Industrial Training Report

On

IDENTITY CLUSTERING

Submitted in Partial Fulfilment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

Submitted By

VIkas Kumar Pandey

(University Roll no. 1609510063)

Amazing Training Basket Pvt Ltd.

Plot no - A-40, Unit No Tower A 207, I- Thum IT Tower, Industrial Area, Sector 62, Noida, Uttar Pradesh 201309

Under the supervision of
Mr. Dheeraj Singh

Submitted To

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MGM's College of Engineering & Technology, Noida

September, 2019

CERTIFICATE

This is to certify that Project Report entitled “Identity Clustering” which is submitted by Vikas Kumar Pandey in partial fulfilment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J Abdul Kalam Technical University, is a record of the candidate own work carried out by them under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Training Coordinator **Head of the Department**

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Industrial Training Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Mrs. Monika Jaglan, Department of Computer Science & Engineering, MGM College of Engineering & Technology, Noida for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day. We also take the opportunity to acknowledge the contribution of Mrs. Archana Sar, Head of Department (Computer Science & Engineering), MGM College of Engineering & Technology, Noida for her full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Date :

Signature :

Name : Vikas Kumar Pandey

Roll No. : 1609510063

Abstract

This project aims to build an “Identity Clustering System” which cluster face images according to their identity. The clustering problem is composed of two key parts firstly representation and secondly similarity metric for face images, and choice of the partition algorithm. This system has two important applications firstly grouping a collection of face images when no external labels are associated with images, and secondly indexing for efficient large scale face retrieval.

In identity clustering we need to perform unsupervised learning because we have only the faces themselves with no names/labels. From there we need to identify and count the number of unique people in a dataset. This system takes a dataset of images as input. This system first extract and quantify the faces in a dataset. Then cluster the faces, where each resulting cluster (ideally) represents a unique individual. This system can be be used by Law enforcement agencies to find all unique faces in video feeds for catching suspects and criminals. The implementation of the system is done using Deep Learning and Python.

List of Figures

TABLE OF CONTENT

Chapters

CHAPTER-1

INTRODUCTION

1.1 Problem Definition

Cameras are everywhere, embedded in billions of smart phones and hundreds of millions of surveillance systems. Surveillance cameras, in particular, are a popular security mechanism employed by government agencies and businesses alike. This has resulted in the capture of suspect facial images in high profile cases such as the 26/11 Mumbai Terror Attacks. But, getting to the point of locating suspect's facial images typically requires manual processing of large volumes of images and videos of an event which is very time consuming. To solve this problem some sort of automation is strongly needed. The automatic processing of images and videos to perform the clustering will solve this problem with a great extent.

1.2 Introduction

Video surveillance methods become increasingly widespread and popular in many organizations, including law enforcement, traffic control and residential applications. In particular, the police perform investigations based on searching specific people in videos and in pictures. Because the number of such videos is increasing, manual examination of all frames becomes impossible. Some degree of automation is strongly needed.

Clustering is a technique to divide a set of objects in different groups or clusters resulting in each cluster having identical objects and different clusters contain objects with different characteristics. Face clustering is the process of grouping the faces of people present on a set of photos or videos. Ideally this process results in each person having his/her own cluster containing images of this particular person. By doing this we can answer certain questions such as who and how many different people were present in a particular photo or video?

This project aims to build an "Identity Clustering System" which cluster face images according to their identity. The clustering problem is composed of two key parts firstly representation and secondly similarity metric for face images, and choice of the partition algorithm. This system has two important applications firstly grouping a

collection of face images when no external labels are associated with images, and secondly indexing for efficient large scale face retrieval.

In identity clustering we need to perform unsupervised learning because we have only the faces themselves with no names/labels. From there we need to identify and count the number of unique people in a dataset. This system takes a dataset of images as input. This system first extract and quantify the faces in a dataset. Then cluster the faces, where each resulting cluster (ideally) represents a unique individual. This system can be used by Law enforcement agencies to find all unique faces in video feeds for catching suspects and criminals. The implementation of the system is done using Deep Learning and Python.

In this project we are using a density-based or graph-based clustering algorithm that can not only cluster the data points but can also determine the number of clusters as well based on the density of the data. For this we are using DBSCAN algorithm that works by grouping points together that are closely packed in an N-dimensional space. Points that lie close together will be grouped together in a single cluster. DBSCAN also naturally handles outliers, marking them as such if they fall in low-density regions where their "nearest neighbours" are far away.

1.3 Objective

The objective of the project is to design an Identity Clustering System which cluster face images according to their identity. The goals of this application are

- To find the number of unique faces in dataset.
- To run an automated search to compare faces to a known dataset of images.
- To speed up the task of finding the unique faces in large dataset.
- The main objective of this system is to cluster the faces where each resulting cluster represents a unique individual.

CHAPTER 2

SYSTEM ANALYSIS AND FEASIABILTY STUDY

2.1 Software Engineering Paradigm

The **software engineering paradigm** which is also referred to as a **software** process model or **Software** Development Life Cycle (SDLC) model is the development strategy that encompasses the process, methods and tools.

2.1.1 Agile Development Model

A discussion issue among software engineering main players is how much to involve customers with the different stages of the SDLC and how much to overlap these phases. This is the issue that led to the development of the Agile development model. The motivation of the development of the Agile development model is the customer

2 Concept - Projects are envisioned and prioritized

3 Inception - Team members are identified, funding is put in place, and initial environments and requirements are discussed

4 Iteration/Construction - The development team works to deliver working software based on iteration requirements and feedback

5 Release - QA (Quality Assurance) testing, internal and external training, documentation development, and final release of the iteration into production

6 Production - Ongoing support of the software

7 Retirement - End-of-life activities, including customer notification and migration

This view presents the full Agile lifecycle model within the enterprise. In any enterprise there may be projects operating simultaneously, multiple sprints/iterations being logged on different product lines, and a variety of customers, both external and internal, with a range of business needs.

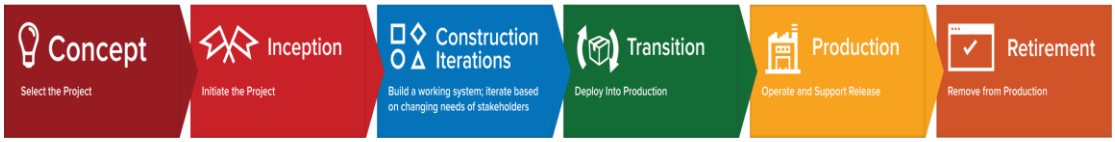


Fig 2.1 Agile Software Development Lifecycle

2.2 Phases of Development Life Cycle:

The Agile software development lifecycle is dominated by the iterative process. Each iteration results in the next piece of the software development puzzle - working software and supporting elements, such as documentation, available for use by customers - until the final product is complete. Each iteration is usually two to four weeks in length and has a fixed completion time. Due to its time-bound nature, the iteration process is methodical and the scope of each iteration is only as broad as the allotted time allows.

