Face Recognition: A Comparative Study of Deep Models

By

Mohammad Sakhawat Hossain

Roll No: 2014123022

Supervisor:

Md. Zainal Abedin

Assistant Professor, CSE

SUST

Department of Computer Science and Engineering

Faculty of Science, Engineering and Technology

Shahjalal University of Science and Technology

Sylhet, Bangladesh

September 2017

Face Recognition: A Comparative Study of Deep Models

By

Mohammad Sakhawat Hossain

Roll No: 2014123022

A Project Report Submitted in Partial Fulfillment of the Requirements of the Shahjalal University of Science and Technology for the Degree of Master of Science in Mathematics

Supervisor:

Md. Zainal Abedin

Assistant Professor, CSE

SUST

Department of Computer Science and Engineering

Faculty of Science, Engineering and Technology

Shahjalal University of Science and Technology

Sylhet, Bangladesh

September 2017

THIS PAGE IS INTENTIONALLY LEFT BLANK

Abstract

Face is the front part of the head in humans extends from the forehead to the chin and includes the mouth, nose, cheeks, eyes etc. Face recognition is the identification of a face from an image source or a video or even in real time in the study of computer vision. It becomes very popular and has wide range of applications by the development of computing power.

The aim of this study is to recognize faces with deep models. As, the ability to recognize faces is so important in human that the brain appears to have an area solely devoted to the task. Here, we studied three deep models in terms of face recognition. These are: DeepFace, FaceNet & OpenFace. And we learn Haar Cascade features and compact Euclidean space to measure the distance. From distance we calculate the confidence and thus the accuracy.

In this study, the models tested on both custom dataset and open-source datasets-generated images and database files by us. For DeepFace it is VGGFace2 and for FaceNet & OpenFace it is Labeled faces of wild (LFW). Which results shows us the computational efficiency and performance analysis. In our comparative study on deep models – DeepFace achieved 99% accuracy, FaceNet achieved 90% accuracy and OpenFace achieved 90% accuracy. Thus, the best result that have achieved in DeepFace model and the model result is 99% accuracy.

Keywords

Face Recognition, Deep Neural Network, Convolutional Neutral Network, Database, Methods.

Declaration

This certifying that the project titled “Face Recognition: A Comparative Study of Deep Models” is the result of my study in partial fulfillment of the M.Sc. degree under the supervision of **Md. Zainal Abedin**, Assistant Professor, Faculty of Science, Engineering and Technology, Shahjalal University of Science & Technology (SUST) and it has not submitted elsewhere for any other degree or diploma.

Signature of the Students

_____

Mohammad Sakhawat Hossain

_____

Signature of the Supervisor

_____

Md. Zainal Abedin

Assistant Professor

Faculty of Science, Engineering & Technology

Shahjalal University of Science & Technology

Approval Certificate

The project entitled “Face Recognition: A Comparative Study of Deep Models” submitted by **Mohammad Sakhawat Hossain** (Roll No: 201423022) has been accepted as satisfactory in partial fulfillment of the requirements for the degree of M.Sc. in Mathematics on September, 2017.

Board of Examination Committee

Md. Zainal Abedin

Assistant Professor

Faculty of Science, Engineering and Technology

Shahjalal University of Science & Technology

Most. Tahamina Khatoon

Senior Lecturer

Faculty of Science, Engineering and Technology

Shahjalal University of Science & Technology

Mrs. Sharmin Akter

Lecturer

Faculty of Science, Engineering and Technology

Shahjalal University of Science & Technology

Israth Jahan Chowdhury

Lecturer

Faculty of Science, Engineering and Technology

Shahjalal University of Science & Technology

Acknowledgement

First, I would like to give thanks to almighty for kindness for which I have completed my project successfully. I would also like to express our sincere gratitude to a number people who has helped me in course of this work. I would like to offer thanks and deep gratitude to my honorable Supervisor **Md. Zainal Abedin** Assistant professor, department of Computer Science & Engineering. He gave me suggestions, ideas and ad-vice during the development of this project. He helped me to collect necessary data. The project has been completed under his kind supervision. In addition, I would like to give thanks to all other respectable teachers who were always with me to cheer up and made me believe in ourselves.

Finally, I would like to express my thanks to all of my well-wishers and all who has contributed in any way to complete my project.

This Project is Dedicated

To

My Parents

Table of Contents

List of Figures

Chapter 1
INTRODUCTION

1.1. Introduction

In recent years, mainly due to the advances of deep learning, more concretely convolutional networks, the quality of face recognition has been progressing at a dramatic pace.

Face Recognition (FR) is the identification of a face from an image source or videos or even in real time in the study of computer vision. It becomes very popular and has wide range of applications by the development of computing power.

In terms of face-recognition the term identification derives, a test face is compared with a set of faces aiming to find the most likely match.

And computer vision is concerned with the theory behind Artificial systems that extract information from images.

1.2. Overview

Face recognition is the science which involves the understanding of how the faces are recognized by biological systems and how this can be emulated by computer systems. Similarly, computer systems employ different visual devices to capture and process faces as best indicated by each particular application.

1.3. Motivation

Face Recognition (FR) is a big challenge in Deep Learning field. It (FR) has become a hot topic in the research/recent fields. As, a facial recognition system is a technology capable of identifying a person from a digital image or a video frame from a video source and it is used as access control in security systems.

1.4. Objective

- Detect face from an image or video.
- Recognize face using Deep Models.
- Performance analysis.

Chapter 2
LITERATURE REVIEW

2.1. Literature Review

The following section contains literature review of the related works done previously for face recognition. Some of the studies are reviewed here.

In [1] this survey, they provide a comprehensive review of the recent developments on deep Face Recognition (FR), covering broad topics on algorithm designs, databases, protocols, and application scenes. First, they summarize different network architectures and loss functions proposed in the rapid evolution of the deep Face Recognition (FR) methods. Second, the related face processing methods are categorized into two classes: “one-to-many augmentation” and “many-to-one normalization”. Then, they summarize and compare the commonly used databases for both model training and evaluation. Third, they review miscellaneous scenes in deep FR, such as cross-factor, heterogenous, multiple-media and industrial scenes. Finally, the technical challenges and several promising directions are highlighted.

Sun et al. [4] Despite concerns that, in terms of DeepaFace max-pooling layers result in loss of accurate spatial information, the same convolutional network architecture as has also been successfully employed for recognition.

Similarly, in terms of FaceNet and OpenFace the inception layers are repeated many times, leading to a deep model.

There is a vast corpus of face verification and recognition works. Reviewing it is out of the scope of this project so we will only briefly discuss the most relevant recent work.

The study by [5], they collected a new large-scale dataset, VGGFace2, for public release. It includes over nine thousand identities with between 80 and 800 images for each identity, and more than 3M images in total.

And In [6] this section, they review the principle "in the wild" datasets that have appeared recently. In 2007, the Labelled Faces in the Wild (LFW) dataset was released, containing 5,749 identities with 13,000 images.

Zhenyao et al. [8] employ a deep network to "warp" faces into a canonical frontal view and then learn CNN that classifies each face as belonging to a known identity.

In the works of [5,7,9] starting with convolutional neural network (CNN), have typically had a standard structure- stacked convolutional

layers (optionally followed by contrast normalization and maxpooling) are followed by one or more fully-connected layers. Variant of this design are prevalent in the face recognition and have yielded the best result.

In [12] a comparison on public available databases that are at vital importance for both model training and testing. Major Face Recognition (FR) benchmarks, such as LFW [4] VGGFace2 [3] are reviewed and compared.

Chapter 3
PROBLEM-BASED LEARNING

3.1. Deep Neural Network (DNN)

Deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modeling to process data in complex ways. Generally, a Deep neural network is a technology built to simulate the activity of the human brain-specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between.

Deep neural networks (DNN) are able to learn the features that optimally represent the given training data. And, the object of Deep neural network (DNN) approach is to solve problems in the same way that a human brain would.

