

A Project/Dissertation Review-1 Report

on

Facial recognition-based attendance system

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



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

B.Tech Computer Science and Engineering

Under The Supervision of

Mr. Lalit Sharma

Designation

Submitted By

Pushkar Thakur

18SCSE1070007

Vandana Nandan Mishra

18SCSE1010267

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA November,

2021

CANDIDATE’S DECLARATION

We hereby certify that the work which is being presented in the project, entitled “**Facial recognition-based attendance system**” in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**—submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period JULY 2021 to DECEMBER 2021, under the supervision of Mr. Lalit Sharma, Assistant Professor, Department of Computer Science and Engineering, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010267 - Vandana Nandan Mishra

18SCSE107007 - Pushkar Thakur

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor Name

Designation

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Vandana Nandan Mishra: 18SCSE1010267, Pushkar Thakur: 18SCSE107007 has been held on _____ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date: December, 2021

Place: Greater Noida

ABSTRACT

In this digital era, face recognition system plays a vital role in almost every sector. Face recognition is one of the mostly used biometrics. It can be used for security, authentication, identification, and has got many more advantages. Despite of having low accuracy when compared to iris recognition and fingerprint recognition, it is being widely used due to its contactless and non-invasive process. Furthermore, face recognition system can also be used for attendance marking in schools, colleges, offices, etc. This system aims to build a class attendance system which uses the concept of face recognition as existing manual attendance system is time consuming and cumbersome to maintain. And there may be chances of proxy attendance. Thus, the need for this system increases. This system consists of four phases- database creation, face detection, face recognition, attendance updation. Database is created by the images of the students in class. Face detection and recognition is performed using Haar-Cascade classifier and Local Binary Pattern Histogram algorithm respectively. Faces are detected and recognized from live streaming video of the classroom. Attendance will be mailed to the respective faculty at the end of the session.

The technology used in this project was mainly machine learning or rather we can say deep learning. Python language is used in this project and OpenCV library is mainly used for this purpose. And a database is used to store the attendance data of the students and teachers.

Table of Content

Title	Page
	No.
Abstract	2
List of Figures	4
Chapter 1 Introduction	5
1.1 Introduction	
1.2 Formulation of Problem	
1.2.1 Tool and Technology Used	
Chapter 2 Literature Survey/Project Design	12
Chapter 3 Algorithm	34
Chapter 4 Implementation	35
Chapter 5 Code	44
Chapter 4 Result	51
Conclusion	52
References	56

List of Figures

Figure No.	Title
1.	Neural network diagram
2.	Basic Structure of CNN
3.	Example architecture of a CNN for a computer vision task
4.	Layers of Deep Boltzmann Machine
5.	Denoising autoencoder
6.	Flowchart of the process
7.	Examples of Hog Descriptor
8.	Code
9.	Result

CHAPTER - 1

Introduction

Maintaining the attendance is very important in all the institutes for checking the performance of employees . Every institute has its own method in this regard. Some are taking attendance manually using the old paper or file based approach and some have adopted methods of automatic attendance using some biometric techniques. But in these methods employees have to wait for long time in making a queue at time they enter the office. Many biometric systems are available but the key authentications are same is all the techniques.

Every biometric system consists of enrolment process in which unique features of a person is stored in the database and then there are processes of identification and verification. These two processes compare the biometric feature of a person with previously stored template captured at the time of enrollment. Biometric templates can be of many types like Fingerprints, Eye Iris, Face, Hand Geometry, Signature, Gait and voice. Our system uses the face recognition approach for the automatic attendance of employees in the office room environment without employees' intervention . Face recognition consists of two steps, in first step faces are detected in the image and then these detected faces are compared with the database for verification .

The solution for this problem is the attendance through facial recognition and showing the previous attendance record of student at the time of attendance when identified.

The technology used in this project was mainly machine learning or rather we can say deep learning. Python language is used in this project and OpenCV library is mainly used for this purpose. And a database is used to store the attendance data of the students and teachers

In recent decade, a number of algorithms for face recognition have been proposed, but most of these works deal with only single image of a face at a time. By continuously observing of face information, our approach can solve the problem of the face detection , and improve the accuracy of face recognition.

The different techniques used for face detection are Knowledge based method that finds the relationship between facial features, Feature invariant method aims to find structural features of a face, Template matching method describes the several standard patterns, and Appearance based method which captures the facial appearance.

The different techniques used for face recognition are Holistic approach where the whole face is taken as input for recognition purpose using Principal Component Analysis and Linear Discriminate Analysis and Feature based approach where local features on face like nose and eyes are detected.

The evolution of face identification has received many vigilances in recent years. It is a centric application in image analysis, but it is arduous to recreate an automatic model altogether which has the ability to pinpoint human face. ID's of the face are vital in the model Management of Attendance. The non-automatic attendance process is time snatcher, so a highly qualitative and quantitative investment of time were given to get liable output. Out of many answers to this hurdle is the use of model popularly known as biometric attendance structure. Inevitably it is still arduous to verify each pupil in a classroom

as the volume of the class is high, and if the model is not able to detect a pupil, it can disturb the teaching process. There is plethora of requirements for biometric system or any equivalent model which are expensive and need a lot of interaction with students, making it a time snatching model. Due to this, it is being considered by the researchers has a real challenge for building a perfect system which can detect and simultaneously recognize pupils with also a novelty of marking the presence of the students. The work and research till now have presented us with faster processing with adequate amount of efficiency and are also able to merge it with the technology which deals with Computer Vision. Presently, Face identification methods can be found in 2 approaches. Firstly, the method uses localized model which specifically works on the features such as Nose, Mouth, Eyes etc. to match an individual face. On the other hand, the other way is global model. Which considers the whole face when feature extraction. The above mentioning of these models has been implemented with the support of many unique algorithms. The face identification includes many systematic proceedings such as, The Image acquisition that deals with Getting a decent captured image which includes the faces of the pupils. Followed by extraction which includes detection of faces to facial feature extractions to the making of databases. As necessary it sounds, this intermediate process is extremely important because the extraction of feature should be meaningful so that to further buttress the facial recognition process. Finally ending the process with Face matching which consist of face comparison. Albeit it is an ending operation, but every project contains post-processing which starts after face matching. The operation of face matching can be implemented in abundant of ways, with the help of many algorithms. The algorithms which are popularly recognized in helping of detection of pupils are as follows: - 'Haar Cascade', 'Viola Jones' whereas the algorithms which go toe-to-toe with identification process are as follows: - 'Eigen Face', 'PCA', 'Fisher Face', 'LDA', 'LBPH'. There working may differ and end results obtained will also be dependent upon the

application i.e. whether it is used for single face recognition or it is being used for multiple face recognition. The various algorithms which were and are used in current decade is been reviewed in the further sections.

With the rapid development of information technology, technologies in various fields such as computer hardware and software have improved. Artificial intelligence (AI) and automation technology, in particular, have been pushed toward the directions of medicine, health, and life in space technology, working to gradually increase human life expectancy and improve the quality of life. At the time of writing, there have been more than 110 million confirmed cases and 2730000 deaths caused by COVID-19 globally. The total number of confirmed cases in the United States has exceeded 30570000 and the number of deaths is 555000.(1) At present, COVID-19 is still spreading all over the world, not only disrupting people's lives but also raising the awareness of disease prevention. As COVID-19 spreads and mutates rapidly, countries around the world have gradually increased preventive measures. The international economic and social order urgently needs to be restructured and restored. To maintain our livelihoods, we have to cautiously return to the workplace while living with the threat of COVID-19 and practicing self-protection and social distancing. The teaching objective is to guide college graduates to innovate and design useful articles for life.(2,3) During the pandemic, group discussions have been conducted through video conferencing, dividing experiments, mastering group progress, letting students self-integrate their knowledge, and verifying their learning experience and ideas from the past four years, so they can apply what they have learned.(4) The principles and technology of sensing are applied to the research and analysis of business processes such as facial recognition, body temperature testing, and automated attendance recording using web technology. The purpose of this study is to establish a fever detection

file database for returning workers, and use face recognition system technology to identify them and give them a green or red health code to indicate whether their current health allows them to enter the workplace. At the same time, a database for an attendance system is established, which can effectively monitor and control the current situation of disease prevention and contact time series, track the timeline and condition of patients with a fever, and establish a personal and group database. This is to achieve effective disease prevention and organizational care, and to improve the virtuous cycle of interpersonal cohesion and centripetal force. The first step is to screen applications and existing software packages connected to fever detection equipment and database planning. Starting from the standardized method of software system development for engineering courses, the goal is to use Internet mobile phones for faceto-face consultation. Mobile health applications (apps) can be used as part of telehealth to monitor patient-reported outcomes and enhance patient-provider communication. In recent years, international corporations have been looking for new and effective methods for monitoring employees' attendance and clocking in, hoping to move towards the current needs of companies and enterprises, and overcome the defects of the attendance mechanisms of magnetic cards and fingerprint technology. Just as license plate recognition and payment systems have been widely used by enterprises, the AI field, which is gradually adopting face recognition technology, is commonly used in the employee clock-in recognition mechanism and has gradually entered use in corporate employee attendance system databases. Face recognition is an image processing technology to detect human face features, which extracts 128 biological characteristics of human face information. On the basis of an image library or on-the-spot photography, element points of the facial contour structure can be quantified, and then the gender, identity, professional title, and job-related authority of the face can be identified through quantitative comparison and analysis.

