IMPLEMENTATION OF AUTOMATED ATTENDANCE SYSTEM USING DEEP LEARNING AND CNN

A Project report submitted in partial fulfillment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

SK MAHABOOB SUBHANI (317126512167)

MD TAJ NAWAZ (317126512148)

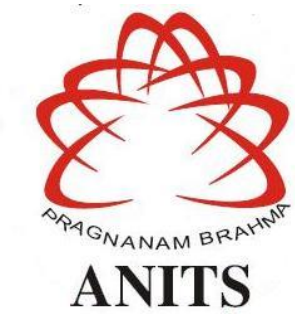
K CHANDRA SUMANTH (31712512139)

PRAMOD POTNURU (317126512159)

Under the guidance of

CH PADMA SREE

ASSISTANT PROFESSOR, ECE DEPARTMENT



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(*Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade*)

Sangivalasa, Bheemili Mandal, Visakhapatnam dist.(A.P)

2020-2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)

Sangivalasa, Bheemili mandal, Visakhapatnam dist.(A.P)

ANITS

CERTIFICATE

This is to certify that the project report entitled **"SMART ATTENDANCE SYSTEM USING DEEP LEARNING"** submitted by K CHANDRA SUMANTH (31712512139), MD TAJ NAWAZ (317126512148), PRAMOD POTNURU (317126512159), SK MAHABOOB SUBHANI (317126512167) in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Electronics & Communication Engineering** of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide
CH PADMA SREE
Assistant Professor
Department of E.C.E
ANITS

Assistant Professor
Department of E.C.E.
Anil Neerukonda
Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department
Dr. V. Rajyalakshmi
Professor & HOD
Department of E.C.E
ANITS

Head of the Department
Department of E C E
Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **CH PADMA SREE** Assistant Professor, Department of Electronics and Communication Engineering, ANITS, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. V. Rajyalakshmi**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa,** for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

SK MAHABOOB SUBHANI (317126512167)

MD TAJ NAWAZ (317126512148)

K CHANDRA SUMANTH (31712512139)

PRAMOD POTNURU (317126512159)

ABSTRACT

When a person is uniquely identified then it is only because of the face which is the crucial part of a human. Using the face, a large number of applications can be implemented like for security purposes at banks, various organizations and also in the areas where there is a large public gathering. In general, various organizations keep on monitoring the performance of the employees or the students by using the attendance. The attendance can be marked either by signature or calling the names of the persons and also through biometric. These ways may sometimes lead to time consuming and common errors by humans. The attendance can be marked by using facial recognition with the help of so called eigen faces and fisher faces algorithms, but these algorithms provide results with less accuracy. So, the proposed solution for this problem is implementing a smart attendance system that automatically marks the attendance of the persons using Face Detection and Face Recognition with easy, less time complexity and also efficiently helps in various administrations. It is achieved by using Deep Learning, CNN, Open Computer Vision Library (Open CV) and Python based face recognition approach.. Deep learning methods, especially convolutional Neural Networks have achieved significant success in the area of computer vision including the difficult face recognition problems. Training of deep models shows exceptional performance with large datasets, but they are not suitable for learning from few samples. The input faces are compared with the images in the data set and will be recognized. The recognized names of the faces along with the time will be automatically updated into a CSV file and this CSV file will be sent to the respective head of the organization through an E-mail. Along with this, the details of the recognized faces will be updated into an attendance system automatically which is designed through web development.

CONTENTS

LIST OF SYMBOLS

$S_{g,t}$	Euclidean distance
W_p	Width of pooling layer
H_p	Height of pooling layer
D_p	Depth of pooling layer
W_c	Width of convolution layer
H_c	Height of convolution layer
D_c	Depth of convolution layer

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

3-D	3-dimensional
AI	Artificial intelligence
RELU	Rectified linear activation function
LBPH	Local binary pattern histograms
ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
DBN	Deep belief network
DNN	Deep neural network
RNN	Recurrent neural network
DL	Deep learning
PCA	Principal component analysis
FDL	Fisher's Linear Discriminant
LDA	Linear Discriminant Analysis
HOG	histograms of oriented gradients
OpenCV	Open Source Computer Vision Library
MNIST	Modified National Institute of Standards and Technology.
CSV	Comma-separated values
k-NN	Kth neural network
ROI	Region Of Interest
ASM	Active Shape Models
AAM	Active Appearance Models
MIXIS	Mixed Interaction Spaces

CHAPTER 1: Introduction

In recent years, deep convolution networks have achieved great success in the area of computer vision. Face recognition has been one of its extensively researched and most interesting applications. The significance of face recognition is due to its technical challenges and wide potential application in video surveillance, identity authentication, multimedia applications, home and office security, law enforcement and different human-computer interaction activities. The appearance of a person in an image is a result of the mutual effects of the illumination, pose, 3-D structure of face, occlusion and background which are uncontrolled constraints that are encountered in real world applications. Different approaches have been proposed to deal with these technical hitches, like in illumination normalization techniques have been proposed and in ANN based classifier is used to resolve this non-linearity. For human identification, the face is important. It is the most recognizable item in an individual. FR is interesting and complicated and affects key applications in many fields, such as security access, personalized identity, law enforcement identification, and banking authentication. Face detection is very easy for humans but for a machine, it is distinct. To date, there is very little knowledge on how an image is autonomized and how the cerebrum encodes it and inner (nose, mouth, eyes) or external highlights (face shape, structure, hairline) used for the effective recognition of the face. Attendance maintenance is a significant function in all the institutions to monitor the performance of the students. Every institute does this in its own way. In Now-a-days every aspect in technology is updated in very quick and rapid manner. When it comes to attendance of an organization marking for 60-80 members at a time in a scheduled work time which consumes larger time and also a uphill task for the person to do, besides this it is a statistical data to store. To avoid this, an attendance system using face detection and face recognition is proposed. With reference to the different face recognition models [1], images are trained, faces are detected and finally attendance is marked to the input image, reference to many theoretical aspects and practical models, this model uses the concept of deep learning [2][3]. The concept of deep learning is used for face detection and recognition with help of CNN. In-order to perform the above process OpenCV is used which consists CNN.CNN (CONVOLUTIONAL NEURAL NETWORKS) is a part of deep learning and another main technology in the recent aspects which gives the feasibility to execute problems in easier, efficient manner. In this project it deals with rectification and giving out better results to make tasks much easier and in a perfect way. The output achieved is a combination of many layers of CNN such as convolution, max pooling, RELU etc. It is widely used in security systems and it can be compared with other biometrics such as fingerprint or eye iris recognition systems. As the number of students in an educational institute or employees at an organization increases, the needs for lecturers or to the organization also increase the complication of

attendance control. This project may be helpful for the explanation of these types of problems. The number of students present in a lecture hall is observed, each person is identified and then the information about the number of students who are present is maintained.

CHAPTER 2: Literature Survey

Over the past few years, the modern researchers have been developing solutions for various kinds of problems in which face detection and recognition has been the major task.

According the various surveys, there a few models which have been developed.

Aiman proposed a model which is used to for face detection. It uses deep learning neural network which with a small data set of images. The images in the data set are applied with gaussian and poison noise which results in doubling the data set. CNN contains the RELU layer which converts the images into better form and hence a good success rate is achieved.

Manna proposed a methodology for face recognition from video using deep learning. It makes use of the CNN architecture and Face Net function of google for facial recognition which has an accuracy of 99.63%. The images are detected using the Euclidean distance between them which gives the good approach for face recognition.

Han held a research for face recognition based on deep learning. It made a study on all the parameters which are required for face detection. It made use of the hidden layers in the CNN for better classification of faces.

Raj proposed an approach face recognition based smart attendance system. It makes use of raspberry pi camera as it provides a better operating system. This model is used to detect the faces of the people present in a room by mounting the camera in the room and finally marking the attendance of the detected faces. It uses LBPH algorithm which has been a promising method for face recognition.

Sudha Narang made a comparison between various face recognition algorithms. It gave a clear insight by putting forward the results generating by using the python and MATLAB. The flow of the model goes with pre-image processing followed by face detection and face recognition with the help of so called LBPH algorithm.

Selvi proposed a model for face recognition based attendance system. The process starts with the pre-image processing techniques like image acquisition, histogram normalization, noise removal, skin calculation and finally face detection. It takes the images from the data base, compares with the input images and finally stores the names of the detected faces into the attendance data base. This gives satisfying results by using the MATLAB software and has a good success rate.

CHAPTER 3: Deep Learning

3.1 Introduction to Deep Learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture. A formal definition of deep learning is- neurons. Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbours. The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

Face recognition is the problem of identifying and verifying people in a photograph by their face. It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it is remained a challenging computer vision problem for decades until recently. Deep learning methods are able to leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as-well and later to outperform the face recognition capabilities of humans.

Fig 3.1-Deep Learning

3.1.1 Architectures:

1. **Deep Neural Network** (DNN) – It is a neural network with a certain level of complexity (having multiple hidden layers in between input and output layers). They are capable of modeling and processing non-linear relationships.
2. **Deep Belief Network (DBN)** – It is a class of Deep Neural Network. It is multi-layer belief networks. **Steps:**
   a. Learn a layer of features from visible units using Contrastive Divergence algorithm.
   b. Treat activations of previously trained features as visible units and then learn features of features.
   c. Finally, the whole DBN is trained when the learning for the final hidden layer is achieved.

3. **Recurrent** (perform same task for every element of a sequence) **Neural Network** – Allows for parallel and sequential computation. Similar to the human brain (large feedback network of connected neurons). They are able to remember important things about the input they received and hence enables them to be more precise.

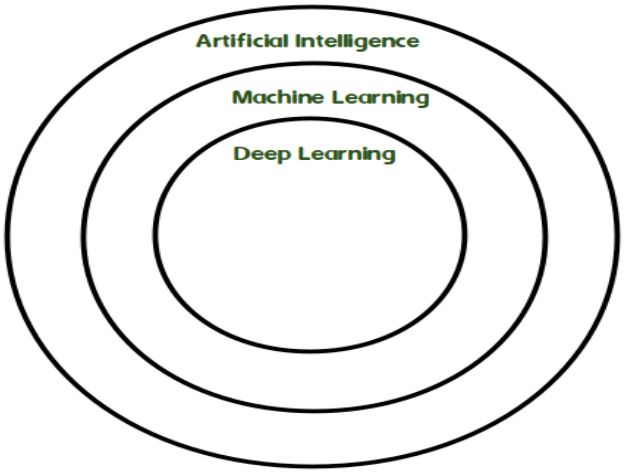


Fig 3.2-Architecture of Deep Learning

3.2 HISTORY:

Fig 3.3- History of Deep Learning

Deep Learning, as a branch of Machine Learning, employs algorithms to process data and imitate the thinking process, or to develop abstractions. Deep Learning (DL) uses layers of algorithms to process data, understand human speech, and visually recognize objects. Information is passed through each layer, with the output of the previous layer providing input for the next layer. The first layer in a network is called the input layer, while the last is called an output layer. All the layers between the two are referred to as hidden layers. Each layer is typically a simple, uniform algorithm containing one kind of activation function.

Feature extraction is another aspect of Deep Learning. Feature extraction uses an algorithm to automatically construct meaningful "features" of the data for purposes of training, learning, and understanding. Normally the Data Scientist, or programmer, is responsible for feature extraction.

The history of Deep Learning can be traced back to 1943, when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain. They used a combination of algorithms and mathematics they called "threshold logic" to mimic the thought process. Since that time, Deep Learning has evolved steadily, with only two significant breaks in its development. Both were tied to the infamous Artificial Intelligence winters.

The earliest efforts in developing Deep Learning algorithms came from Alexey Grigoryevich Ivakhnenko (developed the Group Method of Data Handling) and Valentin Grigor′evich Lapa (author of Cybernetics and Forecasting Techniques) in 1965. They used models with polynomial (complicated equations) activation functions, that were then analyzed statistically. From each layer, the best statistically chosen features were then forwarded on to the next layer (a slow, manual process).

The first "convolutional neural networks" were used by Kunihiko Fukushima. Fukushima designed neural networks with multiple pooling and convolutional layers. In 1979, he developed an artificial neural network, called Neocognitron, which used a hierarchical, multilayered design. This design allowed the computer the "learn" to recognize visual patterns. The networks resembled modern versions, but were trained with a reinforcement

strategy of recurring activation in multiple layers, which gained strength over time. Additionally, Fukushima's design allowed important features to be adjusted manually by increasing the "weight" of certain connections.

3.3 How deep learning works:

First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset. Fifth, Final testing should be done on the dataset.

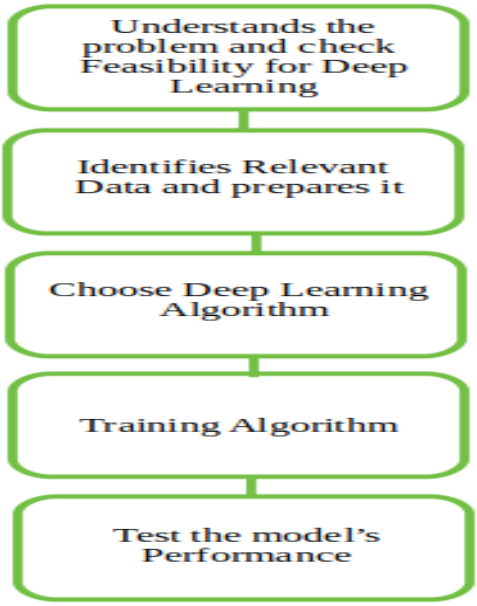


Fig 3.4- Flow chart of Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. Any Deep neural network will consist of three types of layers:

- Input Layer
- Hidden Layer
- Output Layer

1.The Input layer:

It receives all the inputs and the last layer is the output layer which provides the desired output.

2. Hidden Layers:

All the layers in between these layers are called hidden layers. There can be n number of hidden layers. The hidden layers and perceptions in each layer will depend on the use-case you are trying to solve.

3. Output Layers:

It provides the desired output.

3.4 Machine Learning vs Deep Learning:

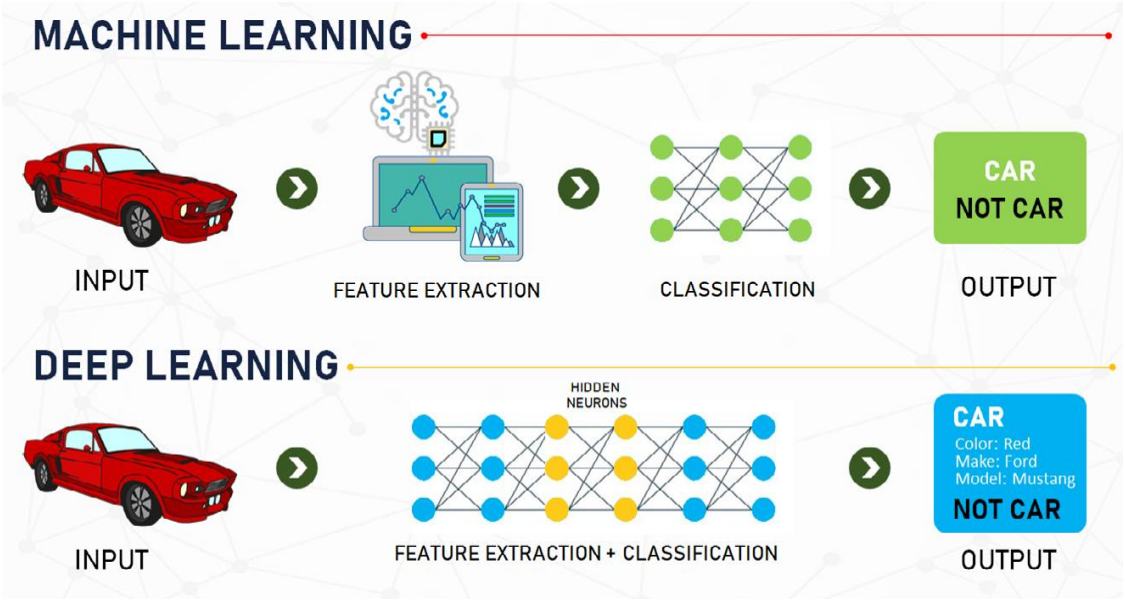


Fig 3.5- Machine Learning vs Deep Learning

Machine Learning:

Machine learning is a subset, an application of Artificial Intelligence (AI) that offers the ability to the system to learn and improve from experience without being programmed to that level. Machine Learning uses data to train and find accurate results. Machine learning focuses on the development of a computer program that accesses the data and uses it to learn from themselves.

