

# **FACE RECOGNITION USING FACENET**

**A Project Report submitted in partial fulfilment of the requirements for the award of  
the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**SUBMITTED BY**

**R. Sai Prakash (1215316248)**

**P. Rahul (1215316240)**

**N. Nikhitha (1215316238)**

**N.J.SRI Harsha (1215316239)**

**Under the esteemed guidance of**

**Sindhu Pusarla**

**Assistant Professor**



**GITAM**

**(Deemed to be University)**

**VISAKHAPATNAM**

**April, 2020**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



**DECLARATION**

We, hereby declare that the Project Review entitled “**Face Recognition Using FaceNet**” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering.

The work has not been submitted to any other college or University for the award of any degree or diploma.

Signature

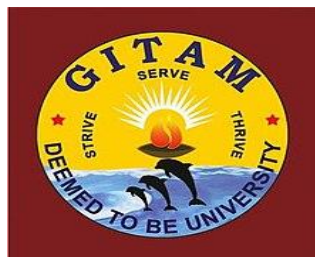
<b>ROLL NO.</b>	<b>NAME</b>	<b>SIGNATURE</b>
<b>1215316248</b>	<b>R. SAI PRAKASH</b>	
<b>1215316240</b>	<b>P. RAHUL</b>	
<b>1215316238</b>	<b>N. NIKITHA</b>	
<b>1215316239</b>	<b>N.J.S. HARSHA</b>	

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



**CERTIFICATE**

This is to certify that the Project Report entitled “**FACE RECOGNITION USING FACENET**” is a bona fide record of work carried out by **R. Sai Prakash(1215316248), P. Rahul(1215316240), N. Nikhitha(1215316238), N. Harsha(1215316239)** submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

**INTERNAL GUIDE**

Sindhu Pusarla

Assistant Professor, Dept of CSE

GIT, GITAM UNIVERSITY

Visakhapatnam

**HEAD OF DEPARTMENT**

Dr. K. Thammi Reddy

HOD, Dept. of CSE

GIT, GITAM UNIVERSITY

Visakhapatnam

# Table of Contents

Acknowledgement	
1. Abstract	
2. Introduction.....	1
2.1 Motivation and Overview.....	1
2.2 Problem Statement.....	2
2.3 Objectives.....	3
2.4 Solution.....	4
3. Literature Survey.....	5
4. Requirements.....	7
5. Dataset.....	8
5.1 Kaggle Dataset.....	8
5.2 Created Dataset.....	8
6. Algorithms.....	10
6.1 Convolutional Neural Networks.....	10
6.1.1 Introduction.....	10
6.1.2 Neural Network Architecture.....	11
7. Implementation Technique.....	17
7.1 Face Net.....	17
7.1.1 Working.....	18
7.1.2 Triplet Loss.....	19
7.2 MTCNN.....	20
8. Implementation.....	22
8.1 Implementing on Kaggle Dataset.....	23
8.2 Implementing on Created Dataset.....	31
9. Testing.....	33

9.1 Testing on Kaggle Dataset.....	33
9.2 Testing on Created Dataset.....	35
9.3 Real Time Testing.....	36
10. Conclusion and Future Scope.....	39
11. References.....	40

## ACKNOWLEDGEMENT

We consider ourselves very honoured to have so many wonderful people who lead me through in the completion of this final year project.

The satisfaction that accompanies the successful completion of the project would be incomplete without mentioning the people who made it possible and whose constant guidance and encouragement crowned all the efforts with success.

We express a profound sense of gratitude to my project guide **Prof. T. Sita Maha Lakshmi**, **Asst Prof. V. S. V. S. Murthy** and **Asst Prof. D. Rama Krishna** of Department of Computer Science, GITAM Institute of Technology, GITAM Deemed to be University for the expert guide and motivation given to me throughout our work. They spared their valuable time for patiently listening to our problems and helping us find solutions. This project report would not have been shaped this form without constant encouragement and cooperation in all aspects.

We also express my thanks to the project reviewer **Sindhu Pusarla**, Assistant Professor, Department of Computer Science and Engineering, GITAM for her valuable suggestions and guidance while reviewing my project.

We also express my deep gratitude to our beloved **Prof K. Thammi Reddy**, Head of the Department, Department of Computer Science and Engineering for encouragement and support given to us.

We take this opportunity to place on record my sincere thanks to our honourable Principal (Incharge) **Prof C. Dharma Raj**, PRINCIPAL(I/C), GITAM Institute of Technology, GITAM for this help and for providing all the required resources for completing project till this stage. Finally, we thank everyone who has supported me directly and indirectly in making the training a successful one.

# **1.ABSTRACT**

Face-Recognition is the uttermost widespread research and analysis space. Face-Recognition is utilized in numerous fields of application such as attendance framework, individuals trailing system, and access management framework. Face-recognition of multiple people from one frame has numerous difficulties for identification and recognition because it is challenging to detect several facial patterns and countenances from single frame edge and it is additionally hard to perceive the facial patterns with low resolution.

In our project, we built a highly efficient Face Recognition Model using FaceNet and MTCNN. It has many real time applications. The applications include Mobile Unlocking, Payments, Security and Access, Criminal Identification, Advertizing, Health Care, Attendance Management System and many more. Therefore, the current system can be utilized in an area where recognition is vital and of high priority.

The Model is built in Python Programming Language platform. The system implements Machine Learning techniques and Computer Vision Logic. This algorithm does high level processing and outputs the recognized person. The system is highly accurate and is very fast and easy to implement in any locality.

## 2. INTRODUCTION

### 2.1 Motivation and Overview

Extraction of helpful data from computerized picture frame is boosting the advancement of technology in many ways which is called Image processing. Image-Processing deals with three things:

- Pictorial data is enhanced and made easy to understand.
- Storing, Retrieving, Transmitting of Pictorial information and features after processing.
- Converting the data into suitable format such that a machine or computer which understands only binary could utilize and handle it.

Smart phones made many individuals use high quality camera features and capture their moments of life through photography and cinematography. Innovation in any field requires a lot of research and analysis. Image-Processing field is no exception. But once we get ahead in research space, we can deploy innovative solutions over a wide range of applications.

One of the first and most promising applications of Image-Processing is Facial recognition. It is the cheapest, easiest and infallible way of human detection. Facial image is intricate and needs smart processing techniques and implementations for recognition. One can argue biometrics have been doing the same for decades, then why go for Face-Recognition. Biometrics is costly and not every organization and government can implement it efficiently. And recent advanced hacking techniques have shown that biometrics can be broken down way too easy for a thing that comes at such costs.





## **2.2 PROBLEM STATEMENT**

Records of people who enter/exit are being maintained by every building, company, and university. Recognizing people provides value to serve customers better for many companies, keep track of student attendance easily and also identify a person who may be a potential threat. Face is a unique feature for an individual and we recognize people through faces. There is a claim stated that crimes become less and service becomes better if we maintain a record of people. In China they are already implementing Face Recognition in every locality and developing their country in all aspects.

The current system is not as efficient as it needs to be and more time is required to store records and retrieve data from it. Our system is designed to solve the complication of keeping track of people through their faces and automatically store it. It also needs to be easy and efficient to implement.

## **2.3 OBJECTIVES**

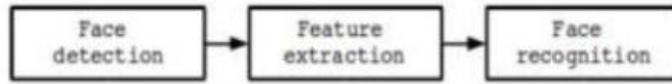
Face recognition and Identification has a broad range of applications similar to pictorial data and visual film processing, Interaction between human and machine, Offender identification and recognition, etc and can solve many issues. There are a lot of researchers trying to construct complex computational systems and algorithms to identify the facial features and other objects, which can be implemented in such a way that the application user has no difficulties in using it and also need not know the background processing and steps involved. Although a facial image has high dimensionality, it has very low dimensional space. So we can consider only a subspace of a face, reduce it's dimensionality and solve the issue. The objective is to build a system that distinguishes a unique face given as input, from large quantities of faces stored in a dataset with real-time dissimilarity.

## **2.4 SOLUTION**

Alternative for keeping records automatically and efficiently is provided by facial recognition. A Facial image is a multifaceted and needs very complex computational analysis. Face detection is a CV subset that helps to identify human faces in images.

### 3. LITERATURE SURVEY

Rishi Kumar[1] stated there are three steps involved in Face recognition:



Face-Recognition takes input from any image capturing device or a video capturing device. It then selects the facial characteristics such as eyes, nose, lips, and mouth. It does this by learning patterns and edges initially and later on higher processing layers it learns face cuts and recognizable facial features. Which are then formatted and matched against existing labels on which the framework has already been trained.

Aung Nway Oo[2] stated Face-Recognition is a challenge of the 21<sup>st</sup> century and we need to develop efficient solutions to solve the problem with high accuracy and implement in any daily applications and improve our world.

