

AUTOMATIC ATTENDANCE MONITORING FOR LABS

**SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF**

Bachelor of Technology
In
Computer Engineering



Under The Supervision of:
Prof. Tanvir Ahmad
Professor

Submitted By:
Antriksh Agarwal-12CSS 12
Sapna-12CSS 55

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING AND TECHNOLOGY
JAMIA MILLIA ISLAMIA, NEW DELHI – 110025
(2015-2016)

DECLARATION

We, Antriksh Agarwal and Sapna, students of Bachelor of Technology, Computer Engineering hereby declare that the project entitled ‘Automatic Attendance Monitoring for Labs’ which is submitted by us to the Department of Computer Engineering, Faculty of Engineering & Technology, Jamia Millia Islamia, New Delhi for the degree of Bachelor of Technology in Computer Engineering, has not been formed the basis for the award of any degree, diploma or other similar title or recognition.

Place : New Delhi

Date: 23rd May, 2016

Antriksh Agarwal
(12CSS-12)
Department of Computer
Engineering
Faculty of Engineering and
Technology
Jamia Millia Islamia

Sapna
(12CSS-55)
Department of Computer
Engineering
Faculty of Engineering and
Technology
Jamia Millia Islamia

CERTIFICATE



This is to certify that the major project report (CEN-891) entitled “**Automatic Attendance Monitoring for Labs**” done by **Mr. Antriksh Agarwal** and **Ms. Sapna**, Roll No. **12CSS-12** and **12CSS-55** is an authentic work carried out by them at **Faculty of Engineering and Technology, Jamia Millia Islamia** under my guidance. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Signature of Guide
19/05/2016
Prof. Tanvir Ahmad
Professor
Department of Computer
Engineering
Faculty of Engineering and
Technology
Jamia Millia Islamia

Signature of HOD
19/05/2016
Prof. Tanvir Ahmad
Head Of Department
Department of Computer
Engineering
Faculty of Engineering and
Technology
Jamia Millia Islamia

ACKNOWLEDGEMENT

We would like to thank our mentor, **Dr. Tanvir Ahmad** (Head & Associate Professor, Deptt. Of Computer Engineering , Jamia Millia Islamia) for providing us with the opportunity to undertake this project. His suggestions in visualizing the project and sustained interest to acknowledged. attain His the objective s invaluable envisaged guidance and are gratefully innovative ideas immensely helped us to surmount difficult situations to bring about successful completion of this project.

We also take this opportunity to express a deep sense of gratitude to **Mr. Sarfaraz Masood** (Assistant Professor, JMI-FET) for his cordial support, valuable information and guidance which helped us in completing this task through various stages.

We also appreciate our Faculty and Jamia Millia Islamia for providing us work culture and environment that has given us the feel of a professional corporate life. Special thanks to towards all our teachers, friends, families and our *Parents* for their valuable support. **Thank you, one and all.**

ANTRIKSH AGARWAL
(12-CSS-12)
Department of Computer
Engineering
Faculty of Engineering &
Technology
Jamia Millia Islamia, New Delhi

SAPNA
(12-CSS-55)
Department of Computer
Engineering
Faculty of Engineering &
Technology
Jamia Millia Islamia, New Delhi

ABSTRACT

Attendance has been under great scrutiny for all types of businesses lately. Everyone wants to reduce the effort and time put in the process of attendance monitoring. In recent times, various automated attendance monitoring systems make use of technological advancements but have been quite cumbersome themselves. This work aims to produce an efficient need-based system to mark attendance. The objective of our project is to detect and recognise people entering and exiting labs and log this data as and when there is an activity outside of the laboratory. The proposed system is based on the face recognition approach with a database of stored images of each individual face on which a neural network is trained. The images captured by the camera / device shall then be detected and recognised using the trained neural network.

The system will be used to monitor attendance automatically by a click. A need-based attempt to reduce manual attendance taking. The system also attempts to reduce the marking of bogus attendance. It involves automatic picture taking and marking attendance by detecting and recognizing faces. The system takes the image of the student, detects the face in the image and tries to recognise the person depicted in the image.

TABLE OF CONTENTS

DECLARATION	2
CERTIFICATE	3
ACKNOWLEDGEMENT	4
ABSTRACT	5
TABLE OF CONTENTS	6
1. Introduction	7
2. Development Process	11
2.1. DataSet	11
2.2. Detection	17
2.2.1. Haar Cascade Detector	17
2.2.2. Results	21
2.3. Recognition	22
2.3.1. Convolutional Neural Network	22
2.3.2. VGG	24
2.3.3. Training	27
2.3.4. Results	29
2.4. Interface	31
3. Conclusion	33

4. Future Scope	34
5. References	35

1. INTRODUCTION

Attendance monitoring system can be used in schools, universities and companies to mark the attendance of students and employees. *Modern attendance systems* may use various methods to mark the attendance. For example, use of a card having a bar code. This card when swiped in a machine, the attendance gets marked in the system. Another attendance monitoring system with more security may require you to give a thumbprint before you occupy your place in the office or class. This is more popularly known as the biometric feature of an attendance system and a common day example of such system is in modern laptops where you have to swipe your finger on a particular metal stripe to unlock the laptop. Similarly, a system of using images to record attendance by marking the recognised faces is going to be less cumbersome and requires less manual intervention.

