-face images. In the final setup, the 16000+ features were reduced to 6000, providing a significant gain in computing.

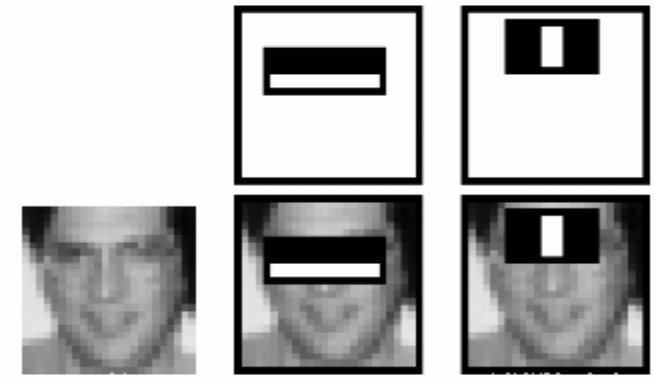


Figure 7 - Haar Cascade Features

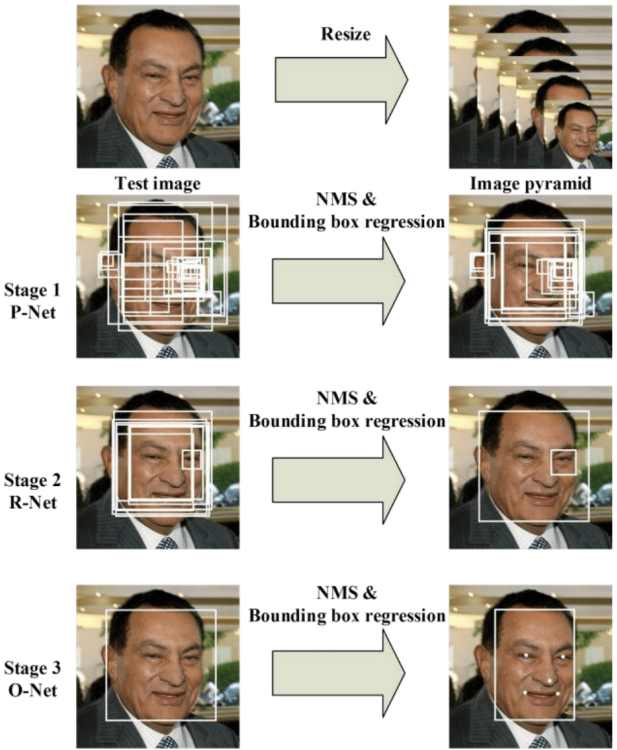
Another feature of this paper is the Attentional Cascade. not all the features need to run on every block. If one of the features does not appear in the block, stop searching in it. The remaining features will not be tested because the machine concludes that there is no face in this block. Then, a new block is taken, and repeat the process. The algorithm sometimes gets fooled and gives false positives, it is not robust and cannot find faces with some occlusions. It is also not good with poor lighting. Deep learning-based methods can be introduced to resolve these problems.

2.2 Multi-task Cascaded Convolutional Networks (MTCNN)

Paper: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks
https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf

Several deep learning methods have been developed and demonstrated for face detection or, in general, object detection like RCNN, fast-RCNN or YOLO.
“Multi-task Cascaded Convolutional Neural Network”, or MTCNN for short, is one of the most popular approaches and current state of the art model.
The MTCNN model is also capable of locating facial landmarks such as eyes and mouth.
The Convolutional Neural Networks (CNNs) have achieved a remarkable performance in many computer vision tasks.
The network is a cascaded structure of three networks.

1. Proposal Network (P-Net)
2. Refine Network (R-Net)
3. Output Network or O-Net



Before providing the image to the network, it is rescaled to different scales to build an image pyramid. In other words, we want to create different copies of the same image in

different sizes to search for different sized faces within the image.
We have a 12 x 12 stage 1 kernel for each scaled copy that will go through every part of