Face-Detection is extracting facial features from an image and eliminates the rest of the pixels. But this type of model has a flaw, it is difficult to determine the number of people in a frame as there may be overlapping faces.

FaceNet is a sophisticated and highly efficient implementation of CNN such that it identifies facial features of a unique individual with high accuracy. It showed marvelous results when implemented on YouTube Faces DB. SVM, along with FaceNet, is the best method of implementation of a framework system used to identify facial features and recognize people. FaceNet outputs a 128 vector embedding which is better classified by SVM than any other algorithm including Neural Networks. SVM + FaceNet combination produces outstanding results.

Florian Schroff[3] stated that he and his team were trying to build a model which would learn Embedding in Euclidean space for a particular image using a deep CNN's. The Model is built such that the squared L2 distances of the embeddings in Euclidean space corresponds to the similarity of facial images. Facial images of the same individual have very less distance when plotted in space between them and facial images of different individuals have large distances.

Many network models were proposed which were compact and cheap to deploy. Usage of an ensemble models over many of such network models, each operating on a dissimilar facial patch obtained a efficient performance on LFW dataset (99.47%).

PCA and a Joint Bayesian model that effectively corresponds to a linear transform in the embedding space are employed. The approach does not involve specific synchronization of 2D/3D. The networks are trained by using a combination of classification and verification loss. Their loss of verification is same triplet loss we employ, in that it minimizes the L2-distance between facial images of the same individual and raises the difference between the distances of facial images of different individuals.

The main difference is that only pairs of images are compared, whereas the triplet loss encourages a relative distance constraint.

Kaipeng Zhang [4] stated using facial feature detection/recognition as an auxiliary function to improve face alignment efficiency utilizing deep CNN.

Processing of first network of the MTCNN model easily produces candidate windows via a shallow CNN. Then, the casements are optimized by removing a significant number of non-face casements into a highly nuanced CNN network. Ultimately, a more efficient CNN is used to optimize the output again and generate five facial hallmark locations. Thanks to this multitask learning framework; the efficiency of the algorithm can be enhanced significantly.

Edwin Jose [5] stated Surveillance systems have deployed over many different platforms through the years, mainly in CPU-based embedded systems.

Model designed similar to FaceNet are efficient and provide high accuracies of 99 percentages. A face recognition-based surveillance system is proposed to be implemented using FaceNet.

## **4.REQUIREMENTS**

### **SOFTWARE:**

Modules : Numpy , Pandas, Scikit-Learn, Matplotlib,  
MTCNN, PIL, Pickle

Programming Language : Python

Platform : Kaggle, Google Colaboratory

## 5. DATASET

### 5.1 Kaggle Dataset

**Link :** <https://www.kaggle.com/dansbecker/5-celebrity-faces-dataset>

**Source:** <https://www.kaggle.com/>

### 5.2 Created Dataset

**Link :** <https://www.kaggle.com/saiprakash7/facerecog>

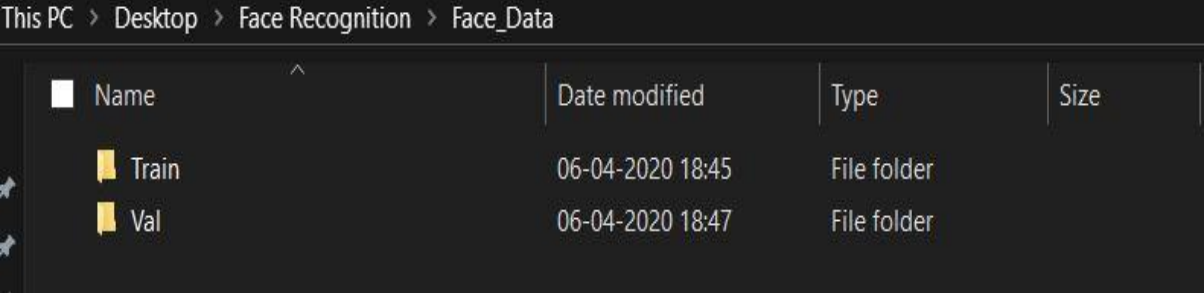
**Source:** <https://www.kaggle.com/>

This is a small dataset created for this project. It has a training directory containing 20 and 19 photos each of the people

- Rahul Paladugula
- Nikitha

For Testing it has 5 photos of each person.

These images are stored as original copies from the camera. They haven't been cropped for consistent aspect ratios.



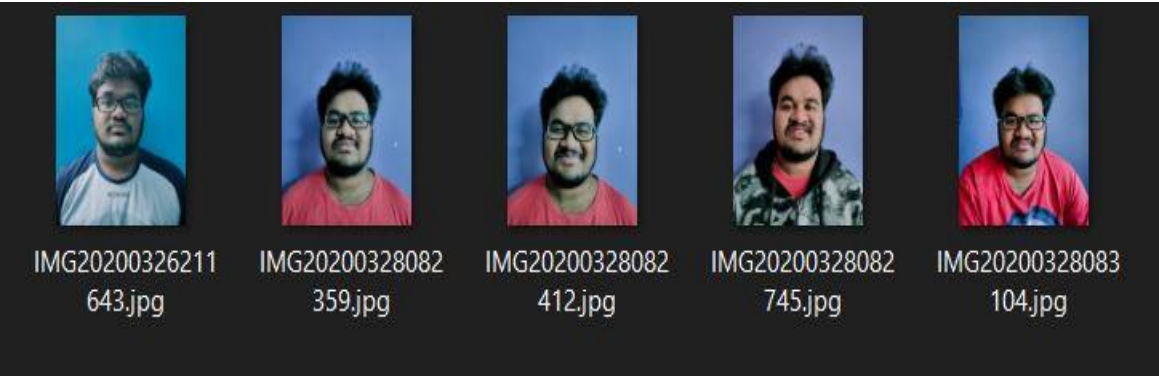
Name	Date modified	Type	Size
Train	06-04-2020 18:45	File folder	
Val	06-04-2020 18:47	File folder	

This PC > Desktop > Face Recognition > Face\_Data > Train

Name	Date modified	Type	Size
Nikitha	06-04-2020 18:46	File folder	
Rahul	06-04-2020 18:46	File folder	



- Training Data**



- Test Data**



## 6. ALGORITHMS

### 6.1 Convolutional Neural Networks

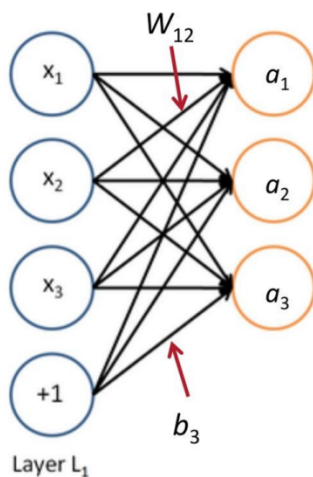
CNN do some very interesting stuff. For example, if you train them over a dataset of distinguished faces, in the first iteration they learn simple stuff like borders, vivid and non-vivid spots. They learn more as the layers deepen and iterations or epochs increase. In our instance, they get familiar with recognizing objects like eyes, nose and mouths. As layers are added, they learn more complex facial features. Similar like every other entity that needs to be identified. When provided with a motion of images on the screen they learn patterns from pixel frames of each image and as it appears. CNN can be built to play games as well. They learn to perform the right move on identifying a particular sequence of patterns. Some cases it plays far better than human. CNN's are powerful. They seem like magic. They sound mystical. Everything they do is based on some very simple concepts implemented in an intelligent way.

They are the best at classifying images. CNN's are weak with data which after rearranging does not provide more information. Their limitation is that they only capture local spatial patterns from data.

### 6.1.1 Neural Network Architecture:

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.

