A PROJECT REPORT

on

"Smart Surveillance System using OpenFace Face Recognition"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE & SYSTEM ENGG.

BY

Gargi Goel	1728199
Borneel Bikash Phukan	1728207
Saptami Das	1728208
Anmol Sharma	1728219
Neelay Choudhury	1728221
VISHAKHA Sinha	1728237

UNDER THE GUIDANCE OF
PROF. GUIDE NAME

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

June 2020

A PROJECT REPORT

on

“Smart Surveillance using OpenFace Facial Recognition System”

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN
COMPUTER SCIENCE & SYSTEM ENGG.

BY

GARGI GOEL	1728199
BORNEEL B. PHUKAN	1728207
SAPTAMI DAS	1728208
ANMOL SHARMA	1728219
NEELAY CHOUDHURY	1728221
VISHAKHA SINHA	1728237

UNDER THE GUIDANCE OF
PROF. TANMAY SWAIN

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAE, ODISHA -751024

June 2020

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"Smart Surveillance System using OpenFace Face Recognition"

submitted by

GARGI GOEL	1728199
BORNEEL B. PHUKAN	1728207
SAPTAMI DAS	1728208
ANMOL SHARMA	1728219
NEELAY CHOUDHURY	1728221
VISHAKHA SINHA	1728237

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & System Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: 10/06/2020

Prof. Tanmay Swain

Acknowledgements

We are profoundly grateful to Prof. Tanmay Swain for his expert guidance and encouragement throughout to see that this project rights its target since its commencement to its completion. The work is a team effort minus which the completion of this project was not possible.

GARGI GOEL
BORNEEL B. PHUKAN
SAPTAMI DAS
ANMOL SHARMA
NEELAY CHOUDHURY
VISHAKHA SINHA

ABSTRACT

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Keywords: FaceNet, OpenFace, LFW Dataset, SVM Classifier, Embeddings Extraction, Triplet Loss

Contents

Chapter 1

Introduction

1.1 Introduction

Security surveillance has always been a prime necessity, especially in criminal prone zones. However, the recent changing trend in the surveillance system has proved that a security camera with the added advantage of recognizing the person on the frame is more preferable to the traditional security cameras. These make the process of tracking culprits easier for the authorities. Therefore we have come up with a surveillance software powered by deep learning which uses the combination of a pre-trained Neural Network Model and Support Vector Classifier to detect the face of the person in frame and classify them according to the face data currently stored in the system.

In this ambitious project, we will extract facial features (also called embeddings) from the person in the frame, process it through the famous FaceNet Neural Network for validation. Then we will build and train a Support Vector Classifier (SVC), an unsupervised learning algorithm that will match and classify our input face data according to the data stored, before returning it with a distinct probability.

The pre-trained FaceNet model weights and parameters are available in the OpenCV repository and are saved with the .caffe extension. As we will find out, FaceNet uses 128d embeddings of the face data as input to the model and the process of conversion is done through a PyTorch implementation of the FaceNet Neural Network. It is this model through which the face embeddings are passed through. Therefore in order to create the entire system, we make use of Pytorch for FaceNet, Scikit-learn library for classification of output, pickle for storing the trained weights of the SVC and OpenCV for visualizing the entire event in action.

1.1.1 Need for Deep Learning in Surveillance Systems

1. A key advantage of deep learning-based algorithms over legacy computer vision algorithms is that deep learning systems can be continuously trained and improved with better and more datasets. Many applications have shown that deep learning systems can "learn" to achieve 99.9% accuracy for certain tasks, in contrast to rigid computer algorithms where it is very difficult to improve a system past 95% accuracy.

2. The second advantage with a deep learning system is the "abnormal" event detection. Deep learning systems have shown a remarkable ability to detect undefined or unexpected events. This feature has the true potential of significantly reducing false-positive detection events that plague many security video analytics systems.

3. Deep learning does not require manual intervention but relies on a computer to extract features by itself. This way it is able to extract as many features from the target as possible, including abstract features that are difficult or impossible to describe. The more features there are, the more accurate the recognition and classification will be. Some of the most direct benefits that deep learning algorithms can bring include achieving comparable or even better-than-human pattern recognition accuracy, strong anti-interference capabilities, and the ability to classify and recognise thousands of features.

4. The algorithmic model for deep learning has a much deeper structure than the two 3-layered structures of traditional algorithms. Sometimes, the number of layers can reach over a hundred, enabling it to process large amounts of data in complex classifications. The higher the layer level, the more specific the components.

5. One of the great benefits of Deep Learning as a technology is the ability to accurately distinguish humans from animals. This makes for a great addition to the security arsenal of agencies where false alarms account for a whopping 95% of all security-related alerts.

1.1.2 Terminologies

- **NEURAL NETWORKS:** Neural networks are a set of algorithms, models loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input The patterns they recognize are numerical, contained in vectors, into which all the real-world data, be it images, sound, text or time series, must be translated.
- **FACENET-** FaceNet is a neural network that learns a mapping from face images to a compact Euclidean space where distances correspond to a measure of face similarity. That is to say, the more similar two face images are the less the distance between them.

- **LFW DATASET-** Labeled Faces in the Wild (LFW) is a database of face photographs designed for studying the problem of unconstrained face recognition. According to the researchers, deep-funnelled images produced superior results for most face verification algorithms compared to the other image types.

- SVM CLASSIFIER- SVM is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

- **OPEN CV-** OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

- **OpenFace**- The OpenFace neural network is a development of the traditional less efficient FaceNet neural network model. Built using pytorch, this model uses 128-D input vectors, goes through 2 layers of NN, with batch normalization and dropout layers and trained with a batch size of 128 and epochs of 100.

Chapter 2

Literature Survey

2.1 Related Works

Face Recognition Vendor Test Demographic Effects Report

On **December 19th, 2019** an FRVT report was released that describes and quantifies demographic differentials for contemporary face recognition algorithms. NIST, a US government initiative has conducted tests to quantify demographic differences for nearly 200 face recognition algorithms from nearly 100 developers, using four collections of photographs with more than 18 million images of more than 8 million people. Using both one-to-one verification and one-to-many identification algorithms submitted to NIST, the report found empirical evidence for the existence of a wide range of accuracy across demographic differences in the majority of the current face recognition algorithms that were evaluated.

Facial Emotion Recognition using Face Images

On **October 18th, 2017** a paper emphasizing the detection of seven human emotions along with the positives and negatives in the graph. It uses deep-learning technology to generate models with emotion-based facial expressions to recognize emotions. Thus, seven emotions were used to recognize, such as Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral and also classified the calculated emotion-recognition scores into positive, negative and neutral emotions.