Deep Learning:

Deep Learning is a subset of Machine Learning where the artificial neural network, the recurrent neural network comes in relation. The algorithms are created exactly just like machine learning but it consists of many more levels of algorithms. All these networks of the algorithm are together called as the artificial neural network. In much simpler terms, it replicates just like the human brain as all the neural networks are connected in the brain, exactly is the concept of deep learning. It solves all the complex problems with the help of algorithms and its process.

Machine Learning	Deep Learning
Works on small amount of Dataset for accuracy.	Works on Large amount of Dataset.
Dependent on Low-end Machine.	Heavily dependent on High-end Machine.
Divides the tasks into sub-tasks, solves them individually and finally combine the results.	Solves problem end to end.
Takes less time to train.	Takes longer time to train.
Testing time may increase.	Less time to test the data.

Table-3.1 Differences between machine learning and deep learning

3.5 How Deep Learning works in face recognition

Deep learning is one of the most novel ways to improve face recognition technology. The idea is to extract face embeddings from images with faces. Such facial embeddings will be unique for different faces. And training of a deep neural network is the most optimal way to perform this task.
Depending on a task and timeframes, there are two common methods to use deep learning for face recognition systems:

Use pre-trained models such as **dlib**, **DeepFace**, **FaceNet**, and others. This method takes less time and effort because pre-trained models already have a set of algorithms for face recognition purposes. We also can fine-tune pre-trained models to avoid bias and let the face recognition system work properly.

Develop a neural network from scratch. This method is suitable for complex face recognition systems having multi-purpose functionality. It takes more time and effort, and requires millions of images in the training dataset, unlike a pre-trained model which requires only thousands of images in case of transfer learning.

Encoding the faces using OpenCV and deep learning

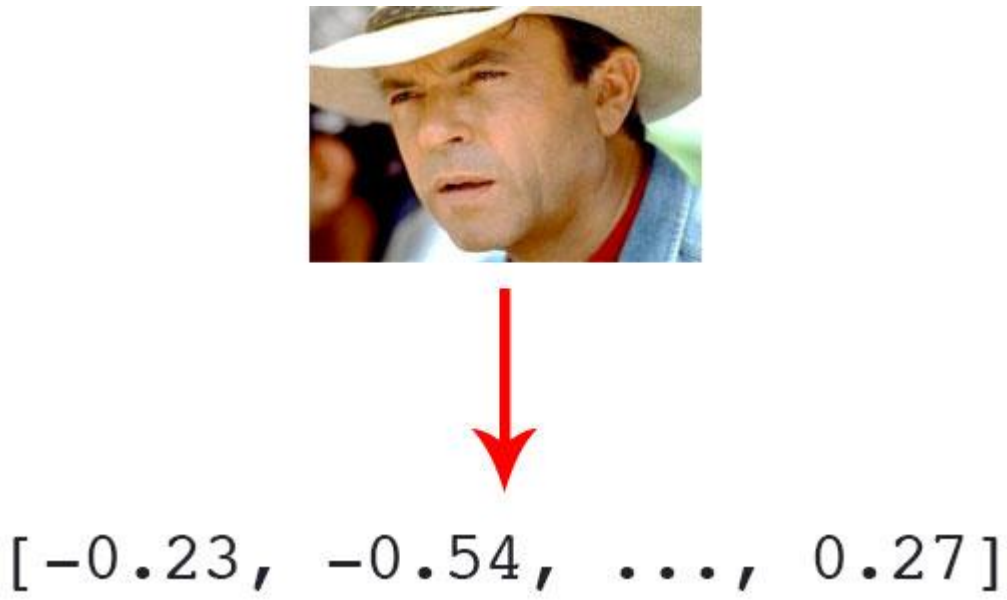


Fig 3.6- Facial Encodings

Facial recognition via deep learning and Python using the face_recognition module method generates a 128-d real-valued number feature vector per face.

Before we can recognize faces in images and videos, we first need to quantify the faces in our training set. Keep in mind that we are not actually training a network here — *the network has* **already been trained** *to create 128-d embeddings* on a dataset of ~3 million images.

We certainly **could** train a network from scratch or even fine-tune the weights of an existing model but that is more than likely overkill for many projects. Furthermore, you would need a **lot** of images to train the network from scratch.

Instead, it's easier to use the pre-trained network and then use it to construct 128-d embeddings for each of the 218 faces in our dataset.

Then, during classification, we can use a simple k-NN model + votes to make the final face classification. Other traditional machine learning models can be used here as well.

3.6 ADVANTAGES

No Need for Feature Engineering

Feature engineering is the process of extracting features from raw data to better describe the underlying problem. It is a fundamental job in machine learning as it improves model accuracy. The process can sometimes require domain knowledge about a given problem.

To better understand feature engineering, consider the following example. In the real estate business, the location of a house has a significant impact on the selling price. Suppose the location is given as the latitude and the longitude. Alone these two numbers are not of any use but put together they represent a location. The act of combining the latitude and the longitude to make one feature is feature engineering. One of deep learning's main advantages over other machine learning algorithms is its capacity to execute feature engineering on it own. A deep learning algorithm will scan the data to search for features that correlate and combine them to enable faster learning without being explicitly told to do so. This ability means that data scientists can sometimes save months of work. Besides, the neural networks that a deep learning algorithm is made of can uncover new, more complex features that human can miss.

Best Results with Unstructured Data

According to research from Gartner, up to 80% of a company's data is unstructured because most of it exists in different formats such as texts, pictures, pdf files and more. Unstructured data is hard to analyze for most machine learning algorithms, which means it's also going unutilized. That is where deep learning can help. Deep learning algorithms can be trained using different data formats, and still derive insights that are relevant to the purpose of its training. For example, a deep learning algorithm can uncover any existing relations between pictures, social media chatter, industry analysis, weather forecast and more to predict future stock prices of a given company.

No Need for Labeling of Data

Getting good-quality training data is one of the biggest problems in machine learning because data labelling can be a tedious and expensive job. Sometimes, the data labelling process is simple but time-consuming. For example, labelling photos "dog" or "muffin" is an easy task, but an algorithm needs thousands of pictures to tell the difference. Other times, data labelling may require the judgments of highly skilled industry experts, and that is why, for some industries, getting high-quality training data can be very expensive.

Let's look at the example of Microsoft's project Inner Eye, a tool that **uses computer vision** to analyse radiological images. To make correct, autonomous decisions, the algorithm requires thousands of well-annotated images where different physical anomalies of the human body are clearly labelled. Such work needs to be done by a radiologist with experience and a trained eye. According to Glassdoor, an average base salary for a radiologist is $290.000 a year, which puts the hourly rate just short of $200. Given that around 4-5 **images can be analysed** per hours, proper labelling of all images will be expensive. With deep learning, the need for well-labelled data is made obsolete as deep learning algorithms excel at learning without guidelines. Other forms of machine learning are not nearly as successful with this type of learning. In the example above, a deep learning algorithm would be able to detect physical anomalies of the human body, even at earlier stages than human doctors.

Efficient at Delivering High-quality Results

Humans need rest and fuel. They get tired or hungry and make careless mistakes. That is not the case for neural networks. Once trained correctly, a deep learning brain can perform thousands of repetitive, routine tasks within a shorter period of time than it would take a human being. The quality of its work never diminishes, unless the training data includes raw data that does not represent the problem you are trying to solve.

3.7 APPLICATIONS

- Self Driving Cars
- News Aggregation and Fraud News Detection
- Natural Language Processing
- Virtual Assistants
- Entertainment
- Visual Recognition
- Fraud Detection
- Healthcare
- Personalisations
- Detecting Developmental Delay in Children
- Colourisation of Black and White images
- Adding sounds to silent movies
- Automatic Machine Translation

- Automatic Handwriting Generation
- Automatic Game Playing
- Language Translations
- Pixel Restoration
- Photo Descriptions
- Demographic and Election Predictions
- Deep Dreaming

CHAPTER 4: Face Recognition

4.1 Introduction

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection itself can have very useful applications. The most successful applications of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

Head pose estimation is another application that heavily relies on face detection. Estimating the head pose is useful in the settings of automated guided cars, where an in-car device runs the head pose estimation algorithm to detect the drowsiness of the driver. It is a real-time head pose estimation system that can identify five poses on a mobile platform.

Another good usage of face detection/tracking is the support of mobile device interaction. In Ref. [16], the authors propose an interaction method with the mobile device based on tracked face position. With mixed interaction spaces (MIXIS), their system allows the mobile device to track the location of the device in relation to a tracked feature in the real world. The main idea of MIXIS is to use the space and information around the mobile device as input instead of limiting interaction with mobile devices to the device itself. To this end, they track the location of the user's face to estimate the location of the device relative to the tracked face, which forms a 4D input vector, i.e., *x*, *y*, *z*, and tilt. The difference with other face-tracking applications is that in their system, it is not the face that moves but the mobile device, which minimizes exhaustion related to frequent head movements. With the face location tracked, the system can support applications such as a Pong game or map panning and zooming. It shows the concept of MIXIS, where the user is able to interact with the mobile device by moving it in different directions. This is achieved through the face-tracking module on the device.

Face detection has attracted significant interest in the literature. In general, it constitutes a difficult problem, especially in cases where the background, head pose, and lighting are varying. Some reported systems use traditional image processing techniques for face detection, such as colour segmentation, edge detection, image thresholding, template matching, or motion information in image sequences, taking advantage of the fact that many local facial sub-features contain strong edges and are approximately rigid.

However, the most widely used techniques follow a statistical modelling approach of face appearance to obtain a binary classification of image regions into the face and non-face classes. Such regions are typically represented as vectors of grey-scale or colour image pixel intensities over normalized rectangles of a predetermined size, often projected onto lower-dimensional spaces, and are defined over a "pyramid" of possible locations, scales, and orientations in the image. These regions can be classified using one or more techniques, such as neural networks, clustering algorithms along with distance metrics from the face or nonface spaces, simple linear discriminants, support vector machines, and Gaussian mixture models, for example. An alternative popular approach uses a cascade of weak classifiers instead, that are trained using the AdaBoost technique and operate on local appearance features within these regions. Notice that if colour information is available, certain image regions that do not contain sufficient number of skin-tone-like pixels can be eliminated from the search.

Once face detection is successful, similar techniques can be used in a hierarchic manner to detect a number of interesting facial features such as the mouth corners, eyes, nostrils, chin, etc. The prior knowledge of their relative position on the face can simplify the search task. Such features are needed to determine the mouth region of interest (ROI) and help to normalize it by providing head-pose information. Additional lighting normalization is often applied to the ROI before appearance-based feature extraction.

Once the ROI is located, a number of algorithms can be used to obtain lip contour estimates. Some popular methods for this task are snakes, templates, and active shape and appearance models. A snake is an elastic curve represented by a set of control points, and it is used to detect important visual features, such as lines, edges, or contours. The snake control point coordinates are iteratively updated, converging toward a minimum of the energy function, defined on the basis of curve smoothness constraints and a matching criterion to desired features. Templates are parametric curves that are fitted to the desired shape by minimizing an energy function, defined similarly to snakes. In contrast, active shape models (ASMs) are statistical models obtained by performing principal component analysis (PCA) on vectors containing the coordinates of a training set of points that lie on the shapes of interest, such as the lip inner and outer contours. Such vectors are projected onto a lower dimensional space defined by the eigenvectors that correspond to the largest PCA eigenvalues, representing the axes of genuine shape variation. Active appearance models (AAMs) are an extension to ASMs that, in addition to the shape-based model, use two more PCAs:

1. The first captures the appearance variation of the region around the desired shape.

2. The final PCA is built on concatenated weighted vectors of the shape and appearance representations.

AAMs thus remove the redundancy due to shape and appearance correlation, and they create a single model that compactly describes shape and the corresponding appearance deformation. ASMs and AAMs can be used for tracking lips or other shapes by means of the algorithm proposed. The technique assumes that, given small perturbations from the actual fit of the model to a target image, a linear relationship exists between the difference in the model projection and image and the required updates to the model parameters. Fitting the models to the image data can be done iteratively or by the downhill simplex method.

4.2 Process of face recognition

Face recognition is often described as a process that first involves four steps; they are: **face detection**, face alignment, feature extraction, and finally face recognition.

1. **Face Detection**. Locate one or more faces in the image and mark with a bounding box.
2. **Face Alignment**. Normalize the face to be consistent with the database, such as geometry and Photometrics.
3. **Feature Extraction**. Extract features from the face that can be used for the recognition task.
4. **Face Recognition**. Perform matching of the face against one or more known faces in a prepared database.

A given system may have a separate module or program for each step, which was traditionally the case, or may combine some or all of the steps into a single process.

The facial recognition process can be split into two major stages: processing which occurs before detection involving face detection and alignment and later recognition is done using feature extraction and matching steps.

1. **FACE DETECTION:**

The primary function of this step is to conclude whether the human faces emerge in a given image, and what is the location of these faces. The expected outputs of this step are patches which contain each face in the input image. In order to get a more robust and easily designable face recognition system. Face alignment is performed to rationalise the scales and orientation of these patches.

2. **FEATURE EXTRACTION:**

Following the face detection step the extraction of human face patches from images is done. After this step, the conversion of face patch is done into vector with fixed coordinates or a set of landmark points.

3. **FACE RECOGNITION:**

The last step after the representation of faces is to identify them. For automatic recognition we need to build a face database. Various images are taken foe each person and their features are extracted and stored in the database. Then when an input image is fed the face detection and feature extraction is performed and its feature to each face class is compared and stored in the database.

How Does Facial Recognition Work?

The computer algorithm of facial recognition software is a bit like human visual recognition. But if people store visual data in a brain and automatically recall visual data once needed, computers should request data from a database and match them to identify a human face.

In a nutshell, a computerized system equipped by a camera, detects and identifies a human face, extracts facial features like the distance between eyes, a length of a nose, a shape of a forehead and cheekbones. Then, the system recognizes the face and matches it to images stored in a database.

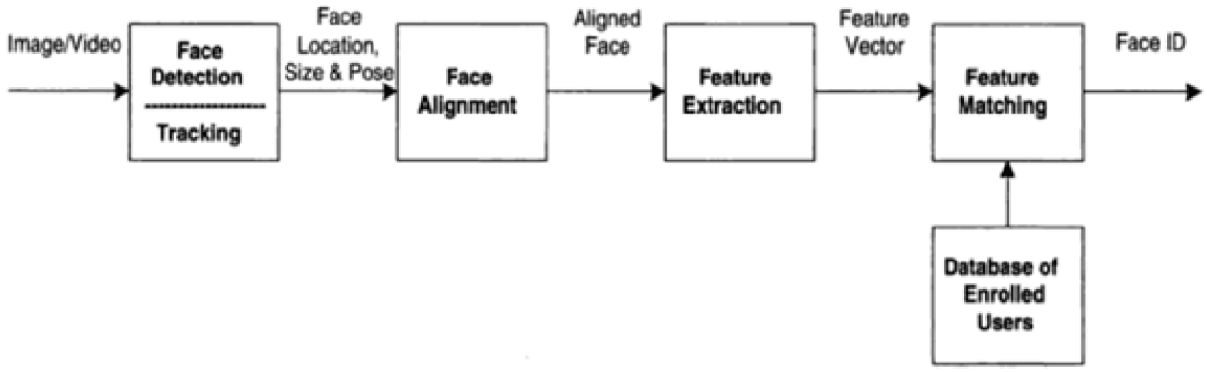


Fig 4.1- Face Recognition Process Flow

Face Recognition Tasks

The task of face recognition is broad and can be tailored to the specific needs of a prediction problem.