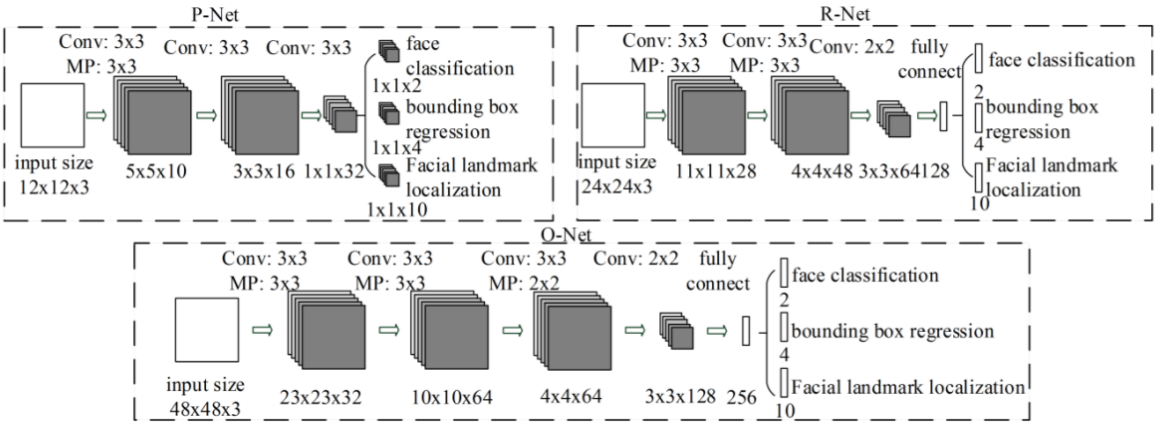
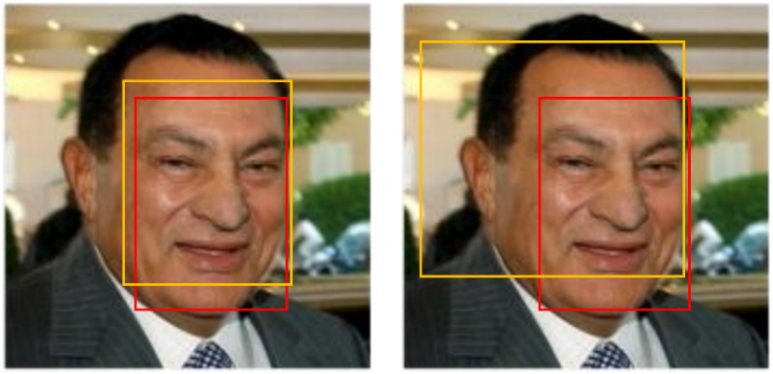


Figure 8 - MTCNN Pipeline and detections

the image, scanning for faces. It starts in the top left corner, a section of the image from (0,0) to (12,12). This portion of the image is passed to P-Net, which returns the coordinates of a bounding box if it notices a face. Then, it would repeat that process with sections (0+2a,0+2b) to (12+2a, 12+2b), shifting the 12 x 12 kernel at a stride of 2.
The weights and biases of P-Net have been trained so that it outputs a relatively accurate bounding box for every 12 x 12 kernel. The P-Net output list is parsed to get confidence levels for each bounding box and delete the boxes with lower confidence. After selecting the boxes with higher confidence, the coordinate system is standardised, converting all the coordinate systems to that of the actual, “un-scaled” image.
However, there are still many bounding boxes left, and a lot of them overlap. Non-Maximum Suppression, or NMS, is a method that reduces the number of bounding boxes. NMS is conducted by first sorting the bounding boxes by their confidence or score. Calculate each kernel area and the overlapping area between each kernel and the kernel with the highest score. The kernels that overlap a lot with the high-scoring kernel get deleted.


Large overlap, yellow box gets deleted

Small overlap, yellow box remains

Figure 9 - Final few detections

For every bounding box, create an array of the same size and copy the pixel values to the new array, resize them to 24 x 24 pixels and normalise them between -1 and 1. All these 24x24 normalised arrays are then fed into the R-Net. The output of R-Net is the coordinates of the new, more accurate bounding boxes and the confidence level of each of these bounding boxes.
The final stage is the O-Net the bounding boxes obtained from R-Net are passed into the O-Net after resizing to 48x48; the O-Net computes the location of 5 facial landmarks - eyes, nose, and endpoints of the mouth.
The loss functions used with this network are the cross-entropy loss function for the face binary classification. The bounding box regression and the facial landmark localisation Euclidean loss have been utilised.
Undoubtedly deep learning has a significant advantage over shallow learning algorithms. This network can detect face even with some occlusions and lighting inconsistencies that other algorithms like haar cascades could not. However, this accuracy comes with the cost that this algorithm is relatively slow on individual frames

compared to the Haar Cascades that ran on almost real-time images; MTCNN barely runs at around ten frames per second. However, it can be fastened with the use of GPUs.
We have used a pre-trained MTCNN model implemented in pytorch. This model was trained on a dataset of millions of images.