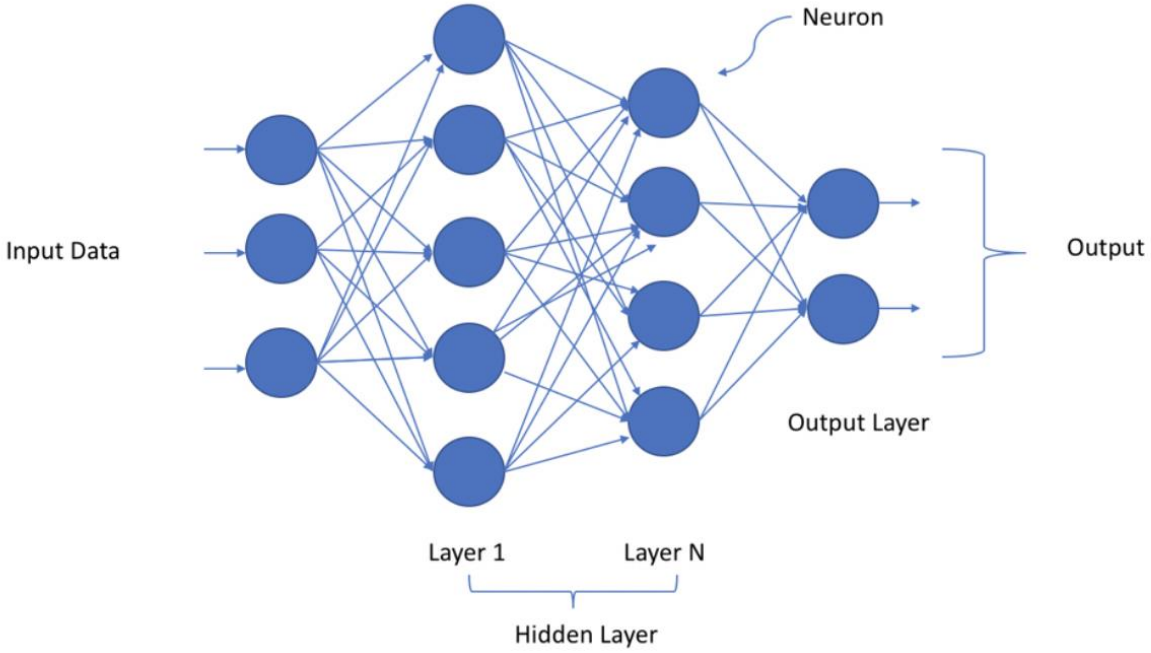

Figure 1: Deep Neural Network

3.2. Convolutional Neural Network (CNN)

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

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a RELU Layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

A convolutional layer within a neural network should have the following attributes:

- Convolutional kernels defined by a width and height (hyper-parameters).
- The number of input channels and output channels (hyper-parameter).
- The depth of the Convolution filter (the input channels) must be equal to the number channels (depth) of the input feature map.

Convolutional layers convolve the input and pass its result to the next layer. Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer.

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

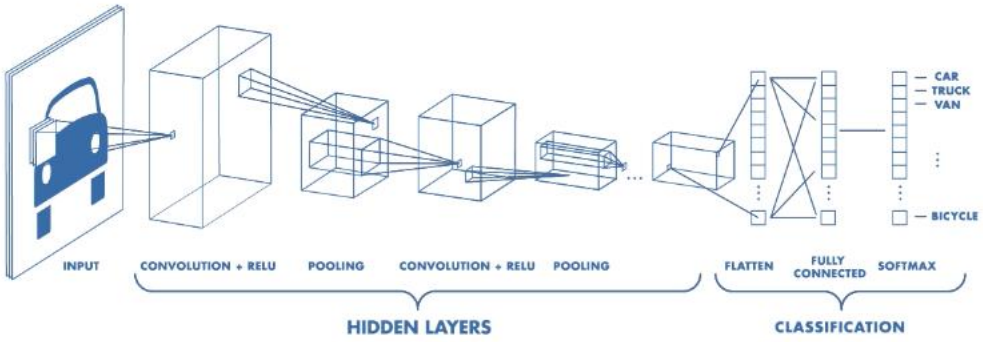

Figure 2: Convolutional Neural Network

3.3. Euclidean Distance

Euclidean distance is the straight lines distance between two data points in a plane.

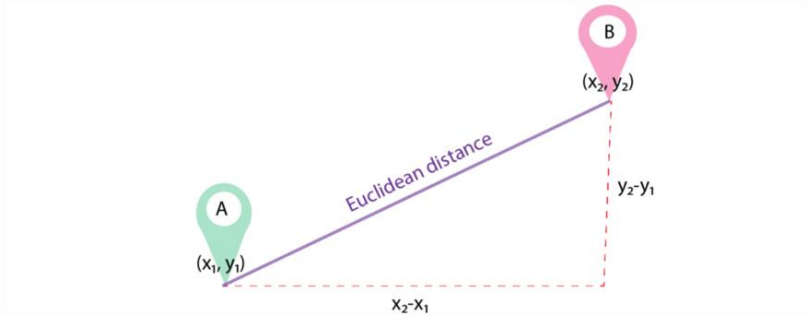

Figure 3: Euclidean Distance

It is calculated using the Minkowski Distance formula by setting 'p' value to 2, thus, also known as the L2 norm distance metric. The formula is:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

This formula is similar to the Pythagorean theorem formula. Thus, it is also known as the Pythagorean Theorem.

3.4. Inception Layer

Inception Layer is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.

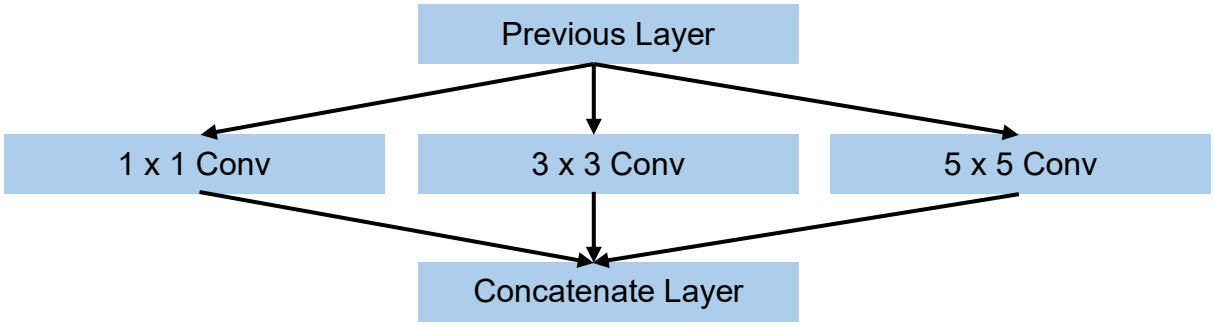

Figure 4: Inception Layer

3.5. Triplet Loss

FaceNet is a pre-trained CNN which embeds the input image into a 128-dimensional vector encoding. It is trained on several images of the face of different people. Although this model is pre-trained. But it still struggles to output usable encoding for unseen data. We want this model to generate encoding such that there is less distance between encoding of the images of same person, and more distance between encoding of the different persons. To achieve above goal on our own images we will train FaceNet model on Triplet Loss function. The triplet loss function takes face encoding of three images anchor, positive and negative. Here anchor and positive are the images of same person whereas negative is the image of a different person.

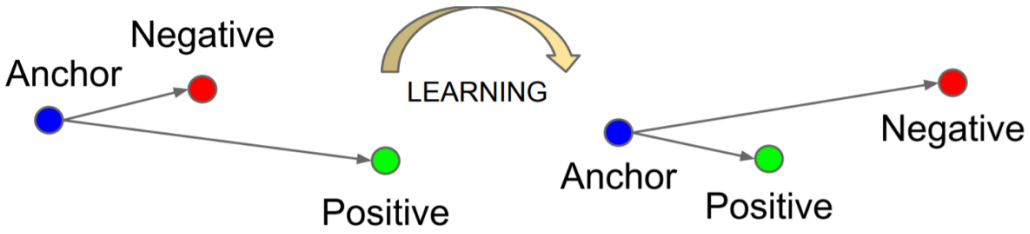

Figure 5: Triplet Loss

Formula of triplet loss:

$$Loss = \sum_{i=1}^{N} \left[\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right]_+$$

Chapter 4
EXPERIMENTAL METHODS

4.1. Training Methods

In the matter of our work, the training methods are:

I. Train with custom dataset: Expressly, It is a collection of documents from web of science that we demark.
II. Train with open-source dataset: In simple terms, this kind of data means, which is open for anyone and everyone for access, reuse, sharing etc.
III. Transfer Learning: In connection with our work, "Transfer Learning " is a method by which we can utilize the experience of pre-trained models to train them for newer and similar datasets.