The design and implementation of a system for facial recognition, fever detection, and

attendance recording based on web technology can meet the requirements of corporate human resource management systems. This system can also provide information for enterprise attendance management and the scientific management of employee health, especially in the case of an emergency or for travellers. Cloud-based solutions have open challenges of interoperability and integration, higher challenges for security and privacy, and may lack 24/7 support for the high availability of health history. Existing portable systems store limited health information for only a specific hospital and do not support mobility of patients across different hospitals. In this paper, we propose a next-generation portable Smart Health Record Management system with secure Near Field Communication (NFC). It will have many advantages, such as early fever warning, convenient health tracking and retrieval of employees, data security, reliable attendance data, a large storage capacity for data, low management cost, and the long service life of conventional equipment. Such a system will be essential to personal health and needs to be widely promoted. Facial recognition is an interdisciplinary research of pattern recognition and computer vision. It originated in the 1960s. At that time, face recognition technology was mainly based on the research of facial contours. In recent decades, recognition technology has analyzed the facial features and contours of each person. A computer can easily distinguish even small differences among people, helping to distinguish them. Presently, face recognition technology is widely used in customs clearance detection systems when entering and leaving a country. Compared with fingerprint and iris recognition technology, facial recognition has similarities but also exceptional characteristics. These recognition technologies are similar in that they can identify each person's unique identity. Although irises and fingerprints can also be used to extract features, they may be damaged and changed as a result of an accident or surgery, whereas facial recognition features will not change with time or the environment.

At present, many research institutions and overseas universities, including MIT, CMU

Robotics Institute, Cornell University, and Berkeley University, have teams that have made significant contributions in the field of facial recognition. Domestic institutions studying facial recognition are the Chinese Academy of Sciences, Tsinghua University, Harbin University of Technology, and Nanjing University of Technology, along with other long-term research teams. There are many kinds of attendance monitoring systems, such as those based on fingerprint recognition,(15) iris recognition,(16,17) and facial recognition.(18) Old attendance monitoring systems have a barcode punch card or a magnetic card recorded in a database and were designed for convenience to replace the use of notes and coins. They have been widely used in various markets because the cost of the cards is low and the recognition accuracy is high. However, they can be lost, damaged, or stolen. The latest technology has been implemented in employee identification systems, inventories, security personnel displays, and RFID items related to patrol tracking records, disaster simulations, and impact factor analyses. In the future, 5G or B5G technology will be able to detect abnormal behaviors of employees due to physical discomfort or unintentional negligence, so disasters can be prevented in time. Recently, face recognition technology has been gradually introduced into large supermarkets for digital currency payment at check-out counters.(19) At present, it is used for convenient cashflow trading, logistics shelf management, accurate customer group marketing, and customer loyalty management. It is expected that face recognition technology will move toward regional strategic operations and analyses in the future.

Theoretical Background According to Literature, Student Attendance System by Face Detection, maintaining attendance is very important and compulsory in all the institutes for checking the performance of students. Every institute has its method in this regard. Some are taking attendance manually using the old paper old file-based approach and some have adopted methods of automatic attendance using some biometric techniques [3]. An automated attendance system based on face recognition is a

biometric system where typically, it registers the attendance of each student present in a class by detecting and identifying all of their faces, and then this recorded information is ideally transmitted to a server device which may compute the attendance of each student and store and update the corresponding data in a database. Automated attendance systems are more reliable, rigid, and efficient than the traditional attendance systems and other biometric attendance systems, leading to better productivity and output of both the teachers and students, as well as better consumption of time [13]. An Automatic Attendance System Using Image Processing, maintaining attendance is very important and compulsory in all the institutes for checking the performance of students. Every institute has its method in this regard. Some are taking attendance manually using the old paper or file-based approach and some have adopted methods of automatic attendance using some biometric techniques. There are many automatic methods available for this purpose i.e. biometric attendance. All these methods also waste time because students have to make a queue to touch their thumb on the scanning device [2]. Face detection and recognition section detect face from the image captured by the camera, and the image of the face is cropped and stored. The element recognizes the images of student's faces, which have been registered manually with their names and ID code in the record. Face recognition data and face identification data are verification into the record.

Literature Survey

Proposed System-

- Face recognition attendance system used to mark the attendance details of students[2]. It can mark the time of lecture and daily presence of the individuals in a premise and generate detailed reports on the same at regular intervals.
- Technically following is the usefulness of this project:
- Enhances security and speed in tracing student attendance and lecture time.
- Easy to set up and use.
- Convenient and inexpensive.
- Helps in managing the time and attendance profiles of students.
- Eliminates proxy punching.
- Manages student attendance records.
- Easily configured according to your requirement.
- Reduces the manual students data entry, register maintenance and monthly requirements.

Python

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages.

NumPy

NumPy (pronounced */ˈnʌmpɑː/ (NUM-py)* or sometimes */ˈnʌmpi/ (NUM-pee)*) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing

Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

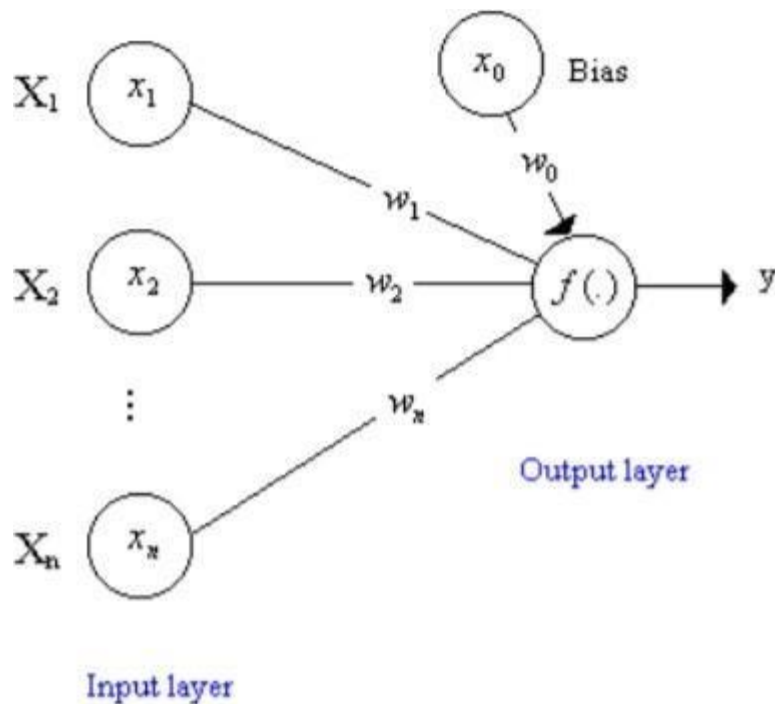
NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[20] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Different approaches which are followed for facial features Recognition:

- **Neural Network Approach** : The neural network contained a hidden layer with neurons. The approach relies on the belief that a neutral face image akin to each image is on the market to the system. Each neural network is trained independently with the employment of on-line back propagation.

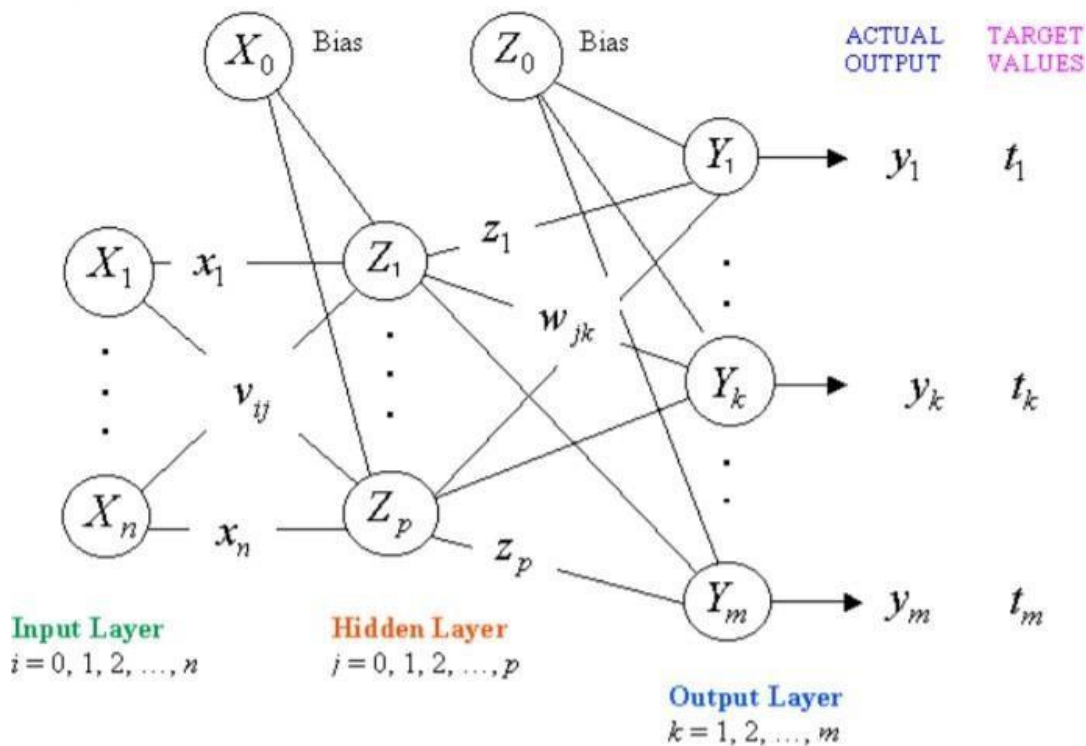


- **Support Vector Machine** : In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and multivariate analysis.

