Face Recognition based Attendance Management System using Machine Learning

Anushka Waingankar[1], Akash Upadhyay[2], Ruchi Shah[3], Nevil Pooniwala[4], Prashant Kasambe[5]

[1,2,3,4,5]*Department of Electronics Engineering, Sardar Patel Institute of Technology, Maharashtra, India.*

---***---

Abstract - *The most arduous task in any organization is attendance marking. In this paper we proposed an automated attendance management system which tackles the predicament of recognition of faces in biometric systems subject to different real time scenarios such as illumination, rotation and scaling. This model incorporates a camera that captures input image, an algorithm to detect a face from the input image, encode it and recognize the face and mark the attendance in a spreadsheet and convert it into PDF file. The system camera of an android phone captures the image and sends it to the server where faces are recognized from the database and attendance is calculated on basis of it.*

***Key Words*: biometric, recognition system, Linux server, Android application, Face Recognition, Deep Learning, Python, Portable Document Format.**

1. INTRODUCTION

Face recognition is a biometric technique which involves determining if the image of the face of any given person matches any of the face images stored in a database. This problem is hard to solve automatically due to the changes that various factors, such as facial expression, aging and even lighting, can cause on the image. Among the different biometric techniques facial recognition may not be the most reliable but it has several advantages over the others[2]. It is widely used in various areas such as security and access control, forensic medicine, police controls and in attendance management system. The various techniques for marking attendance are:

1) Signature based System
2) Fingerprint based System
3) Iris Recognition
4) RFID based System
5) Face Recognition

Amongst the above techniques, Face Recognition is natural, easy to use and does not require aid from the test subject.[1]. It is a series of several related problems which are solved step by step:

1. To capture a picture and discern all the faces in it.
2. Concentrate on one face at a time and understand that even if a face is turned in a strange direction or in bad lighting, it is still the same person.
3. Determine various unique features of the face that can help in distinguishing it from the face of any other person. These characteristics could be the size eyes, nose, length of face, skin colour, etc.
4. Compare these distinctive features of that face to all the faces of people we already know to find out the person's name.

Our brain, as a human is made to do all of this automatically and instantaneously. Computers are incapable of this kind of high-level generalization, so we need to teach or program each step of face recognition separately. Face recognition systems fall into two categories: verification and identification. Face verification is a 1:1 match that compares a face image against a template face images, whose identity is being claimed. On the contrary, face identification is a 1:N problem that compares a query face image.

2. LITERATURE SURVEY

There were many approaches used for dealing with disparity in images subject to illumination changes[3] and these approaches were implemented in object recognition systems and also by systems that were specific to faces. A method for dealing with such variations was using gray-level information to extract a face or an object from shading approach[1].The main reason why gray scale representations are used for extracting descriptors instead of operating on colour images directly is that gray scale simplifies the algorithm and reduces computational requirements. Here in our case, colour is of limited benefit and introducing unnecessary information could increase the amount of training data required to achieve good performance[4].Being an ill-posed problem, these proposed solutions assumed either the object shape and reflectance properties or the illumination conditions[5]. These assumptions made are too strict for general object recognition and therefore it didn't prove to be sufficient for face recognition.

The second approach is the edge map[6] of the image which is a useful object representation feature that is insensitive to illumination changes to certain event.Edge images could be used for recognition and to achieve similar accuracy as gray-level pictures. The edge map information approach possesses the advantage of feature-based approaches, such as invariance to illumination and low memory requirement. It integrates the structural information with spatial information of a face image by grouping pixels of face edge map to line segments. After thinning the edge map, a polygonal line fitting process is applied to generate the edge map of a face [7][8][9].

There is one another approach through which the image disparities due to illumination differences are handled; it is by using a model of several images [12] of the same face which is taken under various illumination conditions. Herein, the images captured can be used as independent models or as a combined model based recognition system [13][14].

3. PROBLEM DEFINITION

Recognizing faces in computer vision is a challenging problem. The illumination problem[3], the pose problem, scale variability, low quality image acquisition, partially occluded faces are some examples of the issues to deal with. Thus face recognition algorithms must exhibit robustness to variations in the above parameters. The existing techniques do not perform well in cases of different illumination, background or rotation. Thus there is a need to address the above mentioned disadvantages. The project aims to design and implement a system which is less sensitive to Illumination, is rotation invariant, scale invariant and robust enough to be implemented in practical applications.