4.2. Methodology

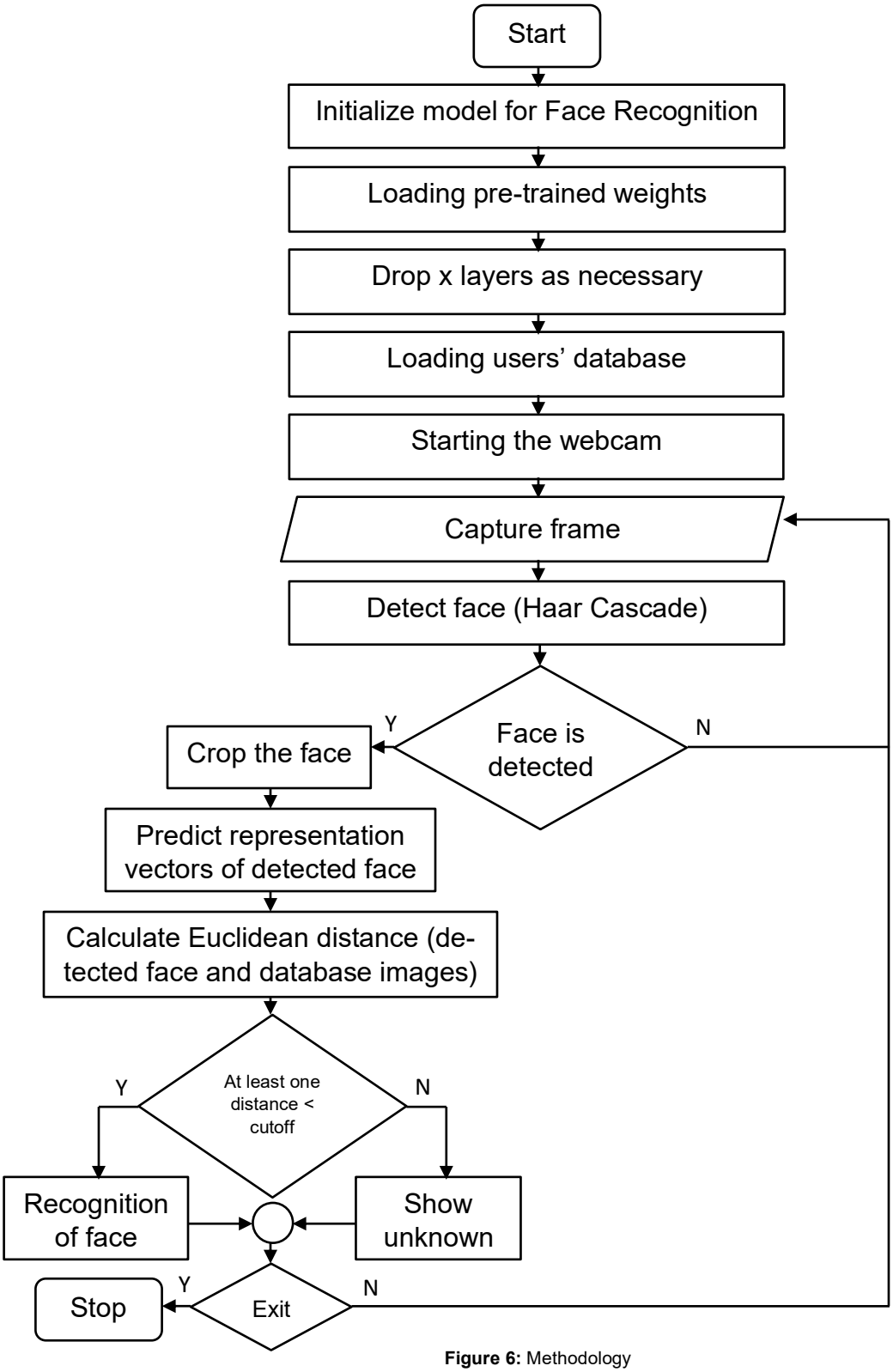

Figure 6: Methodology

4.3. Detection

Detection: Detection is the action or process of identifying the presence of something. Thus, Face Detection is the identify people's face within digital images.

The state or fact of being detected is the detection steps.

Here,

Step 1: Convert input RGB image to Grayscale

Step 2: Crop face from the frame.

Haar-Cascade: We did our detection use case by using "Haar-Cascade" Classifier. It (Haar-Cascade) is an object detection algorithm used to identify objects in an image or video. This is an approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

In this project, we tried to detect the face of individuals using the

haarcascade_frontalface_default.xml

Haar-Features: Haar like features are digital image features used in detection. Thus, this (Haar-Cascade) algorithm has four (04) stages:

I. Haar-feature selection: A Haar feature considers adjacent rectangular regions at a specific location is a detection, sums up the pixel intensities in each region and calculate the difference between these sums.
II. Integral Images: Integral images are used to make this superfast by calculating features.
III. Adaboost: Adaboost selects the best features and trains the classifiers that use them. Here a large number of 'Haar-Features' are necessary to describe an object with sufficient accuracy and are therefore organized into Cascade Classifiers.
IV. Cascade Classifier: Cascading Classifiers are trained with several hundred "positive " images and "negative " images of the same size.

When the classifiers are trained it can be applied to a region of an image and detect.

4.4. Recognition

Recognition is acknowledgement of the existence, validity, or legality of something.

A face recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. Facial Recognition system uses biometrics to map facial features from a photograph or video. It compares the information with a database of known faces to find a match. Facial recognition can help verify personal identity. As facial recognition is a way of recognizing human faces through technology. This system uses biometrics to map facial features from a photograph or video & compare the information with a database of known face to find a match.

Chapter 5
MODEL EVALUATION

5.1. Preface

Face recognition is the problem of identifying and verifying people in a photograph by their face. Face recognition is a process comprised of detection, alignment, feature extraction, and a recognition task. Deep learning models first approached then exceeded human performance for face recognition tasks.

Input: A layer for the input image data models.

Conv 2D: A 2D convolution layer means that the input of the convolution operation is three-dimensional. For example, a color image which has a value for each pixel across three layers: red, green and blue. However, it is called a “2D convolution” because the movement of the filter across the image happens in two dimensions. This layer convolves the input image with a set of (non) learnable filters, each producing one feature map in the output image.

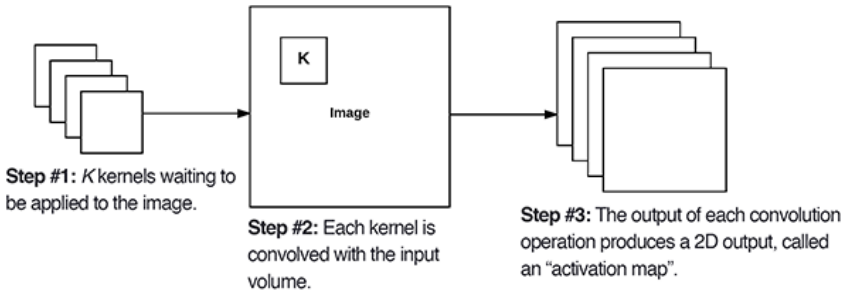

Figure 7: Convolutional 2D

Pooling 2D: Performs 2D pooling which is a form of non-linear down sampling. Its function is to progressively reduce the spatial size of the representation to reduce the amounts of parameters and computation in the network.

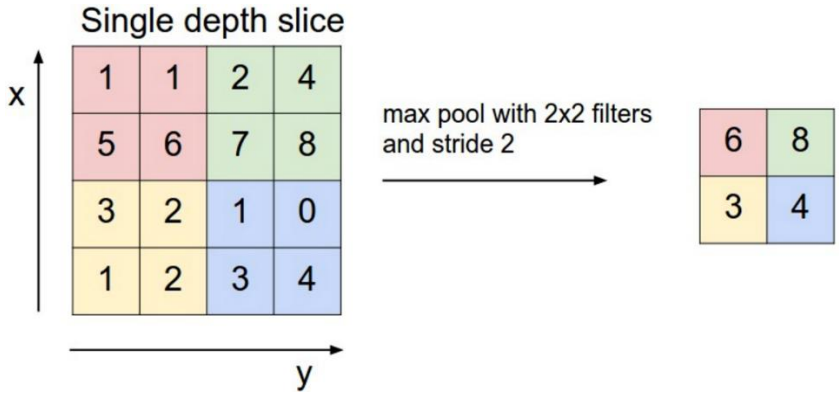

Figure 8: Pooling 2D

Locally-Connected: This type of layer is quite the same as the Convolutional layer explained in this post but with only one (important) difference. In the Convolutional layer the filter was common among all output neurons (pixels). In other words, we used a single filter to calculate all neurons (pixels) of an output channel. However, in Locally-Connected Layer each neuron (pixel) has its own filter. It means the number of parameters will be multiplied by the number of output neurons. It drastically could increase the number of parameters and if you do not have enough data, you might end up with an over-fitting issue. However, this type of layer let network to learn different types of feature for different regions of the input.