$$h(x) = \begin{cases} + & \cdot + \geq 0 \\ - & \cdot + < 0 \end{cases}$$

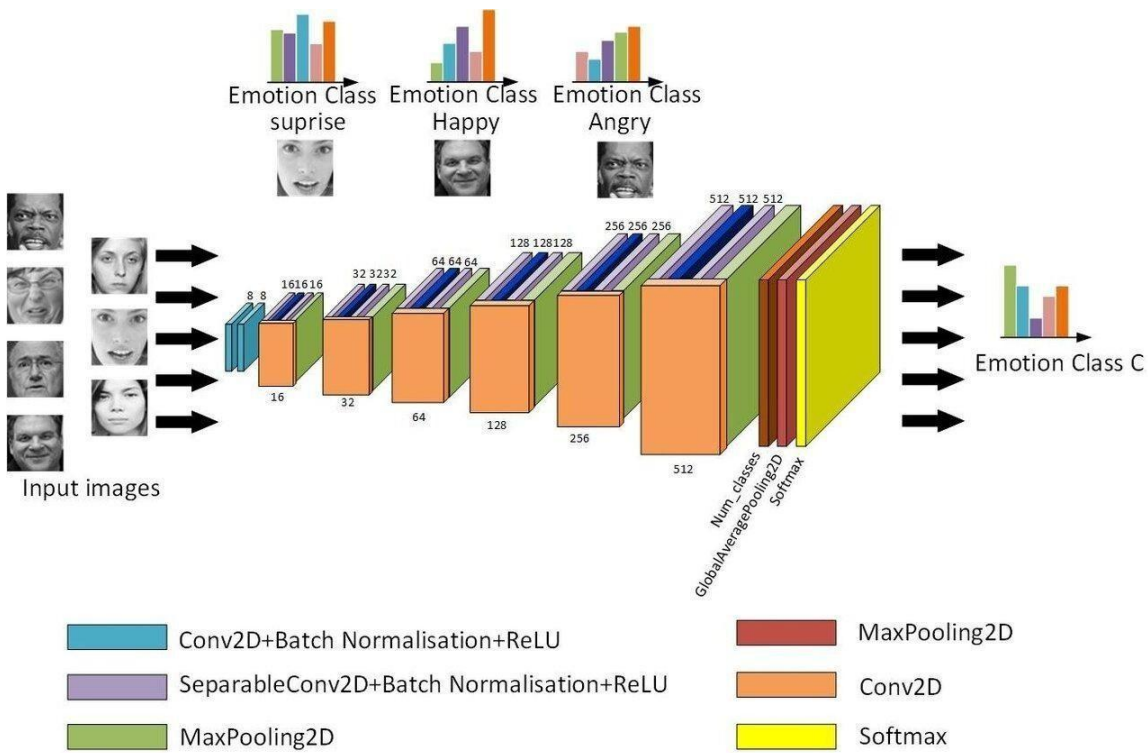
The point above hyperplane will be classified as +1 and point below hyperplane is classified as -1.

$$n \left[\frac{1}{\sum} (\dots - (\dots -)) \right] + \Delta \quad \sqrt{2}$$



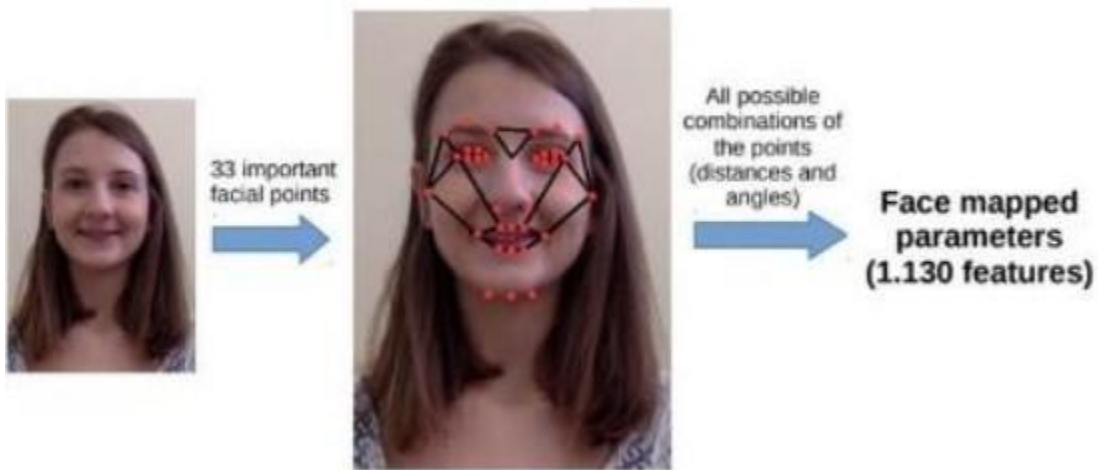
Convolutional Neural Network : A Convolutional Neural is a Deep learning Model used for processing is type of data have grid pattern in it simply data such as images. These networks are built to learn the features and patterns from the images automatically and adaptively. A Convolutional Neural

Network typically comprises three major parts, they are - convolutional block, pooling layers and fully connected networks. CNN accepts the input in the form of tensors with the shape of an image. Then the image is passed through the convolutional layer for the abstraction of features from it with a stack of multiple mathematical operations. Convolutional Neural Network (CNN) has been pre-dominant among various deep learning models. This is a class of artificial neural networks that have produced astonishing results in most of the computer vision tasks. A Convolutional Neural Network is a Deep learning Model used for processing is type of data have grid pattern in it simply data such as images. These networks are built to learn the features and patterns from the images automatically and adaptively. A Convolutional Neural Network typically comprises three major parts, they are - convolutional block, pooling layers and fully connected networks. CNN accepts the input in the form of tensors with the shape of an image. Then the image is passed through the convolutional layer for the abstraction of features from it with a stack of multiple mathematical operations. These convolutional layers convolve the inputs and move the output to the next layer. Pooling layers are used to minimise the dimensions of the input by aggregating the outputs of neuron clusters of a layer into a single neuron in the next layer. The fully connected layers are used to classify the images. The neurons present in one layer are connected to each and every neuron in another layer. The hierarchy of extracted features become more complex as the layers feed their output to another layer as input. It uses back propagation and gradient descent for the optimisation of weights and biases.



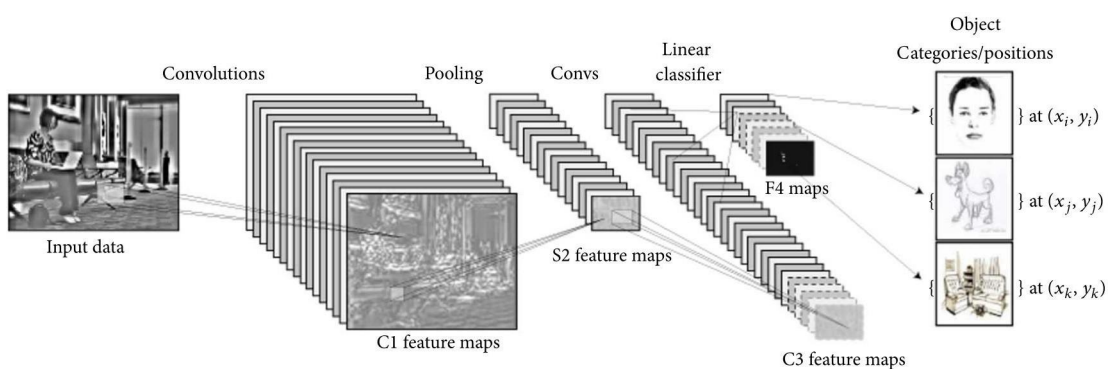
Basic structure of the CNN

Face mapping process



Convolutional Neural Networks (CNNs) were inspired by the visual system's structure, and in particular by the models of it proposed. The first computational models based on these local connectivities between neurons and on hierarchically organized transformations of the image are found in Neocognitron, which describes that when neurons with the same parameters are applied on patches of the previous layer at different locations, a form of translational invariance is acquired. Yann LeCun and his collaborators later designed Convolutional Neural Networks employing the error gradient and attaining very good results in a variety of pattern recognition tasks .

A CNN comprises three main types of neural layers, namely, (i) convolutional layers, (ii) pooling layers, and (iii) fully connected layers. Each type of layer plays a different role. Figure 1 shows a CNN architecture for an object detection in image task. Every layer of a CNN transforms the input volume to an output volume of neuron activation, eventually leading to the final fully connected layers, resulting in a mapping of the input data to a 1D feature vector. CNNs have been extremely successful in computer vision applications, such as face recognition, object detection, powering vision in robotics, and self-driving cars.



Example architecture of a CNN for a computer vision task (object detection).

(i) *Convolutional Layers.* In the convolutional layers, a CNN utilizes various kernels to convolve the whole image as well as the intermediate feature maps, generating various feature maps. Because of the advantages of the convolution operation, several works have proposed it as a substitute for fully connected layers with a view to attaining faster learning times.

(ii) *Pooling Layers.* Pooling layers are in charge of reducing the spatial dimensions (width height) of the input volume for the next convolutional layer. The pooling layer does not affect the depth dimension of the volume. The operation performed by this layer is also called subsampling or downsampling, as the reduction of size leads to a simultaneous loss of information. However, such a loss is beneficial for the network because the decrease in size leads to less computational overhead for the upcoming layers of the network, and also it works against overfitting. Average pooling and max pooling are the most commonly used strategies. A detailed theoretical analysis of max pooling and average pooling performances is given, whereas it was shown that max pooling can lead to faster convergence, select superior invariant features, and improve generalization. Also there are a number of other variations of the pooling layer in the literature, each inspired by different motivations and serving distinct needs, for example, stochastic pooling , spatial pyramid pooling , and def-pooling .

(iii) *Fully Connected Layers.* Following several convolutional and pooling layers, the high-level reasoning in the neural network is performed via fully connected layers. Neurons in a fully connected layer have full connections to all activation in the previous layer, as their name implies. Their activation can hence be computed with a matrix multiplication followed by a bias offset. Fully connected layers eventually convert the 2D feature maps into a 1D feature vector. The derived vector either could be fed forward into a certain number of categories for classification or could be considered as a feature vector for further processing .

