

Project Report On

Smart Attendance Monitoring System

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

Bachelor of Technology
in

***Electronics and Communication
Engineering***

By

**Amal A A (RET16EC023)
Anwin Antony(RET16EC040)
Christy Augustine(RET16EC062)**

**Under the guidance of
Ms. Ramitha Rajesh**



RSET

**RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY**

**Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Rajagiri Valley, Kakkanad, Kochi, 682039
2020**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039



CERTIFICATE

*Certified that project work entitled "**Smart Attendance Monitoring System**" is a bonafide work done by **Amal A A** , **Anwin Antony** , **Christy Augustine** of Eight Semester Electronics and Communication Engineering, in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Electronics and Communication Engineering from APJ Abdul Kalam Technological University, Kerala during the academic year 2019-2020.*

Dr.Rithu James
Head of Department
Associate.Professor
Dept. of ECE
RSET

Ms. Ramitha Rajesh
Seminar Guide
Asst. Professor
Dept. of ECE
RSET

ACKNOWLEDGEMENT

We have immense pleasure and privilege in submitting our project report entitled "Smart Attendance Monitoring System".

We are extremely thankful to our Principal Dr.P.S Sreejith for giving us his consent for this project.

We are also thankful to our Head of the Department Dr. Rithu James, whose help and guidance has been a major factor in completing my journey.

We would specially like to thank our guide, Ms. Ramitha Rajesh, Asst. Professor, Department of Electronics and Communication Engineering, who was a great mentor throughout this project. We thank her for the immense support.

Growth is never by mere chance; it is the result of forces working together. Our seminar is the culmination of many forces joining hands together. For this we thank all our friends for the support and encouragement they have given us during the course of our work.

Above all, we thank God Almighty for his immense love and grace which enabled us to complete this project.

ABSTRACT

In present academic system, regular class attendance of students plays a significant role in performance assessment and quality monitoring. The conventional methods practised in most of the institutions are by calling names or signing on papers, which is highly time-consuming and insecure. Manual maintenance of attendance is inefficient because it takes away a lot of lecture hours and it is prone to proxies or impersonations. The solution to this problem is to make an automatic attendance management system for convenience or data reliability. Smart attendance monitoring system is developed for daily student attendance in colleges or schools. It attempts to record attendance through face detection and recognition. The system is developed by the integration of ubiquitous components to make a portable device for managing the students attendance using face recognition technology. In this study a system monitoring the presence of a student in a class is developed. The proposed face recognition system is based on machine learning. In this experiment AlexNet was used as CNN for detecting the faces successfully.

Contents

| | |
|---|------------|
| Acknowledgements | ii |
| Abstract | iii |
| List of Figures | vi |
| 1 Introduction | 1 |
| 1.1 Scopes and Objectives | 1 |
| 1.2 Overview of Face Recognition | 2 |
| 1.3 Applications | 2 |
| 1.4 Organisation of the Report | 3 |
| 1.5 Summary | 3 |
| 1.6 Major Contributions | 3 |
| 2 Literature Survey | 5 |
| 2.1 A Counterpart Approach to Attendance and Feedback Sys- tem using Machine Learning Techniques | 5 |
| 2.2 Automated Attendance System Using Face Recognition . . . | 5 |
| 2.3 Student Attendance System Using Iris Detection | 5 |
| 2.4 Face Recognition-based Lecture Attendance System | 6 |
| 3 OpenCV | 7 |
| 3.1 Introduction | 7 |
| 3.2 Python OpenCV | 7 |
| 4 Face Detection | 9 |
| 4.1 Overview | 9 |
| 4.2 Applications | 10 |
| 5 Face Detection using Haar Cascades | 11 |
| 5.1 Overview | 11 |
| 5.2 Haar-cascade Detection in OpenCV | 11 |
| 6 Image Preprocessing | 13 |
| 6.1 Overview | 13 |
| 6.2 Parameters | 13 |
| 6.2.1 Pose estimation | 13 |
| 6.2.2 Sharpness | 14 |
| 6.3 Image size or resolution | 15 |
| 6.3.1 Brightness | 15 |
| 6.3.2 Final Score | 15 |
| 6.4 Face landmark detection | 16 |
| 7 Face Recognition | 18 |
| 7.1 Introduction | 18 |
| 7.2 Deep Learning | 18 |

| | | |
|-----------|--|-----------|
| 8 | Convolutional Neural Network | 21 |
| 8.1 | Introduction | 21 |
| 8.2 | Architecture | 21 |
| 8.2.1 | Convolutional layer | 21 |
| 8.2.2 | Pooling layer | 22 |
| 8.2.3 | Fully Connected layer | 23 |
| 8.2.4 | Receptive Field | 23 |
| 8.2.5 | Weights | 23 |
| 8.3 | Different types of CNN architectures | 23 |
| 9 | Experiment and Results | 24 |
| 9.1 | Experiment | 24 |
| 9.1.1 | Dataset creation | 24 |
| 9.1.2 | Dataset Filtering | 24 |
| 9.1.3 | Dataset Training | 24 |
| 9.1.4 | Dataset testing | 24 |
| 9.2 | Result | 25 |
| 10 | Flow chart | 27 |
| 11 | Conclusion | 28 |
| | References | 29 |

List of Figures

| | | |
|------|---|----|
| 1.1 | Pipe line of the system | 1 |
| 4.1 | General face detection system | 9 |
| 5.1 | Common Haar Features | 12 |
| 6.1 | Roll, Yaw and Pitch | 13 |
| 6.2 | Head-pose estimation | 14 |
| 6.3 | Weights for various parameters in computing FQA | 15 |
| 6.4 | Sets of 68 2D landmarks visualised on mean head from different poses. | 16 |
| 6.5 | Visualizing the 68 facial landmark coordinates | 16 |
| 7.1 | Deep learning neural network | 19 |
| 7.2 | Block diagram of the deep neural network | 20 |
| 8.1 | Convolutional nueral network | 22 |
| 9.1 | Python Scripts | 24 |
| 9.2 | Detected Face | 25 |
| 9.3 | Skelton Image | 25 |
| 9.4 | Final output | 26 |
| 10.1 | Flow diagram | 27 |

Chapter 1

Introduction

1.1 Scopes and Objectives

Attendance is defined as the action or state of going regularly to or being present at a place or event. Attendance of every student is being maintained by schools and colleges. The manual attendance record system is inefficient and more time is required to record as well as calculate the attendance of each student. Hence a system is needed which will solve the issue of manual attendance. While the move towards the digital era is being accelerated every hour, biometrics technologies have started affecting people's daily life at each and every instance.

Biometrics technologies use characteristics such as fingerprints, faces, irises, retinal patterns, palm prints, voice, handwritten signatures, and so on for authentication. These techniques employing physical data are increasingly seen as an efficient alternative to conventional security methods such as a password or ID cards. The biometric personal authentication uses data taken from measurements. Such data is unique to the individual and remains so throughout one's life. It is important to identify the correct tools to use in commercial and scientific studies. Barcode readers, Radio Frequency Identification (RFID) system, Bluetooth and NFC (Near Field Communication) are just a few of the examples of such tools. However, they are expensive and therefore they had limited use. Hence a system which does not require a special infrastructure is developed using biometric facial detection and recognition system – Smart Attendance Monitoring System.

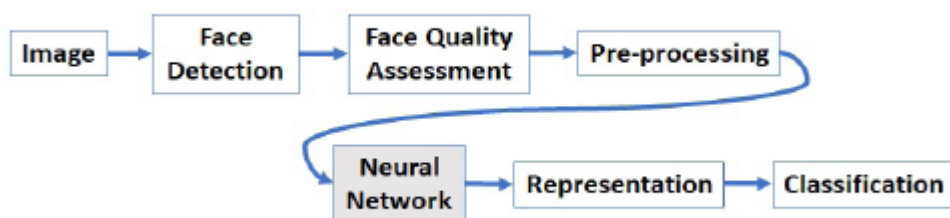


Figure 1.1: Pipe line of the system

1.2 Overview of Face Recognition

With the continuous development of information technology, the demand for security and safety is gradually improving. For the consideration of security, face recognition has been studied for many decades. With the development of information technology, face recognition is widely used in daily life, especially in security systems, information security, human-computer interaction.

Researches are committed to improving the recognition accuracy and response speed of the face recognition system. The state-of-art of face recognition has been significantly improved by the appearance of deep learning. Although these systems perform well on large amounts of web collected facial data, the performance and accuracy are still limited when they are applied in actual scenarios. There is still a long way to go to improve the recognition accuracy of face recognition system in real scenarios.