Multiple iterations will take place during the Agile software development lifecycle and each follows its own workflow. During an iteration, it is important that the customers and business stakeholders provide feedback to ensure that the features meet their needs.

- **Requirements** - Define the requirements for the iteration based on the product backlog, sprint backlog, customer and stakeholder feedback
- **Development** - Design and develop software based on defined requirements
- **Testing** - QA (Quality Assurance) testing, internal and external training, documentation development
- **Delivery** - Integrate and deliver the working iteration into production
- **Feedback** - Accept customer and stakeholder feedback and work it into the requirements of the next iteration

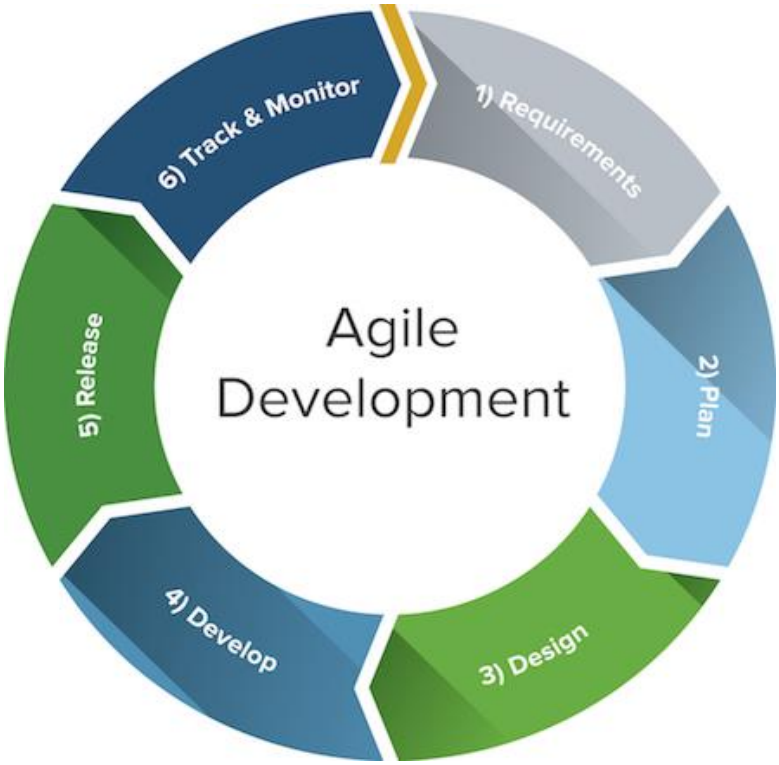


Fig 2.2 Agile Development

2.3 System Analysis

Basically, System Analysis is defined as "The process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". The field of system analysis relates closely to requirements analysis or to operational Research. System Analysis is divided into percent phases:

- Scope Definition: denoting an instrument for observing, viewing or examining.
- Problem Analysis: Analysing the problem that arises.
- Requirements Analysis: determining the conditions that need to be met.
- Logical Design: looking at the logical relationship among the objects.
- Decision Analysis: making a final decision.

You need to implement these blog features if you want to succeed. If you're confused on any of the features, don't worry.

2.3.1 Feasibility Study of the Project

The Feasibility of the project is analysed in this phase and the proposed solution is put forth with a very general plan for the project and some cost estimates. During the system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the organisation/college. For Feasibility analysis, some understanding of the major requirements for the systems is essential. The key considerations involved in feasibility study are: -

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

2.3.2 Operational Feasibility

This refers to the ability of a system to perform all its operations effectively and efficiently. This application is developed using Python. The user interface of the application is kept simple and understandable. A common user can easily understand the functionality of this application. Our system is satisfying the requirements identified in the requirement analysis phase of system development. It is solving the problems well and takes advantage of the opportunities identified during scope development.

2.3.3 Technical Feasibility

This project is technical feasible as working on a laptop with the use of Ubuntu or windows operating system and running python program on these operating system is easy. As python is an open source and high level programming language and easy to understand. The present technology assures technical guarantee of accuracy, reliability and ease of access. The study is carried out to check the technical feasibility i.e. the technical requirements of the system. The hardware needed to carry out this project includes 4GB or above RAM, Processor i3 or above. The software needed to carry out this project includes Windows 7 or higher configuration. So the technology required to carry out the project is easily available and affordable, hence this project is technically feasible.

2.3.4 Economical Feasibility

Economic analysis could also be referred to as cost/benefit analysis. It is the most important method for evaluating the effectiveness of the system. Our system is economically feasible as the project is economically possible in the given resource contents. Operating system (Ubuntu), Python, visual studio, python bindings (Pandas, Numpy, SkLearn, OpenCV, Dlib), python interpreter – anaconda are open source software programs. It requires basis hardware configuration laptop or desktop. As it is a college level project so we can say that it is economically feasible to implement.

2.4 Proposed Modules

2.4.1 Face Extraction:

This face extraction module crops the face from the images in the dataset.

2.4.2 Facial Encoding:

Before we can cluster a set of faces we first need to quantify them. This process of quantifying the face will be accomplished using a deep neural network responsible for:

- Accepting an input image
- And outputting a 128-d feature vector that encode the face

2.4.3 Identity Clustering:

Now that we have quantified and encoded all faces in our dataset as 128-d vectors, the next step is to cluster them into groups.

The problem is, many clustering algorithms such as k-means and Hierarchical Agglomerative Clustering, require us to specify the number of clusters we seek ahead of time.

Therefore, we need to use a density-based or graph-based clustering algorithm that can not only cluster the data points but can also determine the number of clusters as well based on the density of the data**.**

2.5 Hardware and Software Requirements

2.5.1 Hardware requirements

- **Graphical Processing Unit (GPU)** - A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. We require Intel HD Graphics 520 for implementation of our project implementation.

- **Random Access memory (RAM)** - We require a RAM of at least 4 GB because anaconda installation require more than 2 GB free.

- **Storage device (HARD DISK)** -We require a hard disk of minimum 512 GB.

- **Processor**- There is the minimum requirement of Intel Pentium or Above Processor.