Three face recognition tasks:

- **Face Matching**: Find the best match for a given face.
- **Face Similarity**: Find faces that are most similar to a given face.
- **Face Transformation**: Generate new faces that are similar to a given face.

Matching requires that the candidate matching face image be in some set of face images selected by the system. Similarity detection requires in addition to matching that images of faces be found which are similar to a recalled face this requires that the similarity measure used by the recognition system closely match the similarity measures used by humans Transformation applications require that new images created by the system be similar to human recollections of a face.

Two main modes for face recognition, are as:

1.Face Verification.

A one-to-one mapping of a given face against a known identity (e.g. *is this the person?*).

2.Face Identification.

A one-to-many mapping for a given face against a database of known faces (e.g. *who is this person?*).

There is a difference between Face verification and Face recognition some people often take them as same. It is important to know this difference.

4.3 Face Recognition Algorithms

The face recognition algorithm is used to find the characteristics which accurately represents a picture with the features which are already extracted, scaled, and converted into grey scale. There are various algorithms used for facial recognition. Some of them are as follows:

4.3.1.Eigen faces:

Face recognition systems are built on the idea that each person has a particular face structure, and using the facial symmetry, computerized face-matching is possible. The work on face recognition has begun in the 1960's, the results of which are being used for security in various institutions and firms throughout the world. The images must be processed correctly for computer-based face recognition. The face and its structural properties should be identified carefully, and the resulting image must be converted to two-dimensional digital data. An efficient algorithm and a database which consists of face images are needed to solve the face recognition problem. In the

recognition process, an eigenface is formed for the given face image, and the Euclidian distances between this eigenface and the previously stored eigenfaces are calculated. The eigenface with the smallest Euclidian distance is the one the person resembles the most. Simulation results are shown. Simulations have been done using the Matlab program. The success rate for the large database used is found to be 94.74 percent. The face recognition system is similar to other biometric systems. The idea behind the face recognition system is the fact that each individual has a unique face. Similar to the fingerprint, the face of an individual has many structures and features unique to that individual. An automatic face recognition system is based on facial symmetry. Face authentication and face identification are challenging problems. The fact that in the recent past, there have been more and more commercial, military and institutional applications, makes the face recognition systems a popular subject. To be reliable, such systems have to work with high precision and accuracy. In a face recognition system, the database consists of the images of the individuals that the system has to recognize. If possible, several images of the same individual should be included in the database. If the images are selected so that they account for varying facial expressions, lighting conditions, etc., the solution of the problem can be found more easily as compared to the case where only a single image of each individual is stored in the database. A face recognition algorithm processes the captured image and compares it to the images stored in the database. If a match is found, then the individual is identified. If no match is found, then the individual is reported as unidentified. The challenges of face recognition are:

1.Shifting and scaling of the image

2.Differences in the facial look (different angle, pose, hairstyle, makeup, mustache, beard, etc.), x Lighting,

Aging.

The algorithm has to work successfully even with the above challenges.

The basis of the eigenfaces method is the Principal Component Analysis (PCA). Eigenfaces and PCA have been used by Sirovich and Kirby to represent the face images efficiently [11]. They have started with a group of original face images, and calculated the best vector system for image compression. Then Turk and Pentland applied the Eigenfaces to face recognition problem [12]. The Principal Component Analysis is a method of projection to a subspace and is widely used in pattern recognition. An objective of PCA is the replacement of correlated vectors of large dimensions with the uncorrelated vectors of smaller dimensions. Another objective is to calculate a basis for the data set. Main advantages of the PCA are its low sensitivity to noise, the reduction of the requirements of the memory and the capacity, and the increase in the efficiency due to the operation in a space of smaller dimensions. The strategy of the Eigenfaces method consists of extracting the characteristic features on the face

and representing the face in question as a linear combination of the so called ‘eigenfaces’ obtained from the feature extraction process. The principal components of the faces in the training set are calculated. Recognition is achieved using the projection of the face into the space formed by the eigenfaces. A comparison on the basis of the Euclidian distance of the eigenvectors of the eigenfaces and the eigenface of the image under question is made. If this distance is small enough, the person is identified. On the other hand, if the distance is too large, the image is regarded as one that belongs to an individual for which the system has to be trained.

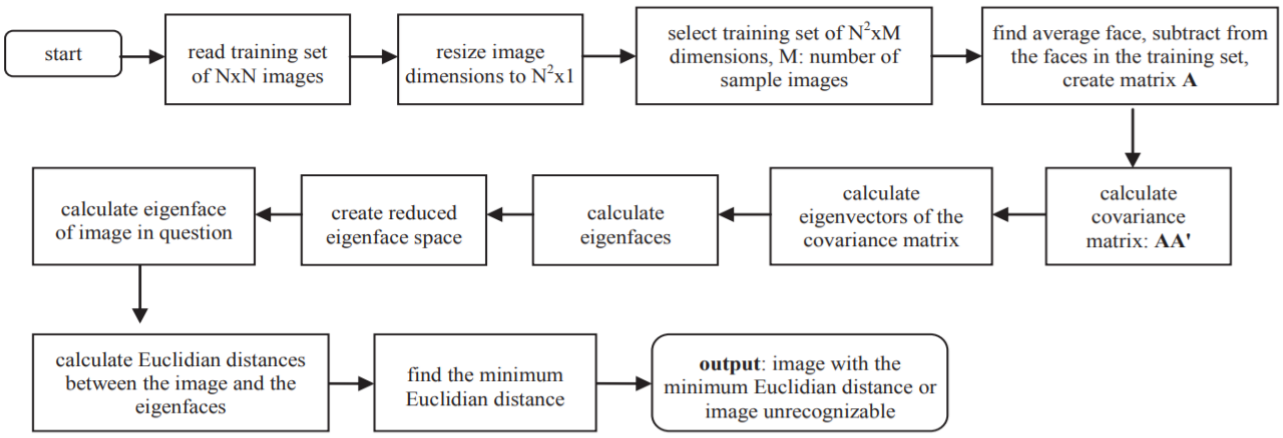


Fig 4.2- Flow Chart for Eigen Faces

The reasons for selecting the eigenfaces method for face recognition are:

- Its independence from the facial geometry,
- The simplicity of realization,
- Possibility of real-time realization even without special hardware,
- The ease and speed of recognition with respect to the other methods,
- The higher success rate in comparison to other methods.

The challenge of the eigenfaces face recognition method is the computation time. If the database is large, it may take a while to retrieve the identity of the person under question.

Fig

Fig 4.3- Sample Data Base

Fig 4.4- Sample Data Base of Our Project

Fig 4.5- Facial Embeddings

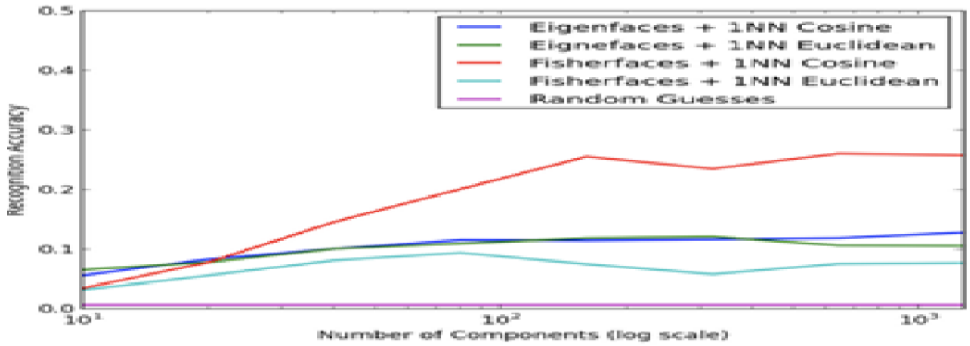


Fig 4.6- Recognition Accuracy vs Number of Components

4.3.2.Fisher faces:

Fisher face is one of the popular algorithms used in face recognition, and is widely believed to be superior to other techniques, such as eigenface because of the effort to maximize the separation between classes in the training process. The purpose of this research is to establish a program of face recognition application using fisher face method by utilizing GUI applications and databases that are used in the form of a Papuan facial image. Image recognition using fisher face method is based on the reduction of face space dimension using Principal Component Analysis (PCA) method, then apply Fisher's Linear Discriminant (FDL) method or also known as Linear Discriminant Analysis (LDA) method to obtain feature of image characteristic. The algorithm used in the process for image recognition is fisher faces algorithm while for identification or matching face image using minimum Euclidean distance.

The face, as a part of human body, is the easiest and the most often used to distinguish the identity of individuals. From the face, humans can be distinguished and recognized more quickly and easily [1]. Therefore the face is used as a means of identification of a person or face recognition [3] Generally, the image recognition system is divided into 2 types, namely: feature-based system and image-based system. In the first system, features extracted from the eye image components, nose, mouth, etc. which are then modeled geometrically to determine the relationship between these features. While in the second system using image pixels which are then represented in certain methods such as Principal Component Analysis, wavelet transformation, etc. which is then used for training and image identification classification [2, 5]. Feature extraction is a process for obtaining characteristics that distinguish a face sample from another face sample. Reliable feature extraction techniques are key in solving pattern recognition problems as the principal component analysis method (PCA) used for face recognition was introduced by Turk and Pentland in 1991 [7]. The PCA method aims to project data in the direction that has the greatest variation (indicated by the eigenvector) corresponding to the largest eigenvalues of the covariance matrix. The weakness of the method is less optimal in the separation between classes [3,7]. In 1991, Cheng et al.

introduces Linear Discriminant Analysis (LDA) method for face recognition. This method tries to find a linear subspace that maximizes the separation of two pattern classes according to Fisher Criterion JF. This can be obtained by minimizing the distance of the within class distribution matrix and maximizing the split matrix spacing between the Sb classes simultaneously resulting in a maximum Fisher Criterion JF. Fisher Linear Discriminant will find subspaces where classes are linearly separated by maximizing the Fisher Criterion JF. If the data dimension is much higher than the number of training samples will cause Sw to be singular. This is a weakness of the LDA method [3]. In 1997, Belheumeur introduced the fisher face method for face recognition. This method is a combination of PCA and LDA methods. The PCA method is used to solve singular problems by reducing the dimensions before being used to perform the LDA process. But the weakness of this method is that when the PCA dimension reduction process will cause some loss of discriminant information useful in the LDA process [3]. But in its development, face recognition with fisher face method still have some problems, such as computation problems and the condition of the face image into input that will be used as image testing. The problem of computation in face recognition using fisher face method becomes a problem because it has a very complicated and very complex computation process. While the problems that affect the condition of the face image is the diversity of the light of the face image, the attributes of the face image, the expression of the face image, and the variation of the position of the image of the face itself. The data used in this study is the image of the Papuan's face. Face images of people of Papua used as data in this study because in addition there has been no research data in the form of face images of the Papuans. Another reason is because the images of the face of the Papuans are generally very similar so there is a possibility of influence on the success of the method used in this study [8].

The Design System Face recognition system using fisher face method is designed to recognize the face image by matching the results of its feature extraction. The system is expected to determine whether the image to be tested is recognized correctly or not. In this research, as many as 200 facial images, taken from 50 students, are used in *.bmp format. Each student have five face images with different expressions. 2.2. Process Design 2.2.1. Data retrieval process. This process aims to collect data in the form of face image. Collection of samples is done with photograph directly the face image. The position of the face is facing toward the front and upright position and not blocked by other objects. A total of 50 students photographed with a distance of ± 100 cm, with the aim to equate the quality and the image of each data taken. 2.2.2. Image Processing Process. The design of this process is divided into two stages: preprocessing stage and processing stage which includes feature extraction and recognition.

• Image Pre-processing

The face image to be used must go through the preprocessing stage first. This stage includes image acquisition, and RBG image conversion to grayscale. Acquisition of face images using camera. The image of this acquisition is a 24-bit RGB image of JPG format with size 92 x 112 pixels. Conversion of face image of acquisition from RGB to 8-bit grayscale, BMP format with size 40 x 40 pixels. Furthermore, the face data is divided into 2 (two) parts i.e; one part of the image will be used as training image (training dataset) and one part of the image will be used as test image (testing dataset).

• Image Processing

At this image processing stage, Fisher face method will be applied to generate feature vector of facial image data used by system and then to match vector of traits of training image with vector characteristic of test image using Euclidean distance formula. 2.2.3. Feature generation process. Features to be extracted is a feature of the face image of people of Papua. The method used is fisher face method is a method that is a merger between PCA and LDA methods.

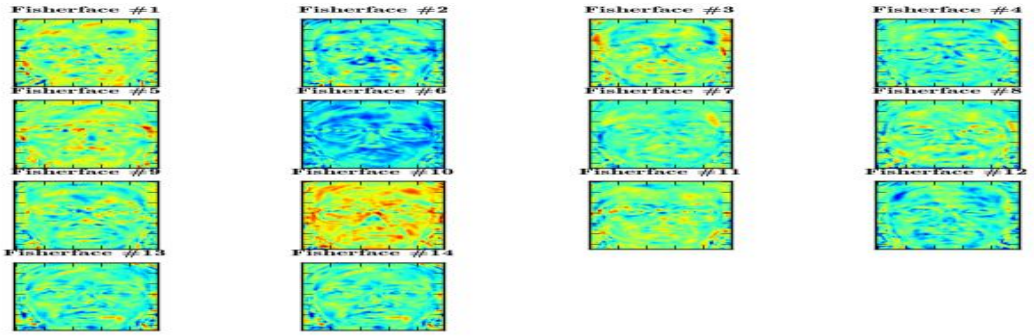


Fig 4.7- Fisher Faces

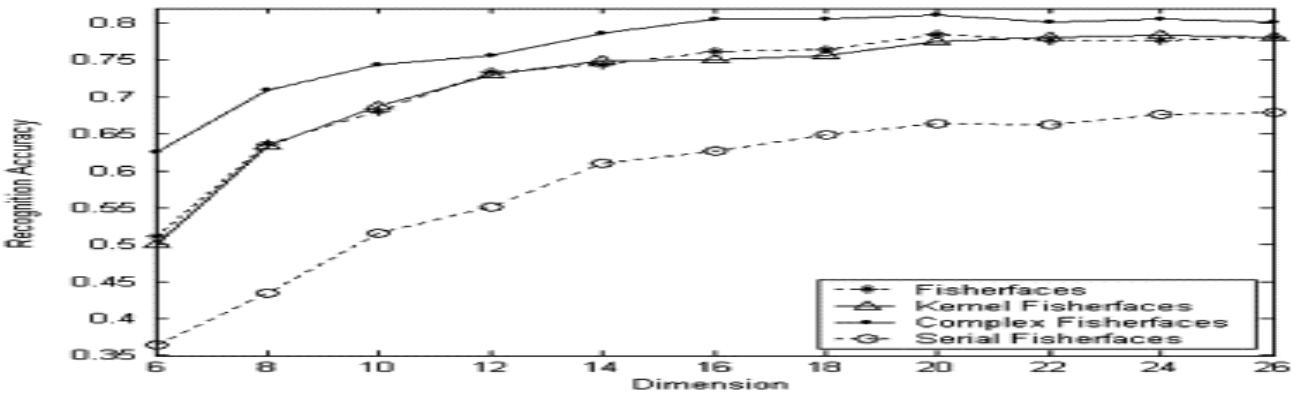


Fig 4.8- Recognition Accuracy vs Dimension

4.3.3.LBPH:

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector. As LBP is a visual descriptor it can also be used for face recognition tasks, as can be seen in the following step-by-step explanation.