1.3 Applications

Face recognition systems identify people by their face images. Face recognition systems establish the presence of an authorized person rather than just checking whether a valid identification (ID) or key is being used or whether the user knows the secret personal identification numbers (Pins) or passwords.

In many of the access control applications, such as office access or computer logon, the size of the group of people that need to be recognized is relatively small. The face pictures are also caught under natural conditions, such as frontal faces and indoor illumination. The face recognition system of this application can achieve high accuracy without much co-operation from user.

Today more than ever, security is a primary concern at airports and for airline staff office and passengers. Airport protection systems that use face recognition technology have been implemented at many airports around the world.

It is also used in image database investigations for searching image databases of licensed drivers, benefit recipients, missing children, immigrants and police bookings.

Like security applications in public places, surveillance by face recognition systems has a low user satisfaction level, if not lower. Free lighting conditions, face orientations and other divisors all make the deployment of face recognition systems for large scale surveillance a challenging task.

1.4 Organisation of the Report

In this report, the Chapter 1, i.e. this chapter gives an overall idea of what this project is all about. It deals with the scope, objective, major contributions of this project, and an overall view of organizing the report.

In Chapter 2, a review of existing attendance monitoring systems using image processing are presented.

Chapter 3 gives an idea about OpenCV.

Chapter 4 discusses Face detection.

Chapter 5 describes Face detection using Haar Cascades.

Chapter 6 explains image preprocessing.

Chapter 7 gives an idea about Face recognition.

Chapter 8 focuses on CNN.

Chapter 9 shows the experiments and results.

Chapter 10 explains the flow chart.

Chapter 11 contains the conclusion.

1.5 Summary

This first chapter was an overview of the project which deals with introduction to face recognition , and it gives an idea about the contents of other chapters.

1.6 Major Contributions

The project work started by the study of Python programming language for coding. This was followed by understanding the basic concepts of OpenCV. A detailed study about various face detection processes was carried out before starting the project. Face detection using Haar cascade classifiers which was proposed by Viola and Jones was thoroughly studied and was implemented.

This was followed by the study of the facial landmarks and its representation. Then a study on how to extract these features were carried out. These image skeletons which were obtained from the detected faces were preprocessed before giving them as input to the CNN.

Then by understanding deep learning, a detailed study about convolutional neural networks and its different architectures was done. It was found that AlexNet was more suitable for image processing purposes. The database was collected and the CNN was trained to get accurate results.

Chapter 2

Literature Survey

Automatic face recognition technologies have made many improvements in the changing world. Smart attendance using real-time face recognition is a real-world solution which comes with day to day activities of handling student attendance system. Face recognition-based attendance system is a process of recognizing the students face for taking attendance by using face biometrics based on high - definition monitor video and other information technology.

2.1 A Counterpart Approach to Attendance and Feedback System using Machine Learning Techniques

This system automatically detects the student performance and maintains the student's records like attendance and their feedback on the subjects like Science, English, etc. Therefore the attendance of the student can be made available by recognizing the face. On recognizing, the attendance details and details about the marks of the student is obtained as feedback.

2.2 Automated Attendance System Using Face Recognition

Automated Attendance System using Face Recognition proposes that the system is based on face detection and recognition algorithms, which is used to automatically detects the student face when he/she enters the class and the system is capable to marks the attendance by recognizing him. Viola-Jones Algorithm has been used for face detection which detect human face using cascade classifier and PCA algorithm for feature selection and SVM for classification. When it is compared to traditional attendance marking this system saves the time and also helps to monitor the students.

2.3 Student Attendance System Using Iris Detection

In this system the student is requested to stand in front of the camera to detect and recognize the iris, for the system to mark attendance for the student. Some algorithms like Gray Scale Conversion, Six Segment Rectangular Filter, Skin Pixel Detection is being used to detect the iris. It helps in preventing the proxy issues and it maintains the attendance of the student in an effective manner, but in one of the time-consuming process for a student or a staff to wait until the completion of the previous members.

2.4 Face Recognition-based Lecture Attendance System

The face recognition-based lecture attendance system takes the attendance automatically recognition obtained by continuous observation. Continuous observation helps in estimating and improving the performance of the attendance. To obtain the attendance, positions and face images of the students present in the class room are captured. Through continuous observation and recording the system estimates seating position and location of each student for attendance marking. The work is focused on the method to obtain the different weights of each focused seat according to its location. The effectiveness of the picture is also being discussed to enable the faster recognition of the image.

Chapter 3

OpenCV

3.1 Introduction

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez .

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C, Perl, Ch, Haskell, and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

3.2 Python OpenCV

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

Chapter 4

Face Detection

4.1 Overview

Face detection is considered as a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution.

Object detection is one of the computer technologies, which connected to the image processing and computer vision and it interacts with detecting instances of an object such as human faces, building, tree, car, etc. The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

Face detection is basically an image segmentation process as the image is to be segmented into two parts: one containing faces and the other representing non-face regions. Face detection takes images/video sequences as input and locates face areas within these images. This is done by separating face areas from non-face background regions. Facial feature extraction locates important feature (eyes, mouth, nose and eye-brows) positions within a detected face. In general face detection system input image is passed to the system for pre-processing. Image may vary in format, size and resolution and can include frames of video. In the next step pre-processing is done, which normalized the image and also remove noise. The classifier decides face and non-face class based on information learned from during training. Finally, the output locates face region.

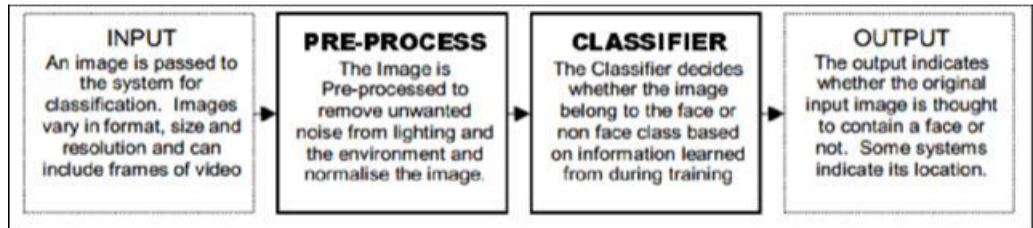


Figure 4.1: General face detection system

4.2 Applications

Face detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc. It is used to detect faces in real time for surveillance and tracking of person or objects. It is widely used in cameras to identify multiple appearances in the frame Ex- Mobile cameras and DSLR's. Facebook is also using face detection algorithm to detect faces in the images and recognise.

Chapter 5

Face Detection using Haar Cascades

5.1 Overview

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. 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 below image 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. Now all possible sizes and locations of each kernel is used to calculate plenty of features. But among all these features calculated, most of them are irrelevant. For this, each and every feature is applied 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. The features with minimum error rate are selected, 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.

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 purpose the concept of Cascade of Classifiers was introduced. Instead of applying all the 6000 features on a window, features were grouped into different stages of classifiers and applied one-by-one.

5.2 Haar-cascade Detection in OpenCV

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class in digital images and videos. OpenCV

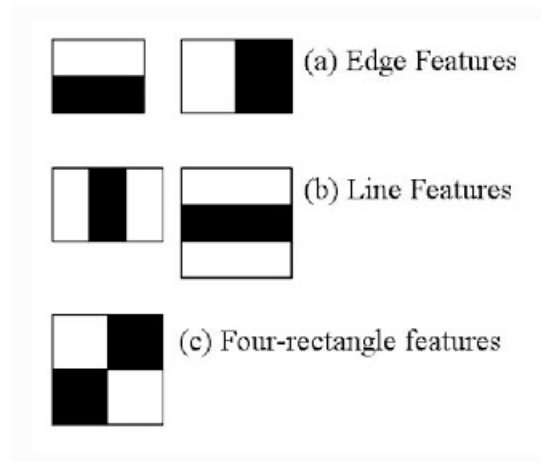


Figure 5.1: Common Haar Features

comes with a trainer as well as detector. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc.