The architecture of CNNs employs three concrete ideas: (a) local receptive fields, (b) tied weights, and (c) spatial subsampling. Based on local receptive field, each unit in a convolutional layer receives inputs from a set of neighboring units belonging to the previous layer. This way neurons are capable of extracting elementary visual features such as edges or corners. These features are then combined by the subsequent convolutional layers in order to detect higher order features. Furthermore, the idea that elementary feature detectors, which are useful on a part of an image, are likely to be useful across the entire image is implemented by the concept of tied weights. The concept of tied weights constraints a set of units to have identical weights. Concretely, the units of a convolutional layer are organized in planes. All units of a plane share the same set of weights. Thus, each plane is responsible for constructing a specific feature. The outputs of planes are called feature maps. Each convolutional layer consists of several planes, so that multiple feature maps can be constructed at each location.

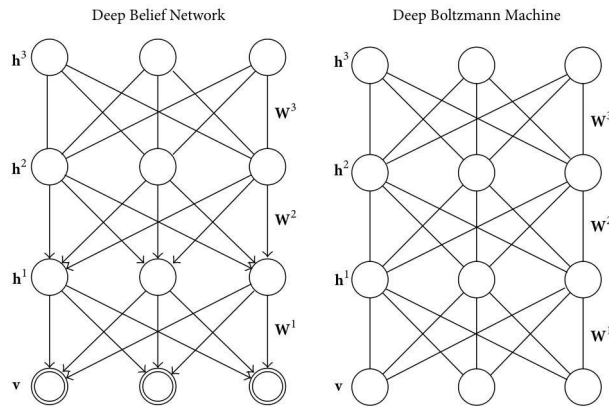
During the construction of a feature map, the entire image is scanned by a unit whose states are stored at corresponding locations in the feature map. This construction is equivalent to a convolution operation, followed by an additive bias term and sigmoid function: where d stands for the depth of the convolutional layer, W is the weight matrix, and b is the bias term. For fully connected neural networks, the weight matrix is full, that is, connects every input to every unit with different weights. For CNNs, the weight matrix is very sparse due to the concept of tied weights. Thus, W has the form of where w_i are matrices having the same dimensions with the units' receptive fields. Employing a sparse weight matrix reduces the number of network's tunable parameters and thus increases its generalization ability. Multiplying with layer inputs is like convolving the input with w_i , which can be seen as a trainable filter. If the input to convolutional layer is of dimension $n \times n$ and the receptive field of units at a specific plane of convolutional layer is of dimension $f \times f$, then the constructed feature map will be a matrix of dimensions $(n-f+1) \times (n-f+1)$.

Specifically, the element of feature map at (i, j) location will be z_{ij} where the bias term is scalar. One of the difficulties that may arise with training of CNNs has to do with the large number of parameters that have to be learned, which may lead to the problem of overfitting. To this end, techniques such as stochastic pooling, dropout, and data augmentation have been proposed. Furthermore, CNNs are often subjected to pretraining, that is, to a process that initializes the network with pretrained parameters instead of randomly set ones. Pretraining can accelerate the learning process and also enhance the generalization capability of the network.

Overall, CNNs were shown to significantly outperform traditional machine learning approaches in a wide range of computer vision and pattern recognition tasks [33], examples of which will be presented in Section 3. Their exceptional performance combined with the relative easiness in training are the main reasons that explain the great surge in their popularity over the last few years.

Deep Belief Networks and Deep Boltzmann Machines

Deep Belief Networks and Deep Boltzmann Machines are deep learning models that belong in the “Boltzmann family,” in the sense that they utilize the Restricted Boltzmann Machine (RBM) as learning module. The Restricted Boltzmann Machine (RBM) is a generative stochastic neural network. DBNs have undirected connections at the top two layers which form an RBM and directed connections to the lower layers. DBMs have undirected connections between all layers of the network. A graphic depiction of DBNs and DBMs can be found in Figure 2. In the following subsections, we will describe the basic characteristics of DBNs and DBMs, after presenting their basic building block, the RBM.



Deep Belief Network (DBN) and Deep Boltzmann Machine (DBM). The top two layers of a DBN form an undirected graph and the remaining layers form a belief network with directed, top-down connections. In a DBM, all connections are undirected.

Restricted Boltzmann Machines

A Restricted Boltzmann Machine is an undirected graphical model with stochastic visible variables and stochastic hidden variables, where each visible variable is connected to each hidden variable. An RBM is a variant of the Boltzmann Machine, with the restriction that the visible units and hidden units must form a bipartite graph. This restriction allows for more efficient training algorithms, in particular the gradient-based contrastive divergence algorithm.

The model defines the energy function $E(v, h)$ where v and h are the model parameters; that is, w_{ij} represents the symmetric interaction term between visible unit i and hidden unit j , and b_i and c_j are bias terms.

A detailed explanation along with the description of a practical way to train RBMs was given, whereas discusses the main difficulties of training RBMs and their underlying reasons and proposes a new algorithm with an adaptive learning rate and an enhanced gradient, so as to address the aforementioned difficulties.

Deep Belief Networks

Deep Belief Networks (DBNs) are probabilistic generative models which provide a joint probability distribution over observable data and labels. They are formed by stacking RBMs and training them in a greedy manner, as was proposed. A DBN initially employs an efficient layer-by-layer greedy learning strategy to initialize the deep network, and, in the sequel, fine-tunes all weights jointly with the desired outputs. DBNs are graphical models which learn to extract a deep hierarchical representation of the training data. They model the joint distribution between observed vector and the hidden layers as follows: where $p(v)$ is a conditional distribution for the visible units at level l conditioned on the hidden units of the RBM at level $l-1$, and $p(v, h)$ is the visible-hidden joint distribution in the top-level RBM.

The principle of greedy layer-wise unsupervised training can be applied to DBNs with RBMs as the building blocks for each layer. A brief description of the process follows: (1) Train the first layer as an RBM that models the raw input as its visible layer. (2) Use that first layer to obtain a representation of the input that will be used as data for the second layer. Two common solutions exist. This representation can be chosen as being the mean activation or samples of h . (3) Train the second layer as an RBM, taking the transformed data (samples or mean activation) as training examples (for the visible layer of that RBM). (4) Iterate steps (2) and (3) for the desired number of layers, each time propagating upward either samples or mean values. (5) Fine-tune all the parameters of this deep architecture with respect to a proxy for the DBN log-likelihood, or with respect to a supervised training criterion (after adding extra learning machinery to convert the learned representation into supervised predictions, e.g., a linear classifier).

There are two main advantages in the above-described greedy learning process of the DBNs. First, it tackles the challenge of appropriate selection of parameters, which in some cases can lead to poor local

optima, thereby ensuring that the network is appropriately initialized. Second, there is no requirement for labelled data since the process is unsupervised. Nevertheless, DBNs are also plagued by a number of shortcomings, such as the computational cost associated with training a DBN and the fact that the steps towards further optimization of the network based on maximum likelihood training approximation are unclear [41]. Furthermore, a significant disadvantage of DBNs is that they do not account for the two-dimensional structure of an input image, which may significantly affect their performance and applicability in computer vision and multimedia analysis problems. However, a later variation of the DBN, the Convolutional Deep Belief Network (CDBN), uses the spatial information of neighboring pixels by introducing convolutional RBMs, thus producing a translation invariant generative model that successfully scales when it comes to high dimensional images, as is evidenced .

Deep Boltzmann Machines

Deep Boltzmann Machines (DBMs) are another type of deep model using RBM as their building block. The difference in architecture of DBNs is that, in the latter, the top two layers form an undirected graphical model and the lower layers form a directed generative model, whereas in the DBM all the connections are undirected. DBMs have multiple layers of hidden units, where units in odd-numbered layers are conditionally independent of even-numbered layers, and vice versa. As a result, inference in the DBM is generally intractable. Nonetheless, an appropriate selection of interactions between visible and hidden units can lead to more tractable versions of the model. During network training, a DBM jointly trains all layers of a specific unsupervised model, and instead of maximizing the likelihood directly, the DBM uses a stochastic maximum likelihood (SML) based algorithm to maximize the lower bound on the likelihood. Such a process would seem vulnerable to falling in poor local minima, leaving several units effectively dead. Instead, a greedy layer-wise training strategy was proposed, which

essentially consists in pretraining the layers of the DBM, similarly to DBN, namely, by stacking RBMs and training each layer to independently model the output of the previous layer, followed by a final joint fine-tuning.

Regarding the advantages of DBMs, they can capture many layers of complex representations of input data and they are appropriate for unsupervised learning since they can be trained on unlabeled data, but they can also be fine-tuned for a particular task in a supervised fashion. One of the attributes that sets DBMs apart from other deep models is that the approximate inference process of DBMs includes, apart from the usual bottom-up process, a top-down feedback, thus incorporating uncertainty about inputs in a more effective manner. Furthermore, in DBMs, by following the approximate gradient of a variational lower bound on the likelihood objective, one can jointly optimize the parameters of all layers, which is very beneficial especially in cases of learning models from heterogeneous data originating from different modalities .

As far as the drawbacks of DBMs are concerned, one of the most important ones is, as mentioned above, the high computational cost of inference, which is almost prohibitive when it comes to joint optimization in sizeable datasets. Several methods have been proposed to improve the effectiveness of DBMs. These include accelerating inference by using separate models to initialize the values of the hidden units in all layer, or other improvements at the pretraining stage or at the training stage .

Stacked (Denoising) Autoencoders

Stacked Autoencoders use the autoencoder as their main building block, similarly to the way that Deep Belief Networks use Restricted Boltzmann Machines as component. It is therefore important to briefly

present the basics of the autoencoder and its denoising version, before describing the deep learning architecture of Stacked (Denoising) Autoencoders.