Now that we know a little more about face recognition and the LBPH, let's go further and see the steps of the algorithm:

1.Parameters: the LBPH uses 4 parameters:

- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

The image below shows this procedure:

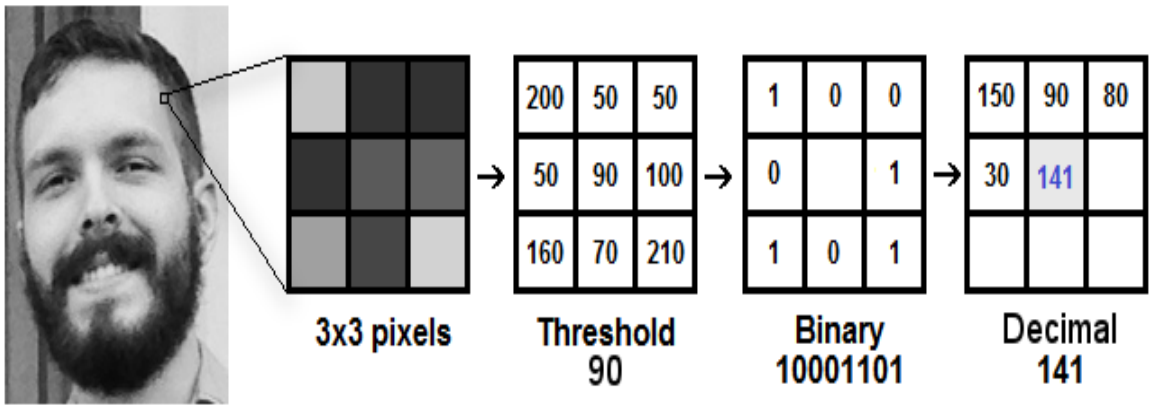


Fig 4.9- Conversion of Low Resolution Pixels into High Resolution Pixels

Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

- **Note**: The LBP procedure was expanded to use a different number of radius and neighbours, it is called Circular LBP.

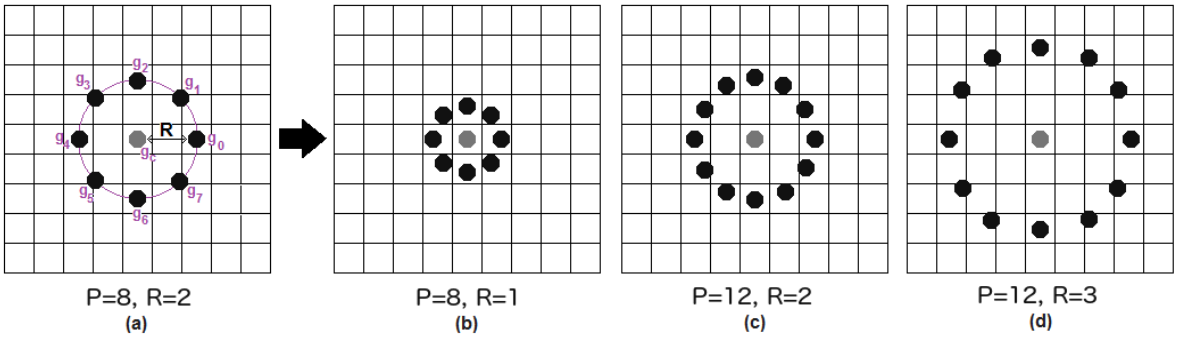


Fig 4.10- Circular LBPH

It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. **Extracting the Histograms**: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:

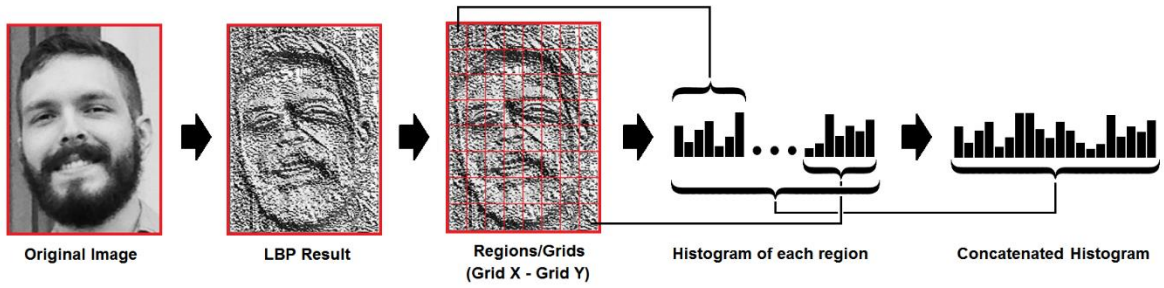


Fig 4.11- Histogram Equalization

Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.

- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

5. Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$S(g , t) = \sqrt{(\sum((g_i - t_i)^2)}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a '**confidence**' measurement.
- **Note**: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

Advantages of LBPH:

- LBPH is one of the easiest face recognition algorithms.
- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic Gray scale transformations.
- It is provided by the **OpenCV** library (Open Source Computer Vision Library).

$LBP_{P,R}$	Percentage of uniform pattern in a Image
8, 1	90%
8, 1	90.6%
8, 2	85.2%
16, 2	70%

Table 4.1. Different LBP radius and neighbor values uniform pattern in the image.

4.4 Comparison of various face recognition algorithms

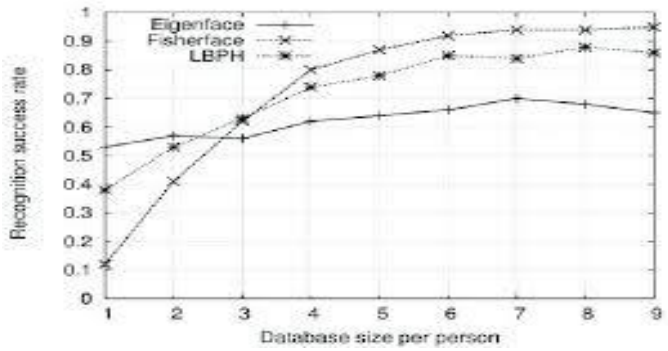


Fig 4.12- Performance of Face Recognition Algorithms

Algorithm	Accuracy	Precision	Recall	F1 Score	Execution Time
Eigen face	99%±0.87%	0.985±0.01	0.99 ± 0.009	0.987 ± 0.010	2.74s
Fisher face	100%	1	1	1	1.37s
LBPH	98%±1.24%	0.97±0.015	0.98 ± 0.012	0.97 ± 0.015	1.84s

Table 4.2. Accuracy, precision, recall, F1 score, and execution time on AT&T dataset.

Algorithm	Accuracy	Precision	Recall	F1 Score	Execution Time
Eigen face	33%±7.18%	0.318±0.072	0.33±0.072	0.305±0.073	2.73s
Fisher face	37%±6.96%	0.398±0.07	0.37±0.07	0.352±0.071	1.37s

LBPH	40%±6.8%	0.36±0.07	0.4±0.068	0.358±0.07	1.95s

Table 4.3. Accuracy, precision, recall, F1 score, and execution time on 5_Celebrity dataset.

Algorithm	Accuracy	Precision	Recall	F1 Score	Execution Time
Eigen face	86%±3.3%	0.803±0.04	0.86±0.033	0.82±0.037	3.0167s
Fisher face	84%±3.51%	0.781±0.041	0.84±0.035	0.801±0.04	1.44s
LBPH	95%±1.96%	0.925±0.024	0.95±0.02	0.934±0.023	2.09s

Table 4.4. Accuracy, precision, recall, F1 score, and execution time on LUDB.

CHAPTER 5: Convolutional Neural Networks

5.1 Introduction

Convolutional neural networks are one of the most common types of neural networks used in **computer vision** to recognize objects and patterns in images. One of their defining traits is the use of filters within convolutional layers. Neural networks are artificial systems that were inspired by biological neural networks. These systems learn to perform tasks by being exposed to various datasets and examples without any task-specific rules. The idea is that the system generates identifying characteristics from the data they have been passed without being programmed with a pre-programmed understanding of these datasets.

Neural networks are based on computational models for threshold logic. Threshold logic is a combination of algorithms and mathematics. Neural networks are based either on the study of the brain or on the application of neural networks to artificial intelligence. The work has led to improvements in finite automata theory.

Components of a typical neural network involve neurons, connections, weights, biases, propagation function, and a learning rule. Neurons will receive an input from predecessor neurons that have an activation, threshold, an activation function f, and an output function. Connections consist of connections, weights and biases which rules how neuron transfers output to neuron. Propagation computes the input and outputs the output and sums the predecessor neurons function with the weight. The learning rule modifies the weights and thresholds of the variables in the network.

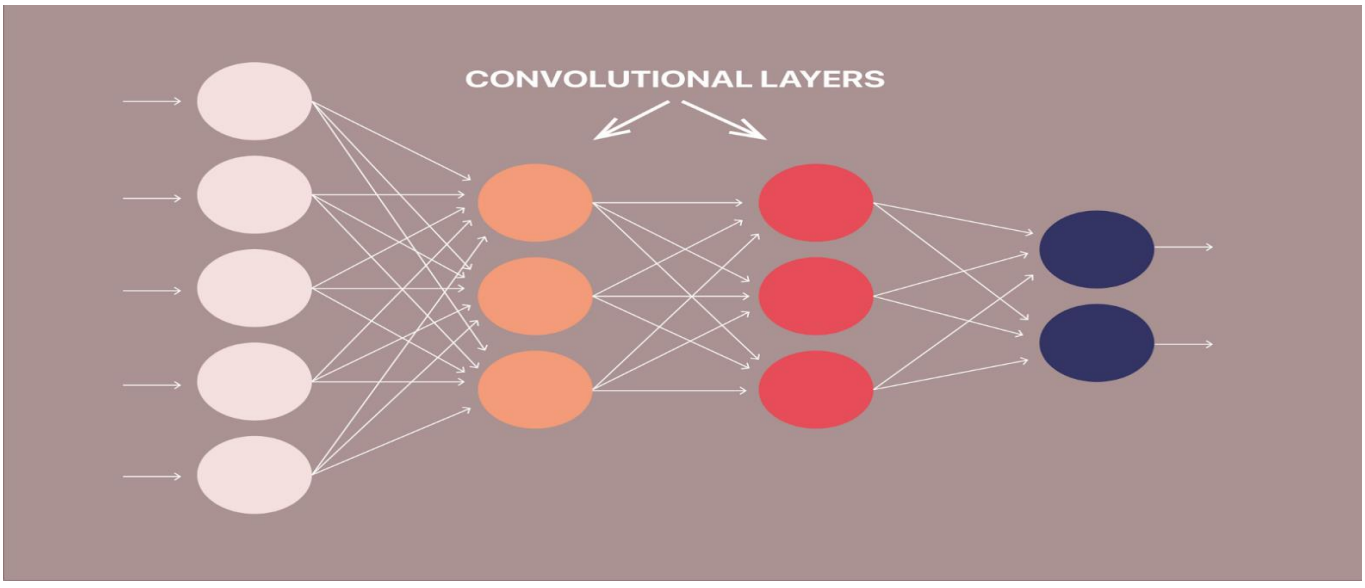


Fig 5.1- Convolutional Layers

Within a convolutional layer, the input is transformed before being passed to the next layer. A CNN transforms the data by using filters.

5.1.1 Evolution of Neural Networks

Hebbian learning deals with neural plasticity. Hebbian learning is unsupervised and deals with long term potentiation. Hebbian learning deals with pattern recognition and exclusive-or circuits; deals with if-then rule.

Back propagation solved the exclusive-or issue that Hebbian learning could not handle. This also allowed for multi-layer networks to be feasible and efficient. If an error was found, the error was solved at each layer by modifying the weights at each node. This led to the development of support vector machines, linear classifiers, and max-pooling. The vanishing gradient problem affects feedforward networks that use back propagation and recurrent neural network. This is known as deep-learning.

Hardware-based designs are used for biophysical simulation and neurotrophic computing. They have large scale component analysis and convolution creates new class of neural computing with analog. This also solved back-propagation for many-layered feedforward neural networks.

Convolutional networks are used for alternating between convolutional layers and max-pooling layers with connected layers (fully or sparsely connected) with a final classification layer. The learning is done without unsupervised pre-training. Each filter is equivalent to a weights vector that has to be trained. The shift variance has to be guaranteed to dealing with small and large neural networks. This is being resolved in Development Networks.

5.1.2 Supervised vs Unsupervised Learning

Neural networks learn via supervised learning; Supervised machine learning involves an input variable **x** and output variable **y**. The algorithm learns from a training dataset. With each correct answers, algorithms iteratively make predictions on the data. The learning stops when the algorithm reaches an acceptable level of performance. Unsupervised machine learning has input data X and no corresponding output variables. The goal is to model the underlying structure of the data for understanding more about the data. The keywords for supervised machine learning are classification and regression. For unsupervised machine learning, the keywords are clustering and association.

5.2 What are Filters in Convolutional Neural Networks?

A filter in a CNN is simply a matrix of randomized number values like in the diagram below.

Fig 5.2- Sample 3 x 3 filter

The number of rows and columns in the filter can vary and is dependent on the use case and data being processed. Within a convolutional layer, there are a number of filters that move through an image. This process is referred to as convolving. The filter convolves the pixels of the image, changing their values before passing the data on to the next layer in the CNN.

5.2.1 How do Filters Work?

To understand how filters transform data, let's take a look at how we can train a CNN to recognize handwritten digits. Below is an enlarged version of a 28x28 pixel image of the number seven from the **MNIST dataset**.

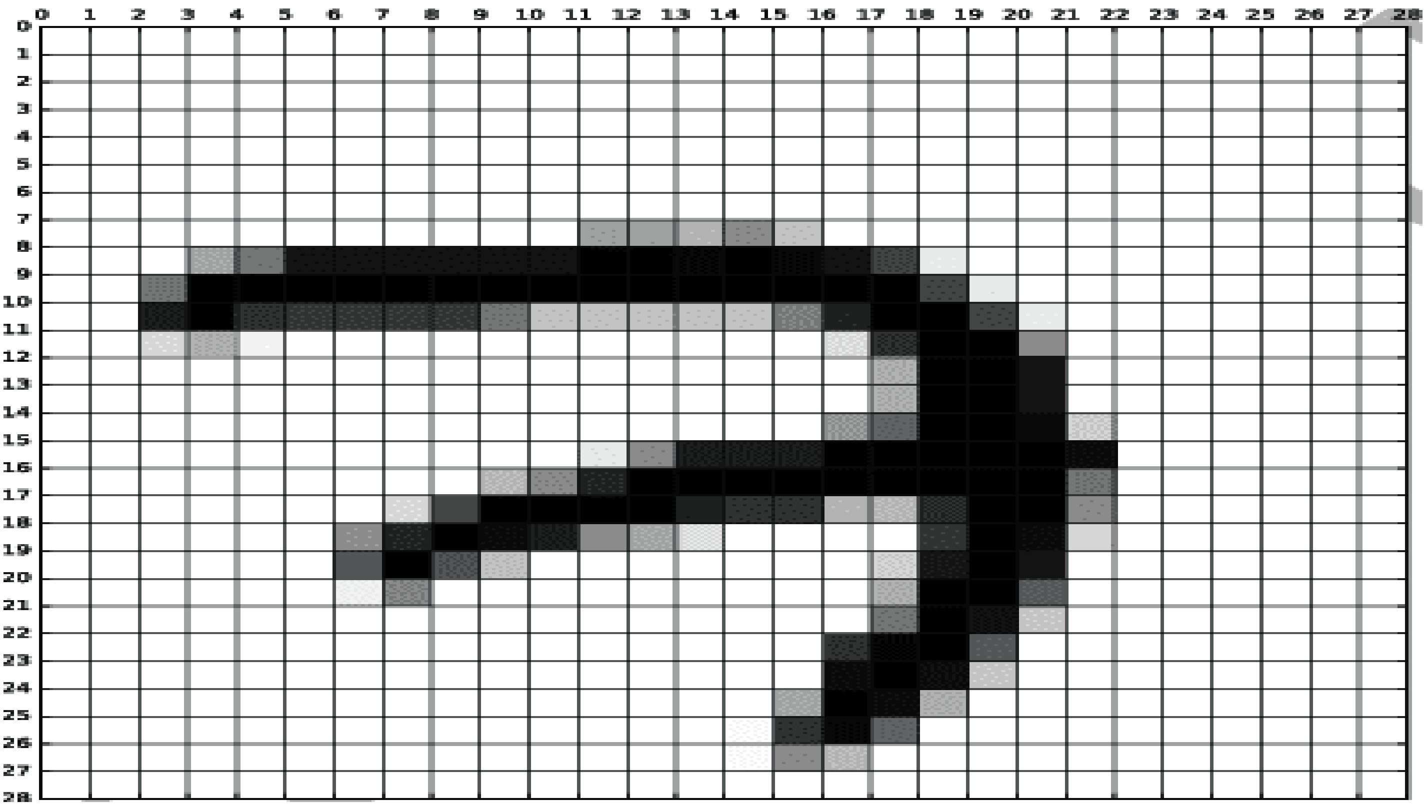


Fig 5.3- Enlarged version of a 28x28 pixel image of the number seven from the **MNIST dataset**

Below is the same image converted into its pixel values.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.4	0.3	0.5	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.4	0.5	0.9	0.9	0.9	0.9	0.9	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.9	1.0	0.8	0.8	0.8	0.8	0.8	0.5	0.2	0.2	0.2	0.2	0.2	0.5	0.9	1.0	1.0	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.3	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.8	1.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.6	1.0	1.0	1.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.9	0.9	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.5	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.7	1.0	1.0	1.0	1.0	0.9	0.8	0.8	0.3	0.3	0.8	1.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.9	1.0	0.9	0.9	0.5	0.3	0.1	0.0	0.0	0.0	0.0	0.8	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.0	0.7	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.9	1.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	1.0	0.9	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	1.0	0.9	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	1.0	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig 5.4- Assignment of pixel values

As the filter convolves through the image, the matrix of values in the filter line up with the pixel values of the image and the **dot product** of those values is taken.