#### Background: Neural Networks



$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$
$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

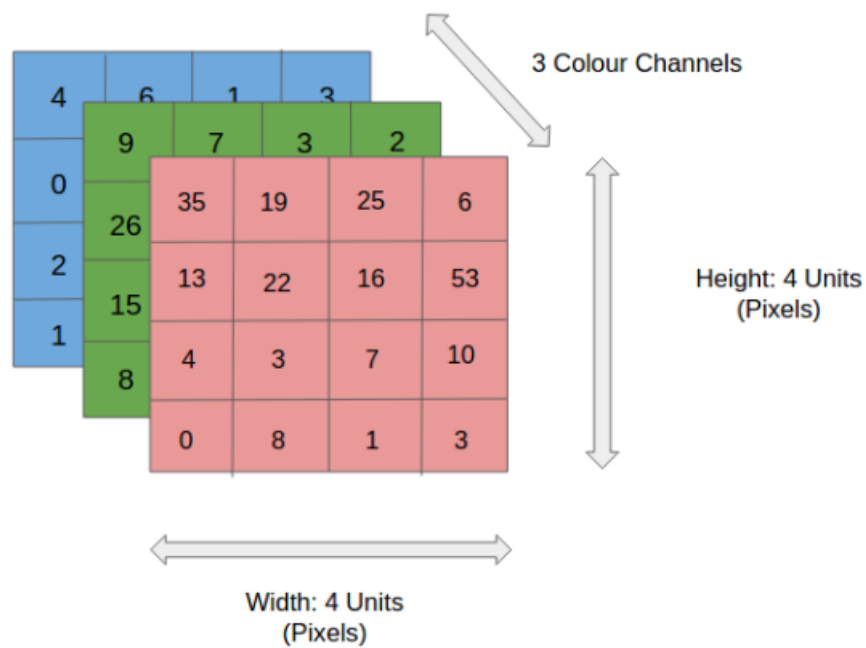
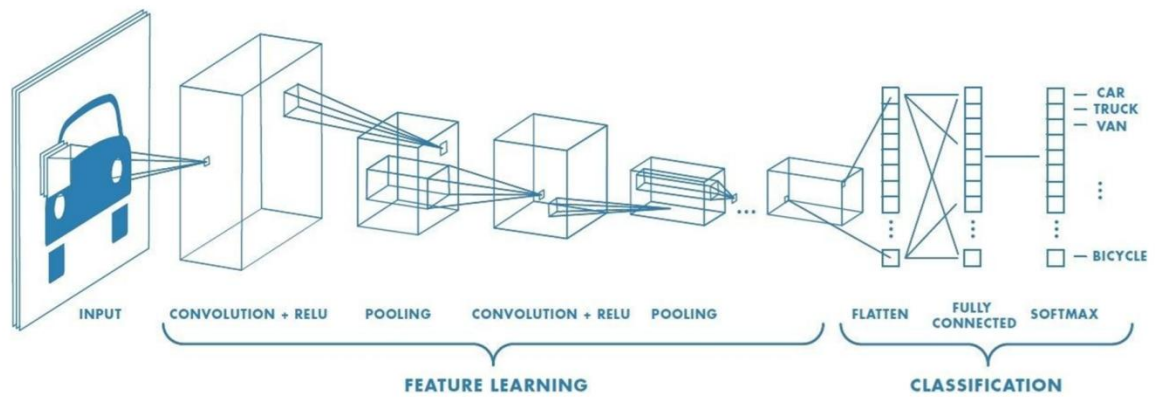
etc.

Express the equations in Matrix form, we have

$$z = Wx + b$$

$$a = f(z)$$

## Sample Architecture



## Convolution Layer :

- It is utilized to extract features from images but performing a dot product between an input matrix of image and a filter.

- An image matrix (volume) of dimension  $(h \times w \times d)$
- A filter  $(f_h \times f_w \times d)$
- Outputs a volume dimension  $(h - f_h + 1) \times (w - f_w + 1) \times 1$

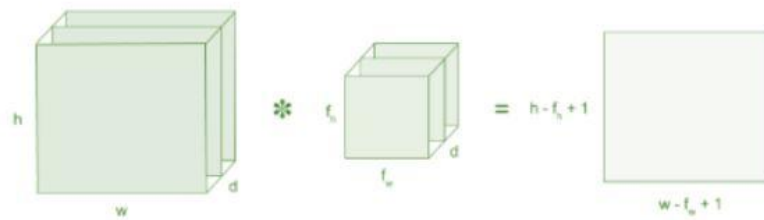
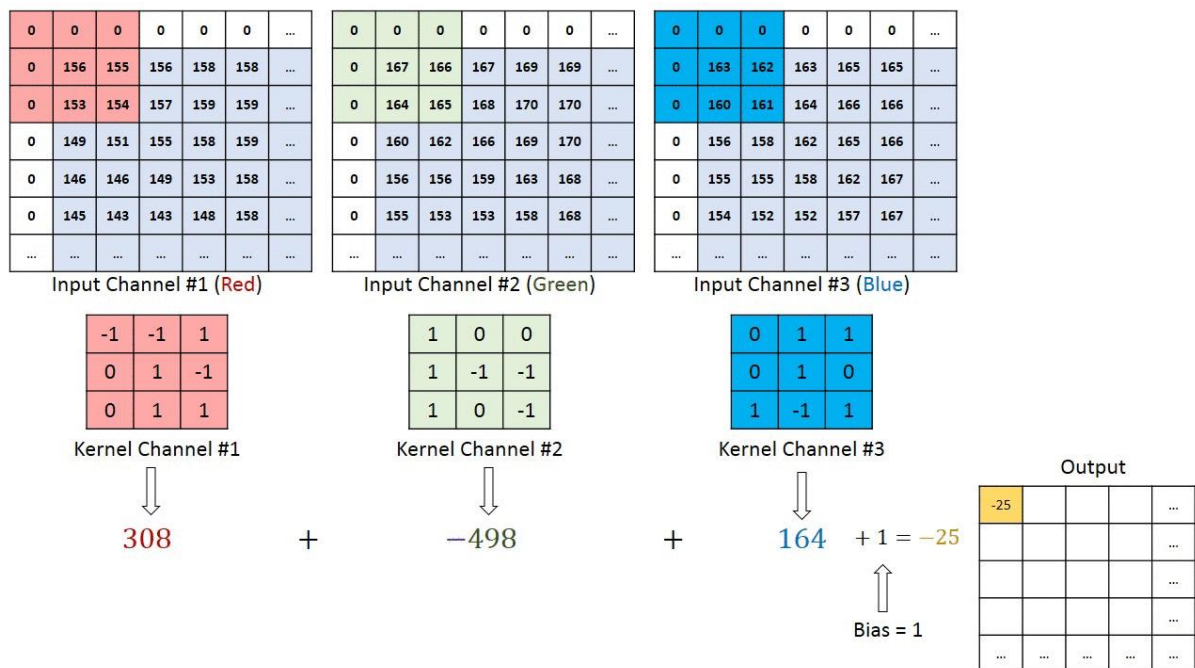
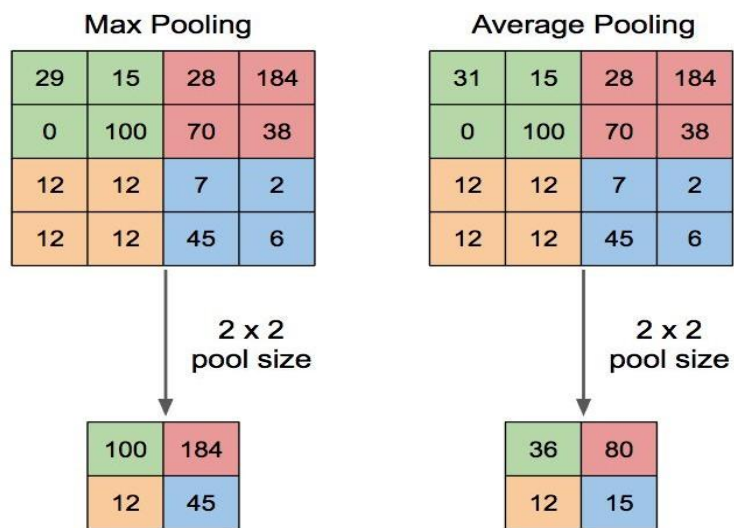