2.5.2 Software Requirements:

• **Operating System** – The low-level software that supports a computer's basic functions, such as scheduling tasks and controlling peripherals. We can work on ubuntu as well as windows 7.

• **Python** –Python is an interpreted language, high level and general purpose language python is dynamically typed and garbage collected used in many programming paradigms includes object oriented programming, procedure programming. We require python 3.3+.

• **Python Bindings**- there are some bindings of python that we need to import

- Pandas 0.25.0
- Numpy 1.17.0
- Dlib 19.17.0
- Sklearn 0.21.3

• **Open CV**- The most popular and probably the simplest way to detect picture using python is by using open-cv. Open CV provides bindings for python. The machine learning algorithm search for faces within the pictures. Faces are very complicated, made up of thousands of small patterns and features which are needed to be matched for a face to be detected.The face recognition algorithm breaks the task of identifying the face into thousands of smaller, bit-sized tasks, each of which is easy to solve ,known as classifiers.

• **Anaconda-python**- together with the list of python packages, tools like editors, python distributions include the python interpreter. Anaconda is a new distribution of python and R data science package. The work environment is used for scientific computing, data science, statically analysis and machine learning.

CHAPTER 3

SYSTEM DESIGN

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. "how to implement?"

It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.

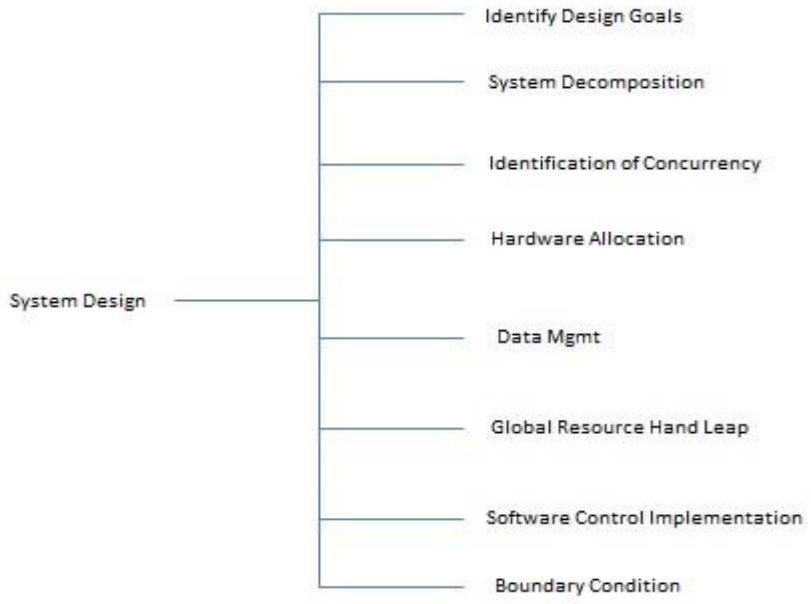


Fig 3.1 System Design

3.2 Features of the system

- Simple method.
- Light weight code.
- It run an automated search to compare faces to a known database of criminals.
- It could be used to aid to law enforcement.
- Easy implementation due to readily available hardware and software.
- Cost effective.

3.3 Future Scope

There is still much work to be done to further improve this algorithm. Since in identity clustering, it's very difficult to get perfect clusters, because of this we performed post-clustering filtering. In the future, we would like to implement additional outlier detection and suppression techniques to experiment with and see which approaches most effectively serve our purposes of removing incorrectly clustered data. One possible approach is to use sum of squared errors with the median face instead of the mean face. Likewise, we could incorporate Scikit-Learn's novelty and outlier detection algorithm to identify images that don't belong in their respective clusters.

Lastly, we would like to test our algorithm on multiple data sets to see how it behaves in various environments. There are several datasets, that other researchers in this area have used. Each data set works under different assumptions, therefore we would be able to observe the limits of our algorithm. Datasets that others researching face image clustering have used include

1. Labelled Faces in the Wild
2. YouTube Faces
3. Web faces
4. CASIA-web faces

CHAPTER 4

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively.

4.1 Coding

- Encode_faces.py

```
# import the necessary packages
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
	help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
```

```
        help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# grab the paths to the input images in our dataset, then initialize
# out data list (which we'll soon populate)
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
        # load the input image and convert it from RGB (OpenCV ordering)
        # to dlib ordering (RGB)
        print("[INFO] processing image {}/{}".format(i + 1,
                len(imagePaths)))
        print(imagePath)
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # detect the (x, y)-coordinates of the bounding boxes
        # corresponding to each face in the input image
        boxes = face_recognition.face_locations(rgb,
                model=args["detection_method"])

        # compute the facial embedding for the face
```

```
            encodings = face_recognition.face_encodings(rgb, boxes)

            # build a dictionary of the image path, bounding box location,
            # and facial encodings for the current image
            d = [{"imagePath": imagePath, "loc": box, "encoding": enc}
                    for (box, enc) in zip(boxes, encodings)]
            data.extend(d)

    # dump the facial encodings data to disk
    print("[INFO] serializing encodings...")
    f = open(args["encodings"], "wb")
    f.write(pickle.dumps(data))
    f.close()
```

- **Cluster_faces.py**

```
    # import the necessary packages
    from sklearn.cluster import DBSCAN
    from imutils import build_montages
    import numpy as np
    import argparse
    import pickle
    import cv2
    import os
```