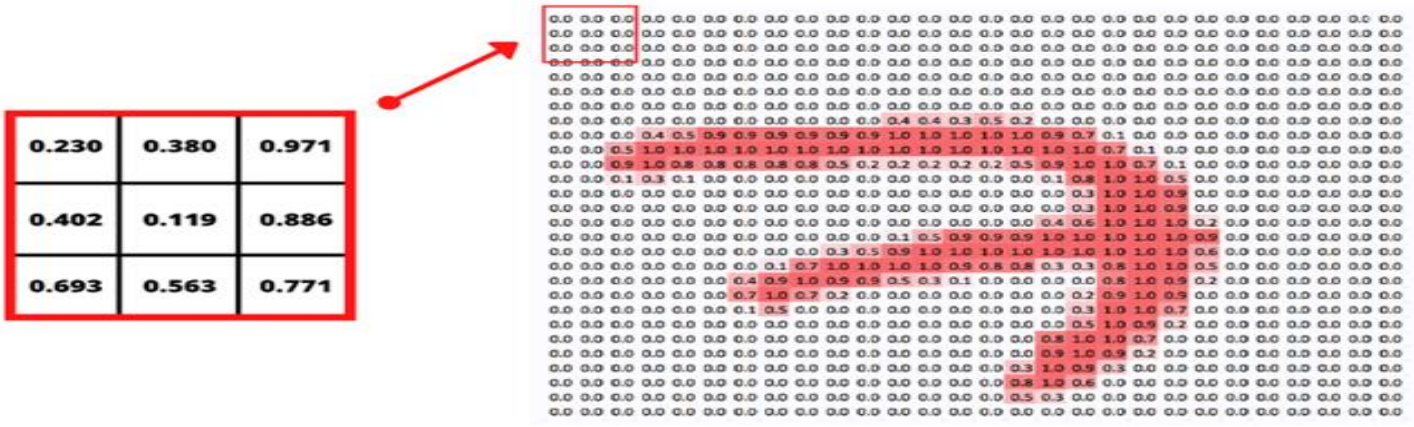


Fig 5.5- Applying the filter

The filter moves, or convolves, through each 3 x 3 matrix of pixels until all the pixels have been covered. The dot product of each calculation is then used as input for the next layer.

Initially, the values in the filter are randomized. As a result, the first passes or convolutions act as a training phase and the initial output isn't very useful. After each iteration, the CNN adjusts these values automatically using a **loss function**. As the training progresses, the CNN continuously adjusts the filters. By adjusting these filters, it is able to distinguish edges, curves, textures, and more patterns and features of the image.

While this is an amazing feat, in order to implement loss functions, a CNN needs to be given examples of correct output in the form of labeled training data.

5.3 CNN-ARCHITECTURE

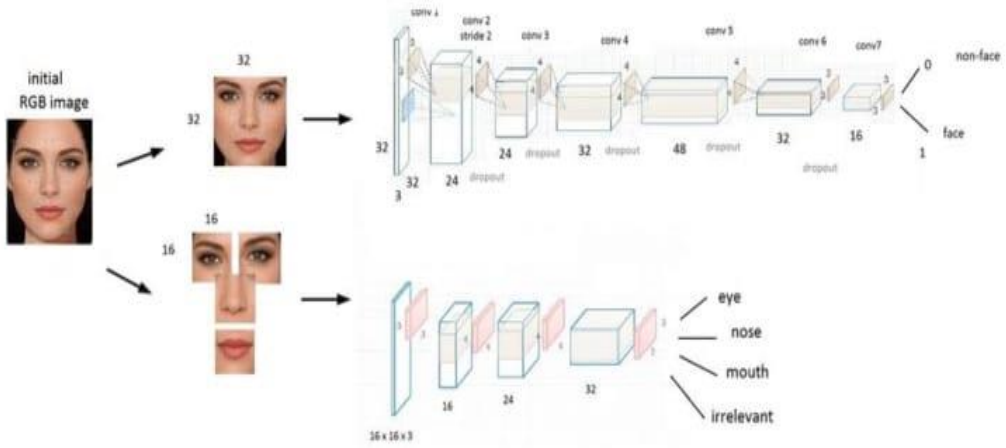


Fig 5.6- CNN Architecture

The training of deep networks provides with more accurate results, but sometimes it is not satisfied with the availability of few samples and may be in need of more training data. In this work, CNN makes the availability of possible training samples and finally they are aggregated in the training set to create an enlarged training set.

1) Face detection using CNN

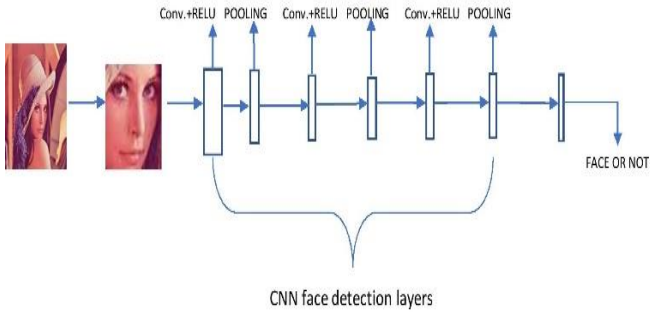


Fig 5.7- Face Detection using CNN

The main aim is to locate and extract the features from the pictures to be used facial recognition algorithm. Whenever an input image is given, it then matches with all the pictures present in the database and gets ready to by extracting the features of the image.

2) Face recognition using CNN

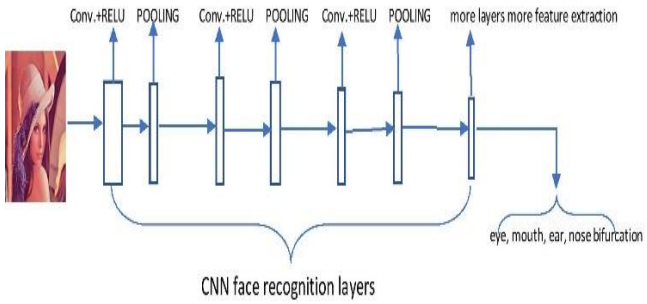


Fig 5.8- Face Recognition using CNN

The face recognition algorithm is used to find the characteristics which accurately represents a picture with the features which are already extracted, scaled, and converted into grey scale.

A convolutional neural network consists of an input layer, **hidden layers** and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final **convolution**. In a convolutional neural network, the hidden layers include layers that perform

convolutions. Typically, this includes a layer that performs a **dot product** of the convolution kernel with the layer's input matrix. This product is usually the **Frobenius inner product**, and its activation function is commonly **RELU**. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

5.3.1 CNN METHODOLOGY

CNNs are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNNs are a type of feed-forward neural networks made up of many layers. CNNs consist of filters or kernels or neurons that have learnable weights or parameters and biases. Each filter takes some inputs, performs convolution and optionally follows it with a non-linearity . A typical CNN architecture can be seen as shown in Fig.5.9. The structure of CNN contains Convolutional, pooling, Rectified Linear Unit (RELU), and Fully Connected layers classification.

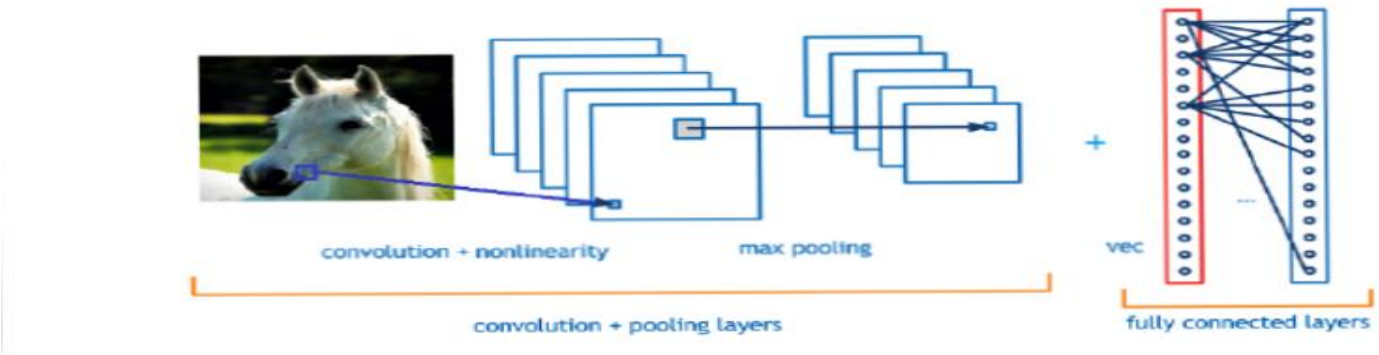


Fig 5.9- CNN Methodology

5.4 DEFINING THE CNN LAYERS

- Input layer
- Convo layer (Convo + RELU)
- Pooling layer

- Fully connected(FC) layer
- Softmax/Logistic layer
- Output layer

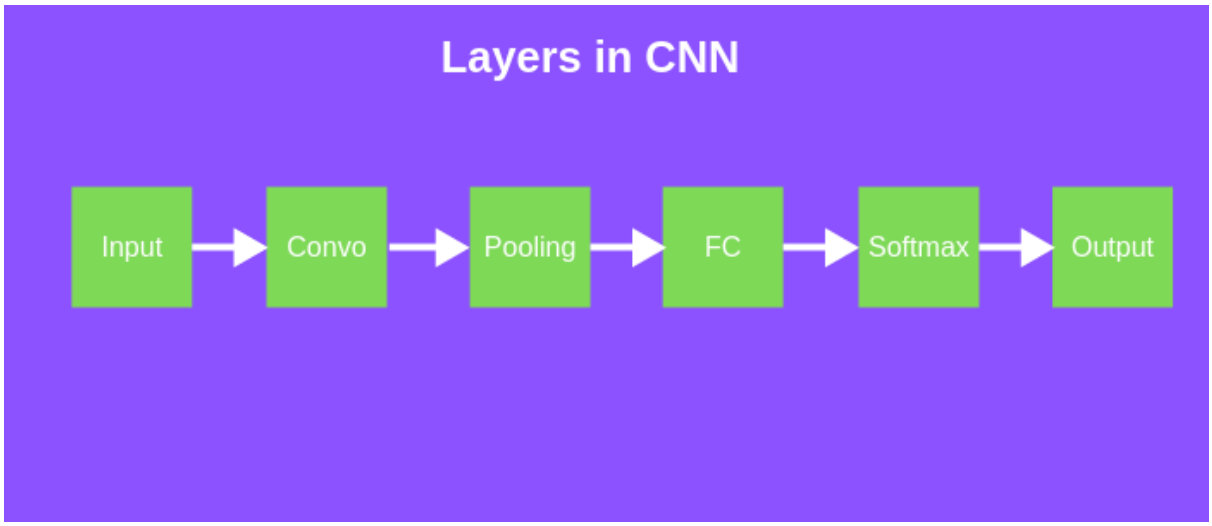


Fig 5.10- Layers in CNN

Input Layer:

The input layer in CNN should contain image data. Image data is represented by a 3-dimensional matrix, as we saw earlier. It would be best if you reshaped it into a single column. For example, suppose you have a picture of dimension 28x28 =784; you need to change it into 784 x 1 before feeding it into an input. If you have"m" training examples, then the input dimension will be (784, m).

Convolution Layer:

The Convo layer is sometimes called the feature extractor layer because the image features are extracted within this layer. First of all, a part of an image is connected to the Convo layer to perform convolution operation as we saw earlier and calculate the dot product between the filter and the receptive field (it is a global part of the input image with the same size as that of filter). The result of the operation is a single integer of the output volume. Then we slide the filter over the following receptive field of the same input image by a Stride, and we repeat the operation again and again. We will continue the same process again and again until we get through the complete picture. Then, the output will be conducted to the input for the next layer. Convo layer contains Rectified Linear Unit activation to make all negative results to zero.

Pooling Layer:

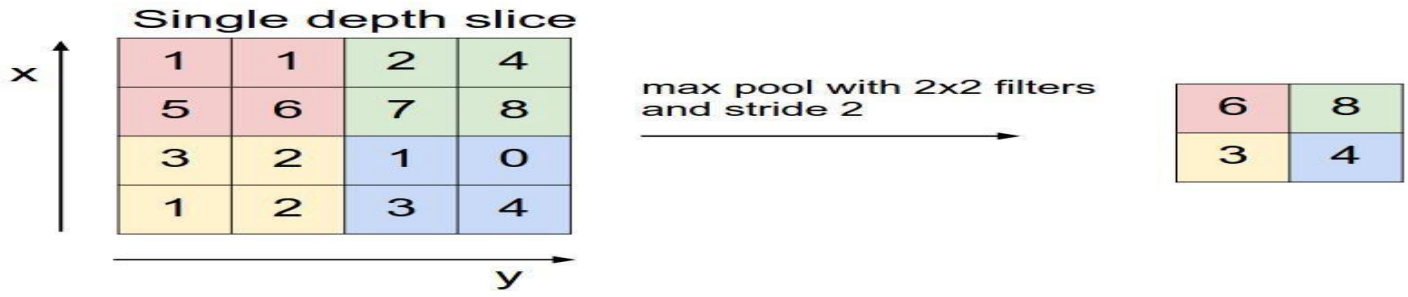


Fig 5.11- Pooling Layer

The pooling layer is used to reduce the spatial volume of the input image after convolution. It is used between two convolution layers. If we apply F.C. after the Convo layer without applying max pooling or pooling, the calculation part will be costly, and we don't want it. So, max pooling is the only way to reduce the spatial volume of an input image. In the above example, we have applied max-pooling with a Stride of 2 in a single depth slice. As a result, we can observe the 4X4 dimension input is decreased to 2X2 dimensions. There is no parameter in the pooling layer, but it consists of two primary hyper parameters — Stride(S) and Filter(F). In general, if we have input dimension Width1XHeight1XDepth1, then Width2=(Width1−Filter)/Stride+1 Height2 = (Height1−Filter)/Stride+1 Depth2 = Depth1 Where Height2, Width2, and Depth2 are the height, width, and depth of output.