4. PROPOSED SOLUTION

The method proposed in this paper is marking attendance using face recognition technique. As shown in Fig - 1, the attendance is recorded by using a camera that will stream video of students, detect the faces in the image and compare the detected faces with the student database and mark the attendance. The attendance gets marked in a spreadsheet which gets converted into PDF file which is mailed to the concerned e-mail Ids.

The project has two main parts:

[A] Development of Face Recognition System.

[B] Development of Attendance System.

Face recognition is achieved using machine learning and the basic pipeline used for it is as follows:

1. Finds face in an image.
2. Analyses facial features.
3. Compares against known faces and makes a prediction.

Development of complete attendance system is achieved using UI and Android application. Here the application takes data like subject details, faculty details, date and time and provides a click to start the attendance. The images of students are clicked and sent to Linux server where python script runs to mark attendance and generate spreadsheet and PDF file which is then mailed.

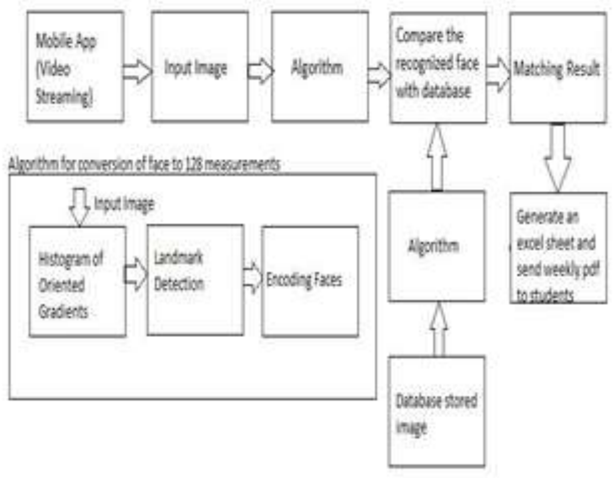

Mobile App (Video Streaming)
Input Image
Algorithm
Compare the recognized face with database
Matching Result
Algorithm for conversion of face to 128 measurements
Input Image
Histogram of Oriented Gradients
Landmark Detection
Encoding Faces
Algorithm
Generate an excel sheet and send weekly pdf to students
Database stored image

Fig -1: Proposed System Block Diagram

1. Find face in the image:

The initial step of our pipeline includes detecting the face for which we will use Histogram of Oriented Gradients (HOG) [15]. One of the most popular and successful “person detectors” out there right now is the HOG with SVM approach.HOG stands for Histograms of Oriented Gradients. HOG is a type of “feature descriptor”. The intent of a feature descriptor is to generalize the object in such a way that the same object (in this case a person) produces as close as possible to the same feature descriptor when viewed under different conditions. This makes the classification task easier. The creators of this approach trained a Support Vector Machine (a type of machine learning algorithm for classification), or “SVM”, to recognize HOG descriptors of people. It is a feature descriptor which is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. To begin with, as shown in Fig -2, we’ll be making our image black and white because we don’t require colour data to find faces. The HOG person detector is fairly simple to understand (compared to SIFT object recognition, for example). One of the main reasons for this is that it uses a “global” feature to describe a person rather than a collection of “local” features. Put simply, this means that the entire person is represented by a single feature vector, as opposed to many feature vectors representing smaller parts of the person. The HOG person detector uses a sliding detection window which is moved around the image. At each position of the detector window, a HOG descriptor is computed for the detection window. This descriptor is then shown to the trained SVM, which classifies it as either “person” or “not a person”. To recognize persons at different scales, the image is sub sampled to multiple sizes. Each of these sub sampled images is searched.

Fig -2: Black and White Image for HOG Algorithm.

Thereafter, we'll look at every single pixel in our image of Fig -3 one at a time. For every single pixel, we want to look at the pixels that are directly surrounding it.

Fig -3 Image for HOG Representation.

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker. If you repeat that process for every single pixel in the image, you end up with every pixel being replaced by an arrow. These arrows are called gradients and they show the flow from light to dark across the entire image. The need of the process is to analyze pixels directly; really dark images and really light images of the same person will have totally different pixel values. But by only considering the direction that brightness changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve. But saving the gradient for every single pixel gives us way too much detail. It would be better if we could just see the basic flow of lightness or darkness at a higher level so we could see the basic pattern of the image. To do this, we'll break up the image into small squares of 16x16 pixels each. In each square, we'll count up how many gradients point in each major direction. Then we'll replace that square in the image with the arrow directions that were the strongest. The end result is we turn the original image into a very similar representation that captures the basic structure of face in a simple way which is shown in Fig -4. Here we solved the problem of brightness as the original image is turned into a HOG representation that captures the major features of the image regardless of image brightness. To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces using which we can now easily find faces in any image.