With the modern features, the attendance monitoring system is no longer limited to marking the attendances of people and does a lot more today. Not only does attendance monitoring system keep a record of their presence or absence from the office/class in a digital format but it also records their leaves, absenteeism, fines, absenteeism patterns, corrective actions and other data. The patterns and trends in the data give a great piece of information to the companies and it all comes from the attendance

monitoring system. To list out some of the common trends found by using an attendance monitoring system -- higher salaries reduce the absenteeism rate, the absenteeism rate for women is more than men, married employees are more punctual than unmarried ones, older employees have a higher presence rate than younger ones etc. There is much more information that can be obtained by using such systems and all it needs is the installation of an application in the system.

When it comes to schools and universities, the attendance monitoring system is a great help for parents and teachers both. Parents are never uninformed of the dependability of their children in the class if the university is using an attendance monitoring system. The registers could easily be exploited by students and if the information was mailed to the parents, there were high chances that the information does not reach them. With the monitoring system in place, the information can easily be printed or a soft copy can be sent directly to parents to their personal email accounts.

In most of the schools and colleges, it is recorded manually or using the biometric monitoring system. These systems consume a lot of time in order to mark attendance and are not modifiable unless self-developed. Thus, we propose a newer and less time-consuming method to monitor attendance which aims to exploit the technological

advancements in computer vision and monitoring using image surveillance.

2. DEVELOPMENT PROCESS

The task of attendance monitoring makes use of a series of steps in order to be able to properly recognise people and store the data in the database in the form of attendance or logging activity as and when it occurs. The attendance monitor we were trying to build was a setup required to go through five steps of hardware as well as software development/setup.

These processes involved setting up the IP camera. The IP camera was selected due to its digital live feeding capability and can be setup with any system and is portable. The IP camera feed was taken and about 500 frames in 19.2 seconds were captured to test it, giving us about 26 frames per second. The next steps involved including the detection, recognition and viewing functionality to the system. This has been explained in the following subsections.

2.1. DATASET

The dataset was collected in two phases. Since the project was development and had to work for people in the department, our mentor considered it better to begin with the development of faces of students and teachers. For a start, I started taking images of students in our class. The

first development phase started when I collected a set of about 400 images of 20 of my classmates. The dataset distribution was as shown in *Figure 1*.