Law Enforcement and finding Missing People

In the dawn of 2019, the government of India adopted a new technology that detects lawbreakers using Facial Recognition System. It consists of the largest face dataset of its citizens in the world and with proper coordination, it has helped the authorities to track down certain lawbreakers (including in social media) and take necessary actions. Also, this methodology has also been proven useful in tracking down missing people.

Chapter 3

Software Requirements Specification

3.1 Hardware & Software Requirements

3.1.1 Hardware Requirements

Components	**Version**
Processor	Intel® Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz
RAM	8.00GB
OS	Windows 10 64-bit, x64-based processor
GPU	Not Necessary

3.1.2 Software Requirements

Software	**Version**
PyCharm Community Edition	2019.3.5
Python	3.6.5
Spyder	4.1.1
VSCode	1.45.1
Google Colab	Adaptive

3.1.3 Libraries Requirements

Libraries	**Version**
OpenCV	4.1.2.30
Pytorch	1.2.0
Numpy	1.17.4
Pickleshare	0.7.5
Argparse	1.4.0

Chapter 4

Requirement Analysis

4.1 Hardware Requirements Analysis

Processor – We are using Intel® Core(TM) i5-7200U CPU @ 2.50GHz 2.71GHz processor for the purpose of speed and better load management that will be needed during the embeddings extraction and training of the SVM model. Processor is a very important factor when it comes to generating accurate results.

RAM – Higher the RAM faster is the process in fetching the data from the secondary memory. The RAM reduces the efforts of fetching data from the secondary memory everytime the code runs an iteration procedure.

4.2 Software & Library Requirements Analysis

IDE – A Python specific Integrated Development Environment (IDE) like VSCode, PyCharm or Spyder has been used for facilitated coding of the entire system. The IDE of choice depends upon the individual using it.

Python – The programming language plays a very important role in every development procedures. The reason for choosing python over other languages is due to the wider acceptability and larger community associated with it. Libraries like OpenCV, Numpy etc all are supported by Python which makes development easier.

OpenCV – OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. In this project, we are using OpenCV for the purpose of detecting faces from the camera and drawing the bounding boxes.

Pytorch – Pytorch is an open-source library that is mostly used for scientific computation in machine learning and natural language processing. The FaceNet model used in this project is made using Pytorch. Just like Tensorflow, Pytorch is renowned for its better performances and adaptability. Here we are using a .caffe saved model of the FaceNet.

Argparse – The Argument Parser is a special tool in Python that is used for taking command line arguments during runtime. This method makes the entire process of taking components easier and thereby increases flexibility of the program.

Pickle – After extracting the embeddings from the images data, we need to save the embeddings so that the SVM can be trained. Therefore in order to store the embeddings, we make use of pickle files. These files provide easy access and loading of data without compromising the integrity.

Numpy – Numpy is a special python tool that is used for performing numerical and scientific computation. In the field of machine learning, this is extremely important as it ensures that the program does not face any space or time complexity issues.

Chapter 5

System Design

5.1 Blue Chart and Components

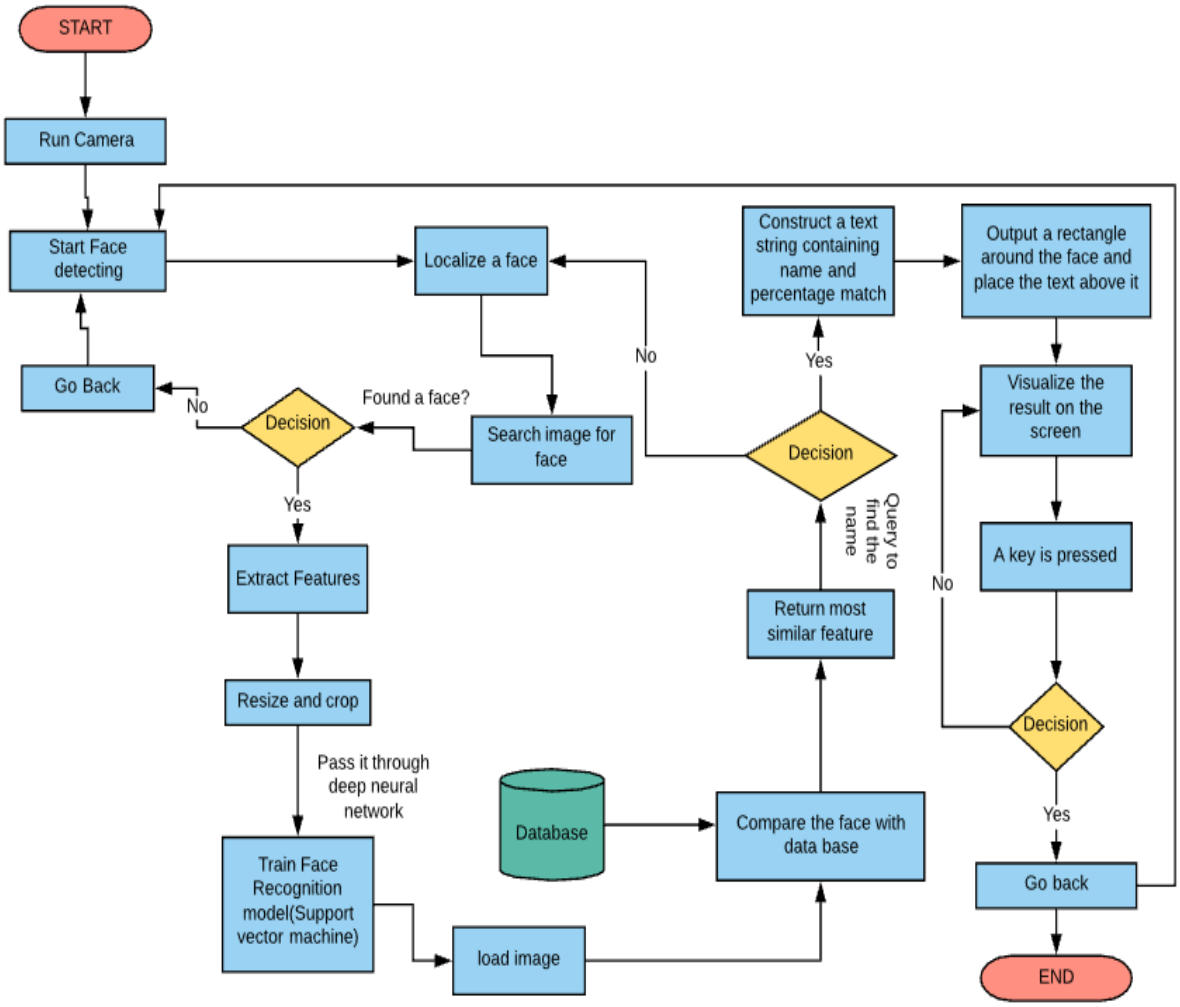


Fig. Blue chart of the system designed

The system is designed using the following major components in mind:

1. **Database** – The database is a very important component that stores the image data of the person who's face is to be detected. Here, the images of every person is stored in an individual sub-folder with the name of the person. Setting the sub-folder as the name of the person is very important because we will be extracting the name of the sub folder and train and use it for identification of the person on frame.