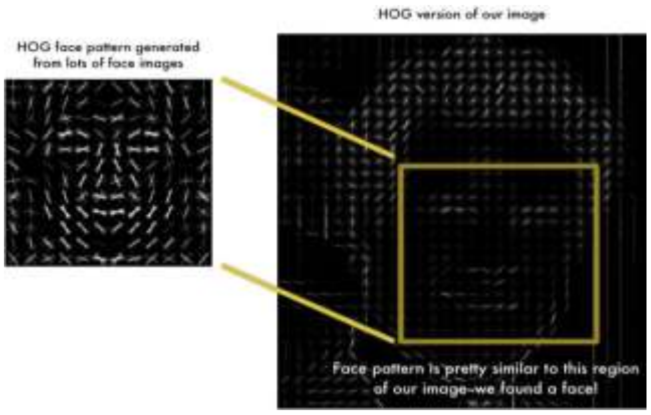

HOG version of our image
HOG face pattern generated from lots of face images
Face pattern is pretty similar to this region of our image–we found a face!

Fig -4: HOG version of Image.

2. Analyze facial features (Posing and Projecting Faces):

After differentiating the faces from the image we have to now deal with the problem of face turning into different directions as shown in Fig -5 which would look totally different to a computer.

Fig -5: Posing and Projecting Faces.

Humans can easily recognize that both images are of the same person, but computer would see these pictures as two completely different people. To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare face in the next steps [15]. For this, we are going to use an algorithm called face landmark estimation.

Herein we would come up with 68 specific points (called landmarks) that exist on every face – the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. as seen from Fig -6.Then we will train a machine learning algorithm to be able to find these 68 specific points on any face. Now that we know where the eyes and mouth is, we'll simply rotate, scale and shear the image so that the eyes and mouth are centered as best as possible. Now no matter how the face is turned, we are able to centre the eyes and mouth so that they are roughly in the same position in the image. This will make our next step a lot more accurate.

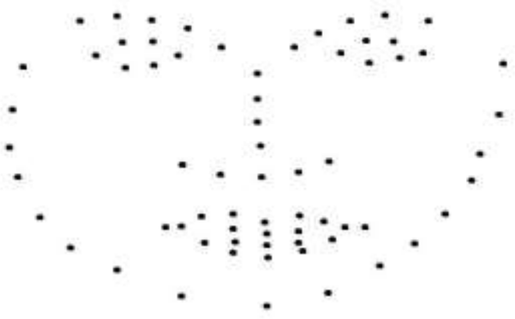


Fig -6: Face Landmark Estimation.

Fig -7: Face Landmarks Detected.

3. Encoding Images:

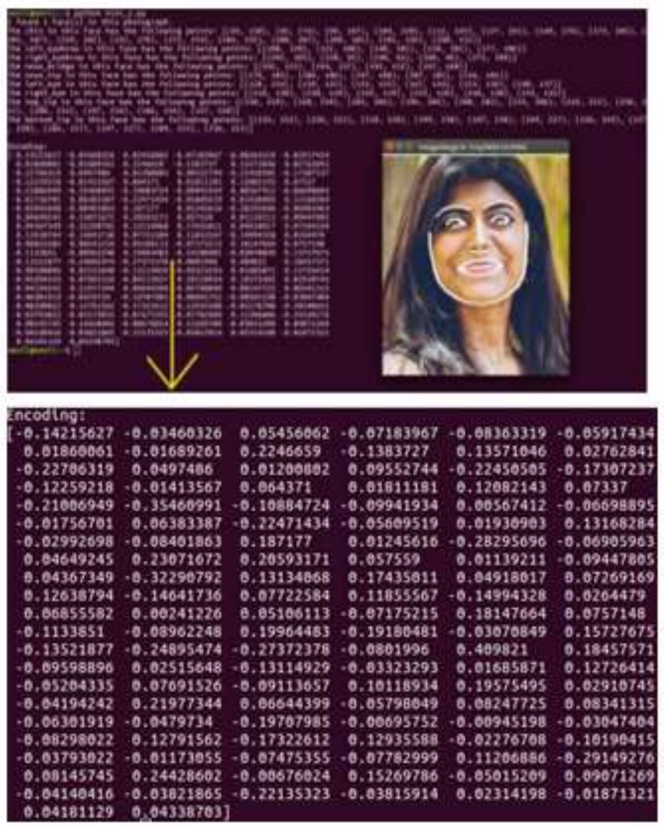
Next step involves recognizing the detected face for which we need to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with closest measurements. The most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure. The solution is to train a Deep Convolution Neural Network to generate 128 measurements for each face.