3. Face Recognition

With the facial images already extracted, cropped, resize, then face recognition algorithm is responsible for finding characteristics that best describe the image. Face recognition is the task of recognising a person based on their facial image. It has become prevalent in the last two decades, mainly because of the new methods developed and the high quality of the current videos/cameras.
Typical face images contain much information (pixels). It is not possible to compare them to identify the person what face recognition algorithms do. It applies some function over the image and extracts helpful information in the form of a vector. Other classifiers like Support Vector Machines can use this vector with critical facial features.
There are different types of face recognition algorithms, for example:

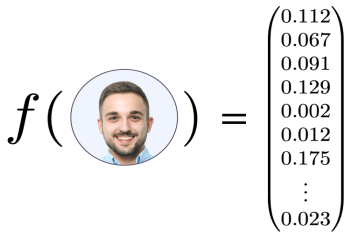


Figure 10 - Face Representation

- Eigenfaces (1991)
- Local Binary Patterns Histograms (LBPH) (1996)
- Fisherfaces (1997)
- Scale Invariant Feature Transform (SIFT) (1999)
- Speed Up Robust Features (SURF) (2006)

Finally, there are deep learning algorithms trained on millions of images for very accurate face recognition. For example:

- Deepface
- VGGface
- Facenet

We have tried to work on two different methods, one each from the above two lists.

3.1 Local Binary Patterns Histogram (LBPH)

Paper: Face Description with Local Binary Patterns
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.456.1094&rep=rep1&type=pdf

The LBPH works on grayscale images. It is a non-holistic type of approach. The LBP has been a powerful tool for texture analysis. Using the LBP combined with histograms can represent the face images with a simple data vector.
The main idea behind the LBP operator is to work on a 3x3 sized block in a grayscale image. The algorithm uses a sliding window of 3x3 to iterate over all parts of the image. The central value of the matrix is taken as the threshold, and the surrounding pixels are converted to 1 if higher than the threshold or 0 otherwise.

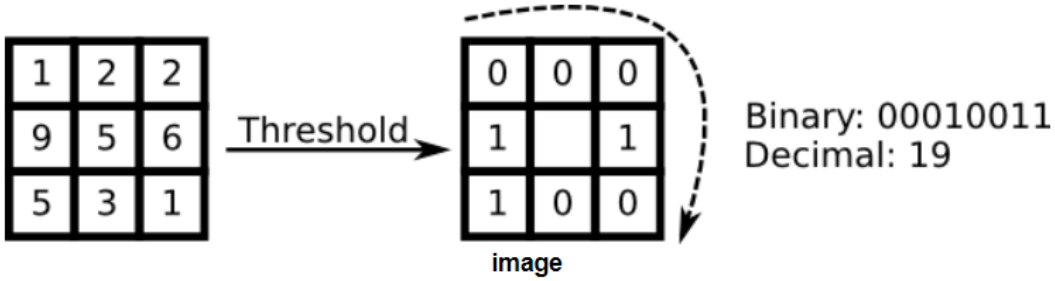


Figure 11 - LBP Operator Explanation

Now the matrix contains only binary values. These values are picked up in order and used to represent a binary number. This decimal value is placed at the central location of the original matrix. This process is repeated with all the subwindows of the image. The figure below shows some sample images after applying the LBP operator on the images. These all are the same faces but with different artificial lighting. This algorithm is quite robust.