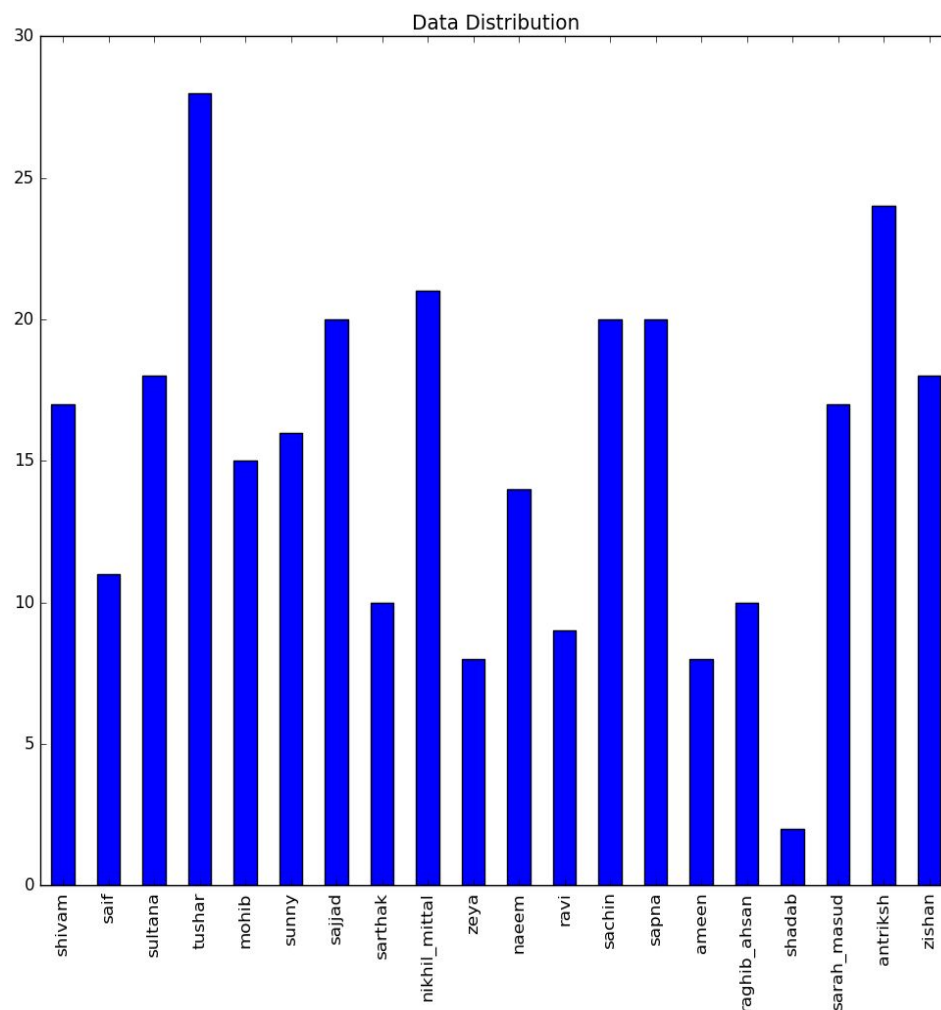


Figure 1. Distribution of Old Dataset

I attempted to train this dataset without any pre-processing of the images but the loss during the training always came down to a *nan* value. The *nan* value denotes a not-a-number which is usually obtained by

division of zero at some point of training. So we had to test what pre-processing might be required. Finally the pre-processing that started training the network was a zero-centering of the data followed by normalisation. A visualisation of the above process can be seen on a data cloud in *Figure 2*.

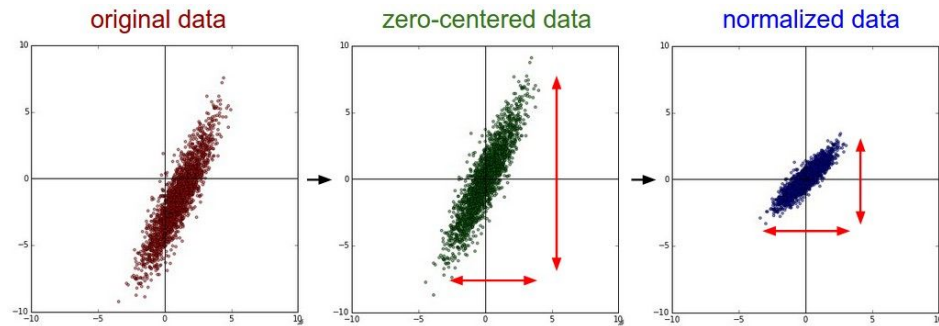


Figure 2: Image Pre-processing

The problem with this data was it was not sparse and there was less randomness in the images as it was collected in the same environment for all making them pose in front of the camera. So, such a dataset was only for initialisation and would never work for real cases. Hence, there was a need to create a dataset with more randomness in the images. So, another dataset was collected later with more number of faces collected from random images of an event involving all of my classmates. This dataset was finally developed manually for training, validation and testing. The

distribution of this data can be seen in the images *Figure 3.*, *Figure 4.*, *Figure 5.*, respectively.