The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person.
2. Load another picture of the same known person.
3. Load a picture of a totally different person.

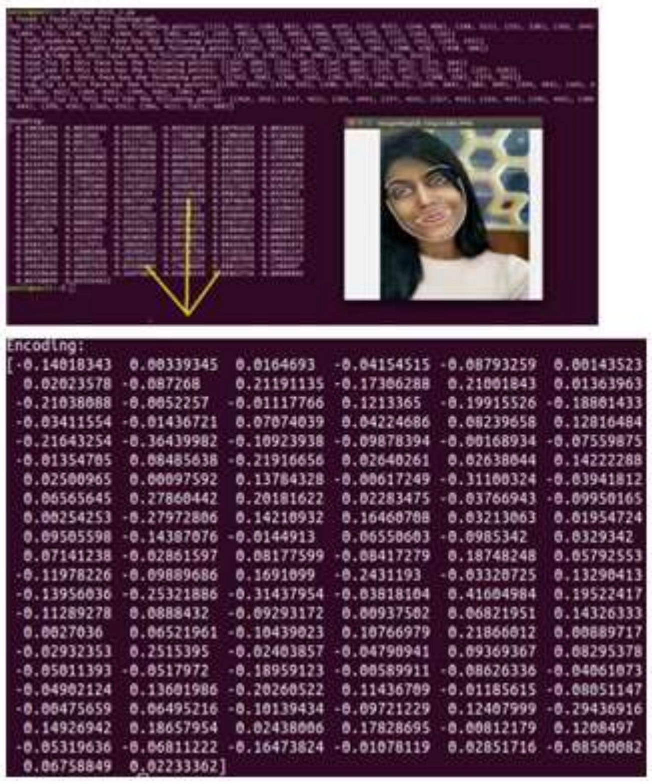
Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart. This is a reduction of complicated raw data like a picture into a list of computer-generated numbers. As shown in fig. 9 and fig. 10, 128 measurements are generated of each faces and it is called as embedding. This process of training a convolution neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Telsa video card, it takes about 24 hours of continuous training to get good accuracy but once the network has been trained, it can generate measurements for any face, even ones it has never seen before. So this step only needs to be done once and several trained networks are already been published by Open Face.

Hence, we need to run our face images through their pre-trained network to get the 128 measurements for each face. The measurements for our test image is shown in Fig -8 and Fig -9 wherein it is seen that the network generates nearly the same numbers when looking at two different pictures of the same person.



```
Encoding:
[-0.14215627 -0.03460326  0.05456062 -0.07183967 -0.08363319 -0.05917434
  0.01860061 -0.01689261  0.2246659  -0.1383727   0.13571046  0.02762841
 -0.22706319  0.0497486   0.01200802  0.09552744 -0.22450505 -0.17307237
 -0.12259218 -0.01413567  0.064371    0.01811181  0.12082143  0.07337
 -0.21006949 -0.35460991 -0.10884724 -0.09941934  0.00567412 -0.06698895
 -0.01756701  0.06383387 -0.22471434 -0.05609519  0.01930903  0.13168284
 -0.02992698 -0.08401863  0.187177    0.01245616 -0.28295696 -0.06905963
  0.04649245  0.23071672  0.20593171  0.057559    0.01139211 -0.09447805
  0.04367349 -0.32290792  0.13134068  0.17435011  0.04918017  0.07269169
  0.12638794 -0.14641736  0.07722584  0.11855567 -0.14994328  0.0264479
  0.06855582  0.00241226  0.05106113 -0.07175215  0.18147664  0.0757148
 -0.1133651  -0.08962248  0.19964483 -0.19180481 -0.03070849  0.15727675
 -0.13521877 -0.24895474 -0.27372378 -0.0801996   0.409821    0.18457571
 -0.09598896  0.02515648 -0.13114929 -0.03323293  0.01685871  0.12726414
 -0.05204335  0.07691526 -0.09113657  0.10118934  0.19575495  0.02910745
 -0.04194242  0.21977344  0.06644399 -0.05798049  0.08247725  0.08341315
 -0.06301919 -0.0479734  -0.19707985 -0.00695752 -0.00945198 -0.03047404
 -0.08298022  0.12791562 -0.17322612  0.12935588 -0.02276708 -0.10190415
 -0.03793022 -0.01173055 -0.07475355 -0.07782999  0.11206886 -0.29149276
  0.00145745  0.24428602 -0.00676024  0.15269786 -0.05015209  0.09071269
 -0.04140416 -0.03821865 -0.22135323 -0.03815914  0.02314198 -0.01871321
  0.04181129  0.04338703]
```