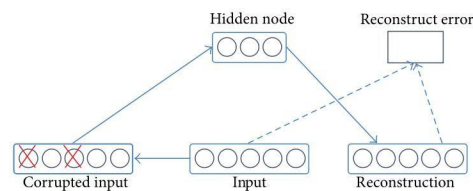
Autoencoders

An autoencoder is trained to encode the input into a representation in a way that input can be reconstructed. The target output of the autoencoder is thus the autoencoder input itself. Hence, the output vectors have the same dimensionality as the input vector. In the course of this process, the reconstruction error is being minimized, and the corresponding code is the learned feature. If there is one linear hidden layer and the mean squared error criterion is used to train the network, then the hidden units learn to project the input in the span of the first principal components of the data. If the hidden layer is nonlinear, the autoencoder behaves differently from PCA, with the ability to capture multimodal aspects of the input distribution. The parameters of the model are optimized so that the average reconstruction error is minimized. There are many alternatives to measure the reconstruction error, including the traditional squared error: where function is the *decoder* and is the reconstruction produced by the model.

If the input is interpreted as bit vectors or vectors of bit probabilities, then the loss function of the reconstruction could be represented by cross-entropy; that is, The goal is for the representation (or *code*) to be a distributed representation that manages to capture the coordinates along the main variations of the data, similarly to the principle of Principal Components Analysis (PCA). Given that is not lossless, it is impossible for it to constitute a successful compression for all input . The aforementioned optimization process results in low reconstruction error on test examples from the same distribution as the training examples but generally high reconstruction error on samples arbitrarily chosen from the input space.

Denoising Autoencoders

The denoising autoencoder is a stochastic version of the autoencoder where the input is stochastically corrupted, but the uncorrupted input is still used as target for the reconstruction. In simple terms, there are two main aspects in the function of a denoising autoencoder: first it tries to encode the input (namely, preserve the information about the input), and second it tries to undo the effect of a corruption process stochastically applied to the input of the autoencoder (see Figure 3). The latter can only be done by capturing the statistical dependencies between the inputs. It can be shown that the denoising autoencoder maximizes a lower bound on the log-likelihood of a generative model.



Denoising autoencoder .

In the image above, the stochastic corruption process arbitrarily sets a number of inputs to zero. Then the denoising autoencoder is trying to predict the corrupted values from the uncorrupted ones, for randomly selected subsets of missing patterns. In essence, the ability to predict any subset of variables from the remaining ones is a sufficient condition for completely capturing the joint distribution between a set of variables. It should be mentioned that using autoencoders for denoising was introduced in earlier works, but the substantial contribution of lies in the demonstration of the successful use of the method for unsupervised pretraining of a deep architecture and in linking the denoising autoencoder to a generative model.

Stacked (Denoising) Autoencoders

It is possible to stack denoising autoencoders in order to form a deep network by feeding the latent representation (output code) of the denoising autoencoder of the layer below as input to the current layer. The unsupervised pretraining of such an architecture is done one layer at a time. Each layer is trained as a denoising autoencoder by minimizing the error in reconstructing its input (which is the output code of the previous layer). When the first layers are trained, we can train the n th layer since it will then be possible to compute the latent representation from the layer underneath.

When pretraining of all layers is completed, the network goes through a second stage of training called fine-tuning. Here supervised fine-tuning is considered when the goal is to optimize prediction error on a supervised task. To this end, a logistic regression layer is added on the output code of the output layer of the network. The derived network is then trained like a multilayer perceptron, considering only the encoding parts of each autoencoder at this point. This stage is supervised, since the target class is taken into account during training.

As is easily seen, the principle for training stacked autoencoders is the same as the one previously described for Deep Belief Networks, but using autoencoders instead of Restricted Boltzmann Machines. A number of comparative experimental studies show that Deep Belief Networks tend to outperform stacked autoencoders, but this is not always the case, especially when DBNs are compared to Stacked Denoising Autoencoders.

One strength of autoencoders as the basic unsupervised component of a deep architecture is that, unlike with RBMs, they allow almost any parametrization of the layers, on condition that the training criterion is continuous in the parameters. In contrast, one of the shortcomings of SAs is that they do not

correspond to a generative model, when with generative models like RBMs and DBNs, samples can be drawn to check the outputs of the learning process.

- **OpenCV** : OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

This paper discusses about various frameworks proposed for participation the board utilizing various advancements. In view of this conversation another methodology for participation the board is proposed to be utilized explicitly for customary level schools. The proposed framework comprises of RFID part and Mobile application part. The RFID part is proposed for capturing understudy participation and recording in the back end information base. The application part is planned for imparting attendance data to their parents. The application part utilized as a backup for recording the attendance in the event if there is no power or no enough assets to send the RFID part. This system proposes an automated attendance management system which handles the issue of recognition of faces in biometric frameworks subject to various constant situations, for example, light, revolution and scaling. The model consolidates a camera that takes input image, a calculation to identify a face from the input picture, encode it and perceive the face and imprint the participation in a spreadsheet and convert it into PDF record. The camera of an android phone takes the picture and sends it to the server where faces are recognized from data set and attendance is marked. This system presents another strategy utilizing Local Binary Pattern (LBP) calculation joined with advanced image processing, for example, Contrast Adjustment, Bilateral

Filter, Image Blending and Histogram Equalization to address a portion of the issues hampering face recognition accuracy to improve the LBP codes, hence improve the accuracy of the general face acknowledgment framework. The examination results show that the method is exceptionally precise, solid and powerful for face acknowledgment system that can be basically executed in real-life environment as a programmed attendance management system. This research develops the attendance system which intends to build up the confronting orderly framework to be more viable and the mechanic of the system which students can easily be approved. The trial of this research is to discover the best approach to recognize the face by utilizing the method of Android Face Recognition with Deep Learning which can accurately recognize up to 97%. The data set is associated with Attendance Management System web worker by utilizing cloud storage. The outcome on screen on the application has the goal of students confirming and checking the data. The proposed framework gives highlights such as recognition of faces, extraction of the highlights, discovery of removed highlights, investigation of students' attendance and monthly report. The proposed system incorporates methods, for example, picture contrasts, integral pictures, Ada-Boost, Haar-like highlights and falling classifier for detecting features. Faces are perceived using progressed LBP utilizing the information base that contains pictures of students and is utilized to recognize faces of students utilizing the caught picture. Better precision is achieved in outcomes and the framework considers the progressions that happens in the face throughout the time frame. The proposed system comprises of a high resolution digital camera that is placed on a gate/door to monitor the class. The pictures captured by the camera are lead to a computer application for further analysis. The obtained pictures are compared to reference images that are stored in the database. The references images are of the students. This framework targets giving a system to consequently record the students' participation during class hours in a room utilizing facial recognition innovation rather than the conventional manual methods. The target behind this research is to

completely examine the field of pattern recognition which is vital and is used in different applications like identification and detection. This system uses face detection which is used to recognize human face and concentrate the locale of interest. It further deals with the areas of interest via face recognition methods. This system offers a novel and robust hybrid procedure of skin-color model for face detection and indistinct neural network to recognize the faces which are already detected. Proposed hybrid algorithm would be given with high precision and low false positive outcome by using skin shading model and speed processing to recognize the detected picture by utilizing fuzzy neural network. The system starts the procedure through Create Training Dataset by using face detection method. Create Training Dataset refers to applying face detection algorithm to the selected images from database and storing the face in the database. Work-flow of the proposed system for detecting face is, load pictures from Yale base, apply pre-processing which includes rgb to gray transformation and the process of histogram equalization. Haar classifier is applied on handled picture to identify face and the detected faces are stored .

ALGORITHMS

Step 1 : Read the images from the dataset using openCV

Step 2 : Store the images and the names assigned to it.

Step 3 : Change the image format from BGR to RGB.

Step 4 : Find the encoding of the images using `face_encoding()` function of `Face_recognition` module and store the encodings of the images.

Step 5 : Read the camera for input using openCV

Step 6 :Change the incoming image format to RGB.

Step 7:find the number of faces in the frame using `face_locations()` function of `face_recognition` module and also find their encoding using `face_encoding()`.

Step 8 :Match the incoming face to the ones in our database using `compare_face()`

Step 9 :If the face matches then store the name and date and time in the csv file(or database).

IMPLEMENTATION

Working of Hog descriptor :

Feature engineering is a game-changer in the world of machine learning algorithms. It's actually one of my favorite aspects of being a data scientist! This is where we get to experiment the most – to engineer new features from existing ones and improve our model's performance.

Some of the top data scientists in the world rely on feature engineering to boost their leaderboard score in hackathons. I'm sure you would even have used various feature engineering techniques on structured data.

Can we extend this technique to unstructured data, such as images? It's an intriguing riddle for computer vision enthusiasts and one we will solve in this article. Get ready to perform feature engineering in the form of feature extraction on image data.

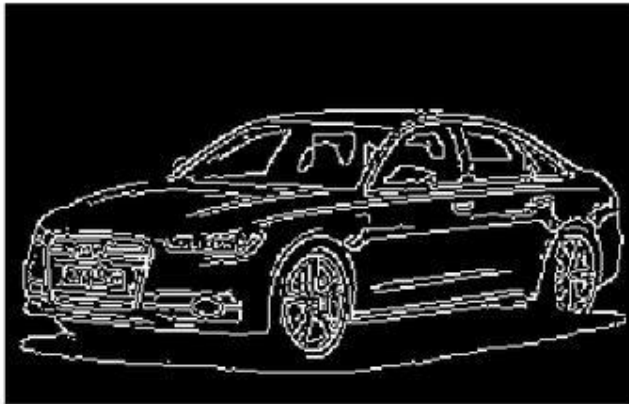
What is a Feature Descriptor?

You might have had this question since you read the heading. So let's clear that up first before we jump into the HOG part of the article.

Take a look at the two images shown below. Can you differentiate between the objects in the image



We can clearly see that the right image here has a dog and the left image has a car. Now, let me make this task slightly more complicated – identify the objects shown in the image below:



Still easy, right? Can you guess what was the difference between the first and the second case? The first pair of images had a lot of information, like the shape of the object, its color, the edges, background, etc.

On the other hand, the second pair had much less information (only the shape and the edges) but it was still enough to differentiate the two images.

Do you see where I am going with this? We were easily able to differentiate the objects in the second case because it had the necessary information we would need to identify the object. And that is exactly what a feature descriptor does:

It is a simplified representation of the image that contains only the most important information about the image.