Chapter 6

Image Preprocessing

6.1 Overview

Pre-processing is a common name for operations with images at the lowest level of abstraction - both input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

6.2 Parameters

6.2.1 Pose estimation

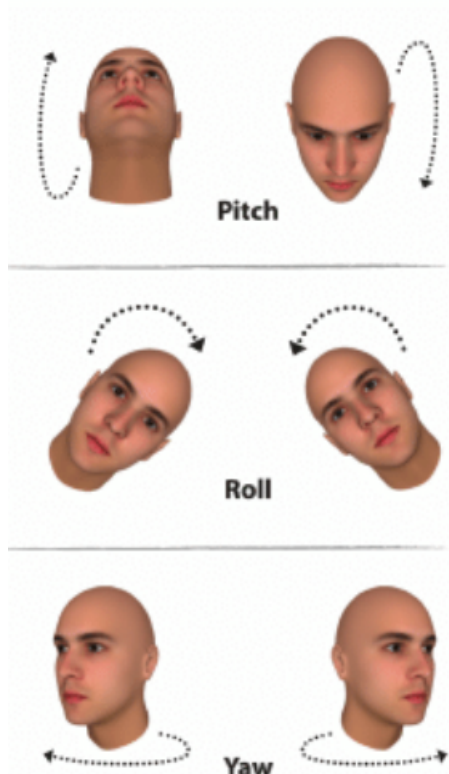


Figure 6.1: Roll, Yaw and Pitch

Since people move around and look at different directions in front of the real-time camera, it is possible to have a wide range of head poses oriented at different angles. But for the sake of biometrics, it is important to have least rotated face as a standout in the entire face-log. Thus it is important to include this feature in face quality assessment. We determined the head pose using three angles: Roll, Yaw, Pitch. All these angles are typically between - 90 to + 90.

The roll and pitch are adjusted by aligning technique during face-log generation, so the only concern is yaw angle. Using face landmarks detection, the coordinates of nose tip and also the point between the eyebrows are calculated. If (x_1, y_1) and (x_2, y_2) are such points, then yaw angle is computed as:

$$yaw = abs(\arctan 2(y_2 - y_1, x_2 - x_1)) \quad (6.1)$$

| Range Of Yaw Angle | 0 to 10 | 10 to 20 | 20 to 30 | More than 30 |
|--------------------|---------|----------|----------|--------------|
| NHP | 1 | 0.5 | 0.33 | 0.25 |

Figure 6.2: Head-pose estimation

6.2.2 Sharpness

It is very likely to have blurry images in real time video sequences because the faces are moving. Thus it is important to include this feature in face quality assessment. To compute the sharpness of an image, the variance of an image Laplacian is utilized. This is defined as:

$$Sharpness = \sum_{(i,j) \in \mathcal{U}(x,y)} (\Delta I(i,j) - \overline{\Delta I})^2 \quad (6.2)$$

where $\overline{\Delta I}$ is the mean value of image Laplacian within $\mathcal{U}(x,y)$. A grey-scale channel of an image is convolved with the (3x3) kernel and took the variance of that result.

$$kernel = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (6.3)$$

If the variance falls below a pre-defined threshold, then the image is considered as a blur. This threshold depends on the working environment and can be set accordingly. Then it is normalized with a pre-defined threshold,

$$NormalizedSharpness(NS) = \frac{Sharpness}{threshold} \quad (6.4)$$

6.3 Image size or resolution

Since the face tracking technique is employed, the camera tracks the face as long as the face is in the scene. But, as the face moves far away from the camera, there will be a large distance between the camera and the face. Thus, the size of the face becomes smaller. Thus, it is important to include this feature in face quality assessment. Using face landmark detection, the position of the eye corners in a face is calculated. Let (x_L, y_L) be the coordinates of the left eye corner and (x_R, y_R) be the coordinates of the right eye corner. The distance between them is given by:

$$Resolution = \sqrt{(x_L - x_R)^2 + (y_L - y_R)^2} \quad (6.5)$$

Then the obtained resolution is normalised with the threshold for interocular distance as,

$$NormalizedResolution(NR) = \frac{Resolution}{threshold} \quad (6.6)$$

Larger the distance, smaller will be the size of the face and hence, lesser will be the resolution.

| Parameter | Head-pose | Sharpness | Brightness | Resolution |
|-----------|-----------|-----------|------------|------------|
| Weight | 17 | 9 | 6 | 8 |

Figure 6.3: Weights for various parameters in computing FQA

6.3.1 Brightness

Changes in Lightening conditions are quite common in real-time applications like surveillance cameras. It is easier to apply Local feature extraction on brighter faces than on darker faces. Thus, it is necessary to include this feature in face quality assessment. To obtain this parameter, we calculated the mean of all the intensities of various channels (R, G, B) present in the image.

$$Brightness = (R + G + B)/3 \quad (6.7)$$

$$NormalizedBrightness(NB) = Brightness/100 \quad (6.8)$$

6.3.2 Final Score

In order to get the best quality image in the real-time video sequence, we need to assign weights to each of the normalized parameters (NHP, NS, NR, NB). We gave the highest priority to the head pose followed by other parameters as shown in Table II. So, we computed the Face Quality Assessment (FQA) as,

$$FQA = NHP*17 + NS*9 + NB*6 + NR*8 \quad (6.9)$$

Greater is the value of FQA, greater will be the quality of face which will be stored in Face-log. We can set the threshold for these FQA values of generated images to prevent the bad quality images from being stored in facelog.

6.4 Face landmark detection

Facial landmark points capture rigid and non-rigid deformation of faces in a very compact description and are therefore valuable for many different face analysis tasks. For face recognition or different categorisation tasks such as gender, age, ethnicity or expressions, a rough pose normalisation is needed in order to apply other algorithms. For 3D face tracking or reconstruction facial landmarks are often used as initialisation for more sophisticated approaches. As input, landmark detectors use the image itself and a face box provided by a previous detector. In the field these consecutive detection steps are often looked at separately.

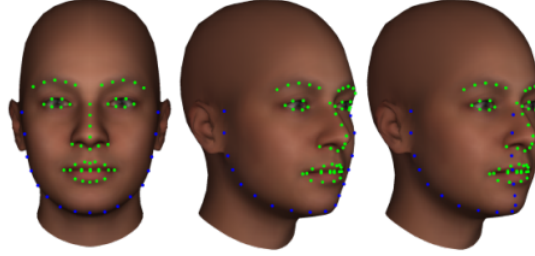


Figure 6.4: Sets of 68 2D landmarks visualised on mean head from different poses.



Figure 6.5: Visualizing the 68 facial landmark coordinates

One focus of this work is to evaluate the entire landmarking pipeline including face detectors. We evaluate how precision, robustness and temporal stability of landmark detectors change using face boxes provided by different face detectors. Temporal stability of landmarks is an issue becoming increasingly important with more applications shifting from single images to videos. There are only a few datasets with ground truth landmarks for videos. Landmarking methods are tested exhaustively in several experiments and compared regarding their performance.

Face landmark detection is the process of finding points of interest in an image of a human face. Landmark detection starts with face detection, finding faces in the image and their extents (bounding boxes). Facial detection has long been considered a solved problem, and OpenCV contains one of the first robust face detectors freely available to the public.

Chapter 7

Face Recognition

7.1 Introduction

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems can be used to identify people in photos, video, or in real-time.

Face recognition systems use computer algorithms to pick out specific, distinctive details about a person's face. These details, such as distance between the eyes or shape of the chin, are then converted into a mathematical representation and compared to data on other faces collected in a face recognition database. The data about a particular face is often called a face template and is distinct from a photograph because it's designed to only include certain details that can be used to distinguish one face from another.

Some face recognition systems, instead of positively identifying an unknown person, are designed to calculate a probability match score between the unknown person and specific face templates stored in the database. These systems will offer up several potential matches, ranked in order of likelihood of correct identification, instead of just returning a single result. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape.