2. **openface_nn4.small2.v1.t7** – This is the pytorch implementation of the FaceNet model. The model makes use of Google's FaceNet architecture for feature extraction and uses a triplet loss function to test how accurate the neural net classifies a face. This model is trained using 50,000 images of the Labelled Faces in the Wild Home (LFW) Dataset and the weights stored as a caffe model.

3. **res10_300x300_ssd_iter_140000.caffemodel** – The caffemodel is used for storing the trained weights generated after training the openface neural network. It contains a deploy.prototext file which stores the meta data of the model. In this project, we will be using and overwriting these weights with our own custom face image data to generate an entirely new facial recognition model.

4. **Support Vector Classifier** – Once the process of triplet loss calculation has been completed, the process of classifying the different face images according to the person on frame is initiated. For this, we can either use multiple classification, random forest classification or support vector classification. Out of all these classification technique, the "sklearn-svc" or Support Vector Classifier has proven to be the most accurate in terms of classification.

5. **Creating the FaceBlob -** Once the classification procedure has been accomplished, it is time to initiate the creation of a Faceblob around the face on frame. This is done using the OpenCV framework where we create a square around the face embeddings and implement it on a loop.

6. **Computing the Softmax probabilities –** Once the detection and creation of the face blob is completed, we have to compute the softmax probabilities. A softmax probability is used for determining the degree of accuracy of prediction by the system. It is a comparision between the absolute value and the real value. Higher the softmax probability, more is the probability that the prediction of the person is correct.

Chapter 6

System Testing

For the purpose of system testing of the detection and identification, we have put forward three test cases and measured the softmax probability of detection. The softmax probability formula is:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

where,
xi = input features
xj = output features

The three cases are-1. Obstruction or Overlapping
2. Maximum distance of detection
3. Obstruction with a different face

6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	Obstruction or Overlapping	Face covered with a mask	69.85%	100.00%
T02	Maximum distance of detection	Detection from maximum 8 feet distance	77.57%	100.00%
T03	Obstruction with a different face	Target face behind an unknown face	58.78%	100.00%

Note: The testing has been performed manually.

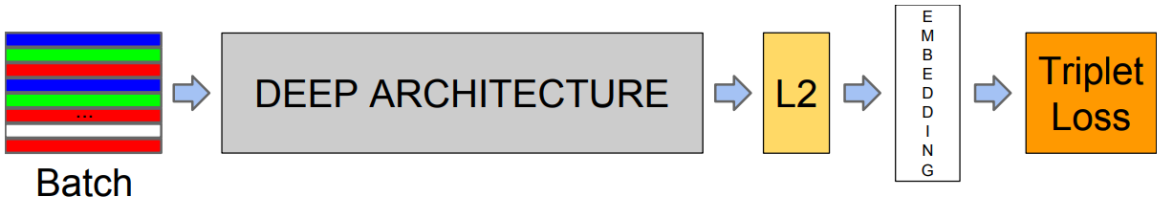
Chapter 7

Project Planning

7.1. Cumulative Architecture

In this project, we are using OpenFace Neural Network which is inspired by the Google made FaceNet model in order to extract embeddings and classify according to their mappings. The proposed architecture generates a high quality face mapping from the images stored using a deep learning architecture such as ZF-Net and Inceptionneural network.

Following processing through the deep learning network, and adding L2-normalization, the calculation of triplet loss is done to train this architecture. The arcchitecture is



given below:

- The model employs end to end learning in its architecture. It uses ZF-Net or Inception as its underlying architecture. It also adds several *1*1* convolutions to decrease the number of parameters.
- These deep learning models outputs an embedding of the image *f(x)* with *L2* normalization performed on it. These embeddings then passed into the loss function to calculate the loss.
- The goal of this loss function is to make the squared distance between two image embedding is independent of image condition and pose of the same identity is small, whereas the squared distance between two images of different identity is large. Therefore a new loss function called **Triplet loss** is used.
- The idea of using triplet loss in our architecture is that it makes the model to enforce a margin between faces of different identities.

7.1.1 **Calculation of Triplet Loss:**

The embedding of an image is represented by *f(x)* such as $x \in \mathbb{R}$. This embedding is in the form of vector of size *128* and it is normalized such that

$$\|f(x)\|_2^2 = 1$$

We want to make sure that the anchor image ($x^i a$) *of a person is closer to a positive image*($x^i p$) (image of the same person) as compared to a negative image (xin) (image of another person) such that:

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathrm{T}$$

where α is the margin that is enforced to differentiate between positive and negative pairs and T are the image space.
Therefore the loss function is defined as the following :

$$\mathrm{L} = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]$$

7.1.2 **Triplet Selection method:**

In order to ensure faster learning, we need to take triplets that violate the equation above. This means for givenwe x_i^a need to select triplets such that is maximum and $\|f(x_i^a) - f(x_i^n)\|_2^2$ is minimum .

It is computationally expensive to generate triplets based on whole training set.
There are two methods of generating triplets:

- Generating triplets on every step on the basis of previous checkpoints and compute minimum and maximum on a subset of data.
- Selecting hard positive (x_i^p) and hard negative (x_i^p) by using minimum and maximum on a mini batch.

Fig. Triplet Loss and Learning

7.2 Architecture of the Neural Network

Training :
This model is trained using Stochastic Gradient Descent (SGD) with backpropagation and AdaGrad. This model is trained on a CPU cluster for *1k-2k hours*. The steady decrease in loss (and increase in accuracy) was observed after 500 hours of training. This model is trained using two networks :

- **ZF-Net:**
  The visualization below indicates the different layers of ZF-Net used in this architecture with their memory requirement:

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Fig. ZF-Net architecture

As we notice that there are *140 million* parameters in the architecture and 1.6 billion FLOPS memory is required to train this model.

- **Inception:**
  The visualization below indicates the different layers of Inception model used in this architecture with their memory requirement:

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Fig. Inception Architecture

In our model, we have used the ZF-Net architecture version of the OpenFace model.