Fig – 8: Generation of 128 measurements for Image 1



```
Encoding:
[-0.14018343  0.00339345  0.0164693  -0.04154515 -0.08793259  0.00143523
  0.02023578 -0.087268    0.21191135 -0.17306288  0.21001843  0.01363963
 -0.21038088 -0.0052257  -0.01117766  0.1213365  -0.19915526 -0.18801433
 -0.03411554 -0.01436721  0.07074039  0.04224686  0.08239658  0.12816484
 -0.21643254 -0.36439982 -0.10923938 -0.09878394 -0.00168934 -0.07559875
 -0.01354705  0.08485638 -0.21916656  0.02640261  0.02638044  0.14222288
  0.02500965  0.00097592  0.13784328 -0.00617249 -0.31100324 -0.03941812
  0.06565645  0.27860442  0.20181622  0.02283475 -0.03766943 -0.09950165
  0.00254253 -0.27972806  0.14210932  0.16460708  0.03213063  0.01954724
  0.09505598 -0.14387076 -0.0144913   0.06550603 -0.0985342   0.0329342
  0.07141238 -0.02861597  0.08177599 -0.08417279  0.18748248  0.05792553
 -0.11978226 -0.09889686  0.1691099  -0.2431193  -0.03320725  0.13290413
 -0.13956036 -0.25321886 -0.31437954 -0.03818104  0.41604984  0.19522417
 -0.11289278  0.0888432  -0.09293172  0.00937502  0.06821951  0.14326333
  0.0027036   0.06521961 -0.10439023  0.10766979  0.21866012  0.00889717
 -0.02932353  0.2515395  -0.02403857 -0.04790941  0.09369367  0.08295378
 -0.05011393 -0.0517972  -0.18959123 -0.00589911 -0.08626336 -0.04061073
 -0.04902124  0.13601986 -0.20260522  0.11436709 -0.01185615 -0.08051147
 -0.00475659  0.06495216 -0.10139434 -0.09721229  0.12407999 -0.29436916
  0.14926942  0.18657954  0.02438006  0.17828695 -0.00812179  0.1208497
 -0.05319636 -0.06811222 -0.16473824 -0.01078119  0.02851716 -0.08500082
  0.06758849  0.02233362]
```

Fig -9: Generation of 123 measurements for Image 2

4. Anticipation of the Users Identity:

For discovering the person's name from the encoding, a basic machine learning classification algorithm is used i.e. simple linear Support Vector Machine (SVM) classification. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyper plane. In other words, given labelled training data (*supervised learning*), the algorithm outputs an optimal hyper plane which categorizes new examples.SVM classifiers are effective in high dimensional spaces and in cases where number of dimensions is greater than the number of sample. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. The classifier is trained to take in the measurements from a new test image and tells which known person is the closest match and the result of the classifier is the person's name which takes milliseconds to run and present the result.The classifier is trained to take in the measurements from a new test image and tells which known person is the closest match and the result of the classifier is the person's name which takes milliseconds to run and present the result.SVM it capable of doing both classification and regression. Here we would focus on using SVM for classification. Non-linear SVM means that the boundary that the algorithm calculates doesn't have to be a straight line. The benefit is that one can capture much more complex relationships between the data points without having to perform difficult transformations on your own. The downside is that the training time is much longer as it's much more computationally intensive. Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the Fig -10 below. In this example, the objects belong either to class green or red. The separating line defines a boundary on the right side of which all objects are green and to the left of which all objects are red. Any new object (white circle) falling to the right is labelled, i.e., classified, as green (or classified as red should it fall to the left of the separating line).