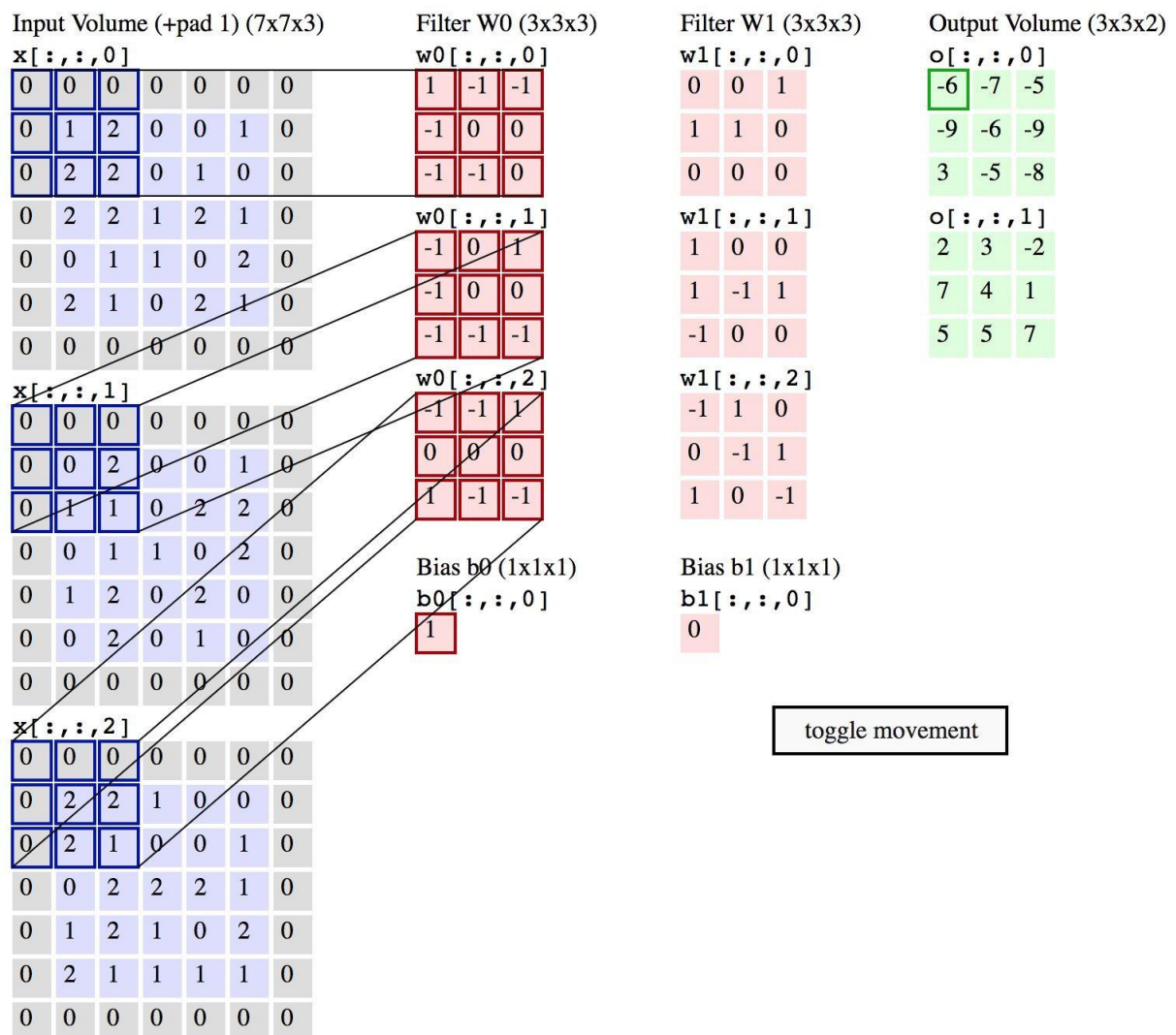
Figure 3: Image matrix multiplies kernel or filter matrix



## Pooling Layer :

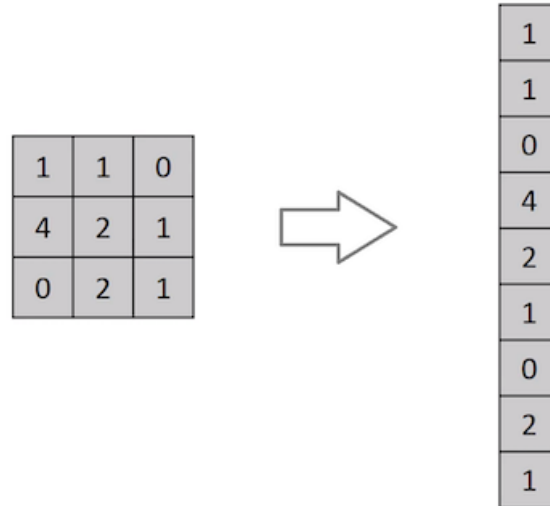
It reduces the size of the Feature output of Convolution Layer. It improves computation speed. It does this without losing important information of the convolved feature. Although there are two types, Max-Pooling is recommended and better.





## FC Layer :

We implement resizing or flattening of the output into a column vector. This is sent to a fully connected neural network and back-propagated over certain epochs and used for classification. We use softmax activation for the weightage of each output in classification.



Softmax is an activation function used at the end of Fully-connected. It converts the output vector of FC into probability vector.

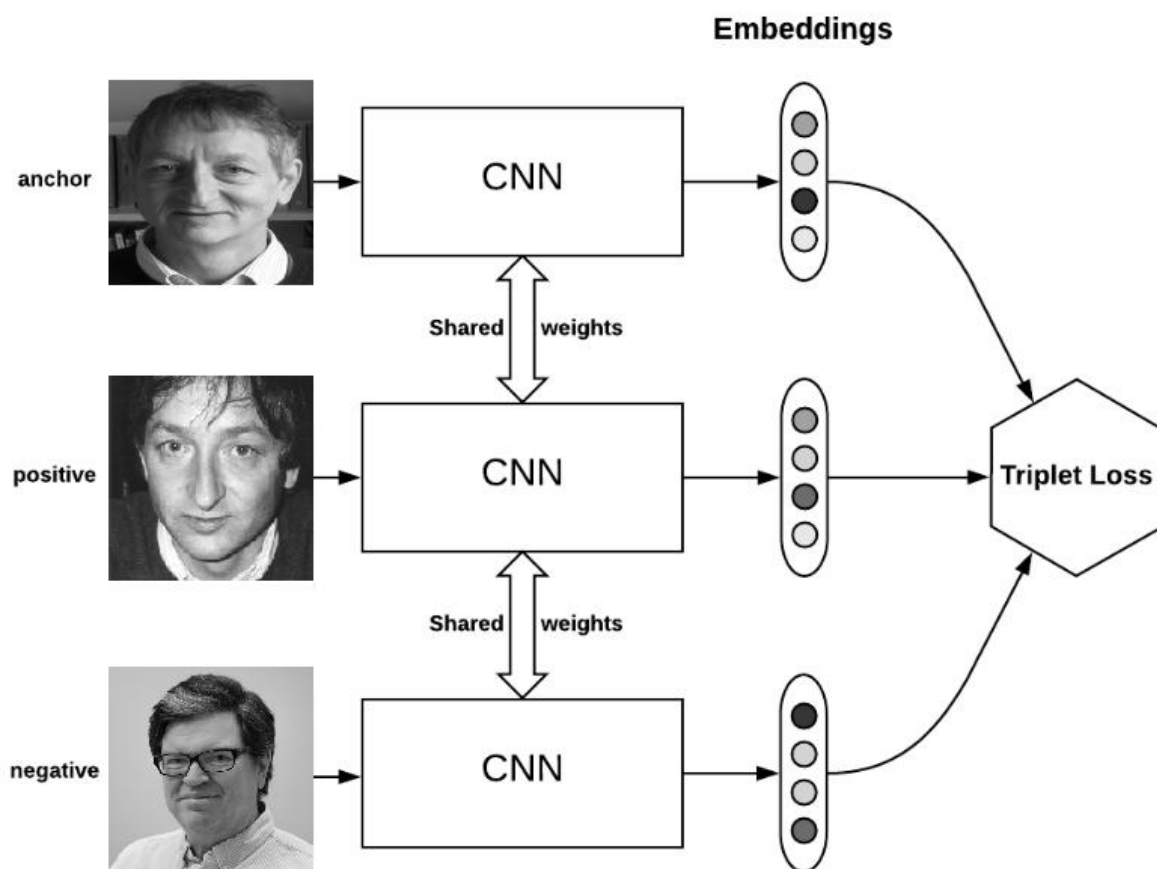
$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^T \mathbf{w}_k}}$$

## 7. IMPLEMENTATION TECHNIQUE

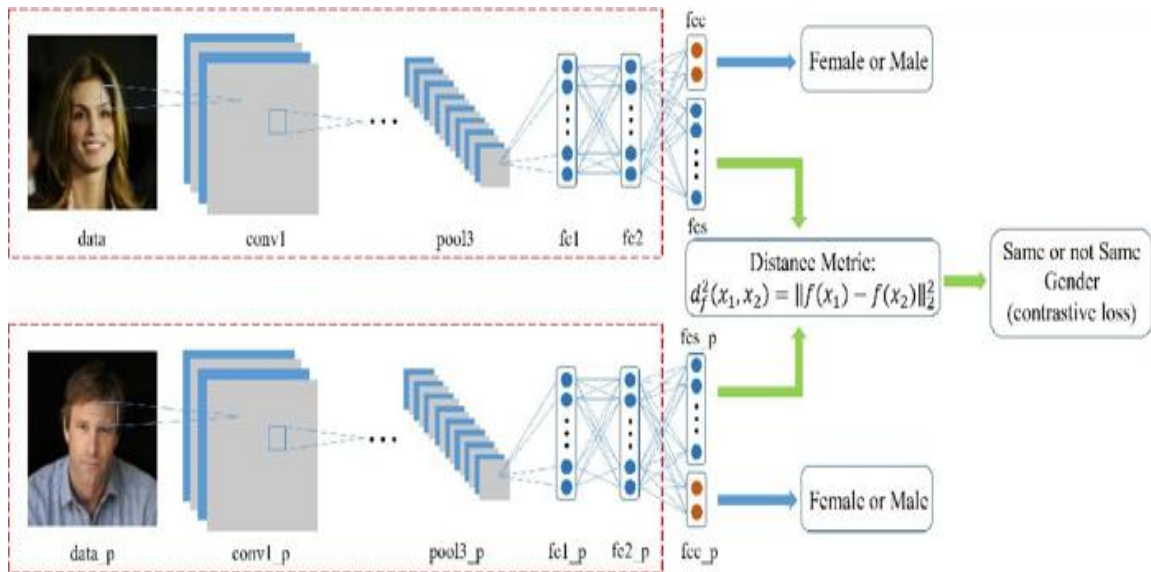
### 7.1 FACE NET – One Shot Learning

Traditional Neural Networks require a large amount of data and time to obtain high accuracy. This is inefficient for many cases and needs improvisation to implement in large sector industries. One-shot learning is a convenient solution for this problem. It is the best method where we have less data.