Chapter 8

Implementation

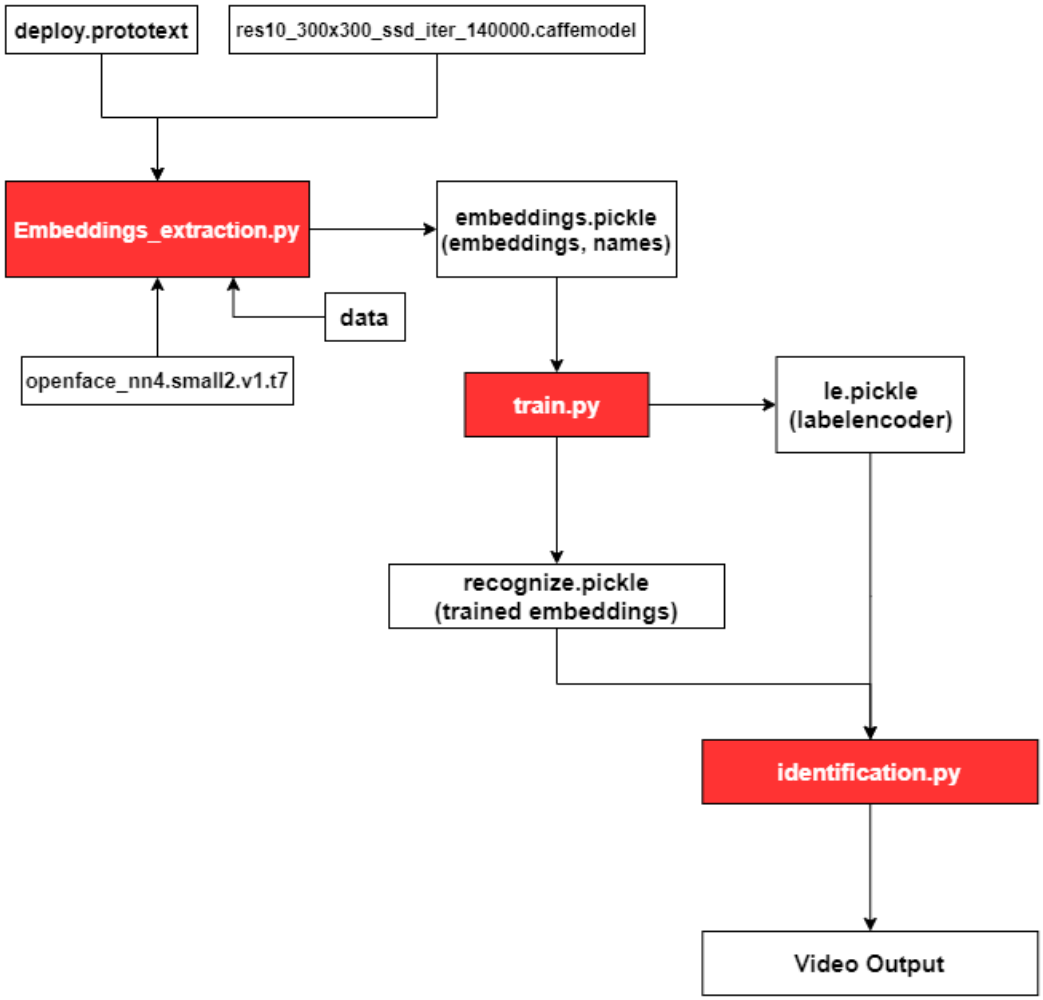


Fig. Simplified demonstration of the working of the system.

To enable the system to function properly, we need to run 3 scripts. The embeddings_extraction.py uses four components to extract facial embeddings (or features like face alignment, distance between the eyes, etc). The embeddings extraction process stores the extracted output onto a pickle file called “embeddings.pickle”.

Next, the train.py file takes in the embeddings data stored in the embeddings.pickle file. Then it adds a “labelencoder” to the embeddings and generates the recognize.pickle file. This pickle files contain trained embeddings and along with the saved labelencoders in “le.pickle” file, these are fed into the identification.py file which in turn detects the face from the camera.

STEP 1: EXTRACT EMBEDDINGS FROM FACE DATA SET

- First, we import the required packages and process the command line arguments.

```
from imutils import paths
import numpy as np
import argparse
import imutils
import pickle
import cv2
import os

#Argument Parsing
'''
--data -> path of input directory of images
--embeddings -> path of face embeddings after training
--detector -> path to pre-trained Caffe deep learning model provided by OpenCV to
detect faces. This model detects and localizes faces in an image.
--embedding-model -> path for pre-built and pre-trained FaceNet Neural Network.
--confidence -> minimum probability to filter weak detections
'''

argp = argparse.ArgumentParser()
argp.add_argument("-i", "--dataset", required=True)
argp.add_argument("-e", "--embeddings", required=True)
argp.add_argument("-d", "--detector", required=True)
argp.add_argument("-m", "--embedding-model", required=True)
argp.add_argument("-c", "--confidence", type=float, default=0.5)
args = vars(argp.parse_args())
```

- Then we load the following:
  - **Face detector:** Using a Caffe based DL face detection to localize the faces in an image.
  - **E**

    mbedder: Responsible for extracting embed-dings via deep learning feature extraction.

```
#loading the face detector
print("Loading Face Detection Algorithm (1)...")
proto_path = os.path.sep.join([args["detector"], "deploy.prototxt"])
model_path = os.path.sep.join([args["detector"],
"res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(proto_path, model_path)

#loading the face embeddings
print("Loading Extracted Face Embeddings (2)...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])
```

- Now we will grab the path to each image in the data-set. Our embeddings and corresponding names will be held in two lists: knownEmbeddings and KnownNames. We will also keep the track of no. Of images processed.

```
#grabbing the path of the input images
image_path = list(paths.list_images(args["dataset"]))

#creating a list of extracted face embeddings and person name
flagged_embeddings = []
flagged_names = []

total = 0
```

- We will loop over the image path and extract embeddings from faces found in each image.

```
#looping over every images
for (i, img_path) in enumerate(image_path):
    print("Extracting embeddings from images {}/{}".format(i+1, len(image_path)))
    person_name = img_path.split(os.path.sep)[-2]

#loading image, resizing to width 600px maintaining image ratio and grabbing the dimension
face_image = cv2.imread(img_path)
face_image = imutils.resize(face_image, width=600)
(height, width) = face_image.shape[:2]
```