RELU correction layer

RELU (Rectified Linear Units) refers to the real non-linear function defined by **RELU(x)=max(0,x)**. Visually, it looks like the following:

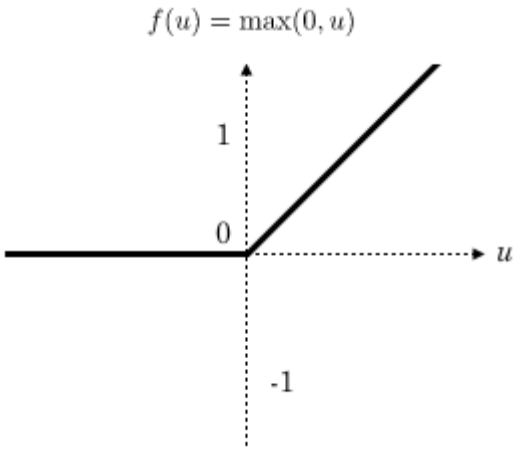


Fig 5.12- RELU Function

Fully Connected Layer:

The fully-connected layer is always the last layer of a neural network, convolutional or not — so it is not characteristic of a CNN. This type of layer receives an input vector and produces a new output vector. To do this, it applies a linear combination and then possibly an activation function to the input values received.

The last fully-connected layer classifies the image as an input to the network: it returns a vector of size N, where N is the number of classes in our image classification problem. Each element of the vector indicates the probability for the input image to belong to a class.

To calculate the probabilities, the fully-connected layer, therefore, multiplies each input element by weight, makes the sum, and then applies an activation function (logistic if N=2, softmax if N>2). This is equivalent to multiplying the input vector by the matrix containing the weights. The fact that each input value is connected with all output values explains the term fully-connected. The convolutional neural network learns weight values in the same way as it learns the convolution layer filters: during the training phase, by *backpropagation of the gradient*. The fully connected layer determines the relationship between the position of features in the image and a class. Indeed, the input table being the result of the previous layer, it corresponds to a feature map for a given feature**:** *the high values indicate the location* (more or less precise depending on the pooling) *of this feature in the image***.** If the location of a feature at a certain point in the image is characteristic of a certain class, then the corresponding value in the table is given significant weight.

Output Layer:

The output layer contains the flag or label, which is of the form one-hot encoded.

5.5 The parameterization of the layers

A convolutional neural network differs from another by the way the layers are stacked, but also parameterized. The layers of convolution and pooling have indeed hyperparameters, that is to say parameters whose you must first define the value. The size of the output feature maps of the convolution and pooling layers depends on the hyperparameters. Each image (or feature map) is W×H×D, where W is its width in pixels, H is its height in pixels and D the number of channels (1 for a black and white image, 3 for a colour image).

The convolutional layer has four hyperparameters:

1. The number of filters K

2. The size F filters: each filter is of dimensions F×F×D pixels.

3. The S step with which you drag the window corresponding to the filter on the image. For example, a step of 1 means moving the window one pixel at a time.

4. The Zero-padding P: add a black contour of P pixels thickness to the input image of the layer. Without this contour, the exit dimensions are smaller. Thus, the more convolutional layers are stacked with P=0, the smaller the input image of the network is. We lose a lot of information quickly, which makes the task of extracting features difficult.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions $W_c \times H_c \times D_c$, where:

$$W_c=(W-F+2P)/S+1$$

$$H_c=(H-F+2P)/S+1$$

$$D_c=K$$

Choosing *P=F-1/2 and S=1* gives feature maps of the same width and height as those received in the input.

The pooling layer has two hyperparameters:

1. The size F of the cells: the image is divided into square cells of size F×F pixels.

2. The S step: cells are separated from each other by S pixels.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions $W_p \times H_p \times D_p$, where:

$$W_p=(W-F)/S+1$$

$$H_p=(H-F)/S+1$$

$$D_p=D$$

Just like stacking, the choice of hyperparameters is made according to a classic scheme:

- For the convolution layer, the filters are small and dragged on the image one pixel at a time. The zero-padding value is chosen so that the width and height of the input volume are not changed at the output. In general, we then choose F=3, P=1, S=1 or F=5, P=2, S=1

- For pooling layer, F=2 and S=2 is a wise choice. This eliminates 75% of the input pixels. We can also choose F=3 and S=2: in this case, the cells overlap. Choosing larger cells causes too much loss of information and results in less good results in practice

5.6 How CNN works for Face Recognition

If the facial recognition system includes unique features, it may be an optimal way in the long run. The key points to pay attention to are:

- The correct selection of CNN architecture and loss function
- Inference time optimization
- The power of a hardware

It's recommended to use convolutional neural networks (CNN) when developing a network architecture as they have proven to be effective in image recognition and classification tasks. In order to get expected results, it's better to use a generally accepted neural network architecture as a basis, for example, ResNet or EfficientNet.

When training a neural network for face recognition software development purposes, we should minimize errors in most cases. Here it's crucial to consider loss functions used for calculation of error between real and predicted output. The most commonly used functions in facial recognition systems are **triplet loss** and **AM-Softmax**.

The triplet loss function implies having three images of two different people. There are two images – anchor and positive – for one person, and the third one – negative – for another person. Network parameters are being learned so to bring the same people closer in the feature space, and separate different people.

- **AM-Softmax function** is one of the most recent modifications of standard softmax function, which utilizes a particular regularization based on an additive margin. It allows achieving better separability of classes and therefore improves face recognition system accuracy.

There are also several approaches to improve a neural network. In facial recognition systems, the most interesting are **knowledge distillation**, transfer learning, quantization, and depth-separable convolutions.

- **Knowledge distillation** involves two different sized networks when a large network teaches its own smaller variation. The key value is that after the training, the smaller network works faster than the large one, giving the same result.

- **Transfer learning** approach allows improving the accuracy through training the whole network or only certain layers on a specific dataset. For example, if the face recognition system has race bias issues, we can take a particular set of images, let's say, pictures of Chinese people, and train the network so to reach higher accuracy.

  Quantization approach improves a neural network to reach higher processing speed. By approximating a neural network that uses floating-point numbers by a neural network of low bit width numbers, we can reduce the memory size and number of computations.

- **Depth-wise separable convolutions** is a class of layers, which allow building CNN with a much smaller set of parameters compared to standard CNNs. While having a small number of computations, this feature can improve the facial recognition system so as to make it suitable for mobile vision applications.

The key element of deep learning technologies is the demand for high-powered hardware. When using deep neural networks for face recognition software development, the goal is not only to enhance recognition accuracy but also to reduce the response time.

5.7 APPLICATIONS

- Image recognition
- Video analysis
- Natural language processing
- Anomaly Detection
- Drug discovery
- Health risk assessment and biomarkers of aging discovery
- Checkers game
- Cultural Heritage and 3D-datasets

CHAPTER 6: Web Development And Project

Django is a widely used free, open-source, and high-level web development framework. It provides a lot of features to the developers "out of the box," so development can be rapid. However, websites built from it are secured, scalable, and maintainable at the same time.

6.1 Basic Design Of Website With Django

Goal

The goal is to build a application where the content can be created and updated through an administration panel,contents are displayed on the page and can be deleted if needed. Overall application provides CRUD(Create,Read,Update,Delete) functionality.

Required Setup

1. **Git Bash:** The user of all operating systems can use it. All the Django related commands and Unix commands are done through it.
2. **Text-Editor:** Any Text-Editor like Sublime Text or Visual Studio Code can be used. For the following project, PyCharm is used.
3. **Python 3:** The latest version of Python is used.

Virtual Environment

Virtual Environment acts as dependencies to the Python-related projects. It works as a self-contained container or an isolated environment where all the Python-related packages and the required versions related to a specific project are installed. Since newer versions of Python, Django, or packages, etc. will roll out, through the help of a Virtual Environment, you can work with older versions that are specific to your project. In Summary, you can start an independent project related to Django of version 2.0, whereas another independent project related to Django of version 3.0 can be started on the same computer. Steps to create a Virtual Environment are

1. You can create the new directory named 'project-' by using 'mkdir' command in your Desktop.
2. Change the directory to 'project-' by using 'cd' command.
3. The virtual environment is created by using 'python -m venv env', where env is our virtual environment shown by 'ls' command.

4. **For Activating your Virtual Environment:** The Virtual Environment can be activated by using the 'source' command where the 'Scripts' folder needs to be enabled or activated. The 'env' will be shown in the parenthesis if you've successfully activated your Virtual Environment.

5. **Installing the required package:** You can use **'pip install django'** to install Django in your specific Virtual Environment.

Creating a Django Project

1. The first step is creating your project by using the **'django-admin startproject project_name'** command, where 'project_name' is 'django_' in your case. Also, it will generate a lot of files inside our newly created project.

2. Change the directory to the newly created project using 'cd' command and to view the created file using 'ls' command.

3. You can run your project by using **'python manage.py runserver'**.

4. The project can be viewed in your favorite browser (Google Chrome, Mozilla Firefox, etc.). You can come into your browser and type '127.0.0.1:8000' in the URL.

Starting the new Project

For creating a new project in the Django, it's always a two-step process, which is shown below.

1. The first step is to create an app by using **'python manage.py startapp app_name'** command, where app_name is '' in your case. In Django, there are many apps to the single project where each app serves as single and specific functionality to the particular project.

2. The second step is to make our project let know about our newly created app by making changes to the 'django_/settings.py' **INSTALLED_APP** section.

3. Changing in our Models

Django uses 'SQLite' as the default database, which is light and only used for small projects, which is fine for this project. It uses **'Object Relational Mapper(ORM)'** which makes it really easy to work with the database. The actual database code is not written, whereas the database tables are created through the help of 'class' keyword in **'models.py'.**Inside **'/models.py'**, you need to create a new model named 'Post'. This is a class that will become

a database table afterward which currently inherits from 'models.Model'. As a certain 'Post' contains a title, which will be a field called **CharField.** It is a text-based column and accepts mandatory argument as **'max_length'**, which happens to be 50 in your case. Also, there is another field named 'content', which is the TextField, which contains the detail text of the 'Post' as in a standard. The double underscore('*str*') method is defined, which overrides the field 'title' and returns the name of actual 'title' instead of some objects.

4. **Making a Migrations**

'python manage.py makemigrations' is a first step process which reads the **'models.py'** after it's creation. It creates a new folder called **'migrations'** where there is a file named **'0001_initial.py'**, which are portable across the database. The second step where **'python manage.py migrate'** reads the newly created folder **'migrations'** and creates the database, and it evolves the database when there is a change in the model.

5. **Registering to the admin**

Let's move to **'/admin.py'** and do an import of the models called **'Post'** by using **'from.models import Post'**. To register models to the admin, the command is **'admin.site.register(Post)'**.

6. **Creating Super user and Viewing in the Administration panel**

You need to create a Super user before accessing the 'admin' panel. To do so, use **'winpty python manage.py createsuperuser'**. Run your server in the background in bash by command **python manage.py runserver**. Head over to the browser and type the following in the URL. View your admin panel afterward with our newly created models 'Post'. Change the content of the 'Post' by clicking the 'Add' button. Fill out the information and 'Save' the detail.Changing in views and urls by moving to **'/views.py'** and make the changes as shown below. Add the function **'_list'**, which takes in the request. The query is made, which gets all the objects created using **'Post.objects.all()'** and save it to the post. There is a newly created dictionary as 'context' where the object can be passed as key and obtained through template **'-list.html'**, which is done by returning the response with the help of render. Make a new file called **'urls.py'** in **'django_/'** and add the following changes. There is relative import to the views '_list' also a **'urlpatterns'**, which is a list of the path to a specific page on the website. Currently, the <b'path'< b="" style="box-sizing: border-box;">contains the empty string and name of the view.Let's move to **'django_/urls.py'** and import include and make a change to **'urlpatterns'.** Then add the path to your app URLs through include. Also, when users route through 'posts/' then, it gets directed to our **'.urls.'**

7. **Making and Changing the Templates**

Let's make a templates folder that generally holds the 'HTML' and contains their own templating language called '**Jinja2'.** The folder needs to name '**templates//_list.html**', which is the convention. You can see below there is syntax related to '**HyperTextMarkupLanguage(HTML)** where '**h1**' for big headline and an unordered list(**ul**) with list element **li.** Also, ' for' loop syntax related to '**Jinja 2'** is used where an object called '_list' passed as key from '**/views.py**' with each element called 'list' is iterated. View the 'title' named as 'First Post' on the webpage. Let's add another information from the admin panel the same as above and name your second post **title** as 'Second Post.' After adding the information and reloading the homepage, the information will be updated.

8. Details for Each Individual post

You will be creating each individual page containing information regarding the post title, and it's content. The 'url' will be "localhost:8000/posts/'id" where id indicates the unique number or primary key attached to each 'Post' given by Django itself. Let's create a function as '**_detail**' in '**/view.py**' , which accepts id as a parameter. Also, there is a query made for getting specific id only and saving to 'each_post'. Similarly, as above, the information needed is passed as context to the '**_detail.html'.** The url in '**/urls.py**' is changed where path contains the id, which accepts the unique id in the integer form. Assume that the user comes to 'posts/' only then sees all the posts, but when they to 'posts/1', they only see the information regarding the first created post. Let's create a new file, '**/_detail.html**', and make the following changes. Since _detail is passed as context, the 'title' and 'content' can be accessed using dot notation. Head over to the URL of your browser and type the same thing to get individual posts. Since the 'id' for the first information created is '1' where the second information is to be '2' and so on for newly created information.

9. Deleting the Post

Let's define the **_delete**, which takes in the request and the id. Also, query is made where '**Post.objects.get(id=id)**' gets the object with a unique id and save it to **each_post**. After that the '**each_post.delete()**' is called to delete the 'Post'. Finally **HttpResponseRedirect** is imported from the '**django.http**' module where it is used to redirect the page to the '**/posts/**'. In the '**urls.py**' import '**_delete**' and path is set to '**<id>/delete**' where the id with delete at the end will remove that particular object or information. Let's delete our post by typing the following thing in the '**urls.py'.** At last, the page gets redirected to '**/posts**' when '**posts/1/delete/**' is called where only one post exists on the homepage.

6.2 Linkage Of Our Attendance Project To Webserver

Till now we saw how to design a web server using Django and python. Now we will see how to link the attendees data to the webserver and the steps follow in the below prescribed process.

1.Creation of a python file.

2.To code the program to append the data entries when the face is detected to the webserver.

3.Refreshing the website would add the new attendees to the list.

We have 2 pages admin and viewer and their working as follows:

1. **Admin page:**

   On this page the admin can log on view, edit and alter the attendees data entries which can be seen on the viewer page.

2. **Viewer page:**

   On this page the formulated data from the admin page is displayed and the data can be filtered on the basis of various fields.

6.3 Mailing the attendance files

Sending emails manually is a time-consuming and error-prone task, but it's easy to automate with Python. The following is done using the mailing module

- Set up a secure connection using SMTP_SSL() and .starttls()
- Use Python's built-in smtplib library to send basic emails
- Send emails with HTML content and attachments using the email package

Getting Started

Python comes with the built-in smtplib module for sending emails using the Simple Mail Transfer Protocol (SMTP). smtplib uses the RFC 821 protocol for SMTP. The examples in this tutorial will use the Gmail SMTP server to send emails, but the same principles apply to other email services. Although the majority of email providers use the same connection ports as the ones in this tutorial, you can run a quick Google search to confirm yours.To get started with this tutorial, set up a Gmail account for development, or set up an SMTP debugging server that discards emails you send and prints them to the command prompt instead. Both options are laid out for you below. A local SMTP debugging server can be useful for fixing any issues with email functionality and ensuring your email functions are bug-free before sending out any emails.