15

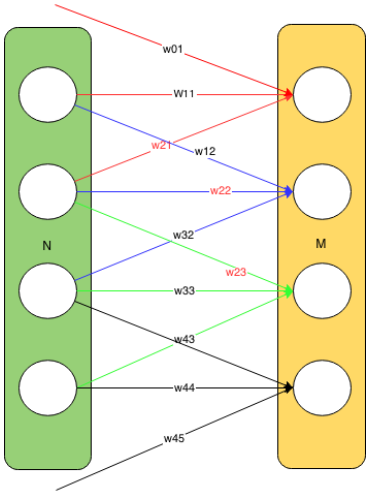

Figure 9: Locally-connected

Dropout: This layer randomly drops its inputs with a probability p. applies dropout to the input. Dropout consists in randomly setting a fraction p of input units to 0 at each update during training time, which helps prevent overfitting.

Flatten: Flattening is converting the data into 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector.

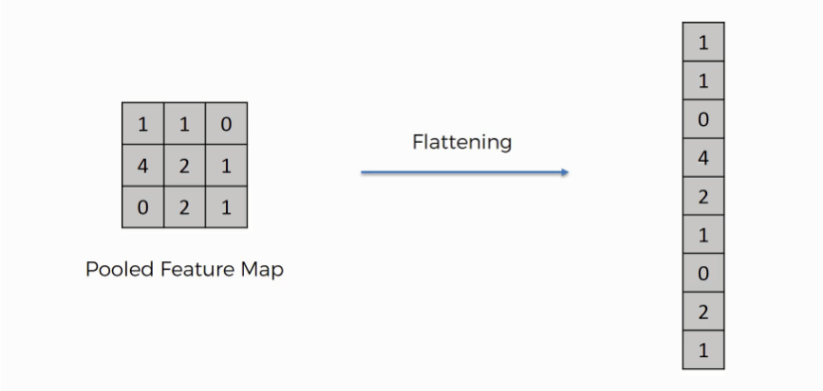

Figure 10: Flatten

Dense: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activation of previous layer a. A dense layer feeds all outputs from the previous layer to all its neuron, each neuron providing one output to the next layer.

ReLU (Rectified Linear Unit): A layer of rectified linear units, f(x)=max(0,x). Where x is the input of a neuron. Using the ReLU is that it does not activate all the neurons at the same time.

Softmax: The softmax activation is normally applies to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh or another activation function. The reason why softmax is useful is because it converts the output of the last layer in our neural network into what is essentially a probability distribution.

5.2. Models

5.2.1. DeepFace: DeepFace is a deep learning facial recognition system created by a research group at Facebook. It (DeepFace) identifies human faces in digital images. It (DeepFace) employs a nine layers neutral network with over 120 million connection weights and was trained on four million images uploaded by Facebook users. Facebook's DeepFace shows serious facial recognition skills.

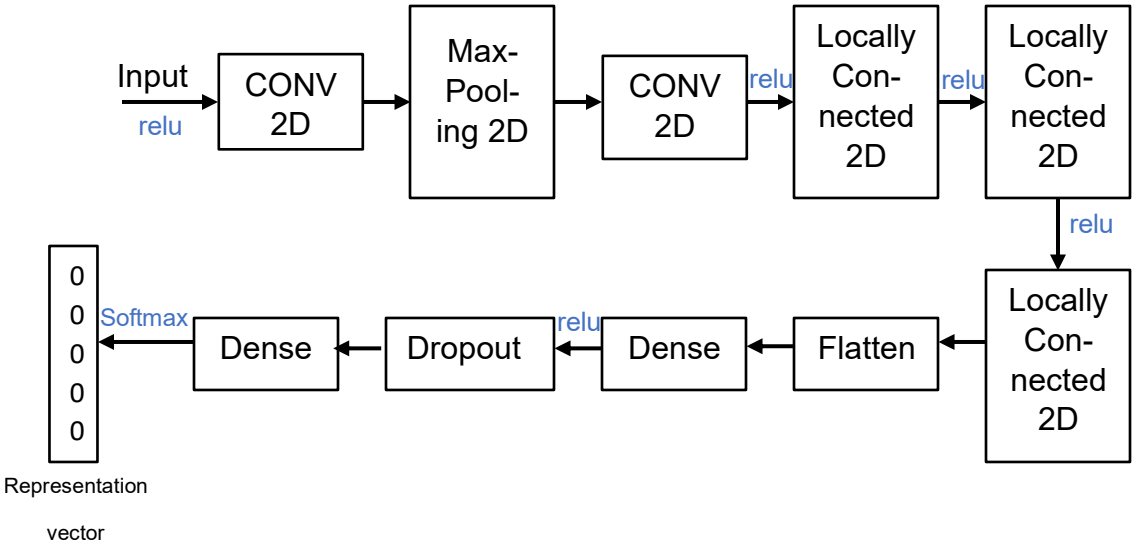

Figure 11: DeepFace model architecture

5.2.2. FaceNet: FaceNet is a start-of-art face recognition, verification and clustering neural network. It is 22-layers deep neural network. It was built on the Inception model. It is used for extracting features from an image of person's face. It was published in 2015 by Google researchers. It takes an image of the person's face as input and outputs a vector of 128 numbers which represents the most important features of a face. This vector is called embedding.

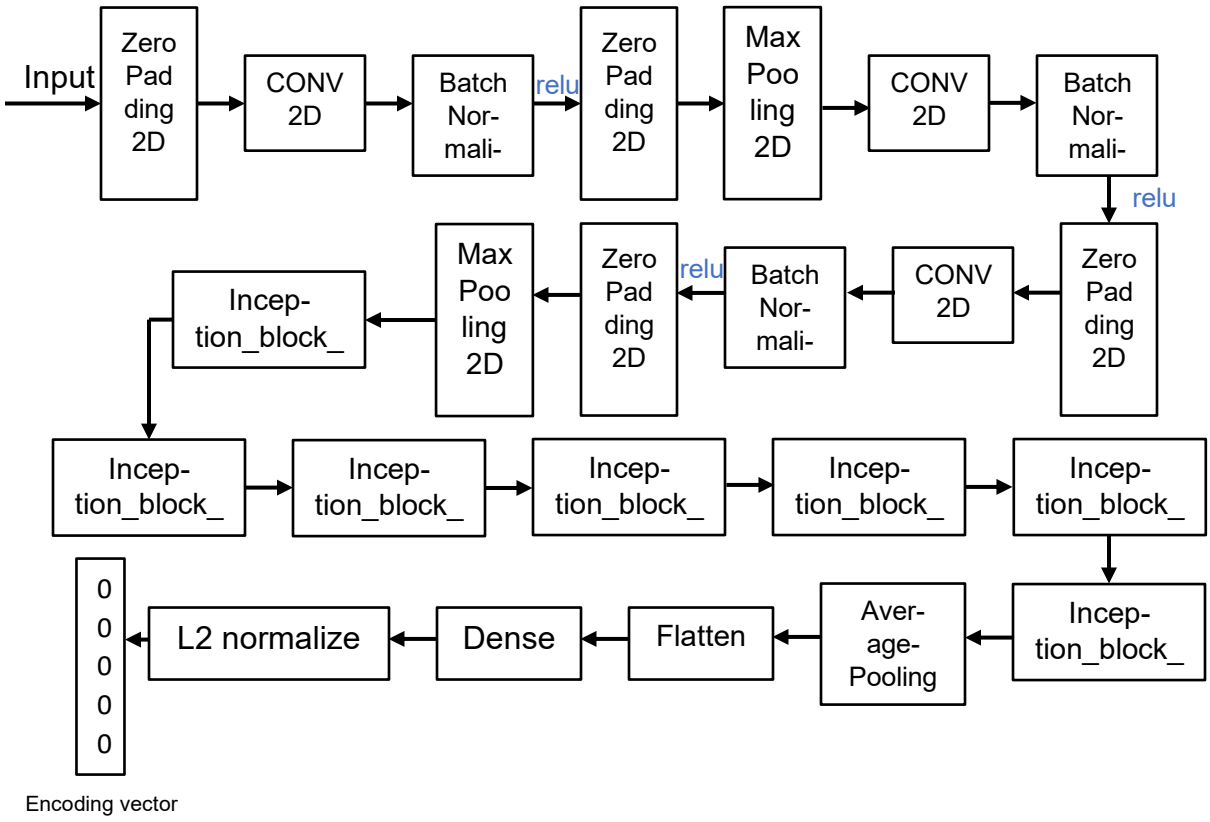

Figure 12: FaceNet model architecture

5.2.3. OpenFace: OpenFace is a source model intended for computer vision. It is a lightweight and minimalist model for face recognition. It gives 128-dimensional output. The model built on Inception. In the output layer, finds the vector representation as embedding and compares it with the representation of the database images that we calculate beforehand.