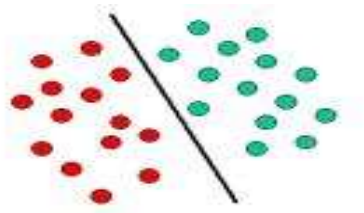


Fig -10: Schematic Example for SVM Classifier.

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyper plane classifiers. Support Vector Machines are particularly suited to handle such tasks.

5. EXPERIMENTATION AND RESULT ANALYSIS:

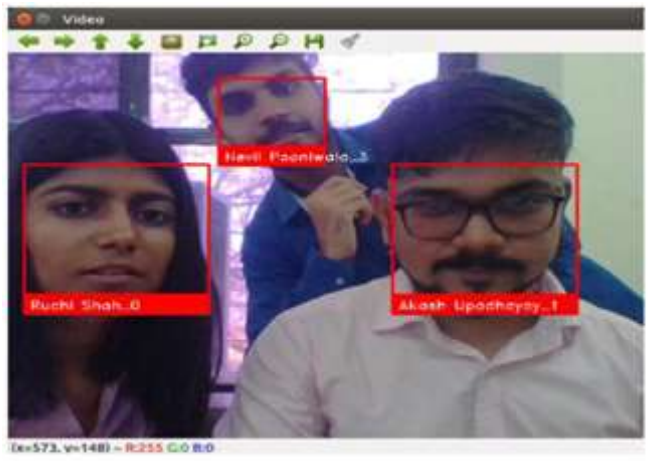


Fig -11: Implementation of Face Recognition in normal lighting condition.

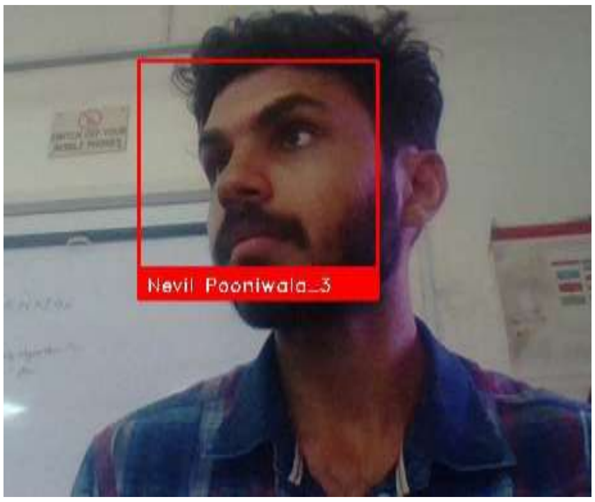
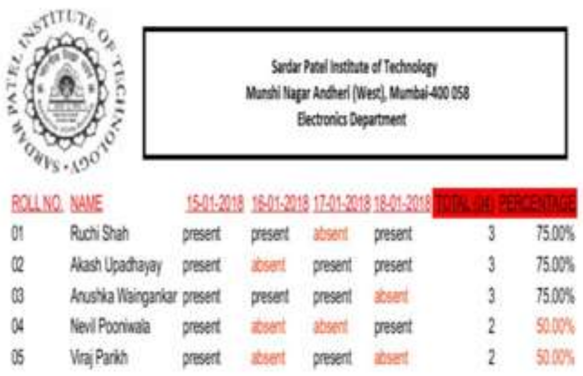


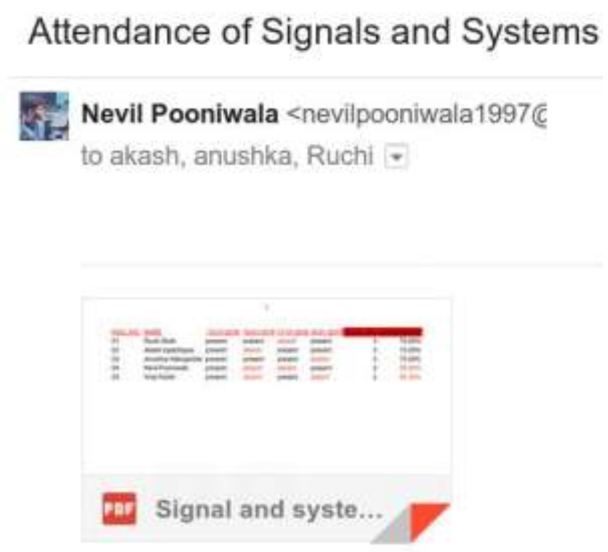
Fig -12: Changing in Posing and Projection of Face.