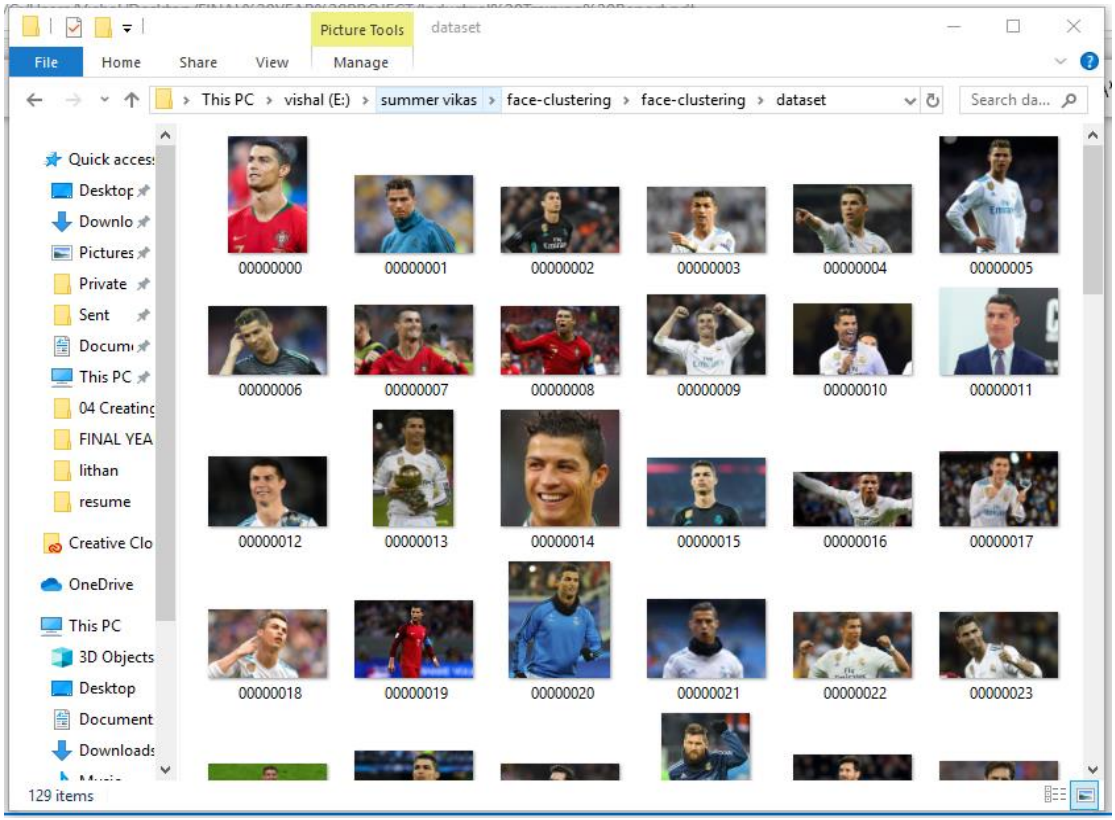
```
# construct the argument parser and parse the argument

ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
help="path to serialized db of facial encodings")
ap.add_argument("-j", "--jobs", type=int, default=-1,
help="# of parallel jobs to run (-1 will use all CPUs)")
args = vars(ap.parse_args())

# load the serialized face encodings + bounding box locations from
# disk, then extract the set of encodings to so we can cluster on
# them
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())
data = np.array(data)
encodings = [d["encoding"] for d in data]

# cluster the embeddings
print("[INFO] clustering...")
clt = DBSCAN(metric="euclidean", n_jobs=args["jobs"])
clt.fit(encodings)
```

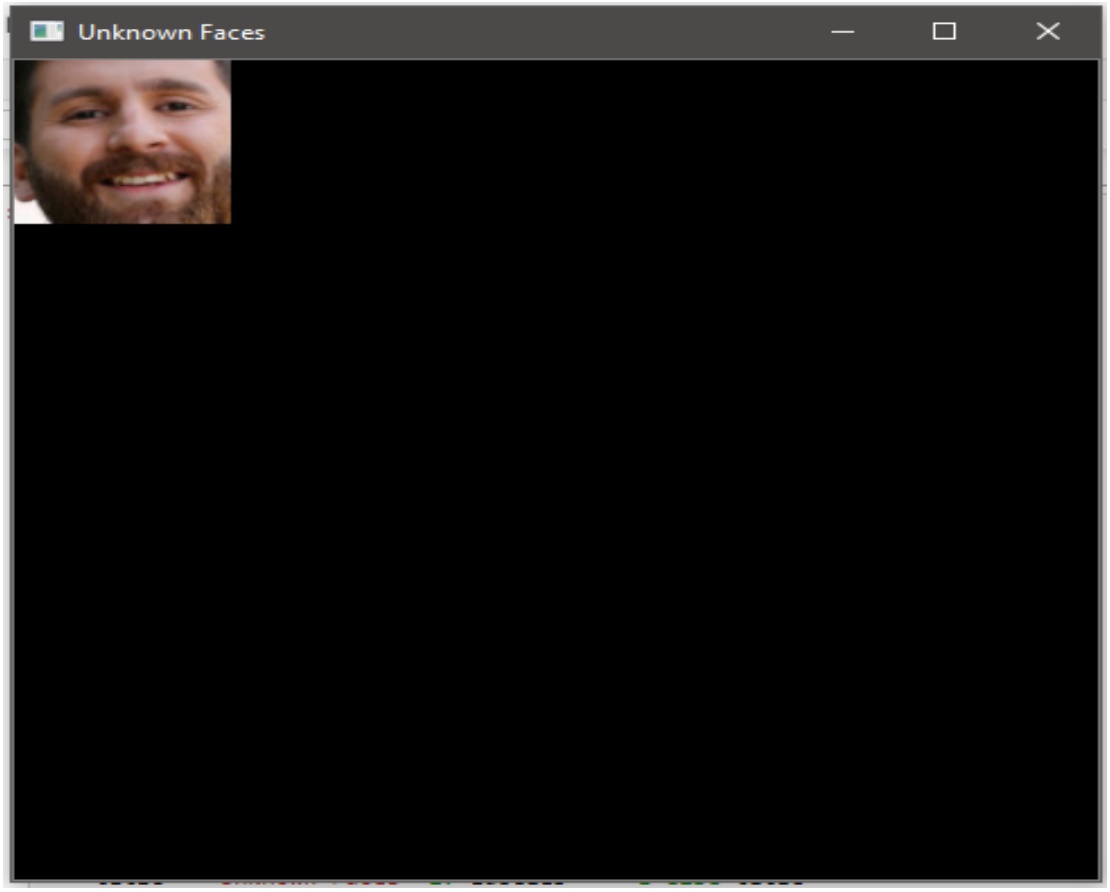
```
# determine the total number of unique faces found in the dataset
labelIDs = np.unique(clt.labels_)
numUniqueFaces = len(np.where(labelIDs > -1)[0])
print("[INFO] # unique faces: {}".format(numUniqueFaces))

for labelID in labelIDs:
target=os.path.join("Clusteredataset",str(labelID))
os.makedirs(target)
idxs = np.where(clt.labels_ == labelID)[0]
i=1
for i in idxs:
image = cv2.imread(data[i]["imagePath"])
imgname="fig"+str(i)+".jpg"
p=os.path.join(target,imgname)
i=i+1
cv2.imwrite(p,image)

# loop over the unique face integers
for labelID in labelIDs:
# find all indexes into the `data` array that belong to the
# current label ID, then randomly sample a maximum of 25 indexes
# from the set
print("[INFO] faces for face ID: {}".format(labelID))
```