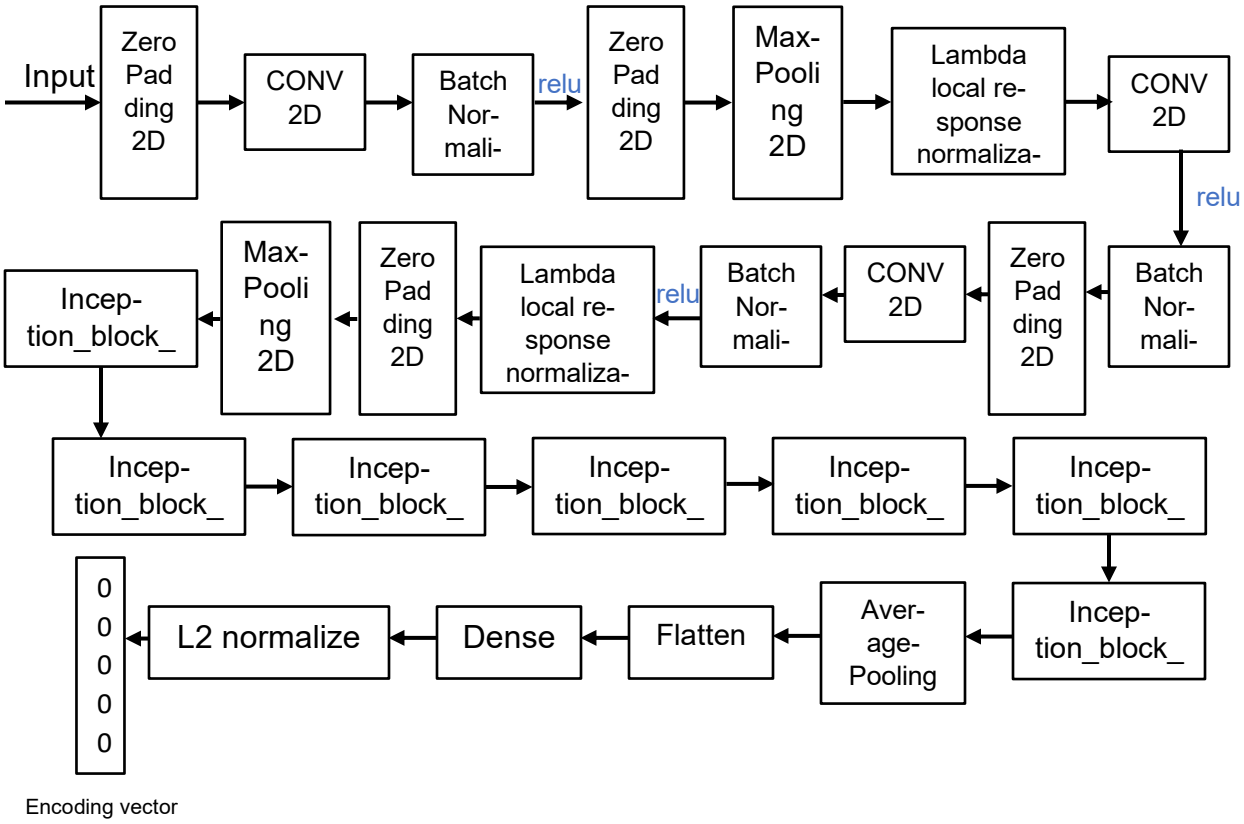

Figure 13: OpenFace model architecture

5.3. Result

DeepFace: Here, we try to find the faces in a video and labelled the known faces with names from our database. And if any face does not match with our database it labelled with unknown.

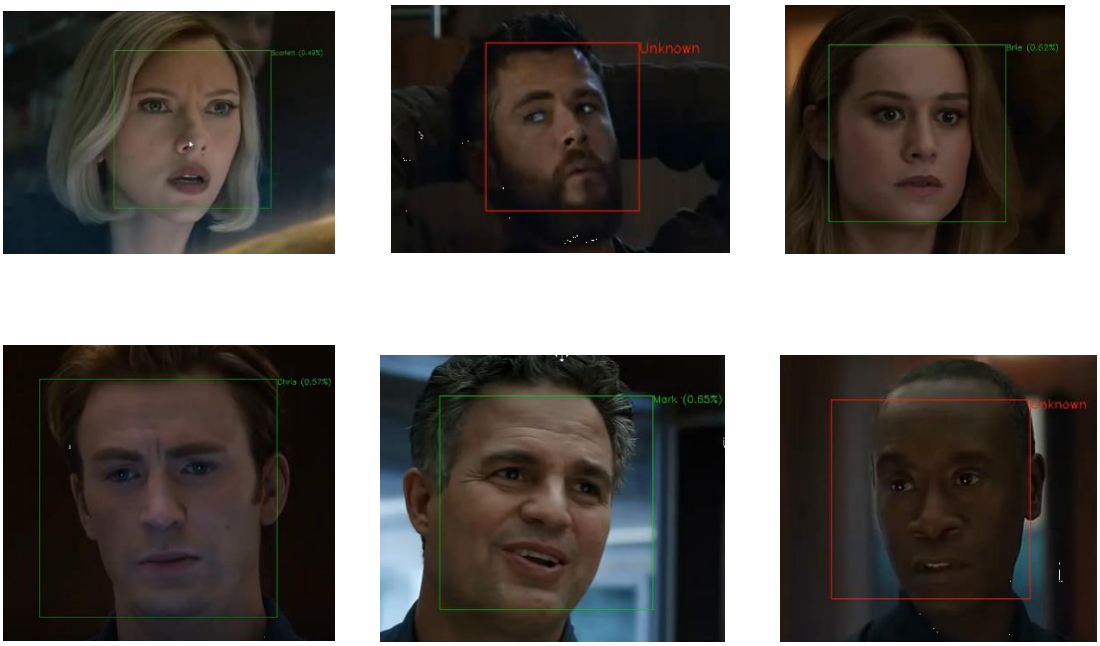

Figure 14: Result of DeepFace

FaceNet: Here, we try to find the faces in a video and labelled the known faces with names from our database. And if any face does not match with our database it labelled with unknown.

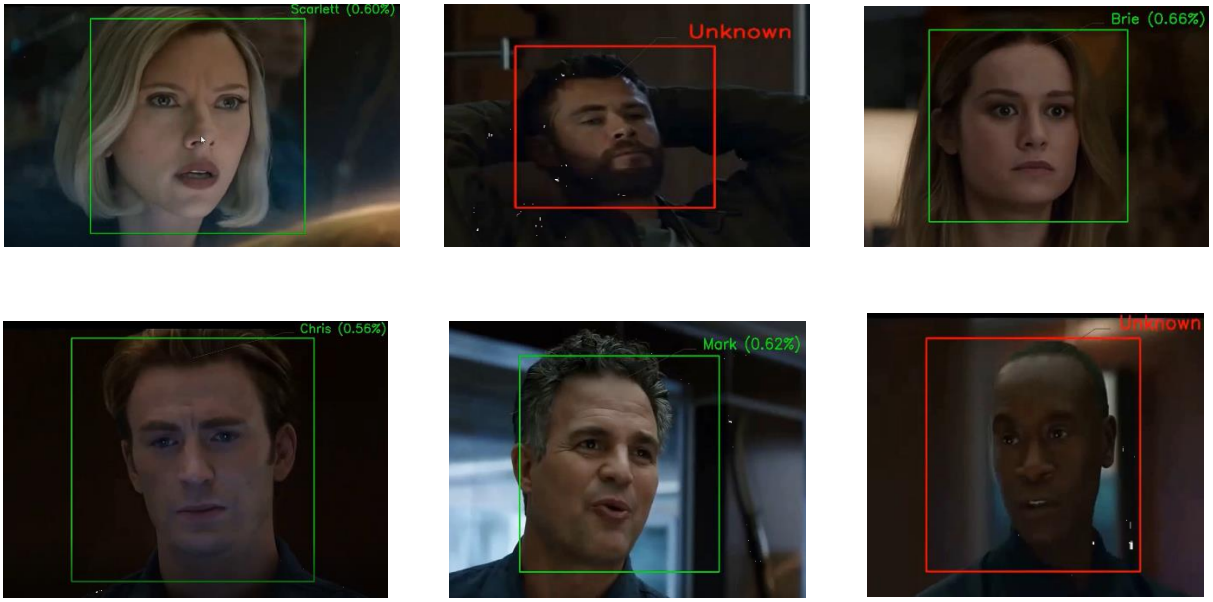

Figure 15: Result of FaceNet

OpenFace: Here, we try to find the faces in a video and labelled the known faces with names from our database. And if any face does not match with our database it labelled with unknown.