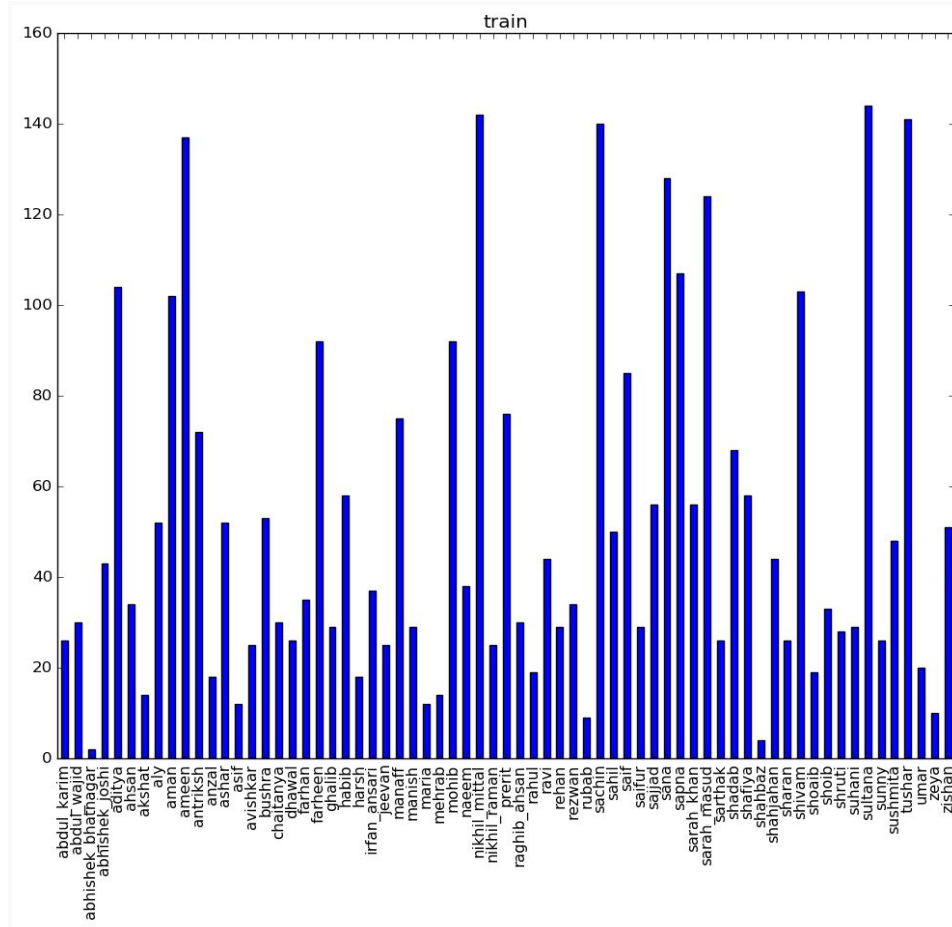


Figure 3. Distribution of Training Data

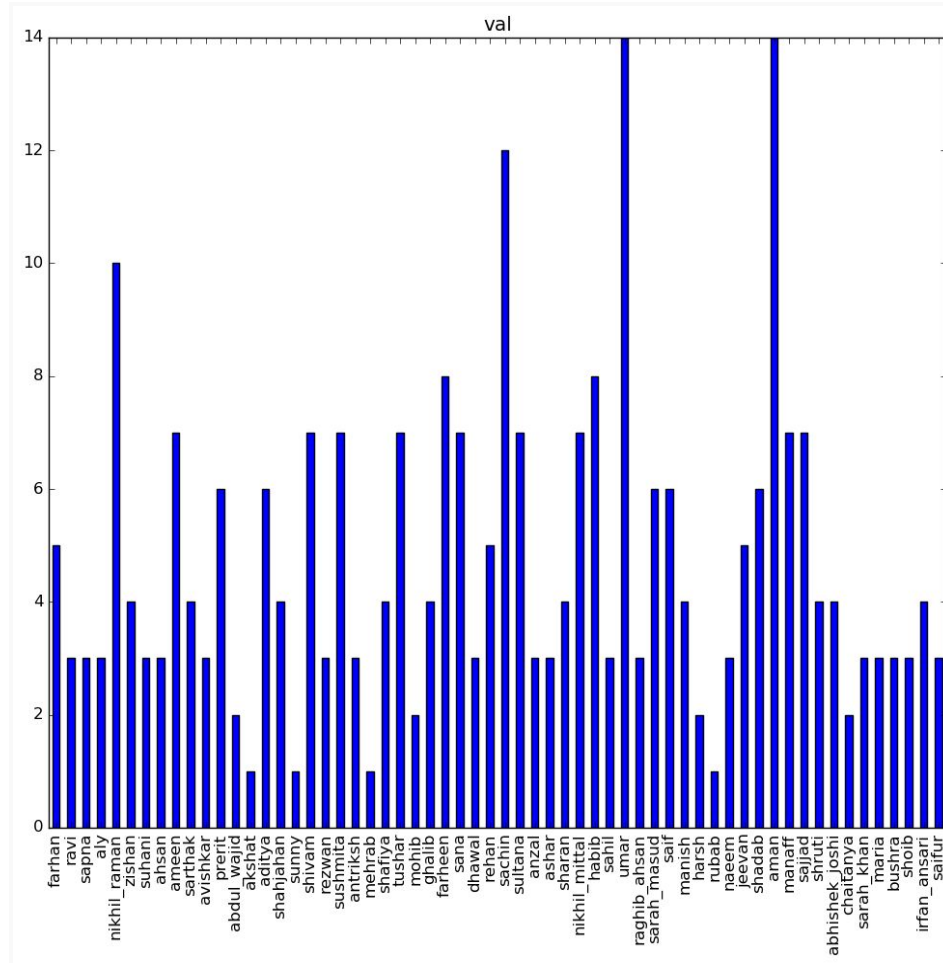


Figure 4. Distribution of Validation data

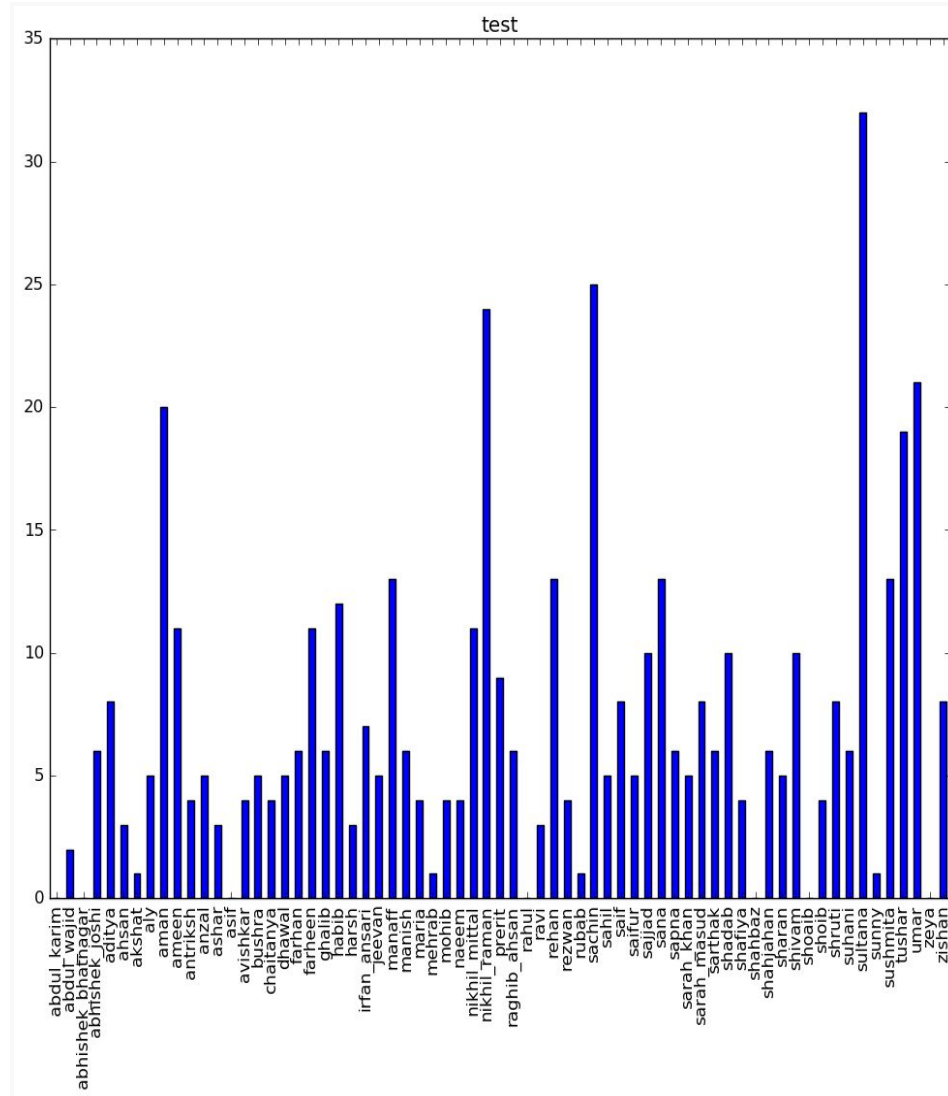


Figure 5. Distribution of Test Data

2.2. DETECTION

The image taken by the camera cannot be directly tested into the recognition network. The network has been trained on face images and will only predict proper results when a face is passed through the network. So, we need to detect the faces in the image before we pass the image through the network. For this a Cascade classifier using Haar feature has been used. The Haar Cascade classifier of OpenCV library has been used in Python language and is claimed to be a very good face detector.

2.2.1. HAAR CASCADE DETECTOR

Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in [1]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in *Figure 6*. are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

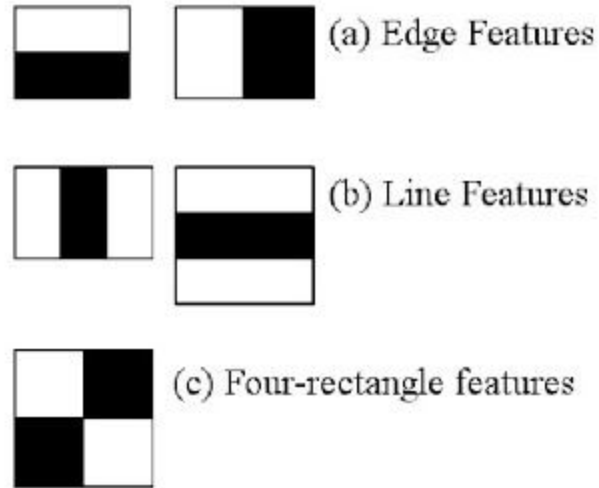


Figure 6. Haar Features

Now all possible sizes and locations of each kernel is used to calculate plenty of features. For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels.

But among all these features we calculated, most of them are irrelevant. For example, consider *Figure 7*. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or

any other place is irrelevant. So to select the best features out of 160000+ features Adaboost is used.

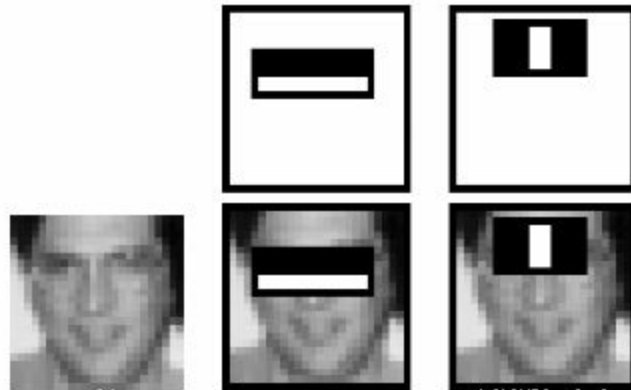


Figure 7. Working of Haar Detector

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images.