There are a number of feature descriptors out there. Here are a few of the most popular ones:

HOG: Histogram of Oriented Gradients

SIFT: Scale Invariant Feature Transform

SURF: Speeded-Up Robust Feature

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection.

Let's look at some important aspects of HOG that makes it different from other feature descriptors:

The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images? In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting the gradient and orientation (or you can say magnitude and direction) of the edges

Additionally, these orientations are calculated in 'localized' portions. This means that the complete

image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections

Finally the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

To put a formal definition to this:

The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image.

Implementing HOG using tools like OpenCV is extremely simple. It's just a few lines of code since we have a predefined function called `hog` in the `skimage.feature` library. Our focus in this article, however, is on how these features are actually calculated.

Process of Calculating the Histogram of Oriented Gradients (HOG)

We should now have a basic idea of what a HOG feature descriptor is. It's time to delve into the core idea behind this article. Let's discuss the step-by-step process to calculate HOG.

Step 1: Preprocess the Data (64 x 128)

This is a step most of you will be pretty familiar with. Preprocessing data is a crucial step in any machine learning project and that's no different when working with images.

We need to preprocess the image and bring down the width to height ratio to 1:2. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations pretty simple. In fact, this is the exact value used in the original paper.

Step 2: Calculating Gradients (direction x and y)

The next step is to calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions. Here, I am going to take a small patch from the image and calculate the gradients on that.

We will get the pixel values for this patch. Let's say we generate the below pixel matrix for the given patch (the matrix shown here is merely used as an example and these are not the original pixel values for the given patch):

I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.

Hence the resultant gradients in the x and y direction for this pixel are:

$$\text{Change in X direction}(G_x) = 89 - 78 = 11$$

$$\text{Change in Y direction}(G_y) = 68 - 56 = 8$$

This process will give us two new matrices – one storing gradients in the x-direction and the other storing gradients in the y direction. This is similar to using a Sobel Kernel of size 1. The magnitude would be higher when there is a sharp change in intensity, such as around the edges.

We have calculated the gradients in both x and y direction separately. The same process is repeated for all the pixels in the image. The next step would be to find the magnitude and orientation using these values.

The gradients are basically the base and perpendicular here. So, for the previous example, we had Gx and Gy as 11 and 8. Let's apply the Pythagoras theorem to calculate the total gradient magnitude:

$$\text{Total Gradient Magnitude} = \sqrt{(G_x)^2 + (G_y)^2}$$

$$\text{Total Gradient Magnitude} = \sqrt{(11)^2 + (8)^2} = 13.6$$

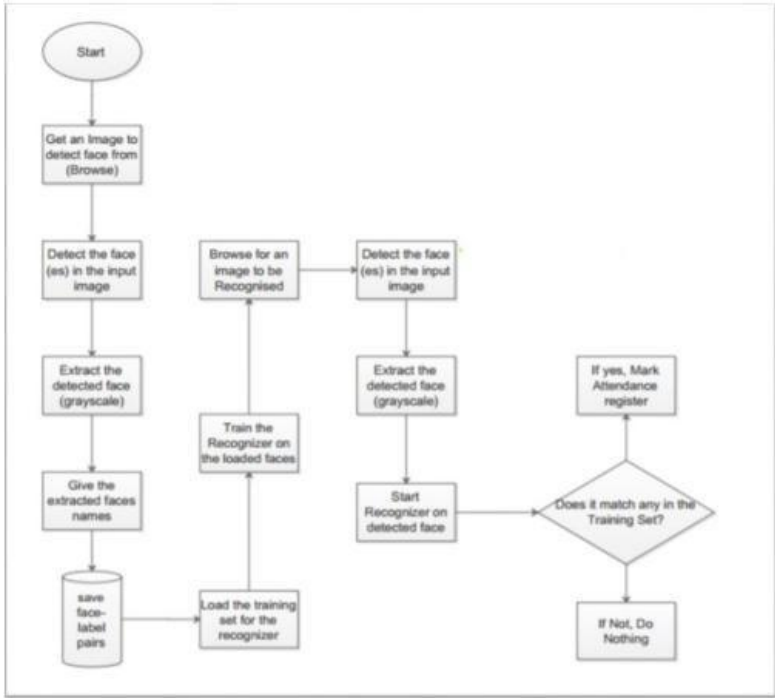
Next, calculate the orientation (or direction) for the same pixel. We know that we can write the tan for the angles:

$$\tan(\Phi) = G_y / G_x$$

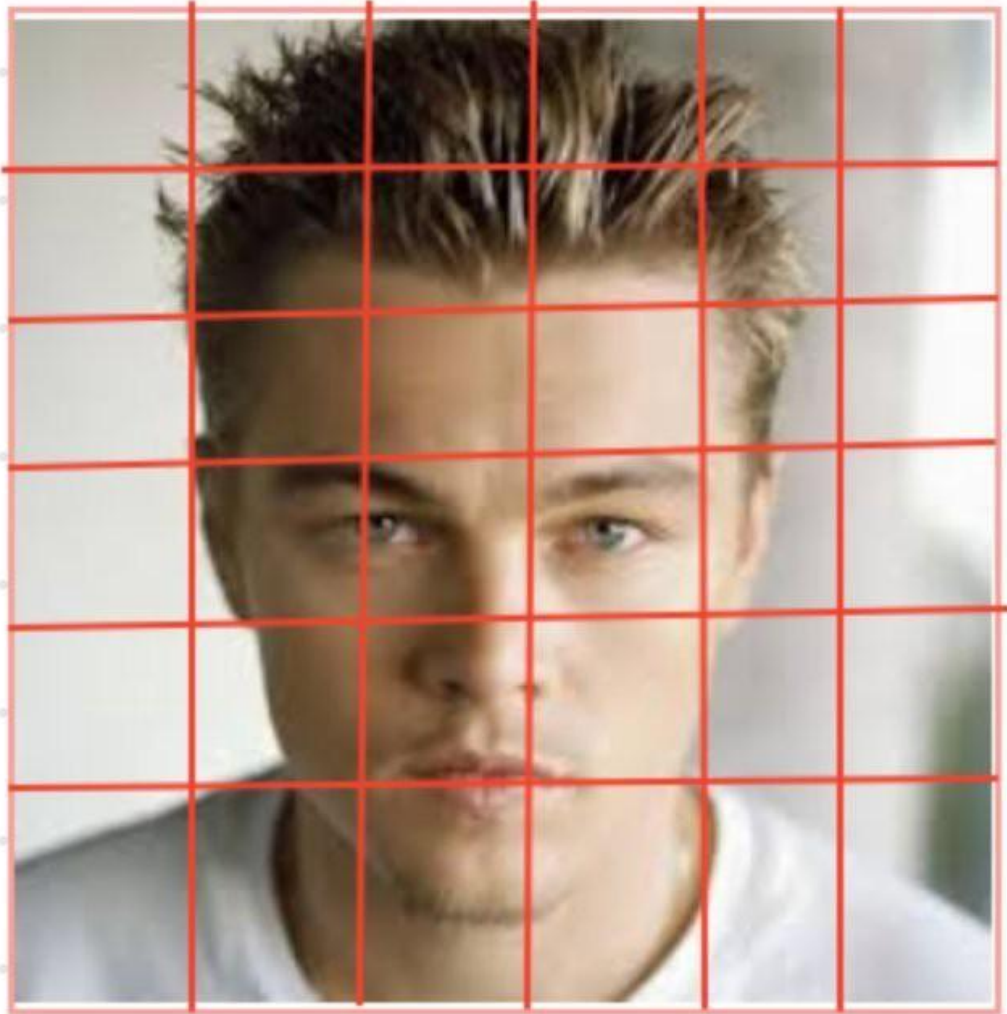
$$\text{Hence, the value of the angle would be: } \Phi = \text{atan}(G_y / G_x)$$

The orientation comes out to be 36 when we plug in the values. So now, for every pixel value, we have the total gradient (magnitude) and the orientation (direction). We need to generate the histogram using these gradients and orientations.

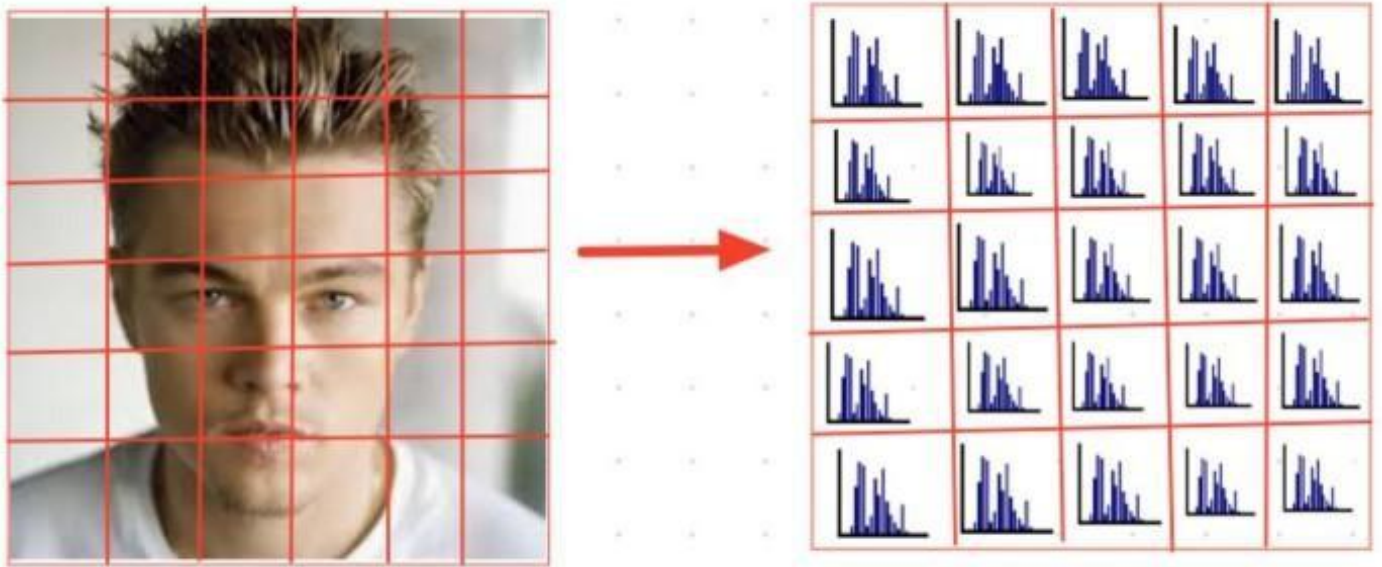
But hang on – we need to take a small break before we jump into how histograms are created in the HOG feature descriptor. Consider this a small step in the overall process. And we'll start this by discussing some simple methods of creating Histograms using the two values that we have – gradients and orientation.