- Then we detect faces by applying Open CV deep-learning based face detector to localize faces in an input image.

```
#making blob around the image
imageBlob = cv2.dnn.blobFromImage(cv2.resize(face_image, (300, 300)), 1.0, (300,300),
(104.0, 177.0, 123.0), swapRB=False, crop=False)

#Using face detector caffe model to detect faces in images
detector.setInput(imageBlob)
detections = detector.forward()
```

- The detection list contains probabilities and coordinates to localize faces in an image.

```
#(detection -> Number of faces)#Ensuring atleast one face is found


if len(detections) > 0:#finding the bounding box with the largest probability
#and filtering out weaker probability of embeddings
   i = np.argmax(detections[0,0,:,2])
   confidence = detections[0,0,i,2]
```

```
    if confidence > args["confidence"]:
#Computing x, y coordinates of the bounding box
        box = detections[0,0,i, 3:7] * np.array([width, height, width, height])
        (start_x, start_y, end_x, end_y) = box.astype("int")

#extracting face ROI from the embeddings and grabbing ROI dimension
        face = face_image[start_y:end_y, start_x:end_x]
        (face_height, face_width) = face.shape[:2]

#face width and face height should be large
        if face_width < 20 or face_height < 20:
            continue
```

- We take advantage of our embedder CNN and extract face embeddings.

```
'''
1. construct a face blob.
2. Pass the blob through the embedder.
3. Use the generate 128d quantification of face
'''

        faceBlob = cv2.dnn.blobFromImage(face, 1.0/255, (96,96), (0,0,0), swapRB=True,
crop=False)
        embedder.setInput(faceBlob)
        face_vectors = embedder.forward()

#adding the person name and face embeddings in a new list
        flagged_names.append(person_name)
        flagged_embeddings.append(face_vectors.flatten())
        total = total + 1
```

- All that's left when the loop finishes is to dump the data to disk and execute the commands to compute the face embeddings with Open CV.

```
#saving the embeddings and person name
print("Saving {} face embeddings...".format(total))
print("Save Complete")

labelled_data = {"embeddings": flagged_embeddings, "names": flagged_names}
file_object = open(args["embeddings"], "wb")
file_object.write(pickle.dumps(labelled_data))
file_object.close()
```

STEP 2: TRAIN FACE RECOGNITION MODEL

At this point, we have extracted 198-d embeddings for each face but now we need to actually recognize a person based on these embeddings.

- First, we import the required packages and parse the command line arguments.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
import argparse
import pickle
import matplotlib.pyplot as plt

argp = argparse.ArgumentParser()
argp.add_argument("-e", "--embeddings", required=True)
argp.add_argument("-r", "--recognizer", required=True)
argp.add_argument("-l", "--labelencoder", required=True)
args = vars(argp.parse_args())
```

- Then we will load the embeddings that were generated previously.

```
#loading the face embeddings
print("Loading Face Embeddings...")
data = pickle.loads(open(args["embeddings"], "rb").read())

#encoding the labels
print("Encoding in process...")
le = LabelEncoder()
labels = le.fit_transform(data["names"])
```

- We will train our SVM model that will accept the 128-d embeddings of the face and then produce the actual face recognition.

```
#training the modelprint("Model Training in progress...")
trainer = SVC(C=1.0, kernel="linear", probability=True)
trained_model = trainer.fit(data["embeddings"], labels)
```

- We write the actual face recognition model and label encoder to disk.

```
#saving the configured face recognition model
file_object = open(args["recognizer"], "wb")
file_object.write(pickle.dumps(trained_model))
file_object.close()

#saving the label encoder
file_object = open(args["labelencoder"], "wb")
file_object.write(pickle.dumps(le))
file_object.close()
```

STEP 3: RECOGNIZING FACES WITH OPEN CV

We are now ready to recognize faces in the image or video streams.

- First, we import the required package and module and parse the command line arguments.

```
from imutils import paths
from imutils.video import FPS
from imutils.video import VideoStream

import imutilsimport pickle
import os
import numpy as np
import cv2
import argparse
import time

argp = argparse.ArgumentParser()
argp.add_argument("-d", "--detector", required=True)
argp.add_argument("-m", "--embedding-model", required=True)
argp.add_argument("-r", "--recognizer", required=True)
argp.add_argument("-l", "--labelencoder", required=True)
argp.add_argument("-c", "--confidence", type=float, default=0.5)
args = vars(argp.parse_args())
```

- Next we load the three models from the secondary memory to main memory:

1. **Detector**: A pre-trained Caffe DL model to detect where in the image the faces are.
2. **Embedder**: A pre-trained Torch DL model to calculate our 128-d face embeddings.
3. **Recognizer**: Our Linear SVM face recognition model.

```
#loading the face detector
print("Loading Face Detection Algorithm (1)...")
proto_path = os.path.sep.join([args["detector"], "deploy.prototxt"])
model_path = os.path.sep.join([args["detector"], "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(proto_path, model_path)

#loading the face embeddings
print("Loading Extracted Face Embeddings (2)...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])

#loading the configured face recognition model with label encoder
recognizer = pickle.loads(open(args["recognizer"], "rb").read())
label_encoder = pickle.loads(open(args["labelencoder"], "rb").read())
```

- Now we load the image and resize it to have proper width and then grab the image dimension.

```
#starting up camera
print("Firing up the WebCam...")
video = VideoStream(src=0).start()
time.sleep(2.0)

#start FPS throughput estimator
fps = FPS().start()

while True:
    frame = video.read()
    frame = imutils.resize(frame, width=600)
    (height, width) = frame.shape[:2]

    #making blob around the image
    imageBlob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0, (300, 300), (104.0,
177.0, 123.0), swapRB=False, crop=False)

    #Using face detector caffe model to detect faces in images
    detector.setInput(imageBlob)
    detections = detector.forward()

#loop over the detections
    for i in range(0, detections.shape[2]):
        confidence = detections[0,0,i,2]

        #filtering weak detection
        if confidence > args["confidence"]:
            #compute the x, y coordinate of bounding box of face
            box = detections[0,0,i,3:7] * np.array([width, height, width, height])
            (start_x, start_y, end_x, end_y) = box.astype("int")

            #extracting ROI of face
            face = frame[start_y:end_y, start_x:end_x]
            (face_height, face_width) = face.shape[:2]

            #face width and face height should be large
           if face_width < 20 or face_height < 20:
                continue

            '''
           1. construct a face blob.
            2. Pass the blob through the embedder.
            3. Use the generate 128d quantification of face
            '''

            faceBlob = cv2.dnn.blobFromImage(face, 1.0/255, (96, 96), (0,0,0), swapRB=True,
crop=False)
            embedder.setInput(faceBlob)
            face_vectors = embedder.forward()