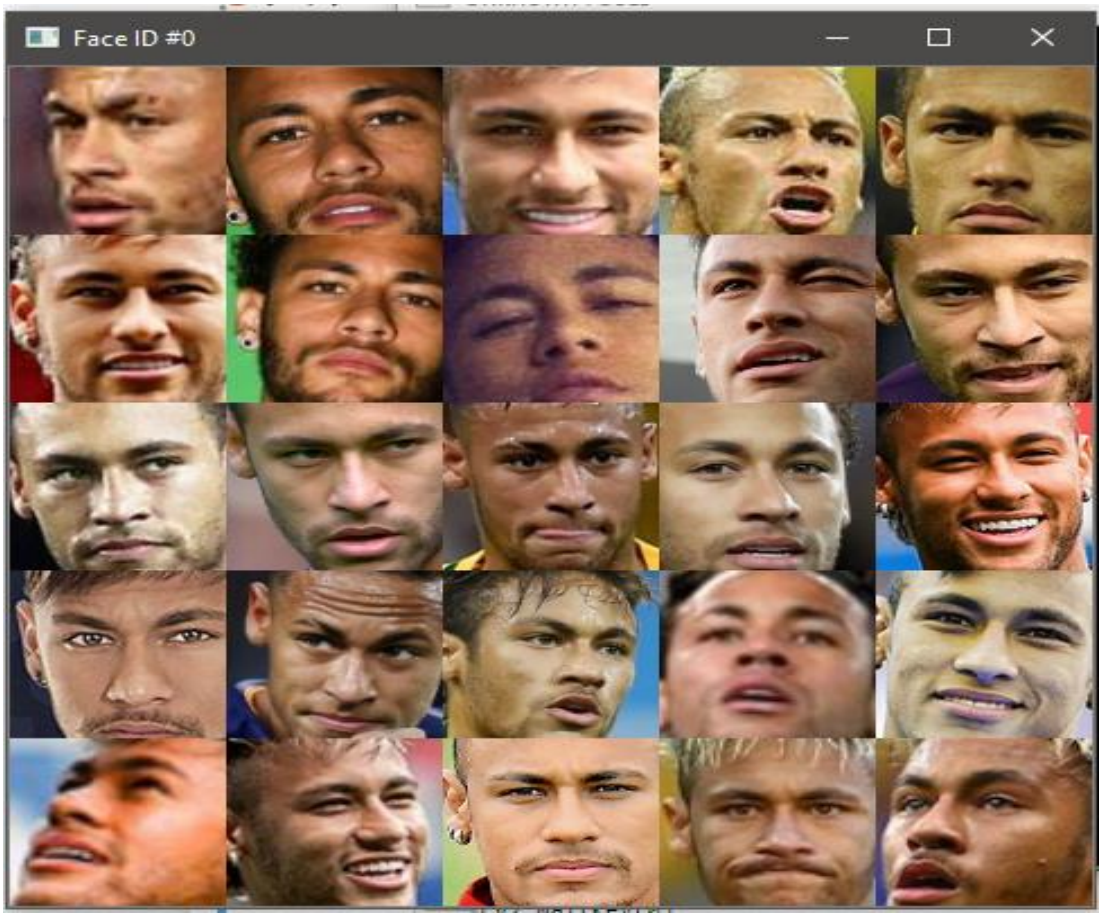
```
idxs = np.where(clt.labels_ == labelID)[0]

idxs = np.random.choice(idxs, size=min(25, len(idxs)),

replace=False)


# initialize the list of faces to include in the montage

faces = []


# loop over the sampled indexes

for i in idxs:

# load the input image and extract the face ROI

image = cv2.imread(data[i]["imagePath"])

(top, right, bottom, left) = data[i]["loc"]

face = image[top:bottom, left:right]


# force resize the face ROI to 96x96 and then add it to the

# faces montage list

face = cv2.resize(face, (96, 96))

faces.append(face)


# create a montage using 96x96 "tiles" with 5 rows and 5 columns

montage = build_montages(faces, (96, 96), (5, 5))[0]


# show the output montage
```

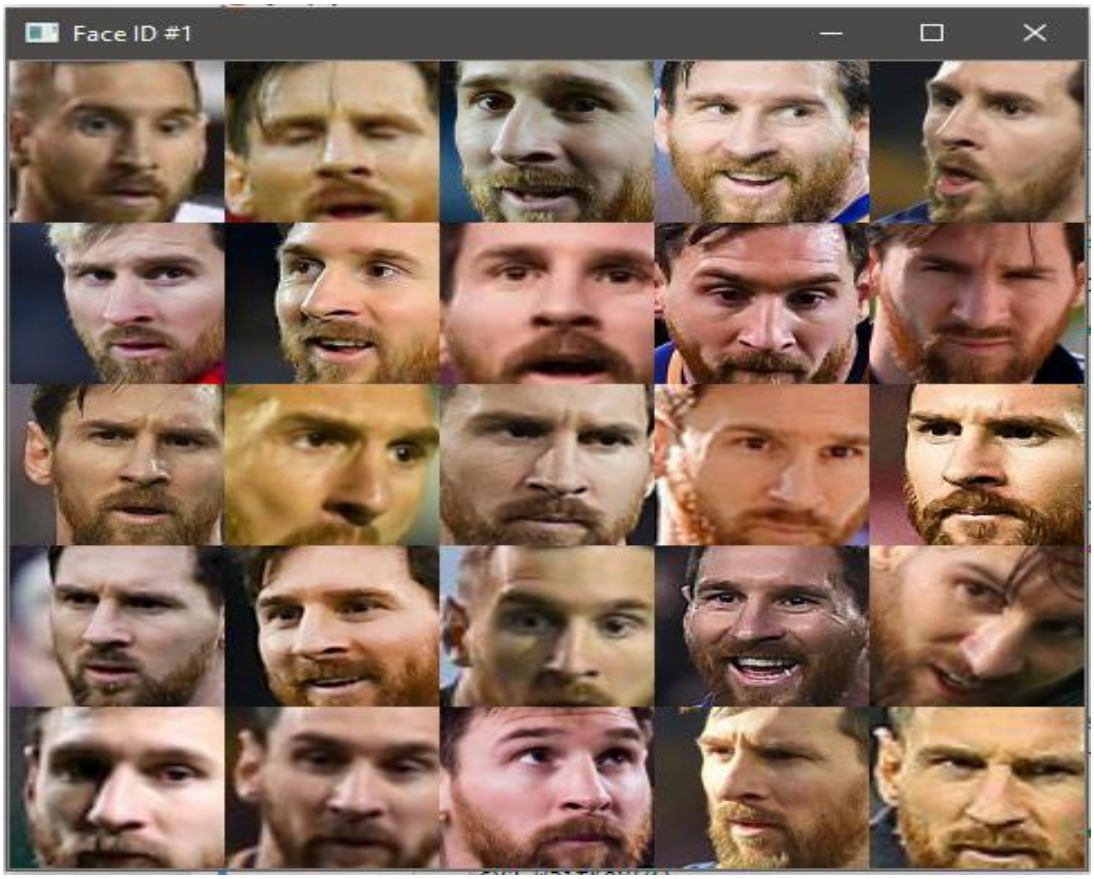
```
title = "Face ID #{}".format(labelID)

title = "Unknown Faces" if labelID == -1 else title

cv2.imshow(title, montage)

cv2.waitKey(0)
```