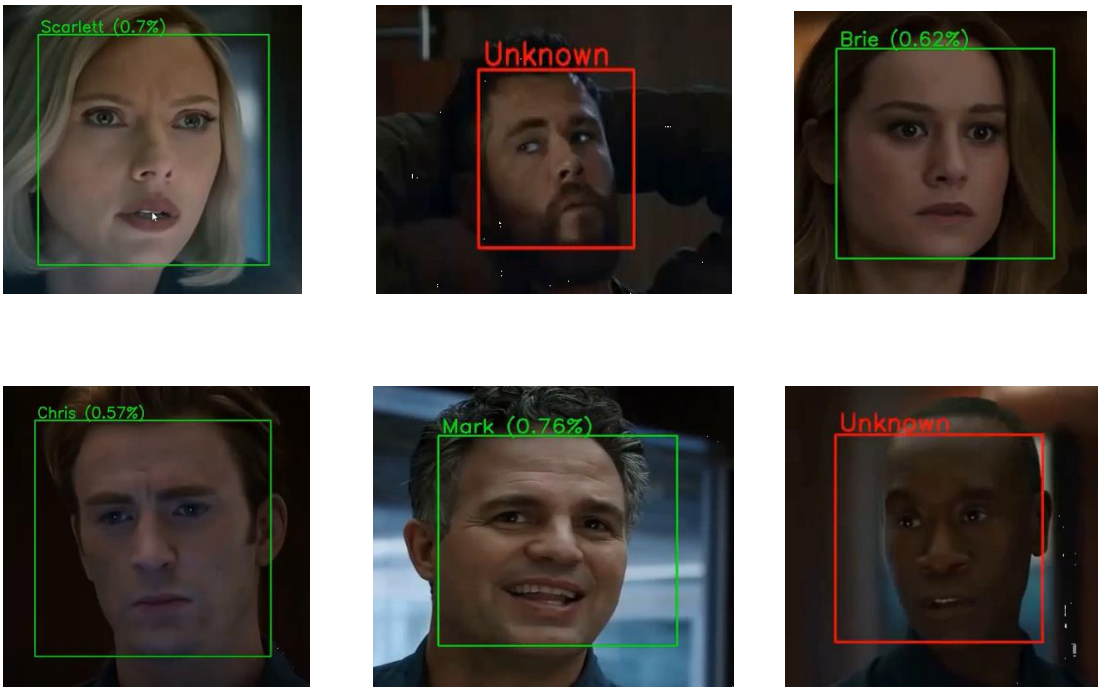

Figure 16: Result of OpenFace

Chapter 6
TOOLS

Tools:

To implement this project, we used many tools. Some of them are as follows:

- Python
- OpenCV
- Keras Deep Learning Framework

Chapter 7
PERFORMANCE ANALYSIS

7.1. Computational Representation

Thus, the calculated computational efficiencies are:

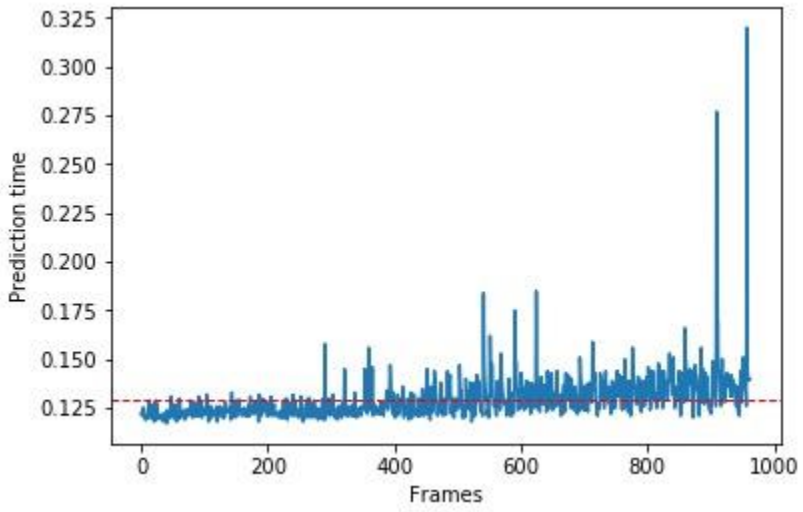

Figure 17: Prediction time / frames graph for DeepFace

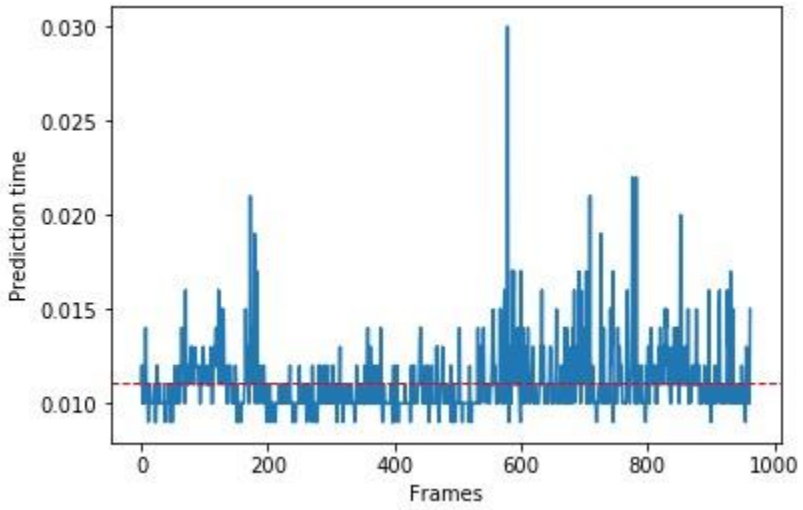

Figure 18: Prediction time / frames graph for FaceNet

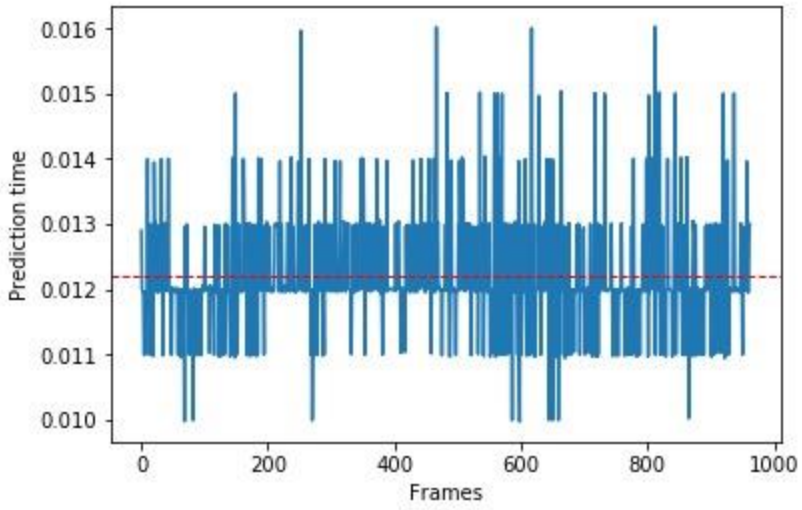

Figure 19: Prediction time / frames graph for OpenFace

7.2. Comparison

Model	Layers	Parameters	Dataset	Computational Efficiency (Sec.)	Accuracy
DeepFace	9	137,774,071	VGGFace2	0.140	99%
FaceNet	22	22,808,144	LFW	0.011	90%
OpenFace	22	3,743,280	LFW	0.012	90%

Figure 20: Comparison Table

Chapter 8
CONCLUSION AND DISCUSSION

8.1. Conclusion

This chapter presents the summary of our comparative study and address some expression for future succession in this area.

8.2. Summary

Face recognition technology has to identify human faces. Facial recognition systems are also highly effective in producing results, all thanks to the use of neural network algorithms. Face recognition, also called identification involves comparing one input image to all images in an image library in order to determine who the input image belongs to. Or if it does not belong to the database at all.

Face recognition is an emerging technology that can provide many benefits. Face recognition can save resources and time, and even generate new income streams, for companies that implement it right. Our work demonstrates the ability to present a marked improvement in face recognition. That is our results seem to yield a solid evidence that approximating the expected optimal results.

8.3. Future Works

The limitation in our work is that the accuracy is not efficient enough in terms of all deep models.

So, in future, we want to make further improvement of our study. Regard, we are planning to introduce the following terms:

- To make the dataset more robust.
- Optimization of the study to improve accuracy.
- Develop the dataset using more recordings.
- Optimize more model for better performance.

References

[1] Mei Wang and Weihong Deng. Deep Face Recognition: A Survey. 2020.

[2] Yaniv Taigman, Ming Yan, Marc' Aurrelio Ranzato and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification.

[3] Florian Schroff, Dmitry Kalenichenko and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. 2015.

[4] Tadas Baltrusaitis, Peter Robinson and Louis-Philippe Morency. OpenFace: an open source facial behavior analysis toolkit.

[5] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of cognitive neuroscience,.vol. 2002.