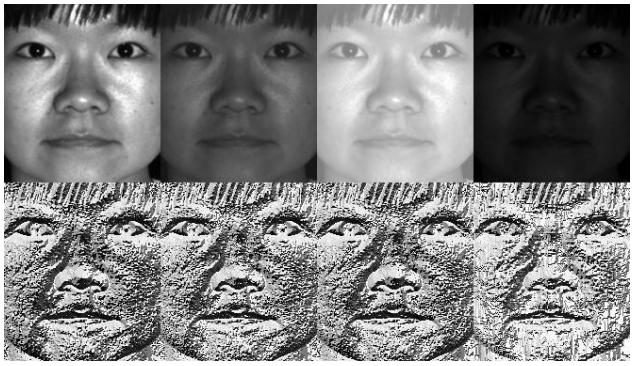


Figure 12 - Sample faces after LBP Operator

The LBP was later extended to use a circular LBP, which uses a radius parameter and the number of sample points to build the circular local binary pattern. In OpenCV implementation, this is done using bilinear interpolation.

To extract spatial information from the image the paper talks about dividing the whole image into parts and compute histograms of each region. The histograms are concatenated to produce the feature vector. This feature vector is compared with another image feature vector to identify the face.

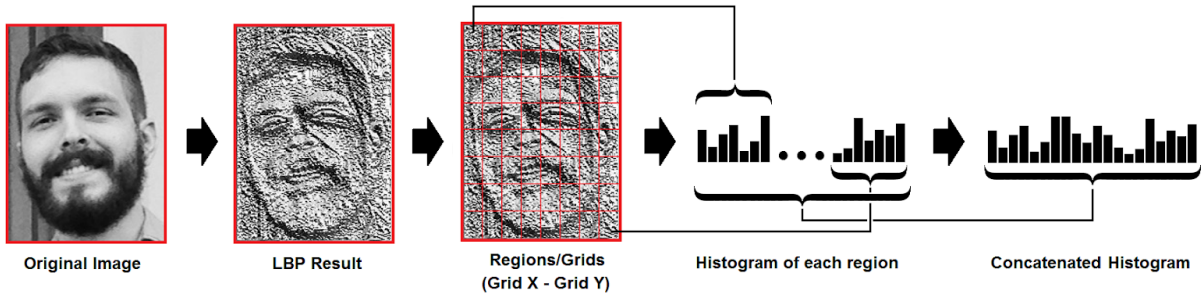


Figure 13 - Face Comparison in LBPH

LBPH algorithm can work with real-time video feed, but similar limitations as with haar cascades apply to this method. It can not work when there is some occlusion to images; for example, if the person is wearing spectacles, the accuracy might drop. Despite all these problems, this technique can be primarily used in machines with low computing power to develop facial attendance systems.

3.2 Deep Learning

Face recognition has been an active area of research in recent years in deep learning in computer vision. Deep learning methods can leverage enormous datasets of faces and learn rich and compact representations of faces, allowing modern models to perform as well and later to outperform the face recognition capabilities of humans. There are perhaps four milestone systems on deep learning for face recognition that drove these innovations: DeepFace, the DeepID series of systems, VGGFace, and FaceNet.
We have used a ResNet-34 based siamese network. ResNets are specially designed networks with many convolutional layers; these solve vanishing gradients problem in deep learning. For specific problems like face recognition and signature verification, we cannot always rely on getting more data to solve this kind of tasks, a new type of neural network architecture called Siamese Networks. It uses only a few numbers of images to get better predictions. The ability to learn from very little data made Siamese networks more popular in recent years.
A Siamese Neural Network is a class of neural network architectures that contain two or more identical subnetworks. They have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks. It is used to find the similarity of the inputs by comparing their feature vectors. Traditionally, a neural network learns to predict multiple classes. So it poses a problem when we need to add/remove new classes to the data. In this case, we have to update the neural network and retrain it on the whole dataset. Also, deep neural networks need a large volume of data to train. Siamese Networks, on the other hand, learn a similarity function. This way, it can classify new classes of data without training the network again.
There are several types of loss functions used in siamese networks:

1. Contrastive Loss: distance-based loss as opposed to more conventional error-prediction losses. This loss is used to learn embeddings in which two similar points have a low Euclidean distance and two different points have a sizeable Euclidean distance.
2. Triplet Loss: It is a loss function where an anchor point is compared to a similar point and a different point. The distance from the anchor to the similar input is minimised, and the distance from the anchor to the different point is maximised.