            #classification for face recognition
            predict = recognizer.predict_proba(face_vectors)[0]
            j = np.argmax(predict)
            proba = predict[j]
            name = label_encoder.classes_[j]
```

```
  #bounding box configuration
          text = "{}: {:.2f}%".format(name, proba*100)
         y = start_y - 10 if start_y else start_y + 10
          cv2.rectangle(frame, (start_x, start_y), (end_x, end_y), (0,0,255), 2)
         cv2.putText(frame, text, (start_x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0,255,0),2)

    #fps updated
    fps.update()

    #output frame display
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == 27:       break
fps.stop()

cv2.destroyAllWindows()video.stop()

print("\n")
print("Developed by Borneel, Anmol")
print("Neelay, Gargi, VISHAKHA and Saptami")
```

Chapter 10

Screenshots of Project

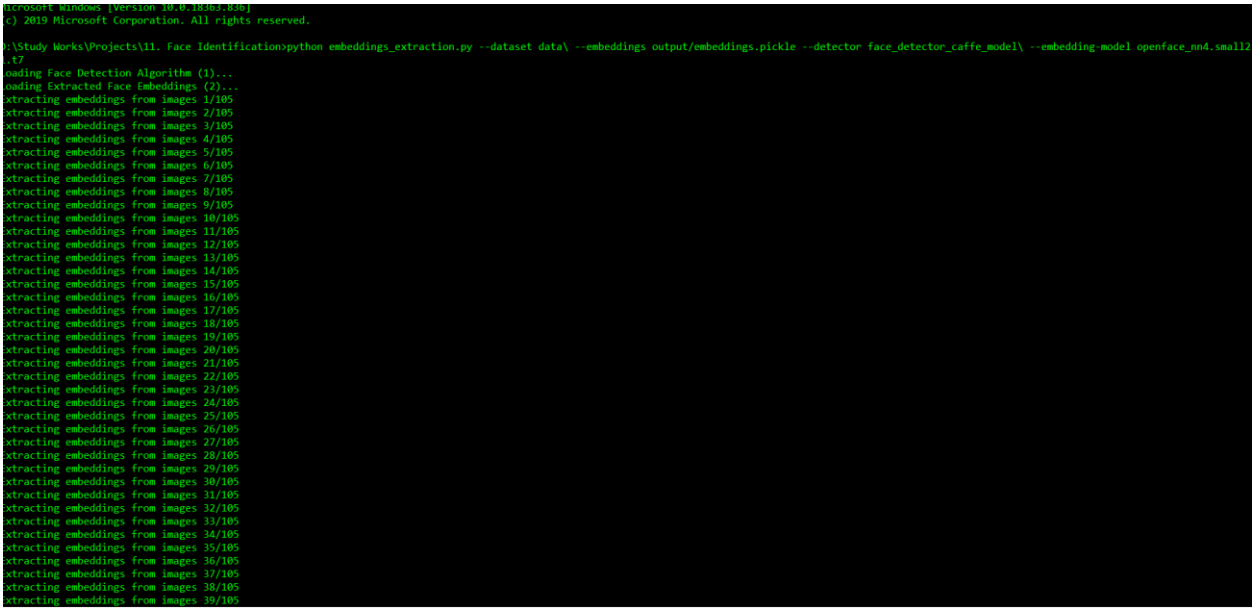


Fig. Extracting Embeddings from each images (The embeddings are stored in the 3 pickle files)

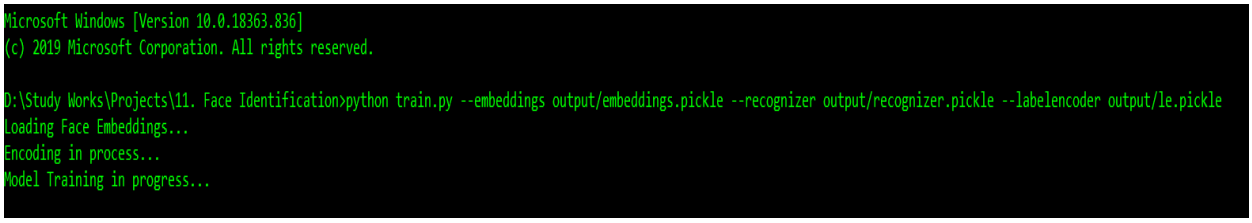


Fig. Training the SVM (train.py)

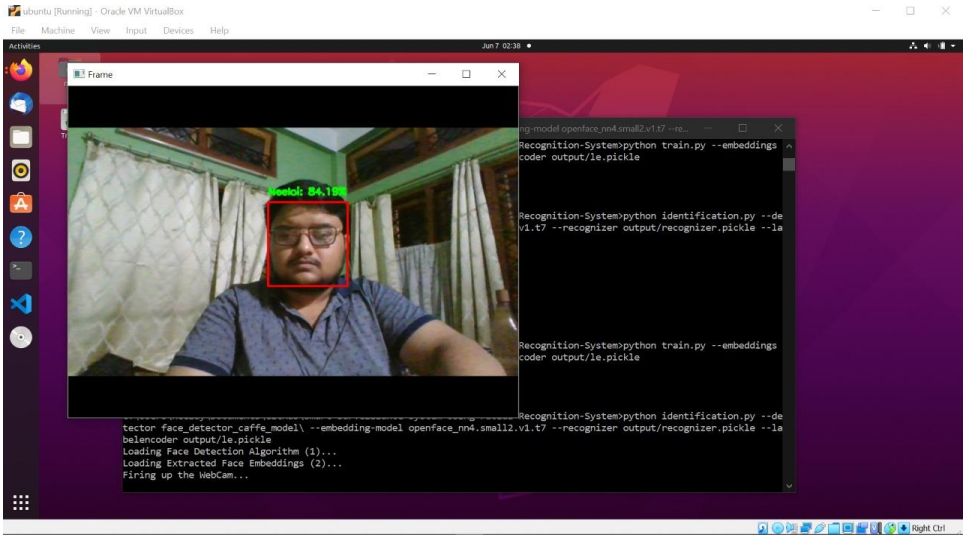


Figure: Detecting Face from a short range

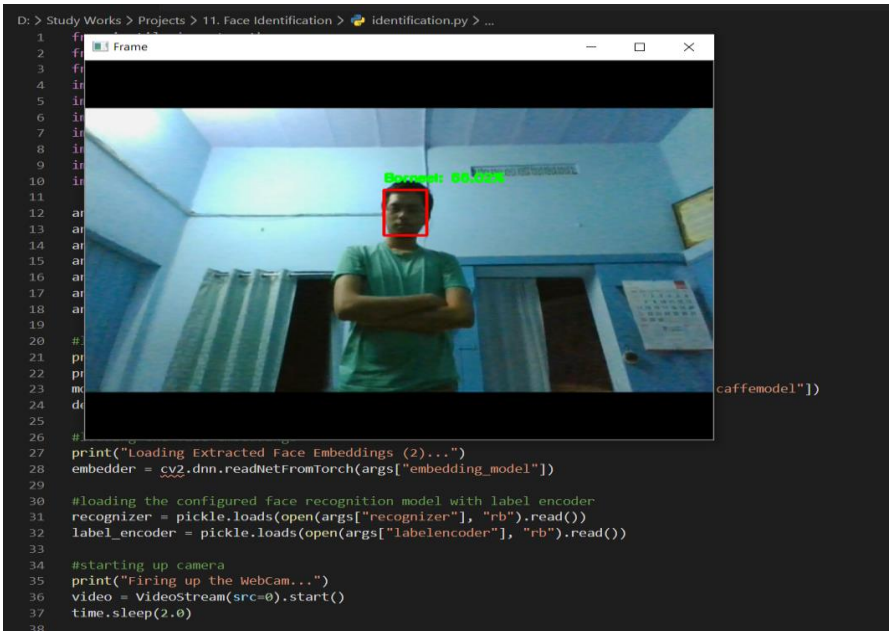


Figure: Detecting Face from a long range (~6ft with 56.02% accuracy)

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

Smart video surveillance system significantly contributes to situation awareness. Such systems transform video surveillance tool to information & intelligence acquisition systems.Real time video analysis provides smart surveillance systems with the ability to react in real-time. Our system senses the intrusion and send notifications to authorized person so that action can be taken in response to the intrusion.

10.2 Future Scope

The scope of Smart Surveillance is unlimited and with the increase in crime rates and unlawfullness, the need for smart surveillance has only grown over time. As far as future scope is concerned, it has multiple use cases like defence, autonomous system or robotics and investigations.

- **Defence Sector** – With the increase in border intrusion and hijacking of important national defence mechanisms, the need for smart surveillance has become very essential in classifying intruders from friendly troops and thereby playing a very important role of alerting command centers. This is highly competent when compared to manual sentry guards.

- **Autonomous Sector** – When the need for automation of multiple sectors are increasing, the need for automation of manual service sectors are increasing as well. Robotics is a very common example. A robot needs to identify the target person and the target task for functioning and the concept of detection and identification used in this project can also be used in autonomous systems for detection and identification tasks.

- **Investigation Bureau Sector** – During an intense investigation procedure, sometimes it becomes very tedious to detect a suspect from raw video footages. Then in this case, a face identification tool comes to help which uses it’s database of criminal records to identify the suspect on frame and thereby get the necessary details stored within the database. This face identification tool shall be based on the same concept that we have used in this project.

References

[1] Florian Schroff, Dmitry Kalenichenko and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. CVPR2015.

[2] "Understanding and Visualizing Resnet34" by Pablo Ruitz https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

[3] "FaceNet using Facial Recognition System" - https://www.geeksforgeeks.org/facenet-using-facial-recognition-system/

[4] "Face Recognition using OpenFace" - https://medium.com/analytics-vidhya/face-recognition-using-openface-92f02045ca2a

[5] "OpenFace Information Repository" - https://cmusatyalab.github.io/openface/

[6] "Understanding Open-Source Facial Recognition Through OpenFace" - https://algorithmia.com/blog/understanding-facial-recognition-openface

[7] "OpenCV reference documentation" - https://docs.opencv.org/4.2.0/

[8] "ML | Unsupervised Face Clustering Pipeline" - https://www.geeksforgeeks.org/ml-unsupervised-face-clustering-pipeline/

[9] "Support Vector Machine — Introduction to Machine Learning Algorithms" - https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[10] "Support Vector Classifier scikit learn documentation" - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[11] "Introduction to FaceNet: A Unified Embedding for Face Recognition and Clustering" - https://medium.com/analytics-vidhya/introduction-to-facenet-a-unified-embedding-for-face-recognition-and-clustering-dbdac8e6f02

[12] How to Develop a Face Recognition System Using FaceNet in Keras - https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/

SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

SMART SURVEILLANCE USING FACIAL RECOGNITION SYSTEM

GARGI GOEL
1728199

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution to project report preparation: I have worked on selection of OpenFace Neural Network and also assisted in bgathering of the theoritical data for the neural network architecture and triplet loss Function. The scope of testing is generally an extension of the requirement analysis phase and mostly considered as a single activity, since they go hand in hand. Once the requirements are out, I have determined what items are to be tested and what not. And finalluy did the debugging and testing part with large data sets.

Individual contribution for project presentation and demonstration: I'm responsible for the introduction and the conclusion part of the report.

Full Signature of Supervisor: Full signature of the student:

……. ……..

BORNEEL BIKASH PHUKAN
1728207