*FaceNet is a system that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.*

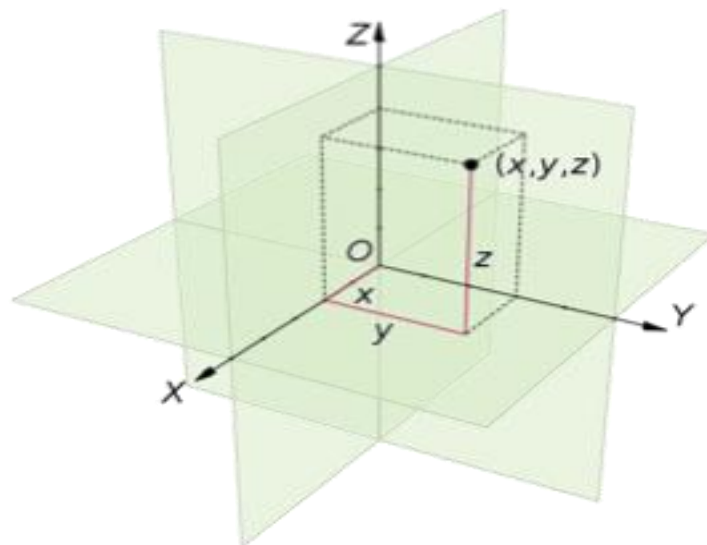






### 7.1.1 WORKING

Face-Net was created by Google. It is a highly efficient model created by Google researchers to identify faces. Face-Net uses CNN to learn features from images. It is a pre-trained network and easy to use. It creates a 128 length vector embedding of the dimensions of an image. The output embeddings when plotted over a 3d graph give results such that similar images have less distance and dissimilar images have large distance between them.

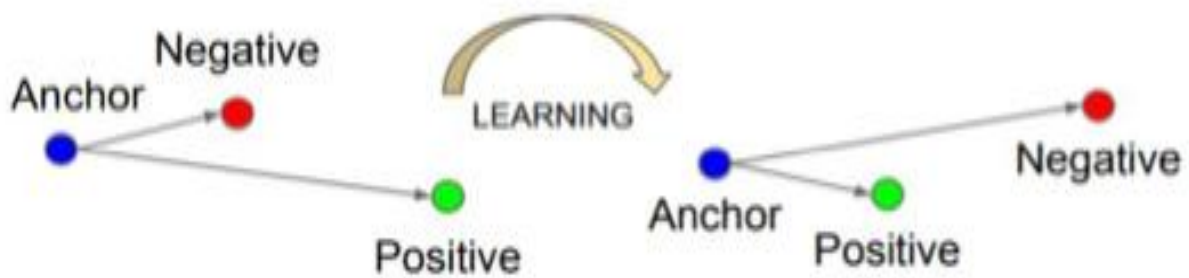




### 7.1.2 TRIPLET LOSS

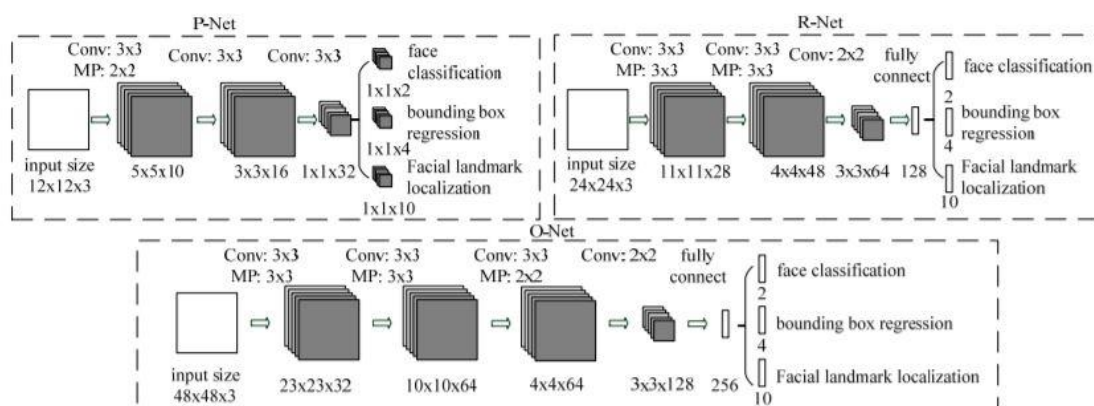
This loss function reduces the L2 distances between images which are similar and increases the L2 distance between dissimilar images. Similar images are forecasted into a solitary point in the space of embedding

It is designed such that it selects an anchor image and marks similar images as a positive and dissimilar image as negative.



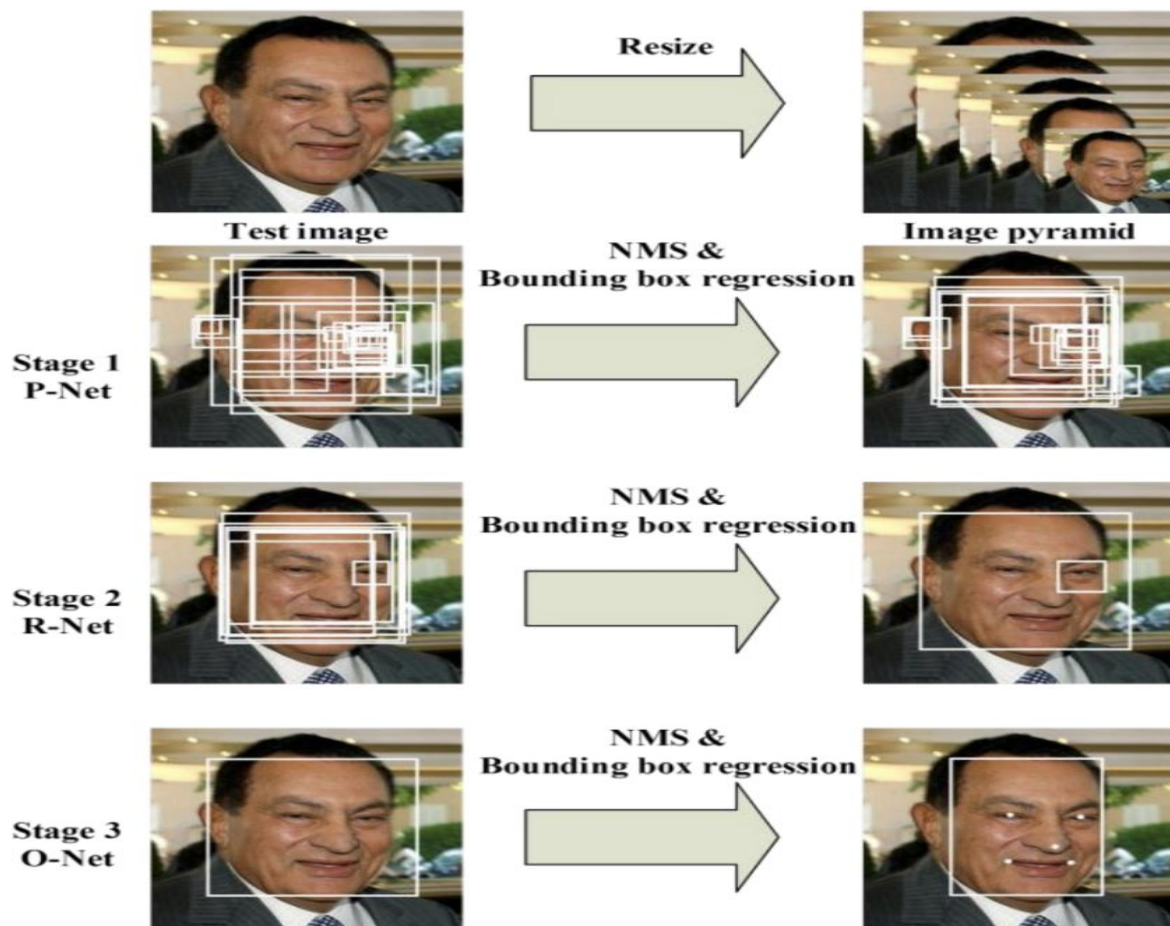
## 7.2 MTCNN

MTCNN stands for “Multi-Task-Cascaded-Convolutional-Neural-Network”. The intramural structure of MTCNN is a nested-cascade of three different networks. First, an image pyramid is created by rescaling the picture input. Then Proposal Network identifies facial areas of the image if any, Then Refine Network identifies the bounding region and filters it out. Later Output Network identifies and outputs facial milestones and critical points.