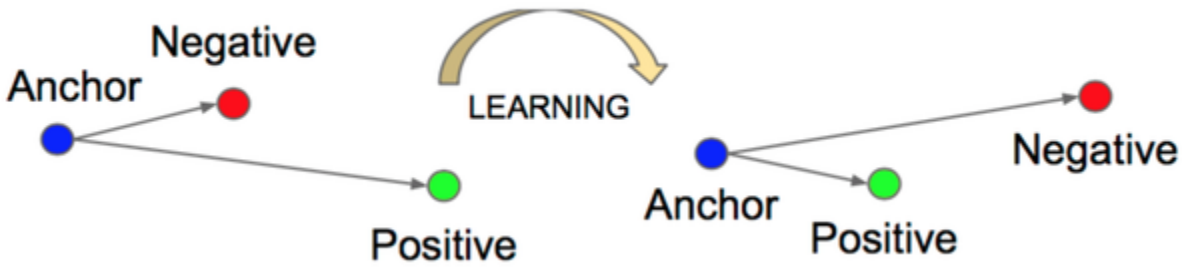


Figure 14 - Interpretation of Loss Functions

The ResNet-34 based model we used was pre-trained on more than 3 million faces. This dataset is derived from several datasets: the face scrub dataset and the Visual Geometry Group dataset. The total number of individual people were 7485. The model was also tested against the LFW benchmark dataset and gave an excellent accuracy.

4. Clustering

Clustering is an unsupervised machine learning technique means it will find a pattern in the dataset and based on this pattern it will cluster the same type of things at the same place. There are different types of Clustering methods available out of which we had used K-Means Clustering.

What is the need of Clustering? During face recognition when we are comparing the incoming person with the person already present in the database then in normal settings we have to compare with each and every person in the database, Hence, the time complexity becomes O(n) which is computationally inefficient for large scale recognition. So we have added an extra clustering based hierarchical method which will first put the incoming face into one of the k clusters and then the IDs present in those clusters are used for further comparison with the facenet Model. So for this the time complexity becomes O(k) where k<N for each case.

Selection of Number of Clusters

Firstly we need to cluster the database into some groups for which the optimal number of clusters is known using the Elbow Method or Silhouette Analysis.

Eqn1 - Describes objective function of clustering Algorithm

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

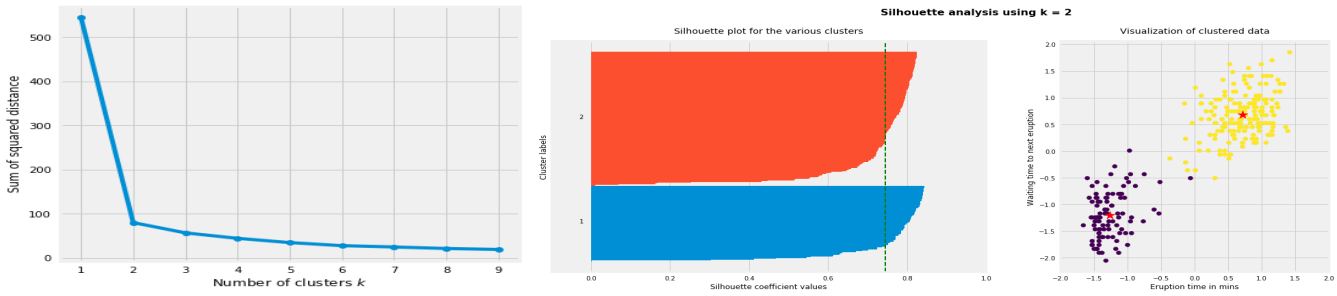


Figure 15 - Example of Elbow method(point of bending is the optimal number of clusters) And Silhouette analysis for determining clusters

5. Tracking

This is the last part of our approach in which we used a KCF Tracking Algorithm implemented in OpenCV [Paper - https://arxiv.org/abs/1404.7584]. It used multiple techniques including the concepts of histogram along with kernelized correlation to track the required entity between subsequent frames. It is a good technique which handles speed-Accuracy tradeoff well. A sample demonstration of tracking using KCF is found below.

Figure 16- Example run by us on some sample videos

OBSERVATIONS AND INNOVATIONS

We have observed several things and based on that we added some innovations accordingly(some of which we already mentioned in our pipeline):

1. Running YOLOv3 continuously decreases overall throughput drastically (1-2 fps on cpu and 15-20 on gpu).
2. Comparison of incoming faces with larger databases decreased the overall speed.
3. We trained a Face Recognition Siamese Model from Scratch on LFW Dataset it was able to perform extremely well on this data but not able to generalize well. For performance comparison we used the CMC(Rankwise comparison) plot to show the deterioration in generalization.