Option 1: Setting up a Gmail Account for Development

If you decide to use a Gmail account to send your emails, I highly recommend setting up a throwaway account for the development of your code. This is because you'll have to adjust your Gmail account's security settings to allow access from your Python code, and because there's a chance you might accidentally expose your login details. Also, I found that the inbox of my testing account rapidly filled up with test emails, which is reason enough to set up a new Gmail account for development. A nice feature of Gmail is that you can use the + sign to add any modifiers to your email address, right before the @ sign. For example, mail sent to my+person1@gmail.com and my+person2@gmail.com will both arrive at my@gmail.com. When testing email functionality, you can use this to emulate multiple addresses that all point to the same inbox. If you don't want to lower the security settings of your Gmail account, check out Google's documentation on how to gain access credentials for your Python script, using the OAuth2 authorization framework.

Option 2: Setting up a Local SMTP Server

You can test email functionality by running a local SMTP debugging server, using the smtpd module that comes pre-installed with Python. Rather than sending emails to the specified address, it discards them and prints their content to the console. Running a local debugging server means it's not necessary to deal with encryption of messages or use credentials to log in to an email server.For the rest we'll assume you're using a Gmail account, but if you're using a local debugging server, just make sure to use localhost as your SMTP server and use port 1025 rather than port 465 or 587. Besides this, you won't need to use login() or encrypt the communication using SSL/TLS.

Sending a Plain-Text Email

Before we dive into sending emails with HTML content and attachments, you'll learn to send plain-text emails using Python. These are emails that you could write up in a simple text editor. There's no fancy stuff like text formatting or hyperlinks. You'll learn that a bit later.

Starting a Secure SMTP Connection

When you send emails through Python, you should make sure that your SMTP connection is encrypted, so that your message and login credentials are not easily accessed by others. SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are two protocols that can be used to encrypt an SMTP connection. It's not necessary to use either of these when using a local debugging server.To start a secure connection with your email server in

this project we started an SMTP connection that is secured from the beginning using SMTP_SSL(). Gmail will encrypt emails using TLS, as this is the more secure successor of SSL. As per Python's Security considerations, it is highly recommended that you use create_default_context() from the ssl module. This will load the system's trusted CA certificates, enable host name checking and certificate validation, and try to choose reasonably secure protocol and cipher settings. If you want to check the encryption for an email in your Gmail inbox, go to More → Show original to see the encryption type listed under the Received header. smtplib is Python's built-in module for sending emails to any Internet machine with an SMTP or ESMTP listener daemon. I'll show you how to use SMTP_SSL() first, as it instantiates a connection that is secure from the outset and is slightly more concise than the .starttls() alternative. Keep in mind that Gmail requires that you connect to port 465 if using SMTP_SSL(), and to port 587 when using .starttls().

Using SMTP_SSL()

The code example below creates a secure connection with Gmail's SMTP server, using the SMTP_SSL() of smtplib to initiate a TLS-encrypted connection. The default context of ssl validates the host name and its certificates and optimizes the security of the connection. Using with smtplib.SMTP_SSL() as server: makes sure that the connection is automatically closed at the end of the indented code block. If port is zero, or not specified. SMTP_SSL() will use the standard port for SMTP over SSL (port 465). It's not safe practice to store your email password in your code, especially if you intend to share it with others. Instead, use input() to let the user type in their password when running the script, as in the example above. If you don't want your password to show on your screen when you type it, you can import the getpass module and use .getpass() instead for blind input of your password. To identify yourself to the server, .helo() (SMTP) or .ehlo() (ESMTP) should be called after creating an .SMTP() object, and again after .starttls(). This function is implicitly called by .starttls() and .sendmail() if needed, so unless you want to check the SMTP service extensions of the server, it is not necessary to use .helo() or .ehlo() explicitly.

The message string starts with "Subject: Hi there" followed by two newlines (\n). This ensures Hi there shows up as the subject of the email, and the text following the newlines will be treated as the message body. For comparison, here is a code example that sends a plain-text email over an SMTP connection secured with .starttls(). The server.ehlo() lines may be omitted, as they are called implicitly by .starttls() and .sendmail().

Including HTML Content

If you want to format the text in your email or if you want to add any images, hyperlinks, or responsive content, then HTML comes in very handy. Today's most common type of email is the MIME (Multipurpose Internet Mail

Extensions) Multipart email, combining HTML and plain-text. MIME messages are handled by Python's email.mime module. For a detailed description, check the documentation. As not all email clients display HTML content by default, and some people choose only to receive plain-text emails for security reasons, it is important to include a plain-text alternative for HTML messages. As the email client will render the last multipart attachment first, make sure to add the HTML message after the plain-text version. In the example below, our MIMEText() objects will contain the HTML and plain-text versions of our message, and the MIMEMultipart("alternative") instance combines these into a single message with two alternative rendering options. We must first define the plain-text and HTML message as string literals, and then store them as plain/html MIMEText objects. These can then be added in this order the MIMEMultipart("alternative") message and sent through your secure connection with the email server. Remember to add the HTML message after the plain-text alternative, as email clients will try to render the last subpart first.

Adding Attachments Using the email Package

In order to send binary files to an email server that is designed to work with textual data, they need to be encoded before transport. This is most commonly done using base64, which encodes binary data into printable ASCII characters. The MIMEultipart() message accepts parameters in the form of RFC5233-style key/value pairs, which are stored in a dictionary and passed to the .add_header method of the Message base class.

Sending Multiple Personalized Emails

Imagine you want to send emails to members of your organization, to remind them to pay their contribution fees. Or maybe you want to send students in your class personalized emails with the grades for their recent assignment. These tasks are a breeze in Python.

Loop Over Rows to Send Multiple Emails

The code example below shows you how to open a CSV file and loop over its lines of content (skipping the header row). To make sure that the code works correctly before you send emails to all your contacts, I've printed Sending email to ... for each contact, which we can later replace with functionality that actually sends out emails.Using open(filename) as file:makes sure that your file closes at the end of the code block. csv.reader() makes it easy to read a CSV file line by line and extract its values. The next(reader) line skips the header row, so that the following line for name, email, grade in reader: splits subsequent rows at each comma,

and stores the resulting values in the strings name, email and grade for the current contact.If the values in your CSV file contain whitespaces on either or both sides, you can remove them using the .strip() method.

Personalized Content

You can put personalized content in a message by using str.format() to fill in curly-bracket placeholders. For example, "hi {name}, you {result} your assignment".format(name="John", result="passed") will give you "hi John, you passed your assignment".

As of Python 3.6, string formatting can be done more elegantly using f-strings, but these require the placeholders to be defined before the f-string itself. In order to define the email message at the beginning of the script, and fill in placeholders for each contact when looping over the CSV file, the older .format() method is used. We can set up a general message body, with placeholders that can be tailored to individuals.

Conclusion

You can now start a secure SMTP connection and send multiple personalized emails to the people in your contacts list. You've learned how to send an HTML email with a plain-text alternative and attach files to your emails. When you're using a Gmail account. If you plan to send large volumes of email, it is worth looking into transactional email services.

CHAPTER 7: Results

Sample 128-d encodings of a face

imgloc: (761, 1573, 1146, 1188)

imgencode: [-1.78218409e-01 1.09680213e-01 8.87243915e-03 -2.79957354e-02 -7.98132196e-02 1.52062252e-03 -7.83939362e-02 -1.30094260e-012.71376789e-01 -1.77497581e-01 1.51594386e-01 1.71610601e-02 -1.79531321e-01 8.85028951e-03]

imgtestloc: (761, 1573, 1146, 1188)

imgtestencode: [-1.78218409e-01 1.09680213e-01 8.87243915e-03 -2.79957354e-02 -7.98132196e-02 1.52062252e-03 -7.83939362e-02 -1.30094260e-01 2.71376789e-01 -1.77497581e-01 1.51594386e-01 1.71610601e-02 -1.79531321e-01 8.85028951e-03 -1.37727894e-02 9.80127007e-02]

Case 1:

Comparison of Similar photos

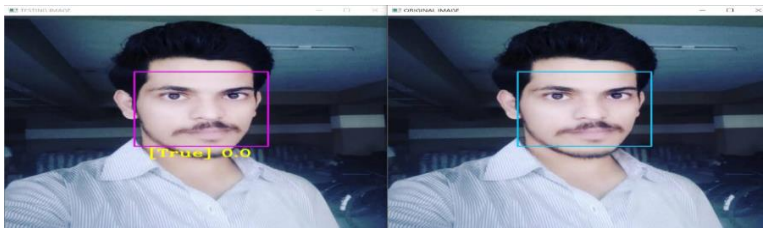


Fig 6.1 Comparison between similar images

Case 2:

Comparison of different photos

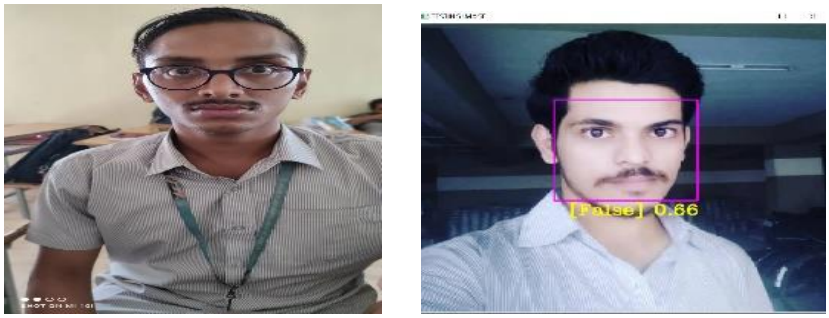


Fig 6.2 Comparison between Different images

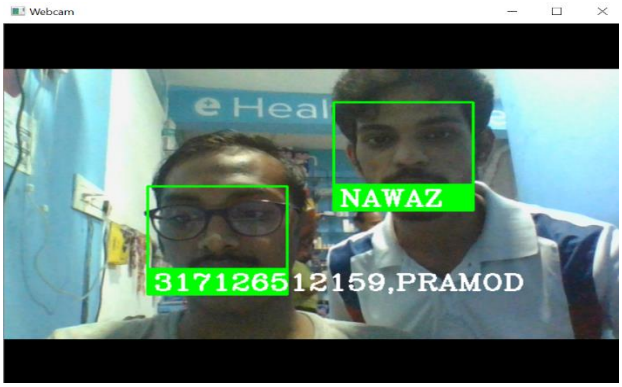


Fig 6.3

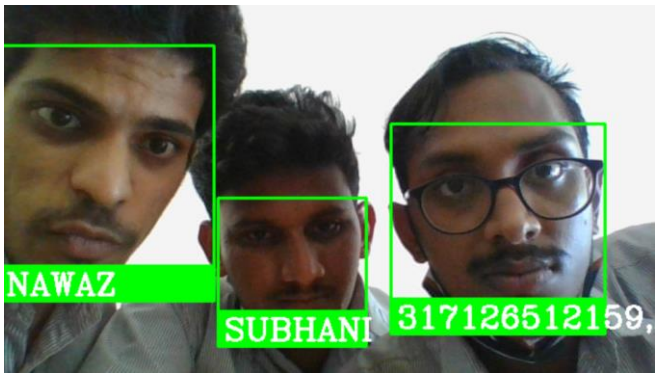


Fig 6.4

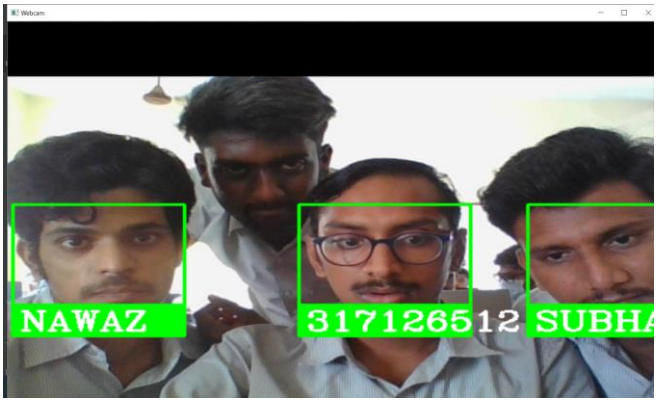


Fig 6.5

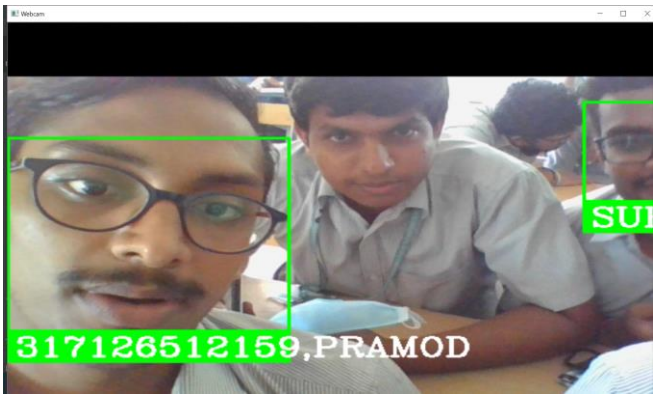


Fig 6.6

Fig 6.3-6.6 Multiple Face Detection

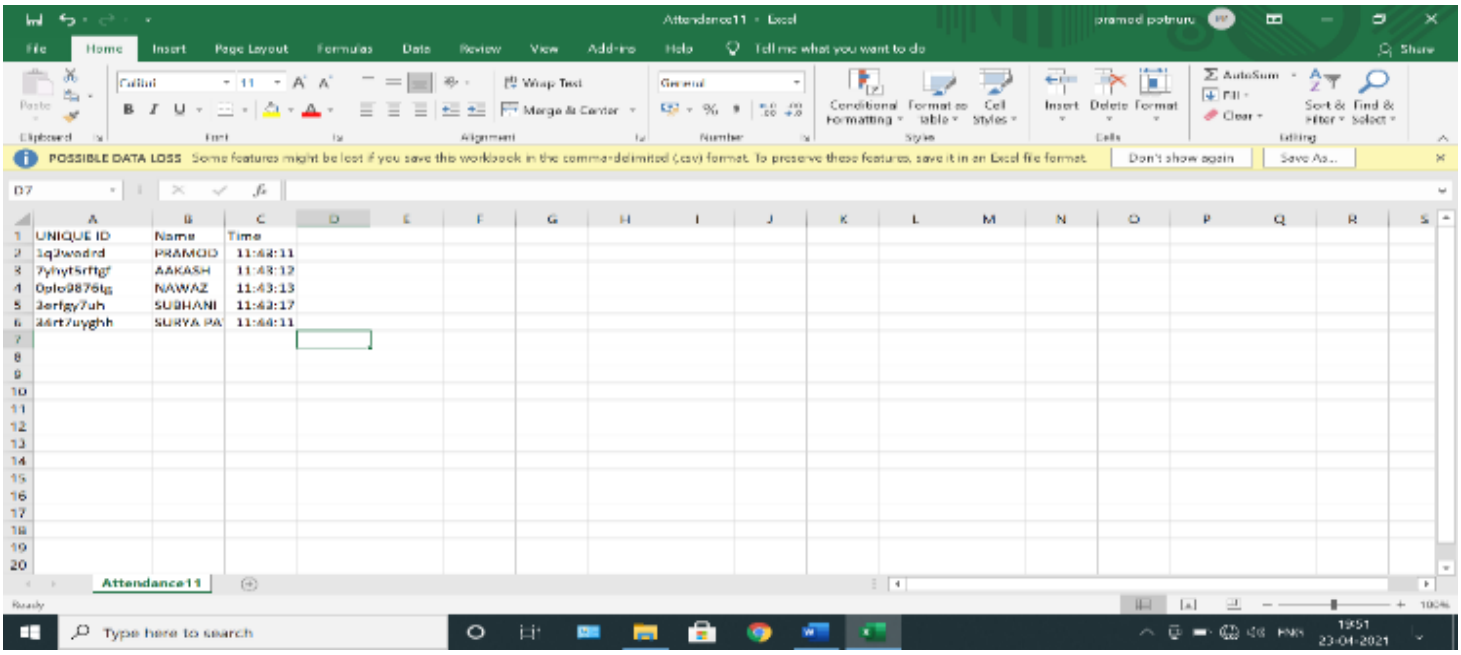


Fig 6.7- Attendance After Face Detection

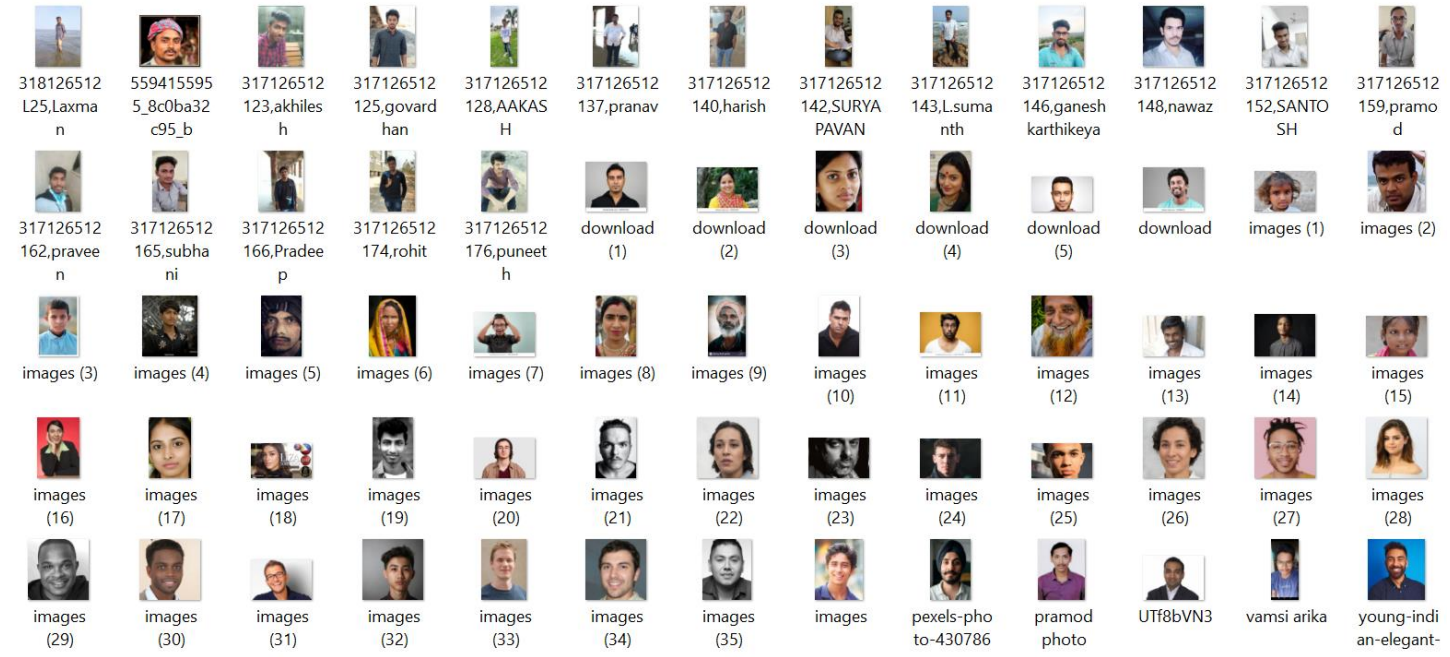


Fig 6.8- Data Base of Images

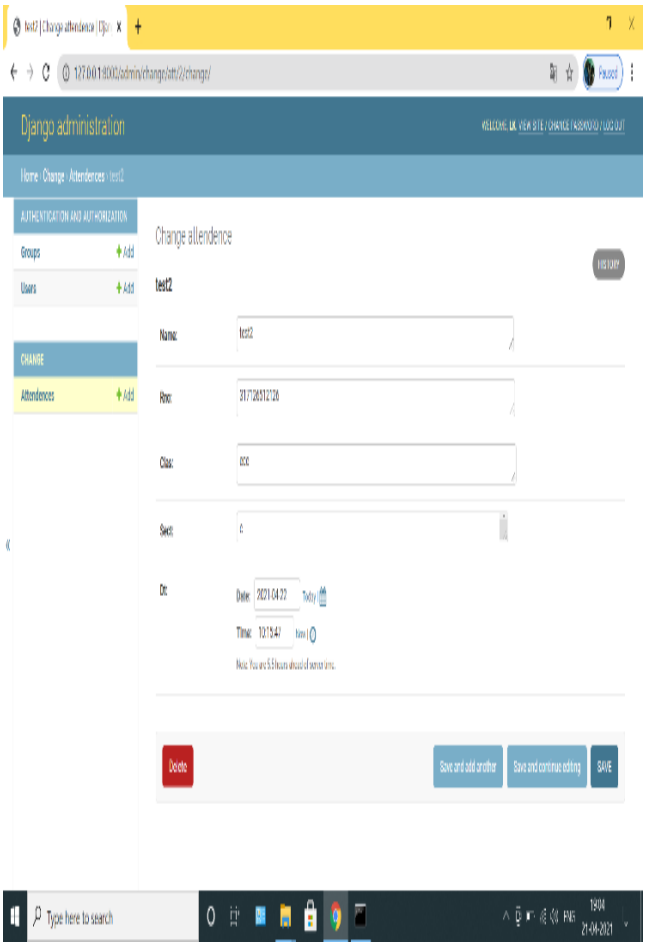


Fig 6.9

Fig 6.10

Fig 6.9 - 6.10 Web outputs

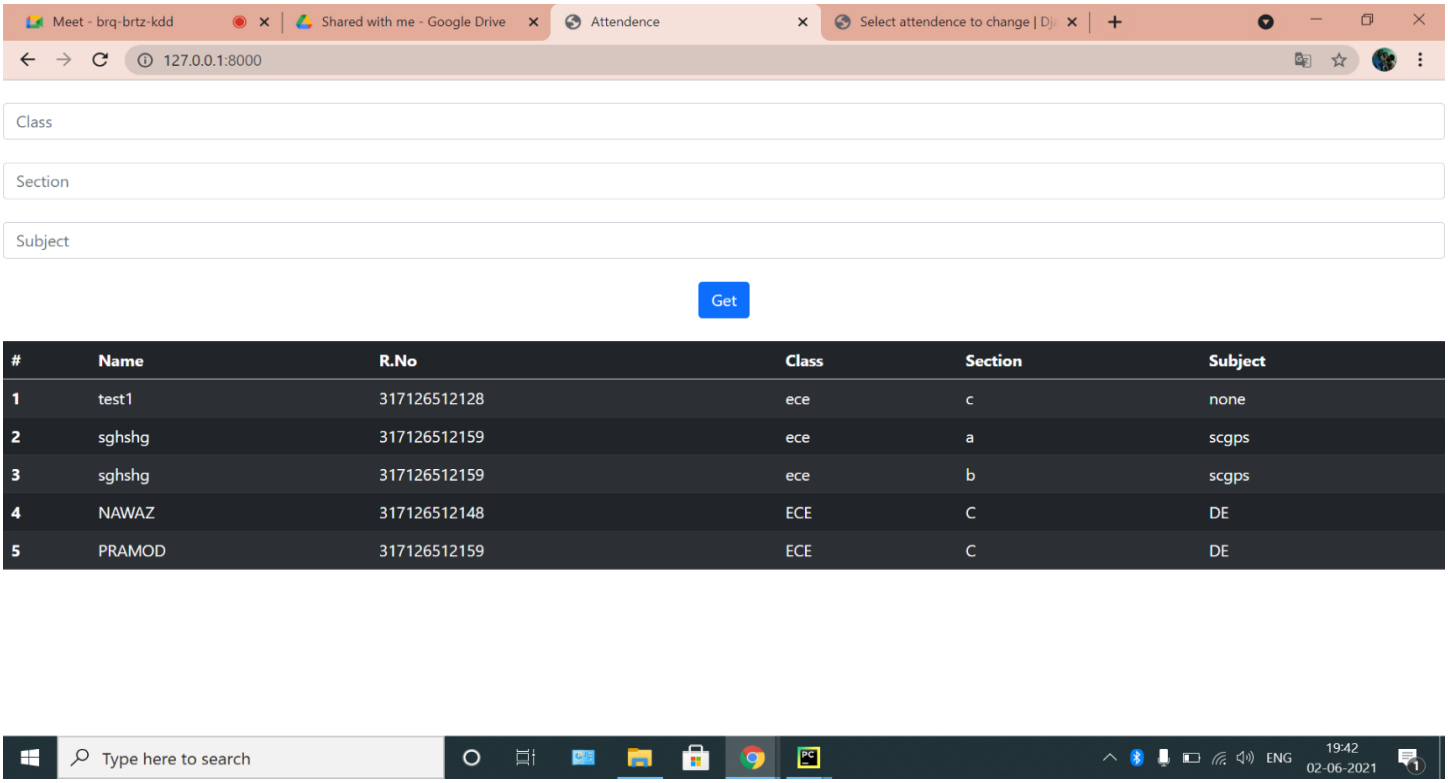


Fig6.11 Website outputs of viewer page

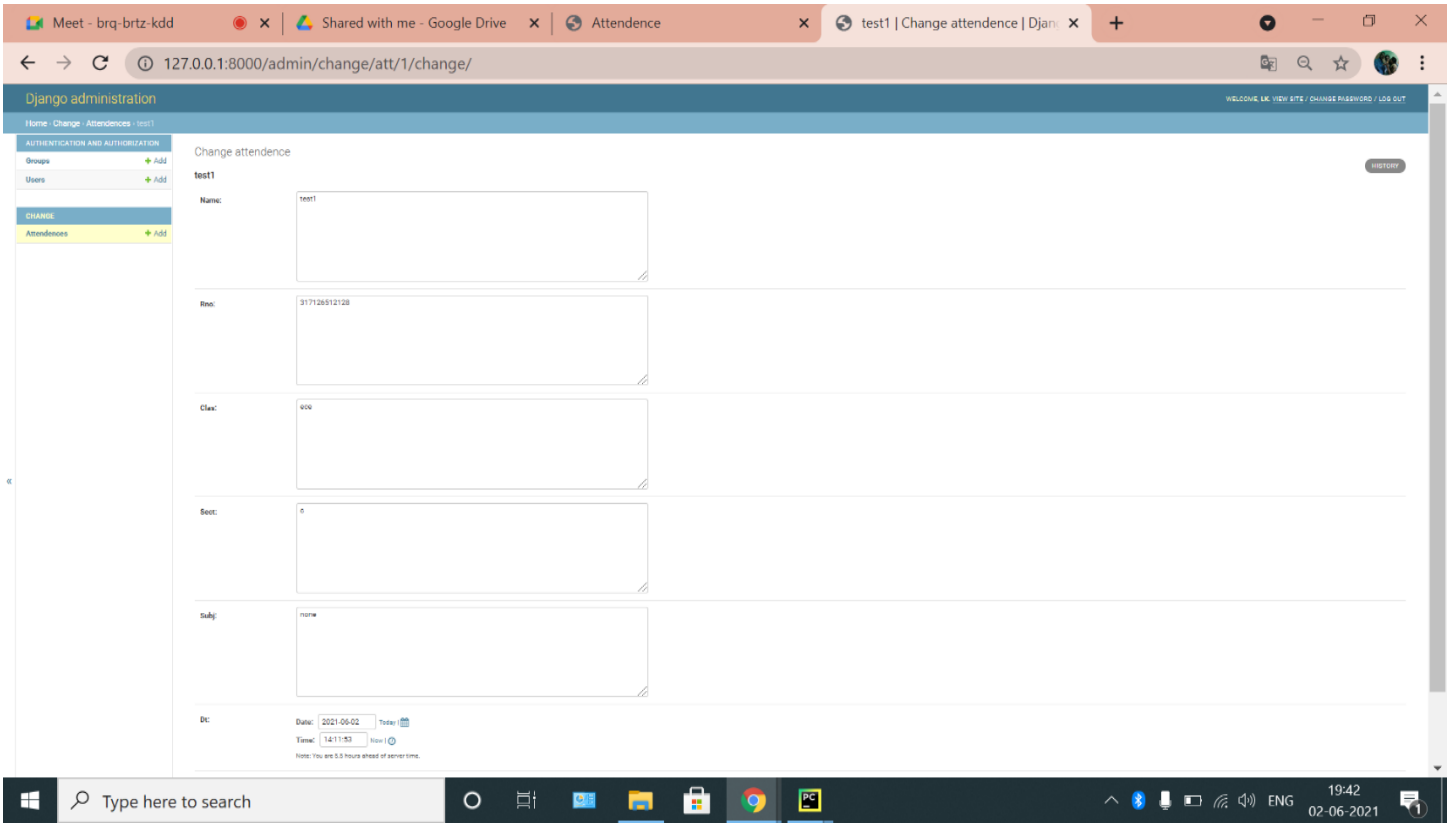


Fig6.12 Website outputs of admin page

CHAPTER 7: Conclusion

The proposed model has the capability of detecting and recognising different faces and images from the camera. The data set which contains the images are pre-trained and tested using deep learning so that the input images would be well detected. This method is secure enough, reliable and available for use. Further CNN adds robustness to the model and using this approach of training data, 95.21% recognition rate has been achieved. Deep learning has advantage over machine learning for other face recognition techniques. The most efficient technique is the LBPH algorithm by which the problems of local features are rectified. The resultant of this entire process is nothing but creating an attendance marking system in which the unique id, name and some more details of the recognised faces could be entered automatically into a CSV file. In further updation this attendance system can be taken towards web development by creating a website and marking the attendance of the people automatically into the website of the organisation so that there would be no need to update or mark attendance manually.

CHAPTER 8: Future Work

Presently this project is developed using Django python and is successfully running on web. In the future updation, a mobile application will be developed in which each and every student are given access with unique login details so that they can track their status of attendance from anywhere round the globe.

CHAPTER 9: References

[1] Umme Aiman1, Virendra P. Vishwakarma2," face recognition using modified deep learning neural network",IEEE- 40222,University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, Sector 16C, Dwarka, New Delhi-110078, India.

[2] Saibal Manna$_1$, Sushil Ghildiyal$_2$, Krishnakumar Bhimani," face recognition from video using deep learning",NIT Jamshedpur, Jharkhand, India, mannasaibal1994@gmail.com, VIT Vellore, Tamil Nadu, India, info.sushil123@gmail.com , Marwadi University, Rajkot, Gujarat, India, info.bhimani@gmail.com.

[3] Xiao Han1, Qingdong Du2 Research on face recognition based on deep learning college of software, Shenyang Normal University, China, 941066866@qq.com, 421532897@qq.com .

[4] Arjun Raj A$_1$, Mahammed Soheb$_2$, Aravind K$_3$, Chetan S$_4$ FACE RECOGNITION BASED ATTENDANCE SYSTEM ECE, PES UNIVERSITY, BANGLORE, INDIA.

[5] SudhaNarang$_1$, Kriti Jain$_2$, MeghaSaxena$_3$, AashnaArora$_4$ comparison of face recognition algorithms using OpenCV for attendance system assistance professor, Undergraduate Student Computer Science and Engineering Department, Maharaja Agrasen Institute of Technology, Delhi, India.

[6] K.Senthamil Selvi1, P.Chitrakala2, A. Antony Jenitha3,"face recognition based attendance marking system". Tech Scholar, Department of Information and Technology, Hindustan University, Chennai, Tamil Nadu, Indiaselvi.kss@gmail.com 2Assistant Professor, Department of Information and Technology, Hindustan University, Chennai, Tamilnadu, India chitrakala@hindustanuniv.ac.in 3M. Tech Scholar, Department of Information and Technology, Hindustan University, Chennai, Tamilnadu, India Jeni.anto23@gmail.com

Paper Publication Details

1. Paper has been sent to the editorial board and waiting for the status