7.2 Deep Learning

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

The adjective "deep" in deep learning comes from the use of multiple layers in the network. Early work showed that a linear perceptron cannot be a universal classifier, and then that a network with a nonpolynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

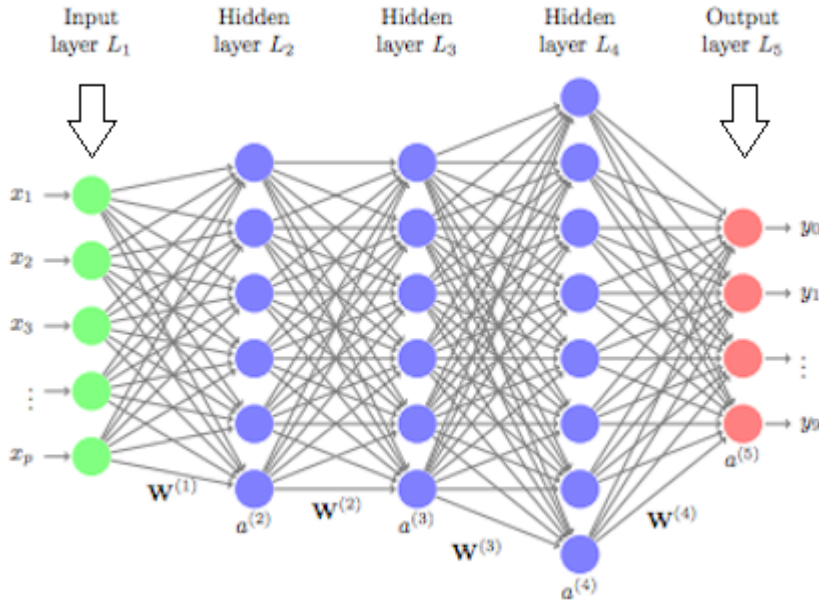


Figure 7.1: Deep learning neural network

Most modern deep learning models are based on artificial neural networks, specifically, Convolutional Neural Networks (CNN)s, although they

can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

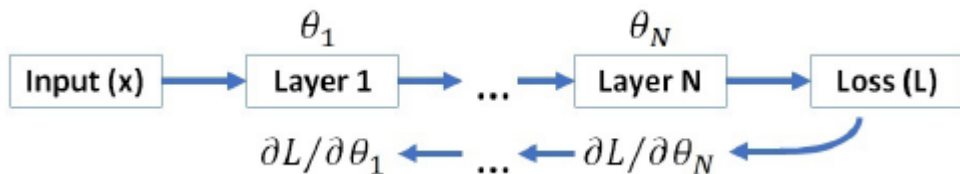


Figure 7.2: Block diagram of the deep neural network

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

Chapter 8

Convolutional Neural Network

8.1 Introduction

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

8.2 Architecture

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a sliding dot product or cross-correlation. This has significance for the indices in the matrix, in that it affects how weight is determined at a specific index point.

8.2.1 Convolutional layer

When programming a CNN, the input is a tensor with shape (number of images) x (image height) x (image width) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes:

[i] Convolutional kernels defined by a width and height (hyper-parameters).

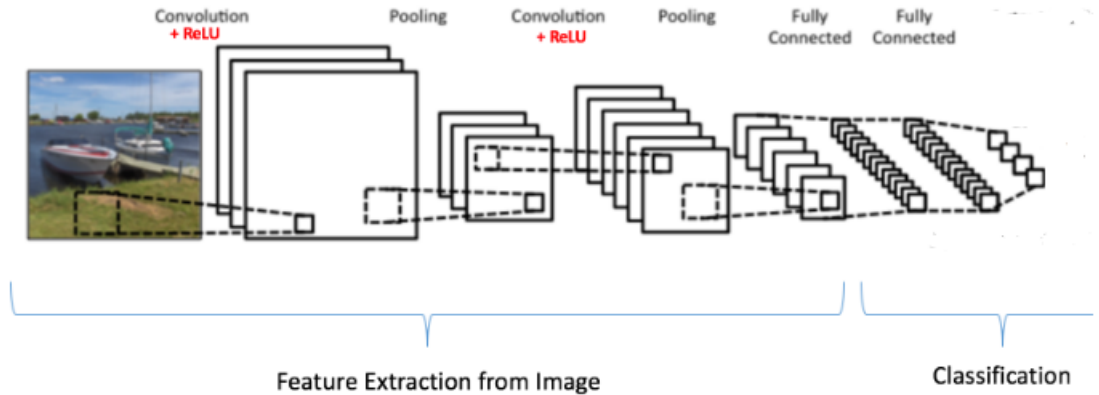


Figure 8.1: Convolutional neural network

- [ii] The number of input channels and output channels (hyper-parameter).
- [iii] The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100×100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5×5 , each with the same shared weights, requires only 25 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks are avoided.

8.2.2 Pooling layer

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2×2 . Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a

cluster of neurons at the prior layer.

8.2.3 Fully Connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

8.2.4 Receptive Field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape. The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer. The subarea of the original input image in the receptive field is increasingly growing as getting deeper in the network architecture. This is due to applying over and over again a convolution which takes into account the value of a specific pixel, but also some surrounding pixels.

8.2.5 Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning, in a neural network, progresses by making iterative adjustments to these biases and weights.

The vector of weights and the bias are called filters and represent particular features of the input. A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces memory footprint because a single bias and a single vector of weights are used across all receptive fields sharing that filter, as opposed to each receptive field having its own bias and vector weighting.

8.3 Different types of CNN architectures

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them are:

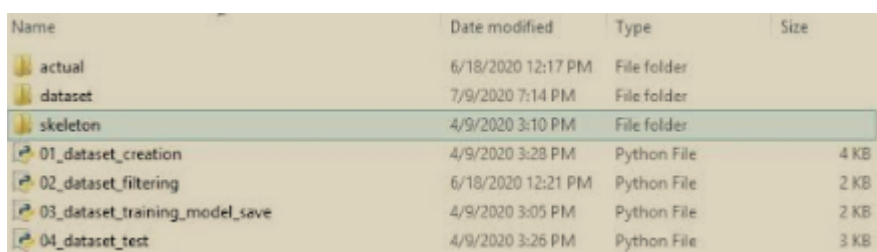
- [i] LeNet
- [ii] AlexNet
- [iii] VGGNet
- [iv] GoogLeNet
- [v] ResNet
- [vi] ZFNet

Chapter 9

Experiment and Results

9.1 Experiment

We will be mainly using 4 python scripts to perform the face recognition process.



| Name | Date modified | Type | Size |
|--------------------------------|--------------------|-------------|------|
| actual | 6/18/2020 12:17 PM | File folder | |
| dataset | 7/9/2020 7:14 PM | File folder | |
| skeleton | 4/9/2020 3:10 PM | File folder | |
| 01_dataset_creation | 4/9/2020 3:28 PM | Python File | 4 KB |
| 02_dataset_filtering | 6/18/2020 12:21 PM | Python File | 2 KB |
| 03_dataset_training_model_save | 4/9/2020 3:05 PM | Python File | 2 KB |
| 04_dataset_test | 4/9/2020 3:26 PM | Python File | 3 KB |

Figure 9.1: Python Scripts

9.1.1 Dataset creation

The first step is to detect the face of a person. For this we are using Haar cascade classifier which uses different face parameters like pose estimation, resolution, brightness and sharpness to detect the face.

9.1.2 Dataset Filtering

From the detected face we are taking out facial landmark points. This landmark points is termed as skeleton points. Here we are taking 10 different skeleton images of a particular person.

9.1.3 Dataset Training

Once the skeleton images are collected then it is fed to CNN(Convolutional Neural Network). In our project we are using AlexNet as our CNN. So the desired face is fed to the CNN for recognition. Here 10 different skeleton images are fed to the CNN for training.

9.1.4 Dataset testing

Finally the presence of person whose face is recognised by the CNN when an image is provided as input is recorded in MySQL.

9.2 Result

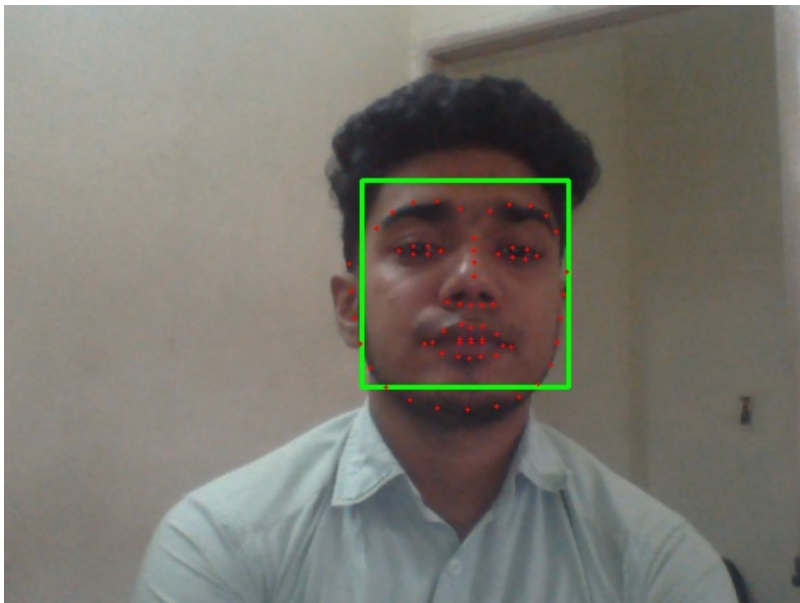


Figure 9.2: Detected Face



Figure 9.3: Skelton Image

SAMS has been designed to register the face of each individual for the first time. Once done, the network trains it automatically for future usage.

| Attendance Monitoring | | | |
|-----------------------|-------|------------|----------|
| ID | Name | Date | Time |
| 7 | Amwin | 09/07/2020 | 20:01:29 |

Figure 9.4: Final output

Chapter 10

Flow chart

Images from a live stream are passed as input to the system. These images are converted to greyscale. From the greyscale images features of the face are extracted. . The features are then compared with existing records to check if there is a match. If the face matches it is displayed and output is in the form of attendance being marked for the person whose face was recognized.