There are three types of predictions in MTCNN. They are:

- Face Classification
- Bounding Box
- Facial Landmark Localization.

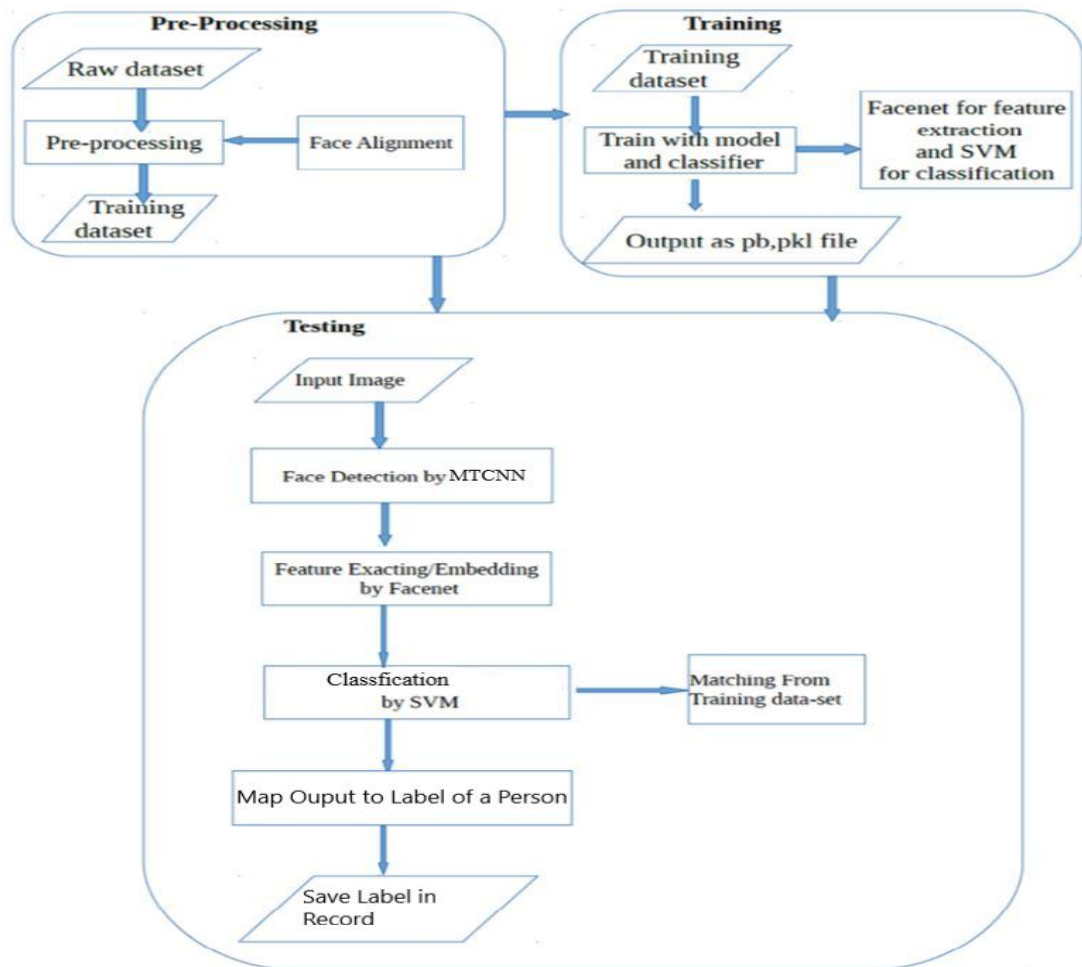


Additional Computations are done in the middle of different network layers. This technique is widely implemented in straining out bounding regions identified by Proposal Network before giving it as an input to Refine Network called NMS

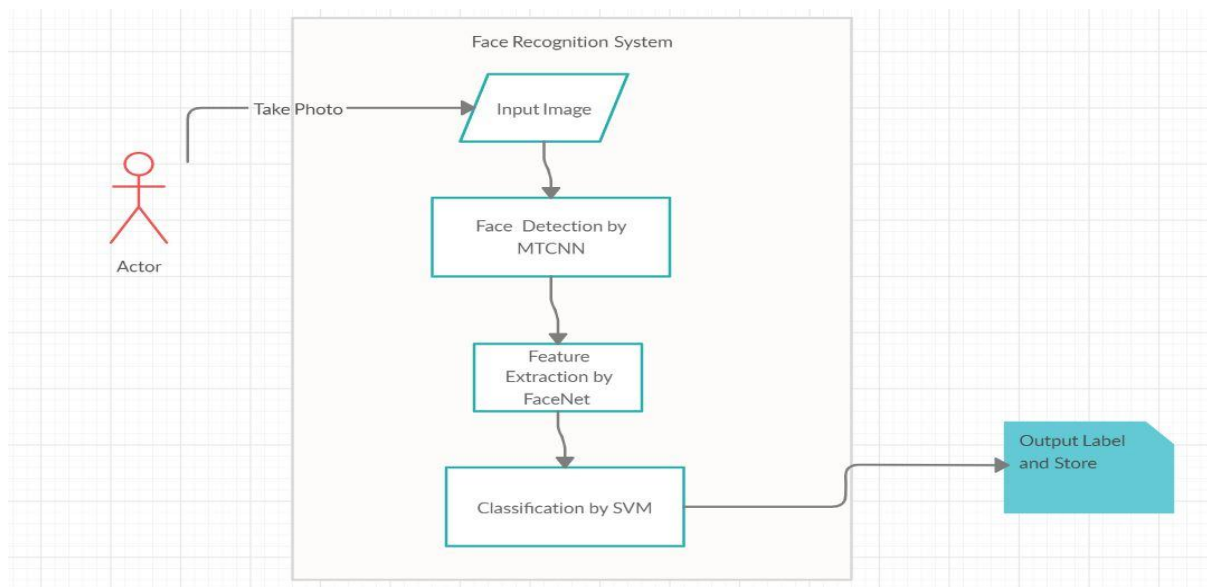
Building MTCNN from scratch is difficult and so we can utilize a package in 'Keras' by importing it in our program.

## 8. IMPLEMENTATION

### SYSTEM FLOW:



### UML DIAGRAM OF REAL-TIME TESTING:



## 8.1 Implementing on Kaggle Dataset

```
In [1]: !pip install mtcnn

Collecting mtcnn
  Downloading https://files.pythonhosted.org/packages/9e/c7/8546b18fbd367b156c5bbbaa8912ab31c8129171523ff8b47b546d70b09/mtcnn-0.0.9.tar.gz (2.3MB)
    |████████████████████| 2.3MB 3.5MB/s
Building wheels for collected packages: mtcnn
  Building wheel for mtcnn (setup.py) ... - - - - - done
  Created wheel for mtcnn: filename=mtcnn-0.0.9-cp36-none-any.whl size=2257690 sha256=198edaa24b616c449987272b442b61f654088b49cde8933be6c88d3d4813d94
  Stored in directory: /tmp/.cache/pip/wheels/85/81/65/6363fa5aafd7a155c896591e0c7c6e27b69642aa82b9cbf076
Successfully built mtcnn
Installing collected packages: mtcnn
Successfully installed mtcnn-0.0.9
```

- Installing MTCNN

```
In [2]: import mtcnn
        print(mtcnn.__version__)
```

Using TensorFlow backend.