[6] X.Liu, L. Song, X. Wu, and T. Tan. Transferring deep representation for nir-vis heterogeneous face recognition. In ICB,pages 1-8 IEEE,2018.

[7] A. R. Chowdhury,Y.-Y. Kin,S. Maji,A. Sarna,I. Mosseri and one-to-many face recognition with bilinear cnns. In WACV, pages 1-9. IEEE,2016.

[8] C. Han,S. Shan,M. Kan,S. Wu, and X. Chen. Face recognition with contrastive convolution. In The European Conference on Computer Vision (ECCV),September 2018.

[9] S. Wang etal.: A Natural Visible and Infrared Facial Expression Database for Expression Recognition and Emotion Inference, IEEE Transactions on Multimedia, Vol. 12, No. 7, pp: 682-691(2010).

[10] R. Cutler: Face recognition using infrared images and eigenfaces, Technical Report, University of Maryland, (2008).

[11] W. L. S. G. Zongwei Wang,Xu Tang. Face aging with identity-preserved conditional generative adversarial networks. In CVPR,pages 7939-7947,2018.

[12] X. Wu, R. He, and Z. Sun. A lightened cnn for deep face representation. In CVPR,Volume 4,2016.

[13] L. Wolf, T. Hassner. and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In CVPR,2011

[14] D. Yi, Z. Lei, and S. Z. Li. Towards pose robust face recognition. In CVPR,2013.

[15] T. Ahonen, A. Hadid. Face description with local binary patterns: Application to face recognition. PAMI,2006.

[16] O. Barkan, J. Weill, L. Wolf,and H. Aronowitz. Fast high-dimensional vector multiplication face recognition. In ICCV,2014.

Appendix

DeepFace

```
import os
from os import listdir
import numpy as np
import cv2
from keras.models import Model, Sequential
from keras.layers import Input, Convolution2D, LocallyConnected2D, MaxPooling2D, Flatten, Dense, Dropout, Activation
from PIL import Image
from keras.preprocessing.image import load_img, save_img, img_to_array
from keras.applications.imagenet_utils import preprocess_input
from keras.preprocessing import image
import matplotlib.pyplot as plt
from keras.models import model_from_json
import math

#------------------------

target_size = (152, 152)

#------------------------

#OpenCV haarcascade module

opencv_home = cv2.__file__
folders = opencv_home.split(os.path.sep)[0:-1]
path = folders[0]
```

```
for folder in folders[1:]:
	path = path + "/" + folder


detector_path = path+"/data/haarcascade_frontalface_default.xml"


if os.path.isfile(detector_path) != True:
	raise ValueError("Confirm that opencv is installed on environment! Expected path
	",detector_path," violated.")
else:
	face_cascade = cv2.CascadeClassifier(detector_path)
#------------------------
def detectFace(img_path, target_size=(152, 152)):
	print(img_path)
	img = cv2.imread(img_path)
	faces = face_cascade.detectMultiScale(img, 1.3, 5)
	if len(faces) > 0:
		x,y,w,h = faces[0]


		margin = 0
	    x_margin = w * margin / 100
		y_margin = h * margin / 100


		if y - y_margin > 0 and y+h+y_margin < img.shape[1] and
		x- x_margin > 0 and x+w+x_margin < img.shape[0]:
			detected_face = img[int(y-y_margin):int(y+h+y_margin), int(x-
			x_margin):int(x+w+x_margin)]
		else:
			detected_face = img[int(y):int(y+h), int(x):int(x+w)]
```

```
            detected_face = cv2.resize(detected_face, target_size)


            img_pixels = image.img_to_array(detected_face)
            img_pixels = np.expand_dims(img_pixels, axis = 0)


            #normalize in [0, 1]
            img_pixels /= 255
            return img_pixels
            else:
            raise ValueError("Face could not be detected in ", img_path,". Please confirm that the picture is a face photo.")
#------------------------
#DeepFace model
base_model = Sequential()
base_model.add(Convolution2D(32, (11, 11), activation='relu', name='C1', input_shape=(152, 152, 3)))
base_model.add(MaxPooling2D(pool_size=3, strides=2, padding='same', name='M2'))
base_model.add(Convolution2D(16, (9, 9), activation='relu', name='C3'))
base_model.add(LocallyConnected2D(16, (9, 9), activation='relu', name='L4'))
base_model.add(LocallyConnected2D(16, (7, 7), strides=2, activation='relu', name='L5')
)
base_model.add(LocallyConnected2D(16, (5, 5), activation='relu', name='L6'))
base_model.add(Flatten(name='F0'))
base_model.add(Dense(4096, activation='relu', name='F7'))
base_model.add(Dropout(rate=0.5, name='D0'))
base_model.add(Dense(8631, activation='softmax', name='F8'))
base_model.load_weights("weights/VGGFace2_DeepFace_weights_val-0.9034.h5")
#Drop F8 and D0 layers. F7 is the representation layer.
```

```
model = Model(inputs=base_model.layers[0].input, outputs=base_model.layers[-3].out-
put)

#-----------------------
def l2_normalize(x):
        return x / np.sqrt(np.sum(np.multiply(x, x)))
def findEuclideanDistance(source_representation, test_representation):
        euclidean_distance = source_representation - test_representation
        euclidean_distance = np.sum(np.multiply(euclidean_distance,
        euclidean_distance))
        euclidean_distance = np.sqrt(euclidean_distance)
        return euclidean_distance
#-----------------------
#putting pictures in this path as name.jpg
employee_pictures = "database/"
employees = dict()

for file in listdir(employee_pictures):
        employee, extension = file.split(".")
        print(employee)
        img_path = 'database/%s.jpg' % (employee)
        try:
                img = detectFace(img_path)
        except:
                print("%s is not valid" %(employee))
                continue
        representation = model.predict(img)[0]
        employees[employee] = representation
```

```
print("employee representations retrieved successfully")
#-----------------------
cap = cv2.VideoCapture('Result/Video.mp4') #webcam
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
out = cv2.VideoWriter('Result/output.avi',cv2.VideoWriter_fourcc('M','J','P','G
'), 10, (frame_width,frame_height))

while(True):
    ret, img = cap.read() #capturing frames from webcam

    if ret == True:
        faces = face_cascade.detectMultiScale(img, 1.3, 5)

        for (x,y,w,h) in faces:
            if w > 0: #discard small detected faces
                cv2.rectangle(img, (x,y), (x+w,y+h), (67, 67, 67), 1) #draw
                rectangle to main image
                detected_face = img[int(y):int(y+h), int(x):int(x+w)] #crop de-
                tected face
                detected_face = cv2.resize(detected_face, target_size)
                #resize to 152x152
                img_pixels = image.img_to_array(detected_face)
                img_pixels = np.expand_dims(img_pixels, axis = 0)
                img_pixels /= 255
                captured_representation =
                model.predict(img_pixels)[0]
            distances = []
```

```
        for i in employees:
            employee_name = i
            source_representation = employees[i]

            distance = findEuclideanDistance(l2_normalize(captured_representation), l2_normalize(source_representation))
            distances.append(distance)
        is_found = False; index = 0
        min_dist = 20
        threshold = 0.55
cutoff = 0.80

        for i in employees:
            if index == np.argmin(distances):
                print(i, ": ", distances[index])

                if distances[index] < min_dist and
                distances[index] < cutoff:
                    print("detected: ",employee_name, "(",distances[index],")")
                    employee_name =  employee_name.replace("_", "")
                    similarity = distances[index]
                    min_dist = distances[index]
                    is_found = True
            index = index + 1

        if is_found:
```

```
display_img = cv2.imread("database/%s.jpg"  % em-
ployee_name)

pivot_img_size = 112

display_img = cv2.resize(display_img,

(pivot_img_size, pivot_img_size))

try:

        resolution_x = img.shape[1]; resolution_y =
        img.shape[0]


        print(employee_name)


        label = employee_name+" ("+"{0:.2f}%".for-
        mat(similarity)+")"


        if y - pivot_img_size > 0 and x + w +
        pivot_img_size < resolution_x:

                cv2.rectangle(img, (x,y), (x+w,y+h),
                (0,255,0), 1)

                cv2.putText(img, label, (x+w, y+10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                (0,255,0), 1)


        elif y + h + pivot_img_size < resolution_y and x
        - pivot_img_size > 0:

                cv2.rectangle(img, (x,y), (x+w,y+h),
                (0,255,0), 1)

                cv2.putText(img, label, (x -
                pivot_img_size, y+h-10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                (0,255,0), 1)
```

```
                                elif y - pivot_img_size > 0 and x -
                                pivot_img_size > 0:

                                    cv2.rectangle(img, (x,y), (x+w,y+h),
                                    (0,255,0), 1)
                                    cv2.putText(img, label, (x -
                                    pivot_img_size, y+10), cv2.FONT_HER-
                                    SHEY_SIMPLEX, 0.5, (0,255,0), 1)

                                elif x+w+pivot_img_size < resolution_x and y +
                                h + pivot_img_size < resolution_y:
                                    cv2.rectangle(img, (x,y), (x+w,y+h),
                                    (0,255,0), 1)
                                    cv2.putText(img, label, (x+w, y+h),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                                    (0,255,0), 1)
                            except Exception as e:
                                print("exception occured: ", str(e))
                        else:
                            label = 'Unknown'
                            cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255),  1)
                            cv2.putText(img, label, (x+w, y+10), cv2.FONT_HER-
                            SHEY_SIMPLEX, 0.5, (0,0,255), 1)
                    out.write(img)
                else:
                    break

#kill open cv things
cap.release()
out.release()
cv2.destroyAllWindows()
```