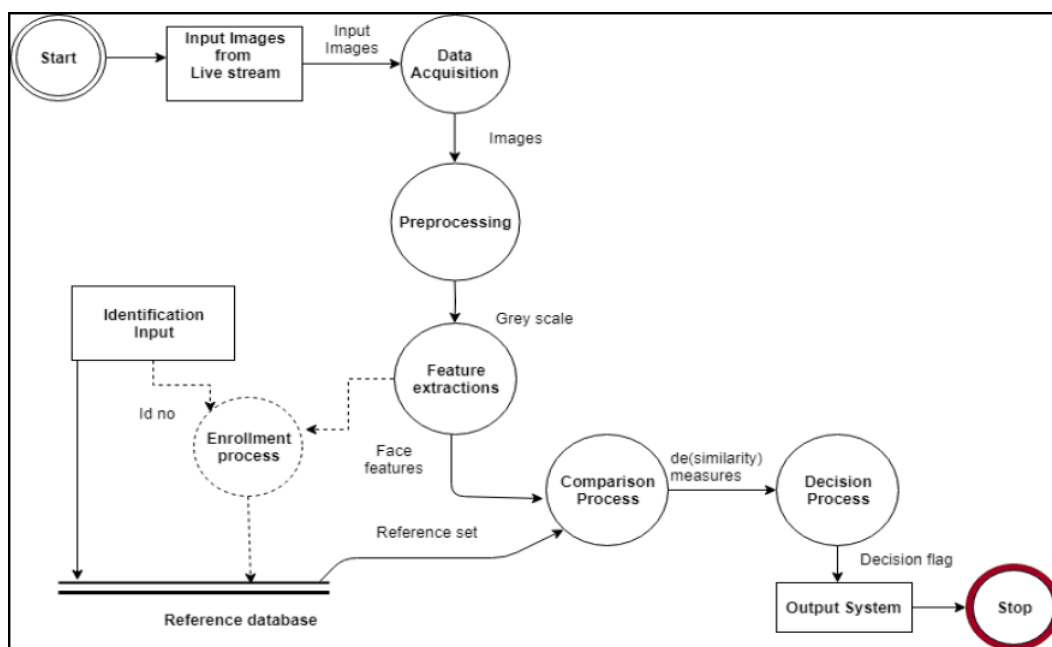


Figure 10.1: Flow diagram

Chapter 11

Conclusion

Smart attendance monitoring system is designed to solve the issues of existing manual systems. We have used face recognition concept to mark the attendance of student and make the system better. The system performs satisfactory in different poses and variations. In future this system need be improved because these system sometimes fails to recognize students from some distance, also we have some processing limitation, working with a system of high processing may result even better performance of this system.

This system perform satisfactorily with different facial expressions, lighting and pose of the person. There is room for improvement since these systems sometimes fail to recognize every face student present in the classroom. We have made the device portable for easy use even when the sessions are on, without disturbing the class. There are future scopes to make a more compact ergonomics to make it a more user-friendly product to make an impact in building a more healthier academic environment.

References

- [1] Li, Xiang-Yu, and Zhen-Xian Lin. "Face recognition based on HOG and fast PCA algorithm." The Euro-China Conference on Intelligent Data Analysis and Applications. Springer, Cham, 2017.
- [2] Rekha, N., and M. Z. Kurian. "Face detection in real time based on HOG." International Journal of Advanced Research in Computer Engineering Technology (IJARCET) 3.4 (2014): 1345-1352.
- [3] Kwolek, Bogdan. "Face detection using convolutional neural networks and Gabor filters." International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2005.
- [4] Kar, Nirmalya, et al. "Study of implementing automated attendance system using face recognition technique." International Journal of computer and communication engineering 1.2 (2012): 100.
- [5] Selvi, K. Senthamil, P. Chitrakala, and A. Antony Jenitha. "Face recognition based attendance marking system." Corresponding Author: S. Rajkumar (2014).
- [6] Joseph, Jomon, and K. P. Zacharia. "Automatic attendance management system using face recognition." International Journal of Science and Research (IJSR) 2.11 (2013): 327-330.
- [7] Patil, Ajinkya, and Mrudang Shukla. "Implementation of classroom attendance system based on face recognition in class." International Journal of Advances in Engineering Technology 7.3 (2014): 974.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

Appendix- A

Program Code

Program Code for Dataset Creation

```
import numpy as np
import sys, os
import time
import cv2
from imutils import face_utils
import imutils
import dlib

pathdir = 'C:/Python35/work/Attendance_FaceRecognition_FacialLandmarks/'
db_pathdir = pathdir + 'dataset/'
db_skl_pathdir = pathdir + 'skeleton/'
db_act_pathdir = pathdir + 'actual/'
print ("Video Capturing starting...")
capture = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier(pathdir + 'haarcascade_frontalface_alt_tree.xml')
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(pathdir + 'shape_predictor_68_face_landmarks.dat')
name = input('Enter your name?: ')
if not os.path.exists(db_pathdir + name): os.makedirs(db_pathdir + name)
if not os.path.exists(db_skl_pathdir + name):
os.makedirs(db_skl_pathdir + name)
if not os.path.exists(db_act_pathdir + name): os.makedirs(db_act_pathdir
+ name)
print ()
print ('System ready to capture')
print ('Press " s " when youre in the middle')
time.sleep(1)
while True:
ret,frame = capture.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.2, 3)
for (x,y,w,h) in faces:
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
cv2.imshow('Recognition', frame)
if cv2.waitKey(10) == ord('s') 0xFF:
cv2.destroyAllWindows()
break
count = 0
time.sleep(1)
print ('Creating db')
while True:
ret,frame = capture.read()
frame = imutils.resize(frame, width=500)
frame_cpy = frame
gray = cv2.cvtColor(frame_cpy, cv2.COLOR_BGR2GRAY)
rect = detector(gray, 1)
```

```

print len (faces), len (rect)
    dimensions = frame_cpy.shape
height = frame_cpy.shape[0]
width = frame_cpy.shape[1]
    whiteFrame = np.ones((height,width, 3), np.uint8)
whiteFrame [:] = 255
    cv2.imshow('Recognition',frame)
key = cv2.waitKey(1) 0xFF
if key == ord('q'):
    capture.release()
cv2.destroyAllWindows()
break
    if len (rect) == 1:
count = count + 1
if count == 11:
    capture.release()
cv2.destroyAllWindows()
break
print len (faces), len (rect), count
    for (i, rect) in enumerate(rect):
shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)
    (x, y, w, h) = face_utils.rect_to_bb(rect)
cv2.rectangle(frame_cpy, (x, y), (x + w, y + h), (0, 255, 0), 2)
    for (x, y) in shape:
cv2.circle(frame_cpy, (x, y), 1, (0, 0, 255), -1)
cv2.circle(whiteFrame, (x, y), 2, (0, 0, 0), -1)
    print (db_pathdir + name + '/' + str(count) + '.jpg')
print (db_skl_pathdir + name + '/' + str(count) + '.jpg')
print()
cv2.imwrite( db_act_pathdir + name + '/' + str(count) + '.jpg', frame)
cv2.imwrite( db_pathdir + name + '/' + str(count) + '.jpg', frame_cpy)
cv2.imwrite( db_skl_pathdir + name + '/' + str(count) + '.jpg', white-
Frame)

```

Program Code for Dataset Filtering

```

import numpy as np
import os
import cv2
import pickle
    DATADIR = "C:/Python35/work/own_model_creation/dataset"
CATEGORIES = ["Christy", "Amal", "Anwin"]
    IMG_SIZE = 70
training_data = []
    X = [] Training data = [] Training label