0.1.0

```
In [3]: import numpy as np
        import pandas as pd
        import cv2
        from mtcnn.mtcnn import MTCNN
        from matplotlib import pyplot as plt
        from keras.models import load_model
        from PIL import Image

        import os
        print(os.listdir("../input"))
```

['facenet-keras', '5-celebrity-faces-dataset']

- Importing Modules and printing Input Data.

```
In [4]: img = cv2.imread('../input/5-celebrity-faces-dataset/data/train/ben_afflek/httpcsvkmeuaecc.jpg')
plt.imshow(img)
plt.show()
print(img.shape)
```



(170, 115, 3)

- Here we took one image from the dataset and read that image using the `imread()` function of CV2 module and printed that image and the shape of the image along with the respected dimensions.

```
In [5]: def extract_face(filename, required_size=(160, 160)):
        image = Image.open(filename)
        image = image.convert('RGB')
        pixels = np.asarray(image)
        detector = MTCNN()
        results = detector.detect_faces(pixels)
        x1, y1, width, height = results[0]['box']
        x1, y1 = abs(x1), abs(y1)
        x2, y2 = x1 + width, y1 + height
        face = pixels[y1:y2, x1:x2]
        image = Image.fromarray(face)
        image = image.resize(required_size)
        face_array = np.asarray(image)
        return face_array
```

- Defining Function that opens an image, converts it into RGB format .We apply the mtcnn detector that detects the pixels around a face.



- We extract those pixels and convert to array and return it. Therefore we extract faces from a given image.

```
pixels = extract_face('../input/5-celebrity-faces-dataset/data/train/ben_affle  
k/httpcsvkmeuaeccjpg.jpg')  
plt.imshow(pixels)  
plt.show()  
print(pixels.shape)
```



(160, 160, 3)

- Applying 'extract\_face' function to a single image and plotting it.



```
In [6]:
def load_face(dir):
    faces = list()
    for filename in os.listdir(dir):
        path = dir + filename
        face = extract_face(path)
        faces.append(face)
    return faces

def load_dataset(dir):
    X, y = list(), list()
    for subdir in os.listdir(dir):
        path = dir + subdir + '/'
        faces = load_face(path)
        labels = [subdir for i in range(len(faces))]
        print("loaded %d sample for class: %s" % (len(faces),subdir) )
        X.extend(faces)
        y.extend(labels)
    return np.asarray(X), np.asarray(y)
```

- Defining functions `load_dataset` and `load_image` that process through each image in Dataset and apply `extract_face` function on each and return a list of results of only the faces of images.
- We calculate the path of the image and put that path in the `extract_face()`.Continue this process for all the faces by appending the faces using `append()`.
- Then create two lists, one is for faces and another is for labels. Name those lists as `X` and `y`. We return those as arrays.

```

trainX, trainy = load_dataset('../input/5-celebrity-faces-dataset/data/train/')
print(trainX.shape, trainy.shape)
testX, testy = load_dataset('../input/5-celebrity-faces-dataset/data/val/')
print(testX.shape, testy.shape)

np.savez_compressed('5-celebrity-faces-dataset.npz', trainX, trainy, testX, testy)

```

```

loaded 19 sample for class: madonna
loaded 14 sample for class: ben_afflek
loaded 22 sample for class: mindy_kaling
loaded 21 sample for class: jerry_seinfeld
loaded 17 sample for class: elton_john
(93, 160, 160, 3) (93,)
loaded 5 sample for class: madonna
loaded 5 sample for class: ben_afflek
loaded 5 sample for class: mindy_kaling
loaded 5 sample for class: jerry_seinfeld
loaded 5 sample for class: elton_john
(25, 160, 160, 3) (25,)

```

- We apply defining functions on Train Dataset and Test Dataset and store the arrays in TrainX, trainy and TestX, testy respectively.
- We save those results in a compressed file '5-celebrity-faces-dataset.npz'

In [7]:

```

data = np.load('5-celebrity-faces-dataset.npz')
trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)

```

```

Loaded: (93, 160, 160, 3) (93,) (25, 160, 160, 3) (25,)

```

- Load the compressed file and print the shapes of all variables.

```
In [8]: facenet_model = load_model('../input/facenet-keras/facenet_keras.h5')
print('Loaded Model')
```

Loaded Model

```
/opt/conda/lib/python3.6/site-packages/keras/engine/saving.py:341: UserWarning: No training configuration found in save file: the model was *not* compiled. Compile it manually.
warnings.warn('No training configuration found in save file: '
```

- Loading FaceNet Model.

```
In [9]: def get_embedding(model, face):
        face = face.astype('float32')
        mean, std = face.mean(), face.std()
        face = (face-mean)/std
        sample = np.expand_dims(face, axis=0)
        yhat = model.predict(sample)
        return yhat[0]
```

- Defining function to compute the face embeddings for all faces in train and test.
- We convert the pixels into float and standardize them
- We expand dimensions to make a single image as an array type and input to the facenet model
- Then FaceNet will extract top quality attributes from the facial image and determine a 128 element vector of these features.
- This feature is called Face Embedding
- We return that embedding.

```

emdTrainX = list()
for face in trainX:
    emd = get_embedding(facenet_model, face)
    emdTrainX.append(emd)

emdTrainX = np.asarray(emdTrainX)
print(emdTrainX.shape)

emdTestX = list()
for face in testX:
    emd = get_embedding(facenet_model, face)
    emdTestX.append(emd)
emdTestX = np.asarray(emdTestX)
print(emdTestX.shape)

np.savez_compressed('5-celebrity-faces-embeddings.npz', emdTrainX, trainy, emdTestX, testy)

```

```

(93, 128)
(25, 128)

```

- We apply the `get_embedding` function to all images in the Train and Test dataset.
- We append the results to a list `emdTrainX` and `emdTestX`
- We convert them into arrays and print the shape.
- We save those results in a compressed file '5-celebrity-faces-embeddings.npz'

```

In [10]:
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Normalizer
from sklearn.svm import SVC

```

- We import modules for classification.

```

print("Dataset: train=%d, test=%d" % (emdTrainX.shape[0], emdTestX.shape[0]))
in_encoder = Normalizer()
emdTrainX_norm = in_encoder.transform(emdTrainX)
emdTestX_norm = in_encoder.transform(emdTestX)
out_encoder = LabelEncoder()
out_encoder.fit(trainy)
trainy_enc = out_encoder.transform(trainy)
testy_enc = out_encoder.transform(testy)

```

- We normalize emdTrainX and emdTestX for fast and easy computation.
- We fit Label-encoder to trainy.
- We transform trainy and testy using the fitted Label-encoder.

```

model = SVC(kernel='linear', probability=True)
model.fit(emdTrainX_norm, trainy_enc)
yhat_train = model.predict(emdTrainX_norm)
yhat_test = model.predict(emdTestX_norm)
score_train = accuracy_score(trainy_enc, yhat_train)
score_test = accuracy_score(testy_enc, yhat_test)
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100))

```

```

Dataset: train=93, test=25
Accuracy: train=100.000, test=100.000

```

- We use Linear Support Vector Machine model to fit emdTrainX\_norm and trainy\_enc.
- We use the trained model to predict emdTrainX\_norm and emdTest\_norm.
- We calculate accuracy score against yhat\_train and yhat\_test and print score.

## 8.2 Implementing on Created Dataset

- As we did earlier we install MTCNN.
- Import all the modules.

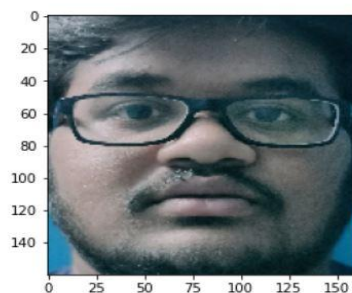
```
In [4]: img = cv2.imread('../input/facerecog/Train/Rahul/IMG20200326211641.jpg')
plt.imshow(img)
plt.show()
print(img.shape)
```



(4608, 3456, 3)

- Taking one image from the dataset, read it using CV2 and plot it.
- We then define the 'extract\_face' function as earlier.

```
In [7]: pixels = extract_face('../input/facerecog/Train/Rahul/IMG20200326211641.jpg')
plt.imshow(pixels)
plt.show()
print(pixels.shape)
```



(160, 160, 3)

- Applying extract\_face on a single face in the dataset and plotting it.