Sardar Patel Institute of Technology
Munshi Nagar Andheri (West), Mumbai-400 058
Electronics Department

ROLL NO.	NAME	15-01-2018	16-01-2018	17-01-2018	18-01-2018	TOTAL (4)	PERCENTAGE
01	Ruchi Shah	present	present	absent	present	3	75.00%
02	Akash Upadhayay	present	absent	present	present	3	75.00%
03	Anushka Waingankar	present	present	present	absent	3	75.00%
04	Nevil Pooniwala	present	absent	absent	present	2	50.00%
05	Viraj Parikh	present	absent	present	absent	2	50.00%

Fig -13: Generation of report in an excel sheet.



Attendance of Signals and Systems

Nevil Pooniwala <nevilpooniwala1997@
to akash, anushka, Ruchi

Fig -14: PDF sent to students after weekly interval.

6. CONCLUSIONS

The purpose of reducing the errors that occur in the traditional attendance taking system has been achieved by implementing this automated attendance system. In this paper, face recognition system have been presented using deep learning which exhibits robustness towards recognition of the users with accuracy of 98.3% .

The result shows the capability of the system to cope with the change in posing and projection of faces. From face recognition with deep learning, it has been determined that during face detection, the problem of illumination is solved as the original image is turned into a HOG representation that captures the major features of the image regardless of image brightness. In the face recognition method, local facial landmarks are considered for further processing. After which faces are encoded which generates 128 measurements of the captured face and the optimal face recognition is done by finding the person's name from the encoding. The result is then used to generate an excel sheet, the pdf of which is sent to the students and professors on weekly interval. This system is convenient to the user and it gives better security.

REFERENCES

[1] B.K.P. Horn and M. Brooks, Seeing Shape from Shading. Cambridge, Mass.: MIT Press, 1989

[2] A.F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2D and 3D face recognition: A survey", Pattern Recognition Letters, vol.28, issue 15, pp.1885-1906, Oct 2007.

[3] Yael Adini, Yael Moses, and Shimon Ullman, "Face Recognition: The Problem of Compensating for Changes in Illumination Direction"

[4] Kanan C, Cottrell GW (2012) Color-to-Grayscale: Does the Method Matter in Image Recognition? https://doi.org/10.1371/journal.pone.0029740

[5] Grundland M, Dodgson N (2007) Decolorize: Fast, contrast enhancing, color to grayscale conversion. Pattern Recognition 40: 2891-2896.

[6] F. Ibikunle, Agbetuvi F. and Ukpere G. "Face Recognition Using Line Edge Mapping Approach." American Journal of Electrical and Electronic Engineering 1.3(2013): 52-59

[7] T. Kanade, Computer Recognition of Human Faces. Basel and Stuttgart: Birkhauser Verlag 1997.

[8] K. Wong, H. Law, and P. Tsang, "A System for Recognising Human Faces," Proc. ICASSP, pp. 1,638-1,642, 1989.

[9] V. Govindaraju, D.B. Sher, R. Srihari, and S.N. Srihari, “Locating Human Faces in Newspaper Photographs,” Proc. CVPR 89, pp. 549-554; 1989

[10] N. Dalal, B. Triggs “Histograms of oriented gradients for Human Detection”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition , Vol. 1, 2005, pp. 886 – 893.

[11] Modern Face Recognition with Deep learning. WebsiteReference: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep- learning.

[12] S.Edelman, D.Reisfeld, and Y. Yeshurun, “A System for Face Recognition that Learns from Examples,” Proc. European Conf. Computer Vision, S. Sandini, ed., pp. 787-791. Springer- Verlag, 1992.

[13] P. Hallinan, “A Low-Dimensional Representation of Human Faces for Arbitrary Lighting Conditions,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 995-999, 1994

[14] M. Turk and A. Pentland, “Eigenfaces for Recognition,” J. Cognitive Neuroscience, vol. 3, pp. 71-86, 1991.

[15] Varying Illumination and Pose Conditions in Face Recognition Manminder Singha,*, Dr. A. S. Arorab a AP, CSE Department SLIET, Sangrur, 148106, India b Professor, EIE Department, SLIET, Sangrur, 148106, India.

[16] Open source computer vision library.[Online] Available: https://opencv.org/

[17] W.Zhao, R.Chellappa, P. J. Phillips, A. Rosenfeld, “Face Recognition: A Literature Survey.” ACM Computing Surveys, 2003, vol. 35, no. 4, pp. 399-458.