```

```

        create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                print (os.path.join(path,img))
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass
            except OSError as e:
                print("OSErrorBad img most likely", e, os.path.join(path,img))
            except Exception as e:
                print("general exception", e, os.path.join(path,img))
        save_training_data():
    X = [] Training data
    y = [] Training label
        for features,label in training_data:
    X.append(features)
    y.append(label)
    X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
        pickle_out = open("X.pickle","wb")
    pickle.dump(X, pickle_out)
    pickle_out.close()
    pickle_out = open("y.pickle","wb")
    pickle.dump(y, pickle_out)
    pickle_out.close()
        create_training_data ()
    save_training_data ()

```

Program Code for Dataset Training

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import pickle
import time
    pickle_in = open("X.pickle","rb")
    X = pickle.load(pickle_in)
    pickle_in = open("y.pickle","rb")
    y = pickle.load(pickle_in)
    X = X/255.0
    dense_layers = [0]
    layer_sizes = [64]

```



```

conv_layers = [3]
    for dense_layer in dense_layers:
for layer_size in layer_sizes:
for conv_layer in conv_layers:
NAME = "-conv--nodes--dense-".format(conv_layer, layer_size, dense_layer,
int(time.time()))
print(NAME)
    model = Sequential()
    model.add(Conv2D(layer_size, (3, 3), input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
    for l in range(conv_layer-1):
model.add(Conv2D(layer_size, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    for _ in range(dense_layer):
model.add(Dense(layer_size))
model.add(Activation('relu'))
    model.add(Dense(1))
model.add(Activation('sigmoid'))
    tensorboard = TensorBoard(log_dir="logs/" .format(NAME))
    model.compile(loss='binary_crossentropy',
optimizer='adam',
metrics=['accuracy'], )
    model.fit(X, y,
batch_size=32,
epochs=10,
validation_split=0.3,
callbacks=[tensorboard])
    model.save('64x3-CNN.model')

```

Program Code for Dataset testing

```

import numpy as np
import sys, os
import time
import cv2
import tensorflow as tf
    from imutils import face_utils
import imutils
import dlib
    import MySQLdb
import datetime
    print("Connecting to database using MySQLdb")
db = MySQLdb.connect(host='localhost', db='attendance_facerecog', user='root',
passwd="")
cursor = db.cursor ()
print("Succesfully Connected to database using MySQLdb!")

```

```

    pathdir = 'C:/Python35/work/Face_Recognition_FacialLandmark_CNN/'
    print ("Video Capturing starting...")
    capture = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(pathdir + 'haarcascade_frontalface_alt_tree.xml')
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor(pathdir + 'shape_predictor_68_face_landmarks.dat')
    CATEGORIES = ["Christy", "Amal", "Anwin"]
    def prepare(filepath):
    IMG_SIZE = 50
    img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 1)
    print ()
    print ('System ready to capture')
    print ('Press " s " when youre in the middle')
    time.sleep(1)
    while True:
    ret,frame = capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.2, 3)
    for (x,y,w,h) in faces:
    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.imshow('Recognition', frame)
    if cv2.waitKey(10) == ord('s') 0xFF:
    cv2.destroyAllWindows()
    break
    count = 0
    time.sleep(1)
    print ("Capturing")
    while True:
    ret,frame = capture.read()

    frame = imutils.resize(frame, width=500)
    frame_cpy = frame
    gray = cv2.cvtColor(frame_cpy, cv2.COLOR_BGR2GRAY)
    rect = detector(gray, 1)
    print len (faces), len (rect)
    dimensions = frame_cpy.shape

    height = frame_cpy.shape[0]

    width = frame_cpy.shape[1]
    whiteFrame = np.ones((height,width, 3), np.uint8)
    whiteFrame [:] = 255
    cv2.imshow('Recognition',frame)
    key = cv2.waitKey(1) 0xFF
    if key == ord('q'):

```