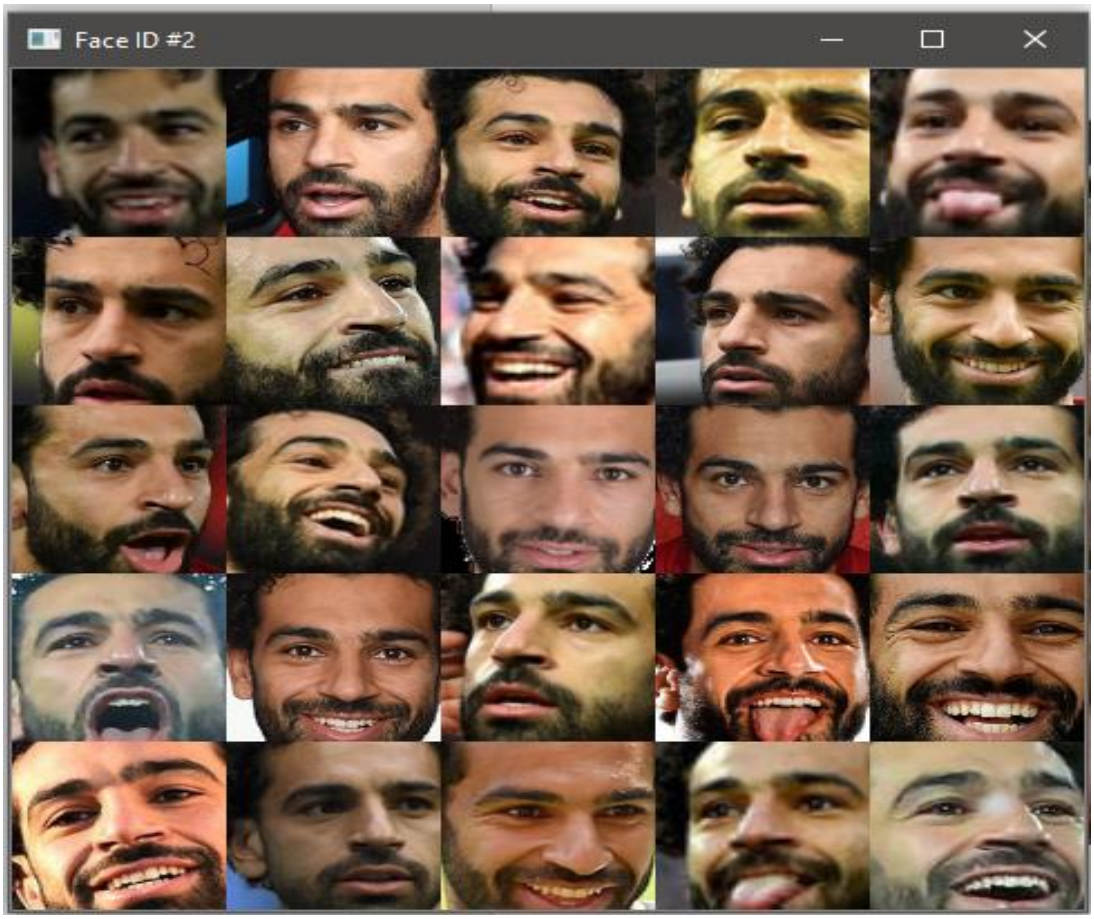
4.2 Project Screenshots

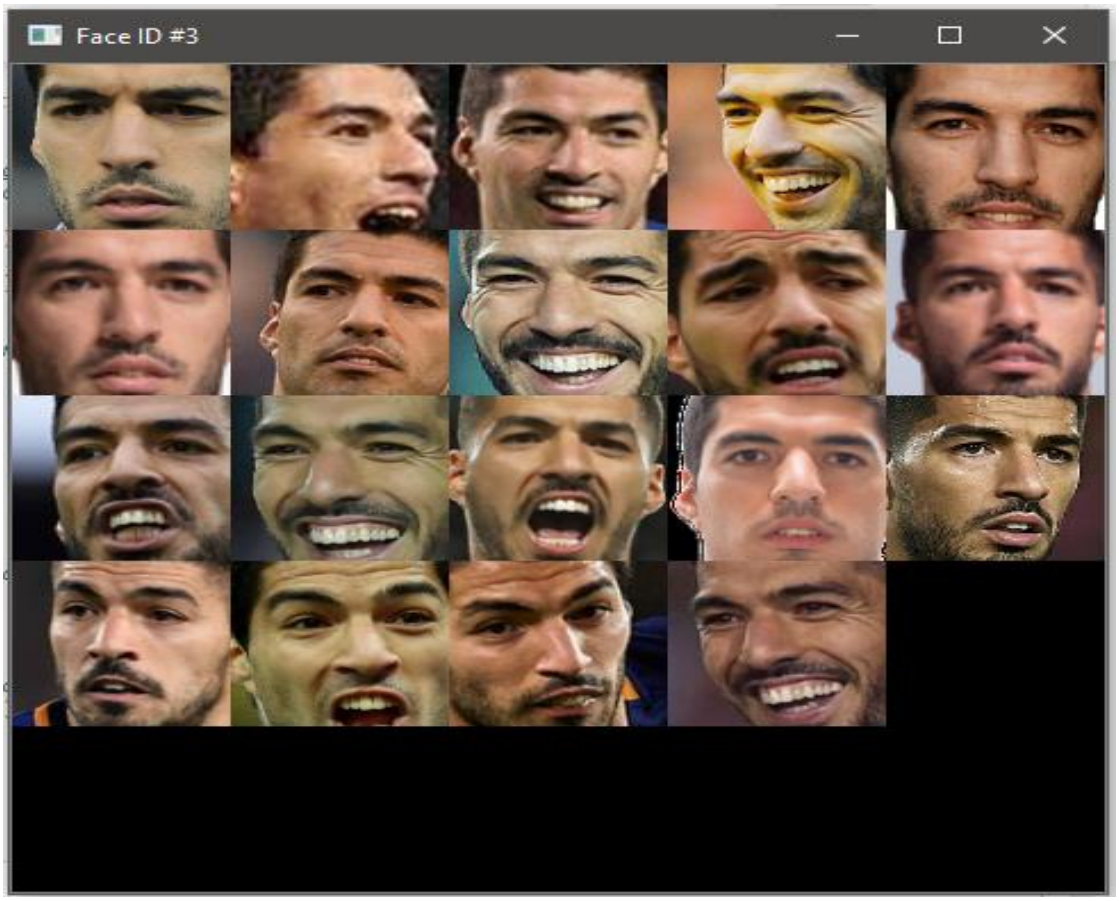


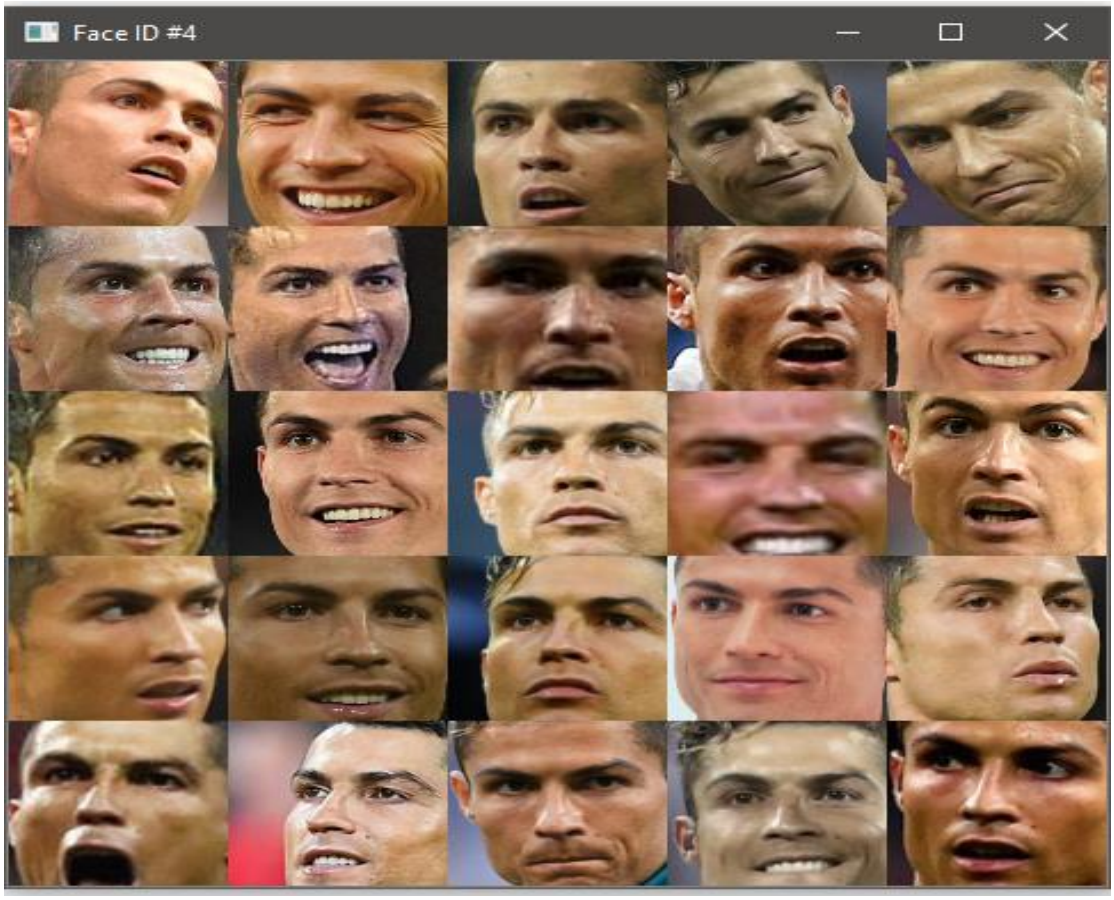

Unknown Faces

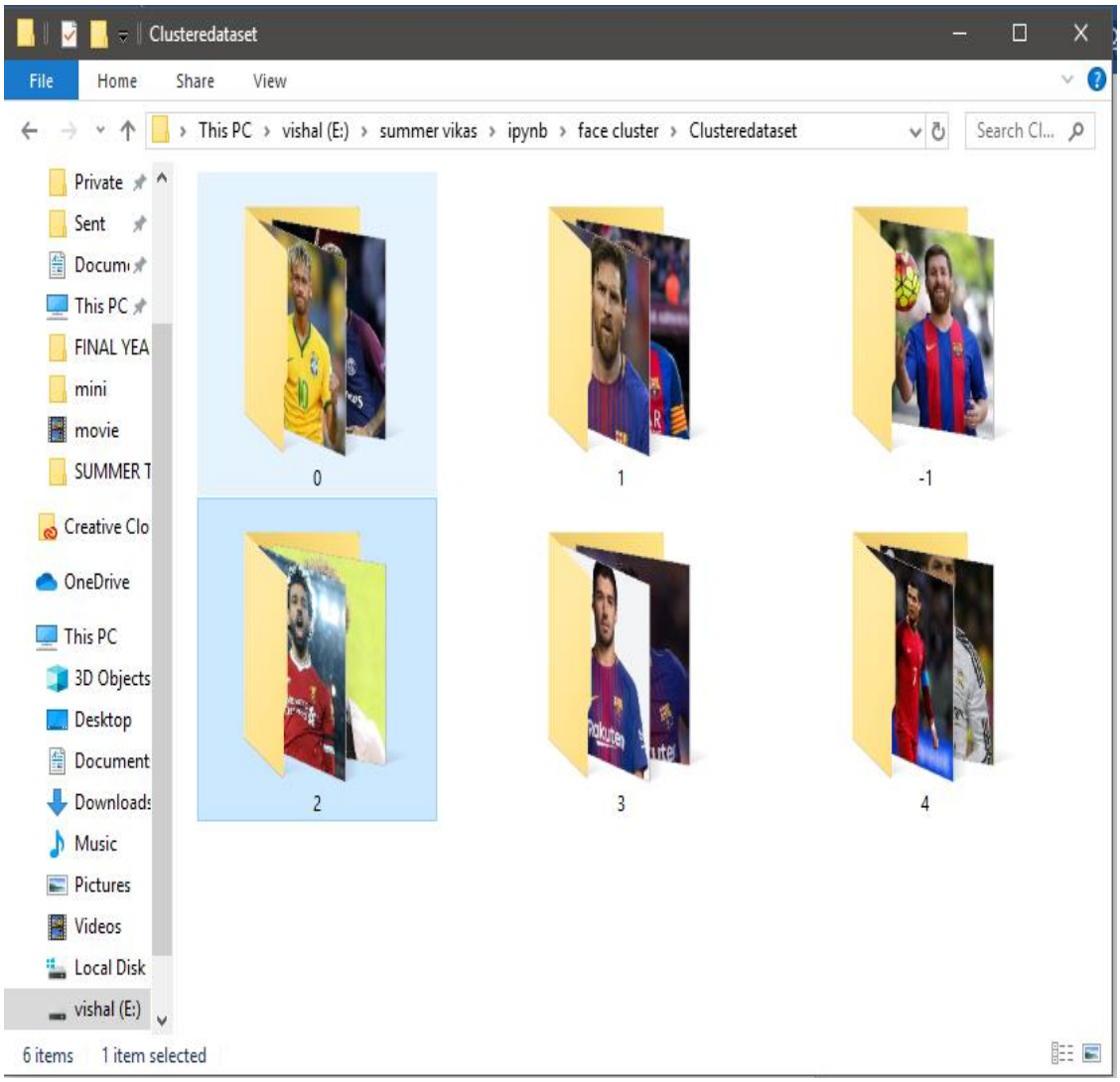

Face ID #0


Face ID #1


Face ID #2


Face ID #3


Face ID #4


Clusteredataset
File
Home
Share
View
This PC › vishal (E:) › summer vikas › ipynb › face cluster › Clusteredataset
Search Cl...
Private
Sent
Documı
This PC
FINAL YEA
mini
movie
SUMMER T
Creative Clo
OneDrive
This PC
3D Objects
Desktop
Document
Downloads
Music
Pictures
Videos
Local Disk
vishal (E:)
0
1
-1
2
3
4
6 items
1 item selected

CHAPTER 5

TESTING

5.1 INTRODUCTION

Testing is the process of identifying the correctness and quality of software program. The purpose is to check whether the software satisfies the specific requirements, needs and expectations of the customer. In other words, testing is executing a system or application in order to find software bugs, defects or errors. The job of testing is to find out the reasons of application failures so that they can be corrected according to requirement

5.2 Scope

The overall purpose of testing is to ensure the “Identity Clustering” meets all of its functional requirements. The purpose of this chapter is to describe the overall test plan and strategy for testing the system.

5.3 Testing Goals

The goals in testing this system include validating the quality, usability, reliability and performance of the application. Testing will be performed from a black-box approach. Tests will be designed around requirements and functionality.

5.4 Testing Techniques

Software testing techniques listed here are the major methods used while conducting various Software Testing Types during various Software Testing Levels.