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution and findings: In this project I have highly contributed to the selection of the Neural Network for the purpose of face detection. From the fundamental harr-cascade face recognition, dlib to the advanced and the newer FaceNet and OpenFace, I have compared all the face recognition models and later I found out that OpenFace has better performance output. Thereby I choose the OpenFace Neural Network that is based on Inception neural network. However, the biggest challenge that we encountered was the implementation of the network architecture. Building the entire model from the scratch was a very tedious job and I even tried building the network alone. But unfortunately due to the lack of processing power as well as knowledge limitations, I had to drive away from that idea. Soon I started looking for pre-trained model available as open-source and to my luck, I found that OpenFace is an open-source project and the trained model is available as a pytorch implementation along with the weights saved as caffe model.

Next job was to import the caffe weights and the model into the work environment, and as the model first extracts the 128D face embeddings vectors, we use the model for extracting the embeddings and store it in certain pickle files. Another problem we encountered was the extracting of the names of the person in frame. Therefore, to counter this, I found out a way to extract the name from the path address where the images of each person are stored.

The only condition here is, we will need to arrange each images in groups of folders named after the person. Next this name is saved as labelencoder in another pickle file that will be used for training the Support Vector Classifier.

Individual contribution to project report preparation: While creating the project report, my role was to develop a simplified version of the system design that demonstrates how the data from the image (embeddings) get processed down the line of 3 files. Along with that, my most important task was to create the content for project planning section. Along with Gargi Goel, we determined how the cumulative architecture of the OpenFace works. We learned and noted the process of dividing the entire data into batches and after processing through the neural network architecture, the process of calculating the Triplet Loss and also how to deermine the Triplet Loss so that it causes minimum drag in the performance of the network. Following this, we also determined how the structures of the two types of neural network that can be used. The process of training using the stochaistic gradient descent and how the hyperparameters were determined, all the information and the necessary changes were collected and included by us. In order to provide a holistic approach and error-free content, we had to refer to sources like GeeksforGeeks and Medium.
Last but not the least, I have written the source code for embeddings extraction in the project which I have included in the implementation section.

Individual contribution for project presentation and demonstration: For the purpose of contributing to the project presentation, I have volunteered to explain the cumulative architecture of the system and also the embeddings extraction code.

Full Signature of Supervisor: Full signature of the student:

……. ……..

SAPTAMI DAS
1728208

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution and findings: I play some role in implementing the project through code.To build our project first we perform face detection, extract face embeddings from each face using deep learning, train a face recognition model on the embeddings, and then finally recognize faces in both images and video streams with OpenCV.The entire code can be divided in three main steps.In step 1,we extract embeddings from the face dataset.Here we import our packages and parsed command line arguments, and also load our face detector and embedder.After that,we grab our image paths and perform initialization and finally we detect and localize face.Our next step is to train our face recognition model which is ,Support vector machine ,which is a common machine learning model.And to implement and train our face recognition mode we will be using scikit-learn, a machine learning library.In the last step which is face recognition with openCV and to do that we import our required packages and load the three main model(face detector mode,serialized face embedding model,and actual face recognition model) from disk into memory.and we also load our label encoder which holds the name of the people our model can recognize,and finally we execute the program in the terminal. Currently we are facing the problem of integrating the entire project into a Web Application. The entire project consists of python files And every one of them needs to be run on a python console.

Unfortunately, we are lacking in the required knowledge and tools for running the same on a web browser so that with further development, this system can we used for remote detection in surveillance systems.

Individual contribution to project report preparation: I helped my project team members in preparation of our project report by making the detailed data flow diagram of our entire project. It demonstrates how our project will perform when it is used in real time,how it is going to take input from the live video stream and the entire work process of how it compares the detected face from our dataset and returns the most similar features and finally how the final result will be shown in the screen.

Individual contribution for project presentation and demonstration: For the purpose of contributing to the project demonstration and presentation ,I have explained how do we actually recognize a person based on embeddings and also the work flow of our entire project.

Full Signature of Supervisor: Full signature of the student:

……. ……..

ANMOL SHARMA
1728219