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper [1] says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

The detector used by [1] had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. According to [1], on an average, 10 features out of 6000+ are evaluated per sub-window.

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. We will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. and we have used one of those for face detection.

2.2.2. RESULTS

Some of the images with a lot of faces were tested with this detector in order to tell if it is actually good or not. *Figure 8.* shows some of the results. As can be seen in the figure, there are some detections which are not faces as well as some faces which are not detected, but most of the faces seem to ha



Figure 8. Detection of faces on the images with a lot of people

2.3. RECOGNITION

The major task of the software was to recognise the person in the image. This could have been carried out by various software implementations. There are researches in all types of Computer Vision trying to extract the features of a face to classify/recognise people. Recent developments in image classification have hinted the use of Convolutional Neural Networks. In all types of classification and recognition problems, convolutional neural networks (CNNs) have worked wonders, bringing down the error rate in object detection to about 3% on smart use of the convolutional layers. Thus, it was indeed a good development to be used in this project for better use of emerging trends.

2.3.1. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks for learning regular Neural Networks still apply.

Regular Neural Nets don't scale well to full images. For example, CIFAR-10 is a dataset (benchmark for object detection) in which images are only of size $32 \times 32 \times 3$ (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have $32 \times 32 \times 3 = 3072$ weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. $200 \times 200 \times 3$, would lead to neurons that have $200 \times 200 \times 3 = 120,000$ weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: **width, height, depth**. (Note that the word *depth* here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions $32 \times 32 \times 3$ (width, height, depth respectively). The neurons

in a layer of the CNN will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions $1 \times 1 \times 10$, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. *Figure 9.* Shows a visualisation of the same.

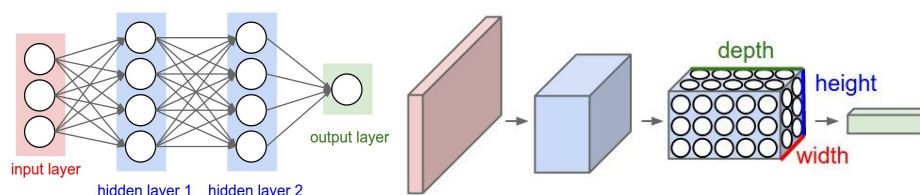


Figure 9. Left: A regular 3-layer Neural Network. Right: A ConvNet

The ConvNet in *Figure 9.* arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

2.3.2. VGG-NET

The network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet as in [2]. Its main contribution was in

showing that the depth of the network is a critical component for good performance.

Figure 10. shows the various ConvNet configurations Andrew Zisserman and Karen Simonyan in [2] tested on to finally show that the best architecture they achieved was ConvNet configuration D. It contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 10. VGG ConvNet Configurations

We used a pre-trained model of this 16-layer architecture which has been pre-trained on various faces due to limited computation available with us. This pre-training was again done by the same people, Karen Simonyan and Andrew Zisserman as in [3]. Using this model saves us weeks or maybe months of training from scratch.

2.3.3. TRAINING

The first set of training was performed on the first dataset we collected. During this time we had no GPU available hence the training had to be performed on a CPU system. For training, Theano [6] and Tensorflow [7] are two of the libraries we worked on during the course of the project. Theano has a very good space utilisation on GPU as well as CPU and is quite fast on GPU. Tensorflow on the other hand is the fastest on a CPU but when you have limited RAM, Tensorflow is not able to limit its resource utilisation. We had to train a VGG Net (140 M parameters) so, we had to use all the space possible as for storing the network and the least for batch size/ chunk size. Thus, we trained the VGG Net for the first dataset on a CPU with a batch size of 4 using Tensorflow to try to speed things up. Keras [5] is a development library developed as a cover for theano and tensorflow and was used in most part of the project.

The network was compiled with categorical crossentropy for loss function (we were using Softmax output) and nesterov Momentum was used. Learning rate was set to $2.5e-4$ which was calculated from [3] after reducing the batch size by a factor of 16 (reduce the learning rate by at least a factor of 4) and freezing all the layers except the final fully connected layers (train at about $\sim 1/10$ th of the initial learning rate). The learning rate was taken down to $1e-4$ if the accuracy on validation set

stopped increasing and then reduced by a factor of ten when loss on validation set stopped reducing. The weight decay was set to $5e-4$ all along according to [3] and the momentum was set to 0.9. *Figure 11* shows a training and loss curve for the same.