Black Box: A software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Test design

techniques include Equivalence partitioning, Boundary Value Analysis, Cause-Effect Graphing.

White Box: A software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Test design techniques include Control flow testing, Data flow testing, Branch testing, Path testing.

5.4.1 Black Box Testing

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. Black-box test design treats the system as a “black-box”, so it does not explicitly use knowledge of the internal structure. Black-box test design is usually described as focusing on testing functional requirements. Synonyms for black-box includes: behavioral, functional, opaque-box and closed-box.

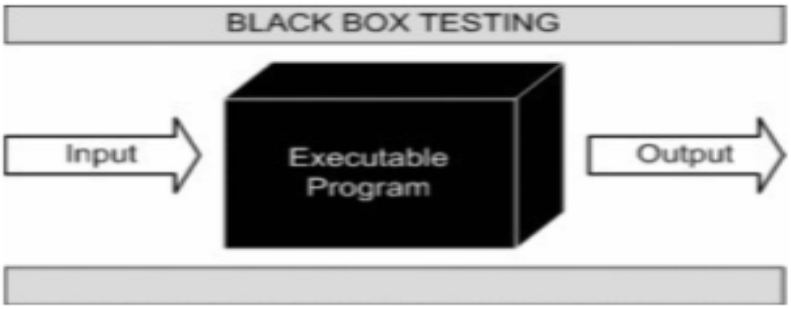


Fig.5.1 Black Box Testing

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see.

CHAPTER 6

ADVANTAGES & DISADVANTAGES

6.1 Advantages

- Simple method.
- Light weight code.
- It run an automated search to compare faces to a known database of criminals.
- It could be used to aid to law enforcement.
- Easy implementation due to readily available hardware and software.
- Cost effective.

6.2 Disadvantages

- System is required to run the code.
- Troubles with images size and quality.

CHAPTER 7

FUTURE ENHANCEMENT

There is still much work to be done to further improve this algorithm. Since in identity clustering, it's very difficult to get perfect clusters, because of this we performed post-clustering filtering. In the future, we would like to implement additional outlier detection and suppression techniques to experiment with and see which approaches most effectively serve our purposes of removing incorrectly clustered data. One possible approach is to use sum of squared errors with the median face instead of the mean face. Likewise, we could incorporate Scikit-Learn's novelty and outlier detection algorithm to identify images that don't belong in their respective clusters.