- We define load\_dataset and load\_face functions.

```
In [9]: trainX, trainy = load_dataset('../input/facerecog/Train/')
print(trainX.shape, trainy.shape)
testX, testy = load_dataset('../input/facerecog/Val/')
print(testX.shape, testy.shape)

np.savez_compressed('Faces-dataset.npz', trainX, trainy, testX, testy)
```

```
loaded 19 sample for class: Nikitha
loaded 20 sample for class: Rahul
(39, 160, 160, 3) (39,)
loaded 5 sample for class: Nikitha
loaded 5 sample for class: Rahul
(10, 160, 160, 3) (10,)
```

- We apply load\_dataset function on Train Dataset and test Dataset.
- We save the result in a compressed file 'Faces-dataset.npz'
- We load Facenet Model.
- We create the get\_embedding function as earlier and apply it on train and test and save in emdTrainX and emdTestX.
- We import modules for classification.
- We normalize emdTrainX and emdTestX and save results in emdTrainX\_norm and emdTestX\_norm.
- We apply Label encoding to trainy and testy.
- Fit Linear SVM model and predict result and print the accuracy.

```
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100))
```

```
Dataset: train=39, test=10
Accuracy: train=100.000, test=100.000
```

## 9. TESTING

### 9.1 Testing on Kaggle Dataset

```
In [11]: from random import choice
selection = choice([i for i in range(testX.shape[0])])
random_face = testX[selection]
random_face_emd = emdTestX_norm[selection]
random_face_class = testy_enc[selection]
random_face_name = out_encoder.inverse_transform([random_face_class])

samples = np.expand_dims(random_face_emd, axis=0)
yhat_class = model.predict(samples)
yhat_prob = model.predict_proba(samples)

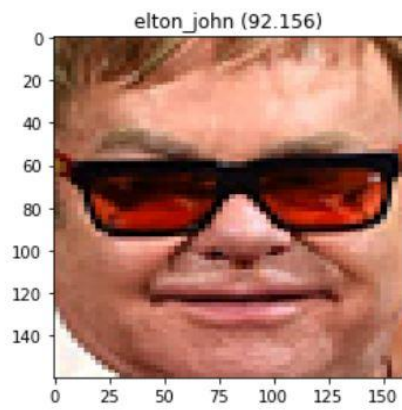
class_index = yhat_class[0]
class_probability = yhat_prob[0,class_index] * 100
predict_names = out_encoder.inverse_transform(yhat_class)
all_names = out_encoder.inverse_transform([0,1,2,3,4])
print('Predicted: %s (%.3f)' % (predict_names[0], class_probability))
print('Predicted: \n%s \n%s' % (all_names, yhat_prob[0]*100))
print('Expected: %s' % random_face_name[0])

plt.imshow(random_face)
title = '%s (%.3f)' % (predict_names[0], class_probability)
plt.title(title)
plt.show()
```

- We select a random image, get its embedding, class and name.
- We expand dimensions and predict output with the model.
- We calculate the probability against each class and plot it.



```
Predicted: elton_john (92.156)
Predicted:
['ben_afflek' 'elton_john' 'jerry_seinfeld' 'madonna' 'mindy_kaling']
[ 0.92100774 92.15561384  0.9672091   1.44549435  4.51067497]
Expected: elton_john
```



## 9.2 Testing on Created Dataset

```
In [14]: from random import choice
selection = choice([i for i in range(testX.shape[0])])
random_face = testX[selection]
random_face_emd = emdTestX[selection]
random_face_class = testy[selection]
random_face_name = [random_face_class]

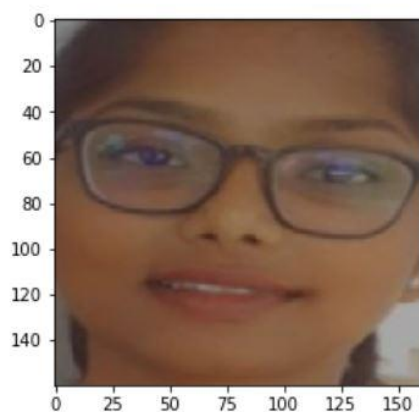
samples = np.expand_dims(random_face_emd, axis=0)
yhat_class = model.predict(samples)
yhat_prob = model.predict_proba(samples)

class_index = yhat_class[0]
class_probability = yhat_prob[0]* 100
predict_names = yhat_class
all_names = [0,1]

print('Predicted: \n%s \n%s' % (all_names, yhat_prob[0]*100))
print('Expected: %s' % random_face_name[0])

plt.imshow(random_face)
plt.show()
```

```
Predicted:
[0, 1]
[95.84154262  4.15845738]
Expected: Nikitha
```



## 9.3 Real-Time Testing

```
In [15]: import pickle
```

```
In [16]: filename = 'Face_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

- We save the model using pickle.
- We upload the Model to google drive.

```
[ ] !pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

```
[ ] auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
[ ] file_obj = drive.CreateFile({'id': '1X7Lf0TxcKGAeu8eWw7MSfwBK4v_A-ow2'})
file_obj.GetContentFile('Face_model_Simple.sav')
```

```
[ ] filename = 'Face_model_Simple.sav'
```

```
[ ] import pickle
model = pickle.load(open(filename, 'rb'))
```

- We load the model from Drive after authentication.

```
from IPython.display import HTML, Audio
from google.colab.output import eval_js
from base64 import b64decode
import numpy as np
import io
from PIL import Image
```

- We import modules to capture image.

```
def take_photo(filename='photo.jpg', quality=1.0, size=(800,600)):
    display(HTML(VIDEO_HTML % (size[0],size[1],quality)))
    data = eval_js("data")
    binary = b64decode(data.split(',')[1])
    f = io.BytesIO(binary)
    return Image.open(f)
```

- We define a function given by Colab to capture an image.

```
[8] img = take_photo() # click
```



- We capture the image from the laptop camera.

```
[9] import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))
plt.imshow(np.asarray(img))
plt.show()
```



- We plot the image

```
[19] pixels = extract_face(img)
plt.imshow(pixels)
plt.show()
print(pixels.shape)
```

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:66: The name tf.get\_default\_graph is deprecated. Please use tf.compat.v1.get\_default\_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:4432: The name tf.random\_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:4267: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:190: The name tf.get\_default\_session is deprecated. Please use tf.compat.v1.get\_default\_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:197: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.



- We apply extract\_face function and plot the image.

```
[27] yhat_class = model.predict(emdTrainX.reshape(1, -1))
yhat_prob = model.predict_proba(emdTrainX.reshape(1, -1))
```

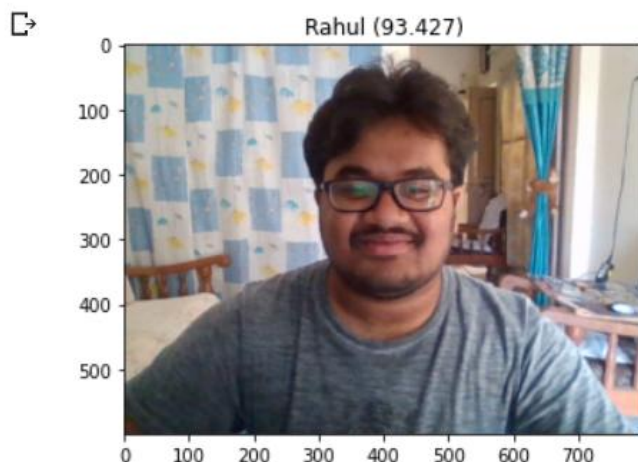
```
[28] class_index = yhat_class[0]
class_probability = yhat_prob * 100
```

```
[29] predicted = {'Nikitha':0, 'Rahul':1}
Prob = int(predicted[class_index])
```

```
[30] print('Predicted: %s (%.3f)' % (class_index, class_probability[0][Prob]))
```

➤ Predicted: Rahul (93.427)

```
▶ plt.imshow(img)
title = '%s (%.3f)' % (class_index, class_probability[0][Prob])
plt.title(title)
plt.show()
```



- We apply the get\_embedding function and predict it. Calculate the probability of for each class and plot the final output image along with predicted class and probability.



## **10. CONCLUSION & FUTURE SCOPE**

Face-Recognition is continuously evolving technology that can give numerous advantages. Resources and time can be saved using Face-Recognition technology, many companies and firms are taking advantage of this new technology to improve their product in many ways and make it more efficient. Many solutions have been deployed that generate income. Today, machines can consequently validate character identification details for safe transfers, for reconnaissance and protection assignments, and validation of access-control in systems, etc.

We built a face detection using FaceNet and SVM classifier to identify people from an image. The system is highly efficient and accurate. We can substitute any system which requires human intervention with an automatic system. It would save money, cut back on the volume of documentation the management has to do.

Experts in the AI field predict that the future of Face-Recognition technologies could provide more sophisticated uses in smart settings where robots and devices become almost like supportive helpers.

## 11. REFERENCES

- [1] Yicheng An, Jiafu Wu, Chang Yue, “CNNs for Face Detection and Recognition”, 2017.
- [2] Dr. Priya Gupta, Nidhi Saxena, Meetika Sharma, Jagriti Tripathi, “Deep Neural Network for Human Face Recognition”, 2018.
- [3] Florian Schroff, Dmitry Kalenichenko , James Philbin “FaceNet: A Unified Embedding for Face Recognition and Clustering” , CVPR 2015.
- [4] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”, 2016.
- [5] Thida Nyein, Aung Nway Oo “University Classroom Attendance System Using FaceNet and Support Vector Machine”, 2019.
- [6] <https://towardsdatascience.com/mtcnm-face-detection-cdcb20448ce0>
- [7] <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>
- [8] <https://towardsdatascience.com/s01e01-3eb397d458d>
- [9] <https://towardsdatascience.com/a-facenet-style-approach-to-facial-recognition-dc0944efe8d1>