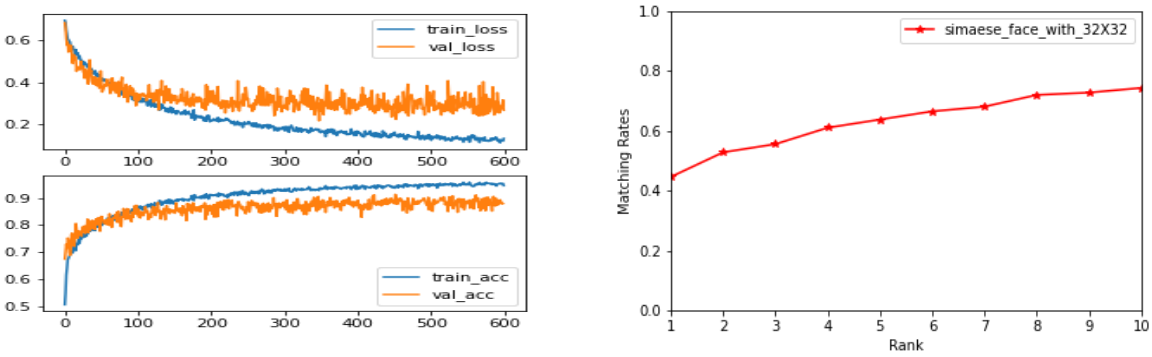


Figure17 - 1st plot showing good performance on individual data but CMC showing less generalization for siamese model trained from scratch

So based on challenges observed we introduced some innovations:

1. Instead of running yolo for each frame we introduced the tracking part to detect once and then track the person till it goes out of frame. This approach increased speed from ~2 fps to 25-30 fps.
2. For comparison bottleneck we introduced the clustering based approach to reduce the searching time efficiently.
3. Based on the above results of siamese trained from scratch, we are decided to use pretrained facenet as it trained on a variety of images with efficient architecture and generalizes well to new images.

RESULTS AND CONCLUSIONS

For object detection:

We extracted the person class from YOLOv3 for our purpose.

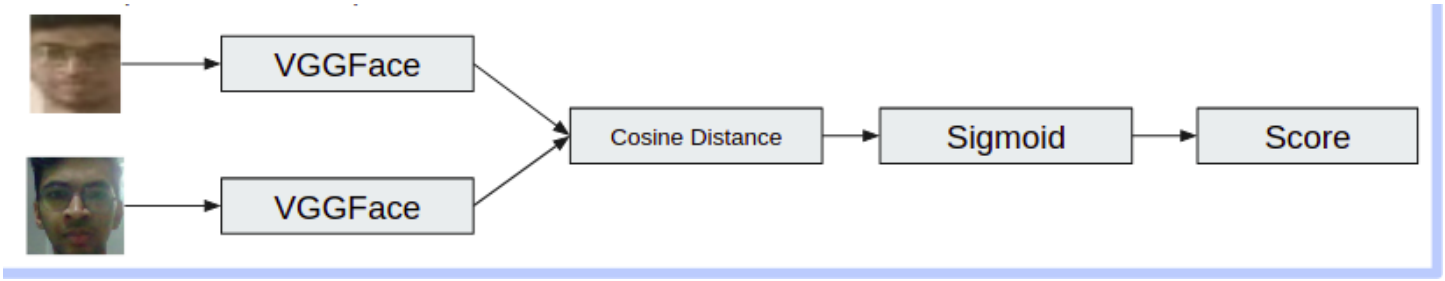
Person Segmentation from Input Image.

For Face detection:

Used MTCNN from the detected person.

Face Detection from Extracted person image.

Face recognition with facenet



Tracking

Using KCF Tracker,tracked that person and recorded timings.

Our approach is presently able to identify, single person in a video frame with tracking efficiently, We have recorded the time as well as fps of tracking. The code is available here at GitHub.

FUTURE WORKS

1. We can add multiple tracking Simultaneously
2. We can add a new approach to increase face resolution known as Super Resolution GANs.
3. We can use a histogram based approach along with clustering for increasing comparison speed.