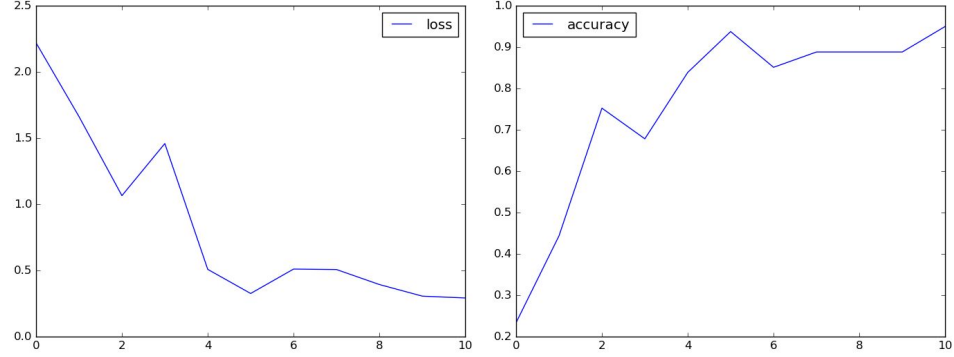


Figure 11. Loss and Accuracy curve for old dataset (10 epochs) on validation set

On obtaining the second dataset, we were given access to a 4GB memory GPU K5000 Quadro which helped us train the network faster. This time we had about 3500 training images which is again a very small dataset, so the above configuration of learning rate was kept intact. *Figure 12* shows the training curve of the network on the second dataset.

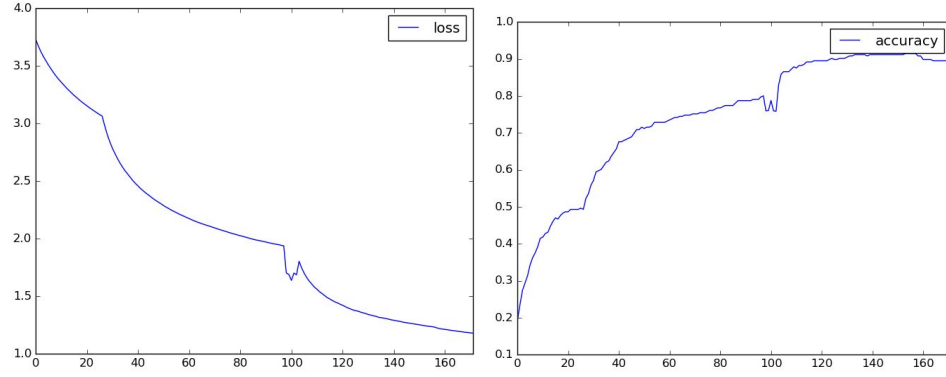


Figure 12. Loss and Accuracy Curve for new dataset (~200 epochs) on validation set

2.3.4. RESULTS

The paper [3] shows about 97% accuracy for the benchmark of Labelled Faces in the Wild, so it was expected that we reach about the same level of accuracy in recognition of the faces in our dataset. This was achieved with the first dataset while it could not be achieved with the second dataset. A glimpse of the classification results are shown in *Figure 13* and *Figure 14*.



Figure 13. Correct Predictions (from top right and moving left to right in a

row; abdul_wajid, abhishek_joshi, aly, aman, ameen, antriksh, anzal, avishkar, bushra, chaitanya, farhan, farheen, ghalib, habib, irfan_ansari, manaff, manish, nikhil_mittal, nikhil_raman, prerit, raghib_ahsan, rehan, rezwan, sachin, sahil, saif, saifur, sajjad, sana, sapna, sarah_khan, sarah_masud, sarthak, shahjahan, shivam, shruti)



Figure 14. Incorrect Predictions (going left to right; shoib predicted aly,

umar predicted antriksh, farheen predicted ashraf, sana predicted bushra, manaff predicted ghalib, raghib_ahsan predicted ghalib, chaitanya predicted habib, mohib predicted habib, nikhil_mittal predicted habib, raghib_ahsan predicted habib, suhani predicted nikhil_raman, sajjad predicted habib)

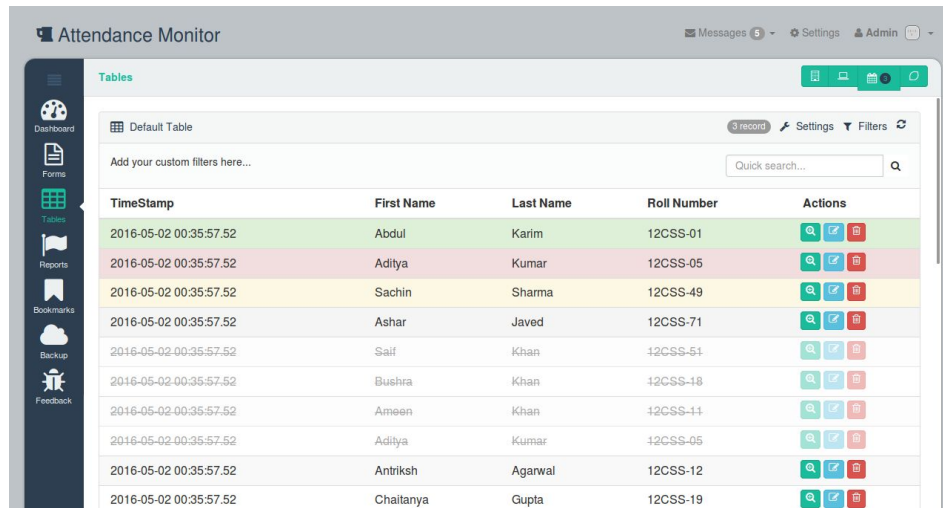
As we can see in *Figure 13* and *Figure 14* the dataset in figure 14 was more likely to be not recognised correctly owing to the blurred images, or half hidden faces in the images or completely dark images. Even though this was what we were looking for in recognition (due to the fact that the area outside the labs might not be illuminated every time or the images of each person may be blurred etc.), we did find a lot of blurred images also recognised correctly. Those which were recognised