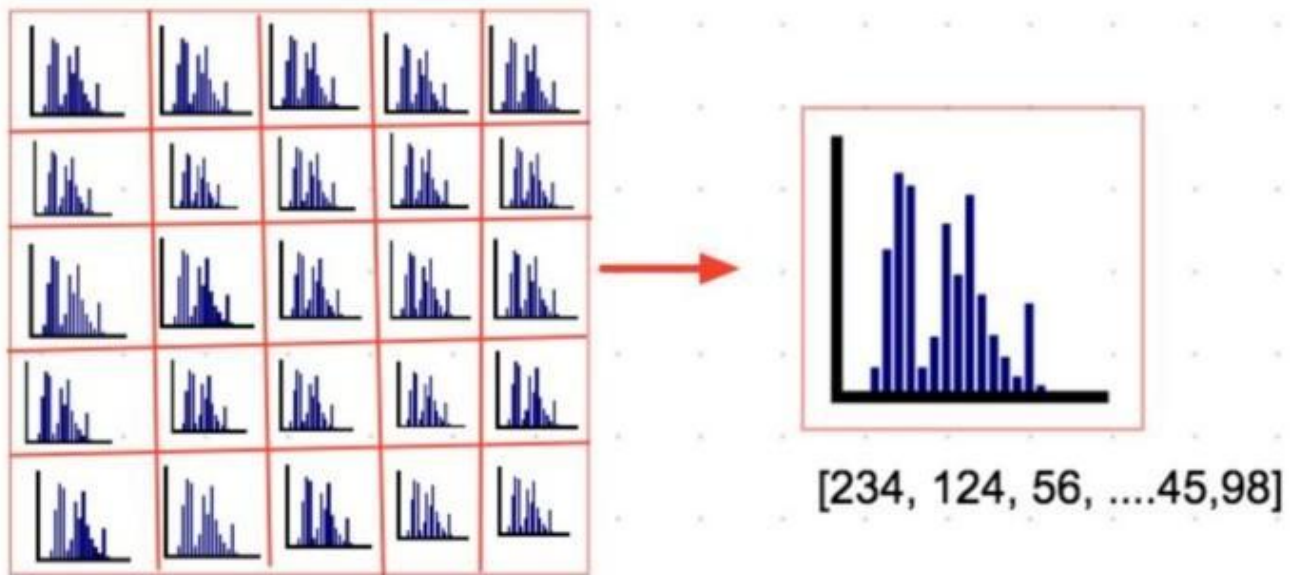
Flowchart of the process



Divide the image into small cells



Compute the histogram of each cell



Combined small histograms into one histogram which is a final feature vector

CODE

```
import cv2

import numpy as np

import face_recognition

import os

from datetime import datetime

path = 'images'

images = []

personNames = []

myList = os.listdir(path)

print(myList)

for cu_img in myList:

    current_img = cv2.imread(f'{path}/{cu_img}')

    images.append(current_img)

    personNames.append(os.path.splitext(cu_img)[0])

print(personNames)
```

```
def faceEncodings(images):
```

```
encodeList = []

for img in images:

    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    encode = face_recognition.face_encodings(img)[0]

    encodeList.append(encode)

return encodeList
```

```
def attendance(name):

    with open('Attendance.csv', 'r+') as f:

        myDataList = f.readlines()

        nameList = []

        for line in myDataList:

            entry = line.split(',')

            nameList.append(entry[0])

        if name not in nameList:

            time_now = datetime.now()

            tStr = time_now.strftime('%H:%M:%S')

            dStr = time_now.strftime('%d/%m/%Y')

            f.writelines(f'\n{name},{tStr},{dStr}')
```

```
encodeListKnown = faceEncodings(images)

print("All Encodings Complete!!!")
```

```
cap = cv2.VideoCapture(1)
```

```
while True:
```

```
    ret, frame = cap.read()
```

```
    faces = cv2.resize(frame, (0, 0), None, 0.25, 0.25)
```

```
    faces = cv2.cvtColor(faces, cv2.COLOR_BGR2RGB)
```

```
    facesCurrentFrame = face_recognition.face_locations(faces)
```

```
    encodesCurrentFrame = face_recognition.face_encodings(faces,
facesCurrentFrame)
```

```
    for encodeFace, faceLoc in zip(encodesCurrentFrame,
facesCurrentFrame):
```

```
        matches = face_recognition.compare_faces(encodeListKnown,
encodeFace)
```

```
        faceDis = face_recognition.face_distance(encodeListKnown,
encodeFace)
```

```
        # print(faceDis)
```



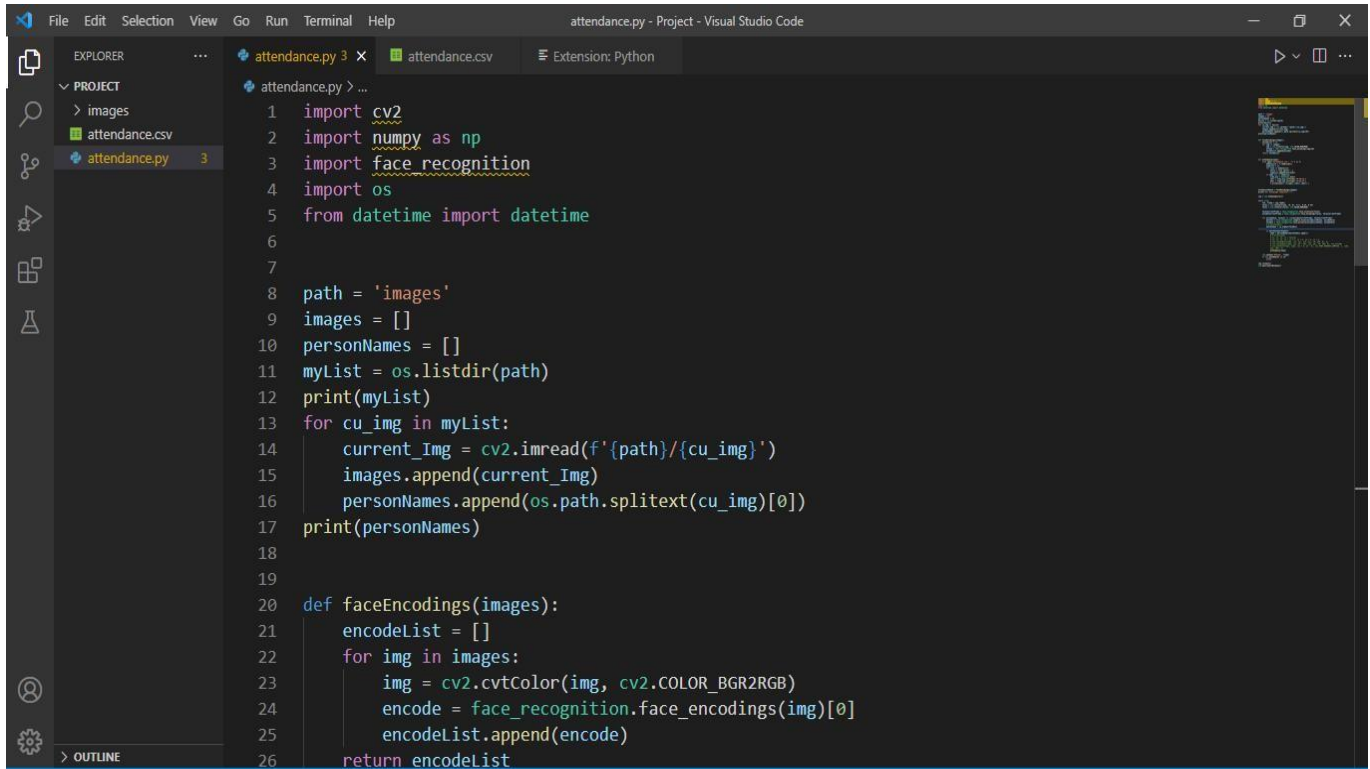
```
matchIndex = np.argmin(faceDis)
```

```
if matches[matchIndex]:  
    name = personNames[matchIndex].upper()  
    attendance(name)
```

```
cv2.imshow("Webcam", frame)  
if cv2.waitKey(0) == 13:  
    break
```

```
cap.release()
```

```
cv2.destroyAllWindows()
```



```
attendance.py - Project - Visual Studio Code
File Edit Selection View Go Run Terminal Help
attendance.py 3 x attendance.csv Extension: Python
EXPLORER
PROJECT
  images
  attendance.csv
  attendance.py 3
  OUTLINE
attendance.py > ...
1 import cv2
2 import numpy as np
3 import face_recognition
4 import os
5 from datetime import datetime
6
7
8 path = 'images'
9 images = []
10 personNames = []
11 myList = os.listdir(path)
12 print(myList)
13 for cu_img in myList:
14     current_Img = cv2.imread(f'{path}/{cu_img}')
15     images.append(current_Img)
16     personNames.append(os.path.splitext(cu_img)[0])
17 print(personNames)
18
19
20 def faceEncodings(images):
21     encodeList = []
22     for img in images:
23         img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
24         encode = face_recognition.face_encodings(img)[0]
25         encodeList.append(encode)
26     return encodeList
```