FaceNet

```
import numpy as np
import cv2
from keras.models import Model, Sequential
from keras.layers import Input, Convolution2D, ZeroPadding2D, MaxPooling2D, Flatten,
Dense, Dropout, Activation
from PIL import Image
from keras.preprocessing.image import load_img, save_img, img_to_array
from keras.applications.imagenet_utils import preprocess_input
from keras.preprocessing import image
import matplotlib.pyplot as plt
from keras.models import model_from_json
from os import listdir

from fr_utils import *
from inception_blocks_v2 import *
#----------------------

face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def preprocess_image(image_path):
    img = load_img(image_path, target_size=(160, 160))
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)

    img = preprocess_input(img)
    return img

#-----------------------

model =faceRecoModel(input_shape=(3, 96, 96))
def triplet_loss(y_true, y_pred, alpha = 0.3):

    anchor, positive, negative = y_pred[0], y_pred[1], y_pred[2]


    # Step 1: Compute the (encoding) distance between the anchor and   the positive,
you will need to sum over axis=-1
    pos_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, positive)), axis=-1)
    # Step 2: Compute the (encoding) distance between the anchor and the negative,
you will need to sum over axis=-1
    neg_dist = tf.reduce_sum(tf.square(tf.subtract(anchor, negative)), axis=-1)
    # Step 3: subtract the two previous distances and add alpha.
    basic_loss = tf.add(tf.subtract(pos_dist, neg_dist), alpha)
```

```
    # Step 4: Take the maximum of basic_loss and 0.0. Sum over the training examples.
    loss = tf.reduce_sum(tf.maximum(basic_loss, 0.0))


return loss
model.compile(optimizer = 'adam', loss = triplet_loss, metrics = ['accuracy'])
load_weights_from_FaceNet(model)
print("model built")

print("weights loaded")

#-----------------------

def findEuclideanDistance(source_representation,test_representation):
    euclidean_distance = source_representation - test_representation
    euclidean_distance = np.sum(np.multiply(euclidean_distance,        euclidean_dis-
tance))
    euclidean_distance = np.sqrt(euclidean_distance)
    return euclidean_distance

#-----------------------

threshold = 21 #tuned threshold for l2 disabled euclidean distance

#-----------------------

#put your employee pictures in this path as name_of_employee.jpg
employee_pictures = "images/"

employees = dict()

for file in listdir(employee_pictures):
        employee, extension = file.split(".")
                img = preprocess_image('images/%s.jpg' % (employee))
        representation = model.predict(img)[0,:]

        employees[employee] = representation

print("employee representations retrieved successfully")
#-----------------------

cap = cv2.VideoCapture(0) #webcam

while(True):
    ret, img = cap.read()
```

```
faces = face_cascade.detectMultiScale(img, 1.3, 5)

for (x,y,w,h) in faces:
    if w > 130: #discard small detected faces

                cv2.rectangle(img, (x,y), (x+w,y+h), (67, 67, 67), 1)
        #draw  rectangle to main image



                detected_face = img[int(y):int(y+h), int(x):int(x+w)]
        #crop detected face

                detected_face = cv2.resize(detected_face, (160, 160))
        #resize to 224x224



                img_pixels = image.img_to_array(detected_face)

                img_pixels = np.expand_dims(img_pixels, axis = 0)

      #employee dictionary is using preprocess_image and it  normalizes in scale of
[-1, +1]

        img_pixels /= 127.5
        img_pixels -= 1

                captured_representation = model.predict(img_pixels)[0,:]
      distances = [ ]

        for i in employees:
            employee_name = i
            source_representation = employees[ i ]

            distance = findEuclideanDistance(captured_representation, source_rep-
resentation)

             #print(employee_name,": ",distance)

             distances.append(distance)


        label_name = 'unknown'
        index = 0
```

```
                for i in employees:
                employee_name = i
                if index == np.argmin(distances):

                        if distances[index] <= threshold:
                            #print("detected: ",employee_name)

                #label_name = "%s (distance: %s)" % (employee_name, str(round(dis-
tance,2)))

                                similarity = 100 + (20 - distance)
                                if similarity > 99.99: similarity = 99.99
                        label_name = "%s (%s%s)" %  (employee_name, str(round(sim-
ilarity,2)), '%')

                        break


                index = index + 1
                cv2.putText(img, label_name, (int(x+w+15), int(y-64)), cv2.FONT_HER-
SHEY_SIMPLEX, 1, (67,67,67), 2)

            #connect face and text                cv2.line(img,(x+w, y-64),(x+w-25, y-
64),(67,67,67),1)
                    cv2.line(img,(int(x+w/2),y),(x+w-25,y-64),(67,67,67),1)

    cv2.imshow('img',img)

    if cv2.waitKey(1) & 0xFF == ord('q'): #press q to quit
        break

#kill open cv things
cap.release()
cv2.destroyAllWindows()
```

OpenFace

```
import cv2
import numpy as np
from utils import image_to_embedding, create_input_image_embeddings
import utils
from model import openFaceModel

np.set_printoptions(threshold=np.nan)
```

```
# load model from model.py
model = openFaceModel

# load weights
weights = utils.weights
weights_dict = utils.load_weights()

# Set layer weights of the model
for name in weights:
  if model.get_layer(name) != None:
    model.get_layer(name).set_weights(weights_dict[name])
  elif model.get_layer(name) != None:
    model.get_layer(name).set_weights(weights_dict[name])

# face recognition happens here

def recognize_face(face_image, input_embeddings, model):

    # from utils.py script, prediction happens here
    embedding = image_to_embedding(face_image, model)

    minimum_distance = 200
    name = None

    # Loop over  names and encodings.
    for (input_name, input_embedding) in input_embeddings.items():

        euclidean_distance = np.linalg.norm(embedding-input_embedding)

        print('Euclidean distance from %s is %s' %(input_name.split('_')[0], euclidean_dis-
tance))

        if euclidean_distance < minimum_distance:
            minimum_distance = euclidean_distance
            name = input_name

    threshold = 0.8

    if minimum_distance < threshold:
        return str(name), minimum_distance
    else:
        return None, None
def recognize_faces_in_cam(input_embeddings):
```

```
 cv2.namedWindow("Face Recognizer")
vc = cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_SIMPLEX
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

while vc.isOpened():
    _, frame = vc.read()
    img = frame
    try:
        height, width, channels = frame.shape
    except AttributeError:
        print("Frame capturing failed.")
        continue

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # Loop through all the faces detected
  #  identities = []
    for (x, y, w, h) in faces:
        x1 = x
        y1 = y
        x2 = x+w
        y2 = y+h

        face_image = frame[max(0, y1):min(height, y2), max(0, x1):min(width, x2)]
        identity,distance = recognize_face(face_image, input_embeddings, model)

        if identity is not None:
            similarity = 100 + (90 - 100*distance)
            if similarity > 99.99: similarity = 99.99

                label = "%s (%s%s)" % (str(identity.split('_')[0]), str(round(similarity,2)), '%')
            img = cv2.rectangle(frame,(x1, y1),(x2, y2),(0,255,0),2)
            cv2.putText(img, label, (x1+5,y1-5), font, 1, (0,255,0), 2)
        else:
            label = "Unknown"
            img = cv2.rectangle(frame,(x1, y1),(x2, y2),(0,0,255),2)
            cv2.putText(img, label, (x1+5,y1-5), font, 1, (0,0,255), 2)

    key = cv2.waitKey(100)
    cv2.imshow("Face Recognizer", img)
```

```
        if key == 27: # exit on ESC
            break
    vc.release()
    cv2.destroyAllWindows()

# program starts from here

# load dataset (function in utils.py)
input_embeddings = create_input_image_embeddings(model)

# real-time face recognition starts here
recognize_faces_in_cam(input_embeddings)
```