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution and findings: My contribution in the project was mostly in managing and keeping records of the issues faced and gathering the required resources for tackling of the different errors. I assisted the team in dubbing and error handling of the codes so and also determining the right proportion of images so that model never overfits or underfits. The idea of using imutils library for taking in the components as command line arguments was solefully mine as it reduces the burden on processing of the code by the interpreter.

Individual contribution to project report preparation: My contribution in creating the project report includes the design of the simplified block diagram version of the system design which explains the working of the 3 python files. We start by running the camera which will start detecting the face,if it localizes the face in the live video stream thenit will search for the faceand if it succeeds;it will extract the features.I t will then resize and crop the image otherwise it will go back to again detecting the face.

After the extraction of face features,the image will go through deep neural network and reach our pre - trained face recognition model which is the SVM(Support Vector Machine).This model will load the image and compare with our dataset and return most similar features.If the face exists in the dataset,it will make a query to find the name.

If it finds the name it will construct a text string containing name and percentage match and output the rectangle around the face with the text above it.

If the name is not found it will go back to the starting position.

After visualizing the result if a key is pressed it will stop showing the result.

Individual contribution for project presentation and demonstration: For the purpose of demonstration, I will be demonstrating the system design and how the application runs and how the image files and how the respective embeddings gets proessed until it can be understood by the openCV framework.

Full Signature of Supervisor: Full signature of the student:

……. ……..

NEELAY CHOUDHURY
1728221

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution and findings: In this project I have contributed to the selection of the Support Vector Machine(SVM) classifier for the purpose of classifying the different face images according to the person on frame. I had tried different classification techniques like multiple classification, random forest classification and support vector classification, but out of all those classification technique, the “sklearn-svc” or Support Vector Classifier has proven to be the most accurate in terms of classification.

Individual contribution to project report preparation: I helped my project team members in preparation of our project report by making the documentation of the identification.py file in the implementation(Chapter 8) chapter of the project, which is the step for Recognizing faces with opencv.

Individual contribution for project presentation and demonstration: For the purpose of contributing to the project presentation, I have volunteered to explain the System Testing part.

Full Signature of Supervisor: Full signature of the student:

……. ……..

VISAKHA SINHA
1728237

Abstract:

Surveillance System has always been around us and with the increase in global crime rate and unlawful activities, it has become very mandatory that these surveillance systems be fitted with some technology that has the capability to detect and identify the person commiting the activity. Although multiple detection system has been developed using basic software engineering methodologies, the model proposed here makes a Deep Learning approach to the problem and makes use of OpenFace Neural Network Architecture for the purpose of detecting, identifying and tracking the person on frame.

Through this paper, we would like to present a comprehensive study of all the researches done on OpenFace Neural Network, compare it with other face recognition system and provide a structured explanation of the implementation of OpenFace neural network in creating a smart surveillance system.

Individual contribution and findings: In the making up of the project,my contribution was to do the documentation part. For that I prepared detailed document which clearly mentions the different parts of the project. In my documentation,I mentioned a brief introduction related to the topic followed by a literary surevy on the project.I also prepared an SRS which clearly mentioned the hardware and software requirements.Blue chart of the system was also documented.Now the main building block of the project that is the test cases and results has been clearly stated in the document.I also mentioned a detailed overview of the project planning which includes claculation for identification with facenet and architecture of the open face neural network. The whole code along with a brief explanation is also documented.Finally the screenshots of the output along with the conclusion and future scope was also attached.

Individual contribution to project report preparation: In this part my role was to prepare a detailed overview of how the coding and compilation part was done during the project creation.Since,the whole project is divided into three parts, So I was assigned the task to write the codes of all the three parts of the project i.e. extracting the embeddings from face net, training the face recognition model, and recognize faces with the OpenCV.

Along with the codes, I also wrote a brief explanation of each part of the code in particular section and also every line of the code has been commented to get an overview of a particular line of code.Along with the codes, the output of the three sections were also shown in the project.

Individual **contribution for project presentation and demonstration:** For project presentation, I was supposed to explain the training of the face recognition model after the embeddings for each face has been extracted.

Full Signature of Supervisor: Full signature of the student:

……. ……..