Reading the images.

```
Go Run Terminal Help • attendance.py - Project - Visual Studio Code
attendance.py 3 • attendance.csv Extension: Python
attendance.py > ...
48 while True:
49     ret, frame = cap.read()
50     faces = cv2.resize(frame, (0, 0), None, 0.25, 0.25)
51     faces = cv2.cvtColor(faces, cv2.COLOR_BGR2RGB)
52
53     facesCurrentFrame = face_recognition.face_locations(faces)
54     encodesCurrentFrame = face_recognition.face_encodings(faces, facesCurrentFrame)
55
56     for encodeFace, faceLoc in zip(encodesCurrentFrame, facesCurrentFrame):
57         matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
58         faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
59         # print(faceDis)
60         matchIndex = np.argmin(faceDis)
61
62         if matches[matchIndex]:
63             name = personNames[matchIndex].upper()
64
65             attendance(name)
66
67     cv2.imshow('Webcam', frame)
68     if cv2.waitKey(0) == 13:
69         break
70
71 cap.release()
72 cv2.destroyAllWindows()
```

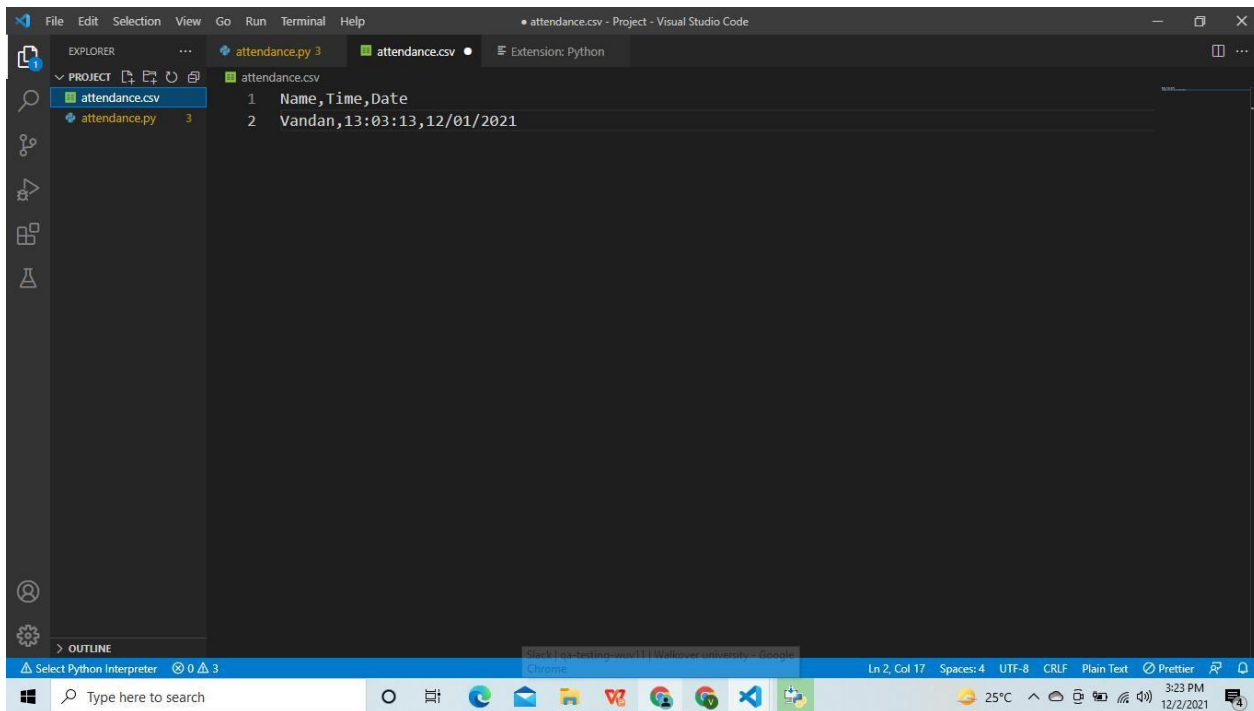
Comparing faces from the camera input to our dataset

```
26     return encodeList
27
28
29 def attendance(name):
30     with open('Attendance.csv', 'r+') as f:
31         myDataList = f.readlines()
32         nameList = []
33         for line in myDataList:
34             entry = line.split(',')
35             nameList.append(entry[0])
36         if name not in nameList:
37             time_now = datetime.now()
38             tStr = time_now.strftime('%H:%M:%S')
39             dStr = time_now.strftime('%d/%m/%Y')
40             f.writelines(f'\n{name},{tStr},{dStr}')
41
42
43 encodeListKnown = faceEncodings(images)
44 print('All Encodings Complete!!!')
45
46 cap = cv2.VideoCapture(1)
47
48 while True:
49     ret, frame = cap.read()
50     faces = cv2.resize(frame, (0, 0), None, 0.25, 0.25)
```

Marking the attendance if the face matches.

Result

Store the name of the students along with date and time:



The screenshot shows the Visual Studio Code interface with a file named 'attendance.csv' open. The file contains the following data:

Line	Content
1	Name,Time,Date
2	Vandan,13:03:13,12/01/2021

The interface also shows the Explorer sidebar with 'attendance.csv' and 'attendance.py' files, and the status bar at the bottom indicating 'Ln 2, Col 17', 'Spaces: 4', 'UTF-8', 'CRLF', 'Plain Text', and 'Prettier'.

Conclusion

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

To achieve this goal computers must be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach -- however, substantial research remains to be done in making person recognition technology work reliably, in widely varying conditions using information from single or multiple modalities.

The Automated Visage Recognition based attendance system has the main goal of automating the process of managing attendance and revolves around the fulcrum of Automation. The system involves mainly three steps: Registration, Authentication and Update. Its fundamental goal is to eliminate time consumption and the need to maintain paperwork. Since the advent of technology, humans have progressed to evolve and adapt to changes based on their convenience. Bringing this

idea to practicality helps the common man to effectively and efficiently progress and this system helps the whole organization to evolve and achieve what is necessary by eliminating the tedious and iterative tasks.

The system is built on a combination of several technologies and has overcome most manual flaws and thus stands apart from the existing systems. However, the database management is an area of concern and needs to be filtered out on a cyclic and timely basis to avoid data overflow.

Furthermore, since the system is built upon the features of Machine learning, effective training and efficient procedures must be followed to achieve 100% accuracy. Finally, we can build the system as an integration of the current attendance systems being used in the organization in order to achieve maximum efficiency.

We used two FaceNet models for training purposes. Initially, we trained both the models with a non-augmented dataset and achieved the following results. The facial recognition process begins with a camera, installed on any compatible device in communication with said camera. This application is then able to use computer vision and a deep neural network to find a prospective face within its stream. There are two primary effective ways to do so: The first is the TensorFlow object detection model and the second is Caffe face tracking. Both these methods have functioned well, and are a part of the OpenCV library. Once a face has been captured, the cropped image will be relayed with an HTTP form-data request to the back end. This facial image is then saved by the API, both on the local file system and in the detection log, appended with a person ID. Based on some papers that we have reviewed, we can conclude that the more datasets that are given to the computer, the higher the success rate will be in identifying face objects. Based on the search results that have been done by the author, we find out that under a controlled environment, the project shows promising results. Some of the controlled environment includes:

1. Background – For the face recognition to work at its best, the background needs to be static. It

is harder to recognize a face in a dynamic background. 2. Lighting – Lighting also imposes an important aspect of recognizing a face. An environment with adequate light shows more accurate results rather than environment. Facial Changes – Some changes to the facial area may cause inconsistency in recognizing the student's faces. For example, students with veil and spectacles are harder to recognize if they did not wear their veil or spectacles.

Based on several papers that we have read, the system can recognize human faces by capturing many examples of faces whose data will be trained as an attempt to recognize faces according to their names. This method is called Deep Learning. So the collection of faces will be studied by the system to experiment to match the name according to his face. This constitutes for RQ1: How the system can identify the face? We attach the identification as shown in Table 2 to be able to answer RQ2. We obtained ten algorithms that were identified based on the results of the main studies that had been extracted. We classify the main research so that it will provide information about the algorithm for face recognition. Each algorithm has its use for face recognition. Based on the paper that has been studied, face recognition has a variety of ways to detect a person's face, so the use of algorithms is needed as needed. Each algorithm also has advantages and disadvantages. Until now the level of accuracy of the algorithm that is best used for face detection is the deep-learning algorithm.

This systematic literature review observes and identifies the Attendance System with Face Recognition. This systematic literature review based on the Deep Learning method, starting with many initial studies collected from online databases. Through the article selection process obtained 37 articles are eligible then analyzed. This study aims at developing and understanding of the Attendance System with Face Recognition. Knowledge of the framework and critical success factors in the implementation of Deep Learning is critical to the successful implementation. FaceNet returns a 128-dimensional vector embedding for each face. The face uses the concept of triplet loss. For RQ to answer the algorithm for

Face detection their's ten algorithms are used. We obtained ten algorithms that were identified based on the results of the main studies that had been extracted. We classify the main research so that it will provide information about the algorithm for face recognition. Each algorithm has its use for face recognition. And until now the level of accuracy of the algorithm that is best used for face detection is a Deep Learning algorithm.

References

- <https://www.kaggle.com/daturks/face-detection-in-images>
- <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>
- <https://learnopencv.com/histogram-of-oriented-gradients/>
- https://en.wikipedia.org/wiki/Face_detection
- https://en.wikipedia.org/wiki/Facial_recognition_system
- https://en.wikipedia.org/wiki/Identity_verification_service