incorrectly were mostly which we ourselves weren't able to recognise.

Thus, the results seemed to be pretty good.

2.4. INTERFACE

Since this is a development project, it was important that we provide an interface to the application for ease of use. Hence, a web interface of the application has been developed and is in working condition. It connects all the recognitions to the database and logs the activities in and out of the labs. The final setup of the hardware and interface is what is being done here.



The screenshot displays the 'Attendance Monitor' web application. On the left is a dark sidebar with navigation icons for Dashboard, Forms, Tables (highlighted), Reports, Bookmarks, Backup, and Feedback. The main content area is titled 'Tables' and shows a 'Default Table' with 3 records. Above the table is a search bar labeled 'Quick search...'. The table has five columns: TimeStamp, First Name, Last Name, Roll Number, and Actions. Each row represents an attendance record with a unique timestamp and roll number, and the Actions column contains three icons (search, edit, delete).

TimeStamp	First Name	Last Name	Roll Number	Actions
2016-05-02 00:35:57.52	Abdul	Karim	12CSS-01	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Aditya	Kumar	12CSS-05	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Sachin	Sharma	12CSS-49	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Ashar	Javed	12CSS-71	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Saif	Khan	12CSS-51	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Bushra	Khan	12CSS-18	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Ameen	Khan	12CSS-11	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Aditya	Kumar	12CSS-05	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Antriksh	Agarwal	12CSS-12	[Search] [Edit] [Delete]
2016-05-02 00:35:57.52	Chaitanya	Gupta	12CSS-19	[Search] [Edit] [Delete]

Figure 15. Interface for logs

The screenshot displays the 'Attendance Monitor' web application. On the left is a dark sidebar with icons for Dashboard, Forms (highlighted), Tables, Reports, Bookmarks, Backup, and Feedback. The top header includes the title 'Attendance Monitor' and links for Messages, Settings, and Admin. The main content area is titled 'Forms' and contains a 'Form Default' section. This section includes a title 'Enter New Student Information' and several input fields: 'Name' (with placeholder 'Enter username'), 'Password' (with placeholder 'Enter password'), 'Roll Number' (with placeholder '.help-block' and a note 'Example block-level help text here.'), 'Image Upload' (with a 'Browse...' button and text 'No file selected.'), 'Year' (with placeholder '.input-lg'), and 'Date Of Birth'.

Figure 16. Interface for adding people to the database

3. CONCLUSION

A system has been developed which will detect, recognise and mark the attendance of people entering the labs automatically. This system is yet to be setup and run for a whole lab period in order to check its real time performance and upgrade the system as needed for future development. This system has to be up and running all the time in order to be able to train the ConvNet when the system is not in use. The system performs better in proper light condition and we hope that during lab times, which it is going to perform as good. Over other types of networks this networks is fast and performs predictions of about 80 images in about 120 seconds on a system with a RAM of 4GB.

4. FUTURE SCOPE

The recognition network i.e. VGG uses a lot of space for storing its model as well as for training of network. It would be better to upgrade to some better versions of a similar neural network such as Residual Networks by Kaiming He. The detection algorithm we employed is not as good as it was expected to be. Maybe, if trained once or twice on the face detection datasets it might improve. If not, it is possible to replace it with a setup of faster RCNN which has been specifically developed for object detection with an mAP of 69.9%. If the system does not perform well in real time, one of the improvements is to replace the VGG Net with Residual Networks by Kaiming He. Even faster RCNN is detects and predicts in real time in about 0.2 seconds on a CPU with ample RAM for the network to be able to run.

5. REFERENCES

- [1] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, 2001.
- [2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
- [3] Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition." *Proceedings of the British Machine Vision* 1.3 (2015): 6.
- [4] Holovaty, Adrian, and Jacob Kaplan-Moss. *The definitive guide to Django: Web development done right*. Apress, 2009.
- [5] Chollet, François. "Keras: Theano-based deep learning library." *Code: <https://github.com/fchollet>. Documentation: <http://keras.io>* (2015).
- [6] Bastien, Frédéric, et al. "Theano: new features and speed improvements." *arXiv preprint arXiv:1211.5590* (2012).
- [7] Abadi, Martin, et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." *arXiv preprint arXiv:1603.04467* (2016).