```

capture.release()
cv2.destroyAllWindows()
break
    if len (rect) == 1:
for (i, rect) in enumerate(rect):
shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)
    (x, y, w, h) = face_utils.rect_to_bb(rect)
cv2.rectangle(frame_cpy, (x, y), (x + w, y + h), (0, 255, 0), 2)
    for (x, y) in shape:
cv2.circle(frame_cpy, (x, y), 1, (0, 0, 255), -1)
    testImage = pathdir + 'test_image.jpg'
print (testImage)
print ()
cv2.imwrite(testImage, frame_cpy)
print ("Going for predict")
    model = tf.keras.models.load_model("64x3-CNN.model")
    prediction = model.predict([prepare(testImage)])
print(prediction)
print(CATEGORIES[int(prediction[0][0])])
    name = CATEGORIES[int(prediction[0][0])]
presentdate = datetime.datetime.now ().strftime ("%d:%m:%Y")
presenttime = datetime.datetime.now ().strftime ("%H:%M:%S")
    query = "INSERT INTO info (Name, date, time)
VALUES ('%s', '%s', '%s') " % (name, str (presentdate), str (presenttime))
cursor.execute (query)
db.commit ()
print("Data inserted succesfully")

```

Appendix- B

IEEE Paper

SMART ATTENDANCE MONITORING SYSTEM

Amal A A , Anwin Antony, Christy Augustine

*Rajagiri School of Engineering and Technology

Abstract—As an important part of class teaching, attendance plays a key role in the evaluation of teaching. At the beginning and end of class, it is usually checked by the teacher, but it may appear that miss someone or some students answer multiple times. To overcome this issue, the latest technology is used to detect the faces and recognize the faces using python programming. Face Detection is one of the significant topics of computer technology in machine learning application. This technology has been available for some years now and is being used all over the world. In this paper, capturing the images of students from the database and faces are detected by the algorithm and then it is acknowledging with the database and at last, the attendance is recorded. For detecting faces Viola-Jones facial identification algorithm is used and for recognizing the faces from the databases.

Index Terms—Python, Viola-Jones algorithm, Machine learning

I. INTRODUCTION

Every schools/college are taking attendance by using an attendance sheet or biometric. But using systems consume so more time. Majorly student appearance sheet which is given to the faculty members is used to take the student attendance, but it may appear that miss someone or some students answer multiple times. In other cases, the appearance sheet may be misplaced or gone. To reduce such problems a smart attendance management system be needed. Manual maintenance of attendance are inefficient due to the following reasons:

[1] It takes away a lot of lecture hours

[2] Prone to proxies or impersonations

To resolve this problem of attendance, many attendance management systems have been introduced in recent years. Jain et. al [1] developed a desktop based application in which students are given attendance by clicking a checkbox next to their name and then by clicking the register button to mark their presence. In 2013, Bhalla et. al [2] have proposed bluetooth based attendance system. Application software installed in mobile phone enables to register the attendance via bluetooth connection and transfer the notification to the instructor. Works of [3] propose a system for employee attendance based on the fingerprint. The system compares one fingerprint template with all previously stored in the database. In [4], Joardar et. al has developed an attendance system based on the palm dorsal subcutaneous vein pattern of individuals.

Lately's video-based face identification has experience with immediate consideration and it will be an important topic to researchers for the people's identification in the field of image handling for public's proof of identity. The main focus for the researchers is face identification, more probably video-based face identification because videos observation, attendance checking in different organization and parallel safety

purposes are based on it. This paper indicates that how to facial identification techniques be effective in the educational field for an effectual attendance scheme will repeatedly record the existence of a registered individual within their own class.

II. METHODOLOGY

A. Face detection

Face detection which is the trending topic in the computer science field in machine learning application. This technology is available for some years now and is being used all over the world.

Viola-Jones algorithm:

This is the algorithm that lies at the foundation of the open CV library and this is one of the most powerful algorithms for computer vision. Training and detection at two stages in viola Jones algorithm.

In detection, the first step is to convert an image into a grey scale. After that, the algorithm will search for a face in the image from the left-top corner and checks as it moves side step by step towards the right. the algorithm checks the human facial characteristics like cheeks, forehead, eyebrows, eyes, etc., if all the characteristics are detected, then it is considered as a face

Haar-like features:

Haar-like characteristics are computerized image characteristics using facial detection. They are three types of features mainly used. They are Edge, line, four rectangle features.

When the Haar-like features are applied to the grayscale image, it will detect the dark shade in the image and the four Haar-like features are applied to specific parts of the image. After features are applied to the image, those specific parts are converted into pixels depending upon the grayscale values (0-255). It will do normalization for optimizing, assign values to the pixels from the range of 0 to 1 with 0 indicating black and 1 indicates white. it will add up the pixel values in each feature and it will differentiate their sums. the difference is used to identify the features of the image.

B. Pre-Processing

Pose Estimation: Since people move around and look at different directions in front of the real-time camera, it is possible to have a wide range of head poses oriented at different angles. But for the sake of biometrics, it is important to have least rotated face as a standout in the entire face-log. Thus it is important to include this feature in face quality assessment. We determined the head pose using three angles: Roll, Yaw, Pitch. All these angles are typically between - 90 to + 90.

The roll and pitch are adjusted by aligning technique during face-log generation, so the only concern is yaw angle. Using

face landmarks detection, the coordinates of nose tip and also the point between the eyebrows are calculated. If (x_1, y_1) and (x_2, y_2) are such points, then yaw angle is computed as:

$$yaw = \arctan 2(y_2 - y_1, x_2 - x_1) \quad (1)$$

| Range Of Yaw Angle | 0 to 10 | 10 to 20 | 20 to 30 | More than 30 |
|--------------------|---------|----------|----------|--------------|
| NHP | 1 | 0.5 | 0.33 | 0.25 |

Fig. 1. Head-pose estimation

Sharpness: It is very likely to have blurry images in real time video sequences because the faces are moving. Thus it is important to include this feature in face quality assessment. To compute the sharpness of an image, the variance of an image Laplacian is utilized. This is defined as:

$$\text{Sharpness} = \sum_{(i,j) \in \mathcal{U}(x,y)} (\Delta I(i,j) - \overline{\Delta I})^2 \quad (2)$$

where $\overline{\Delta I}$ is the mean value of image Laplacian within (x, y) . A grey-scale channel of an image is convolved with the (3x3) kernel and took the variance of that result.

$$\text{kernel} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (3)$$

If the variance falls below a pre-defined threshold, then the image is considered as a blur. This threshold depends on the working environment and can be set accordingly. Then it is normalized with a pre-defined threshold,

$$\text{NormalizedSharpness}(NS) = \frac{\text{Sharpness}}{\text{threshold}} \quad (4)$$

Resolution: Since the face tracking technique is employed, the camera tracks the face as long as the face is in the scene. But, as the face moves far away from the camera, there will be a large distance between the camera and the face. Thus, the size of the face becomes smaller. Thus, it is important to include this feature in face quality assessment. Using face landmark detection, the position of the eye corners in a face is calculated. Let (x_L, y_L) be the coordinates of the left eye corner and (x_R, y_R) be the coordinates of the right eye corner. The distance between them is given by:

$$\text{Resolution} = \sqrt{(x_L - x_R)^2 + (y_L - y_R)^2} \quad (5)$$

Then the obtained resolution is normalised with the threshold for interocular distance as,

$$\text{Normalized Resolution (NR)} = \frac{\text{Resolution}}{\text{threshold}} \quad (6)$$

Larger the distance, smaller will be the size of the face and hence, lesser will be the resolution.

Brightness: Changes in Lighting conditions are quite common in real-time applications like surveillance cameras. It is easier to apply Local feature extraction on brighter faces

| Parameter | Head-pose | Sharpness | Brightness | Resolution |
|-----------|-----------|-----------|------------|------------|
| Weight | 17 | 9 | 6 | 8 |

Fig. 2. Weights for various parameters in computing FQA

than on darker faces. Thus, it is necessary to include this feature in face quality assessment. To obtain this parameter, we calculated the mean of all the intensities of various channels (R, G, B) present in the image.

$$\text{Brightness} = (R + G + B)/3 \quad (7)$$

$$\text{Normalized Brightness}(NB) = \text{Brightness}/100 \quad (8)$$

Final Score: In order to get the best quality image in the real-time video sequence, we need to assign weights to each of the normalized parameters (NHP, NS, NR, NB). We gave the highest priority to the head pose followed by other parameters as shown in Table II. So, we computed the Face Quality Assessment (FQA) as,

$$\text{FQA} = \text{NHP} * 17 + \text{NS} * 9 + \text{NB} * 6 + \text{NR} * 8 \quad (9)$$

Greater is the value of FQA, greater will be the quality of face which will be stored in Face-log. We can set the threshold for these FQA values of generated images to

C. Face Recognition

Face representation is the core of the recognition algorithm used in this system. The face image captured after the quality assessment is needed to be represented in form of feature for further processing. The preprocessed images are too high-dimensional for a classifier to take directly on input. To obtain a low-dimensional distinct feature from the face images we used Convolution Neural Network (CNN), popularly known as deep learning. A deep network is a feedforward network comprising of many function compositions, or layers.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product.

III. EXPERIMENT AND RESULTS

We mainly used 4 python scripts to perform the face recognition process.

Dataset Creation: The first step is to detect the face of a person. For this we are using Haar cascade classifier which uses different face parameters like pose estimation, resolution, brightness and sharpness to detect the face.