Lastly, we would like to test our algorithm on multiple data sets to see how it behaves in various environments. There are several datasets, that other researchers in this area have used. Each data set works under different assumptions, therefore we would be able to observe the limits of our algorithm. Datasets that others researching face image clustering have used include

1. Labelled Faces in the Wild
2. YouTube Faces
3. Web faces
4. CASIA-web faces

CHAPTER 8
CONCLUSION

In this project, we propose a novel algorithm to extract faces from a set of images and cluster them by identity. The two main contributions from this project. The first contribution is the face image clustering algorithm we designed using DBSCAN as the underlying face representation and affinity propagation as the clustering scheme. The second contribution from this project is the face image outlier detection and suppression algorithm that aims to maximize cluster purity by removing any bad clusters present in the clustering and remove all images that are incorrectly clustered. All things considered, our algorithm's performance meets the expectations we had at the set out of this project.

CHAPTER 9

REFRENCES

1. **https://www.anaconda.com/distribution/**
2. **https://beginnersbook.com/2018/01/introduction-to-pythonprogramming/**
3. **https://realpython.com/jupyter-notebook-introduction/**
4. **https://www.digitalocean.com/community/tutorials/an-introductionto-machine-learning**
5. **https://opensource.com/article/18/4/getting-started-anaconda-python**