Dataset Filtering: From the detected face we are taking out facial landmark points. This landmark points is termed as skelton points. Here we are taking 10 different skelton images of a particular person.

Dataset Training: Once the skelton images are collected then it is fed to CNN(Convolutional Neural Network). In our project we are using AlexNet as our CNN. So the desired face is fed to the CNN for recognition. Here 10 different skelton images are fed to the CNN for training.

Dataset Testing: Finally the presence of person whose face is recognised by the CNN when an image is provided as input is recorded in MySQL.

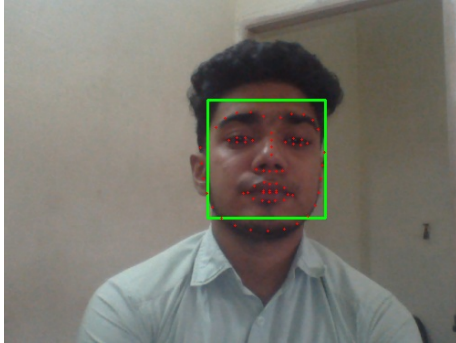


Fig. 3. Input image

| Attendance Monitoring | | | |
|-----------------------|---------|------------|----------|
| ID | Name | Date | Time |
| 7 | Aashish | 09/07/2020 | 20:51:29 |

Fig. 4. Output

SAMS has been designed to register the face of each individual for the first time. Once done, the network trains it automatically for future usage.

IV. CONCLUSION

An automatic attendance management system aims at solving the issues of manual methods of existing systems. We have used the concept of face recognition to implement a system that marks the attendance of a particular person by detecting and recognizing the face. These systems perform satisfactorily with different facial expressions, lighting and pose of the person. There is room for improvement since these systems sometimes fail to recognize every face student present in the classroom. We have made the device portable for easy use even when the sessions are on, without disturbing the class. There are future scopes to make a more compact ergonomics to make it a more user-friendly product to make an impact in building a more healthier academic environment.

REFERENCES

- 1) S. K. Jain, U. Joshi, and B. K. Sharma, "Attendance management system," Masters Project Report, Rajasthan Technical University, Kota, 2011.
- 2) V. Bhalla, T. Singla, A. Gahlot, and V. Gupta, "Bluetooth based attendance management system," International Journal of Innovations in Engineering and Technology (IJJET) Vol. 3, no. 1, pp. 227–233, 2013.
- 3) S. S. Mahat and S. Mundhe, "Proposed framework: College attendance management system with mobile phone detector," International Journal of Research in IT and Management, vol. 5, no. 11, pp. 72–82, 2015.

- 4) S. Joardar, A. Chatterjee, and A. Rakshit, "A real-time palm dorsa subcutaneous vein pattern recognition system using collaborative representation-based classification," IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 4, pp. 959–966, 2015.

Appendix- C

Presentation

Slides

Smart Attendance Monitoring System (SAMS)

Guide: Ms.Ramitha Rajesh

Group Members: Amal A A

Anwin Antony

Christy Augustine

ABSTRACT

- The regular class attendance of students plays a significant role in performance assessment and quality monitoring.
- The system is developed by the integration of various components for managing the students attendance.
- In order to reduce the workmanship, we propose a more sophisticated way of marking attendance by Digital Image Processing(DIP).

TIMELINE

| MONTH | WORK DONE |
|----------|----------------------|
| December | Face Detection |
| January | Image Pre-processing |
| February | Training the CNN |
| March | Face Recognition |
| April | Representation |
| May | Classification |

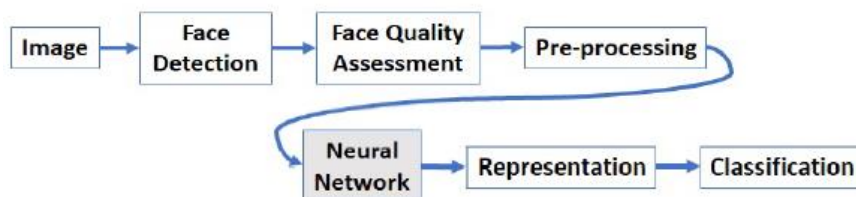
Python

- An interpreted , high level, general purpose programming language.
- Object oriented approach that aims to write clear and logical codes.
- It supports multiple programming paradigms including structured, object oriented and functional programming.

OpenCV

- OpenCV is the most popular library for computer vision.
- Originally written in C/C++, it now provides bindings for Python.
- PythonOpenCV is a Python wrapper for the original OpenCV C++ implementation.

Block Diagram



Face Detection

- It is the first step in face recognition.
- Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.
- The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

Face Detection

- Face detection is basically an image segmentation process as the image is to be segmented into two parts:
 - one containing faces and
 - the other representing non-face regions.
- The classifier decides face and non-face class based on information learned from during training.

Haar Cascade Classifiers

- Haar-like characteristics are computerized image characteristics using facial detection.
- Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones.
- The algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier.

Image Pre-Processing

- Pre-processing is a common name for operations with images at the lowest level of abstraction.
- The aim is improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

Image Pre-Processing

- Face landmark detection is the process of finding points of interest in an image of a human face.
- Landmark detection starts with face detection, finding faces in the image and their extents (bounding boxes).
- OpenCV contains one of the first robust face detectors freely available to the public.

Face Recognition

- The preprocessed images are too high-dimensional for a classifier to take directly on input.
- To obtain a low-dimensional distinct feature from the face images we used Convolution Neural Network (CNN), popularly known as deep learning.
- There are three steps in face recognition :
 - data pre-processing,
 - facial feature extraction, and
 - comparison of features between the target face and faces from database.

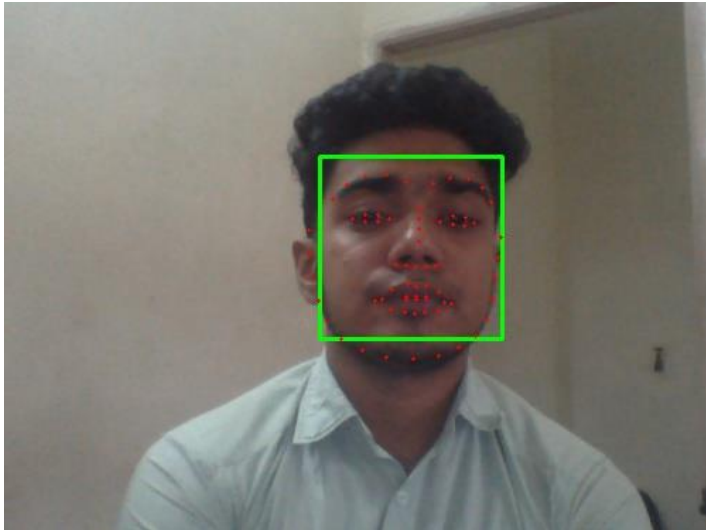
Convolutional Neural Network

- CNN is a class of deep, feed-forward (not recurrent) artificial neural networks that are applied to analyzing visual imagery.
- Images are high-dimensional vectors. It would take a huge amount of parameters to characterize the network.
- To address this problem, convolutional neural networks are used to reduced the number of parameters and adapt the network architecture specifically to vision tasks.

Convolutional Neural Network

- Convolutional neural networks are usually composed by a set of layers that can be grouped by their functionalities.
- There are three main types of layers to build CNN architectures: Convolutional layer, Pooling layer, and Fully-Connected layer.

Results



Results

Attendance Monitoring

| ID | Name | Date | Time |
|----|-------|------------|----------|
| 7 | Anwin | 09.07.2020 | 20.01.29 |

Observation

- SAMS was designed to register the face of every individual.
- The algorithm gives the result when the given image is perfectly matched with the database image.
- AlexNet worked properly and gave desired output.

Contribution

| WORK DONE | GROUP MEMBERS |
|---|-------------------|
| Study of face detection and Haar cascade classifier | Amal A A |
| Code development and study of pre-processing | Anwin Antony |
| Study of CNN and Deep learning | Christy Augustine |

References

- Kwolek, Bogdan. "Face detection using convolutional neural networks and Gabor filters." International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2005.
- Kar, Nirmalya, et al. "Study of implementing automated attendance system using face recognition technique." International Journal of computer and communication engineering 1.2 (2012): 100.
- Patil, Ajinkya, and Mrudang Shukla. "Implementation of classroom attendance system based on face recognition in class." International Journal of Advances in Engineering Technology 7.3 (2014): 974.

THANK YOU

Appendix- D

Questionnaire

[i]What is computer vision?

Computer vision is a field of study which encompasses on how computer see and understand digital images and videos. Computer vision involves seeing or sensing a visual stimulus, makes sense of what it has seen and also extract complex information that could be used for other machine learning activities.

[ii]What is Haar cascade?

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones. 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.

[iii]Explain deep learning.

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. The adjective "deep" in deep learning comes from the use of multiple layers in the network.

[iv]What are basic components in CNN?

CNN has the following five basic components:

- [1] Convolution : to detect features in an image
- [2] ReLU : to make the image smooth and make boundaries distinct
- [3] Pooling : to help fix distored image
- [4] Flattening : to turn the image into a suitable representation
- [5] Full connection : to process the data in a neural network.

[v]List some applications of CNN.

Some of the key applications of CNN are listed here -

- [1] Decoding Facial Recognition
- [2] Understanding Climate
- [3] Analyzing Documents

Appendix- E

Mapping The Project Outcomes With POs And PSOs

Programme Outcomes

| Program Outcomes | PO 1 | PO 2 | PO 3 | PO 4 | PO 5 | PO 6 | PO 7 | PO 8 | PO 9 | PO 10 | PO 11 | PO 12 | Justification |
|--|------|------|------|------|------|------|------|------|------|-------|-------|-------|--|
| Programming Tool usage | 2 | 2 | 1 | - | 3 | - | - | - | 3 | 3 | 1 | 3 | Acquired skills in Python OpenCV. |
| Knowledge on image processing | 3 | 3 | 2 | 3 | 2 | 1 | - | 1 | 3 | 2 | - | - | Acquired knowledge on image processing. |
| Study of haar cascade classifiers | 3 | 2 | 1 | - | 3 | 3 | 1 | - | - | - | - | - | Studied the working of the Haar-cascade classifiers. |
| Understanding of feature extraction methods. | 3 | 2 | 1 | - | 3 | 3 | 1 | 2 | 3 | 3 | - | 3 | Detailed learning approach was applied to learn facial landmark extraction. |
| Understanding CNN and deep learning | 1 | 1 | 2 | 1 | - | - | - | - | 2 | 3 | 1 | 2 | Studied CNN and deep learning. |
| Communication | - | - | 3 | - | - | 2 | 1 | - | 3 | 3 | | 2 | Was able to effectively convey ideas and information to those concerned. |
| Documentation skills | 1 | 2 | 2 | - | 3 | - | - | 1 | 3 | - | - | - | Gained Competency in LATEX and other documentation tools. |
| Teamwork | - | 2 | 3 | 2 | 3 | - | - | - | 3 | 3 | - | 2 | Functioned as a coherent team to achieve project goals and objectives. |
| Presentation skills | 3 | 2 | 2 | 1 | 3 | 1 | 1 | 1 | 3 | 3 | - | 2 | Periodically presented project results using computer technology and tools to a panel of evaluators. |

PROGRAMME-SPECIFIC OUTCOMES

| PROGRAMME SPECIFICATION OUTCOMES | PSO 1 | PSO 2 | PSO 3 | JUSTIFICATION |
|--|-------|-------|-------|---|
| Familiarized with image processing , feature extraction and deep learning. | 2 | 3 | 1 | Studied about image processing ,facial landmark extraction and deep learning. |
| Programming skills | 3 | 2 | 3 | Coded using Python OpenCV. |
| Documentation skills | - | 1 | 2 | Project report, individual project report and IEEE paper was prepared. |
| Planning and Scheduling | - | 1 | 2 | Detailed timeline for the work progression was developed and each member was assigned work Accordingly. |
| Teamwork | - | 2 | 3 | Worked as a team during the entire phases of the project. |