

APPENDIX 1



OBJECT DETECTION USING DEEP LEARNING

A Project Report of Capstone Project – 2

Submitted By

Pushpendra Chaudhary

(1613101528/ 16SCSE101478)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTER SCIENCE&ENGINEERING

Under the Supervision of

Dr.Kuldeep Singh Kaswan(Professor)

APRIL/ MAY 2020

APPENDIX 2



SCHOOL OF COMPUTER SCIENCE&ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report “OBJECT DETECTION USING DEEP LEARNING” is the bonafide work of “PUSHPENDRA CHAUDHARY(1613101528)” who carried out the project work under my supervision.

SIGN. OF HEAD

DR.MUNISH SHABARWAL,

Professor & Dean,

School of Computer Science

& Engineering

SIGN. OF SUPERVISOR

DR.KULDEEP KASWAN,

Professor

School of Computer Science

& Engineering

Abstract

Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision consists of various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, we are using highly accurate object detection-algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms, based on deep learning which is also based on machine learning require lots of mathematical and deep learning frameworks understanding by using dependencies such as TensorFlow, OpenCV, imageai etc, we can detect each and every object in image by the area object in an highlighted rectangular boxes and identify each and every object and assign its tag to the object. This also includes the accuracy of each method for identifying objects.

APPENDIX 3

TABLE OF CONTENTS

<u>CHAPTER NO.</u>	<u>TITLE</u>	<u>PAGE NO.</u>
	Title of Project	1
	Bonafide Certificate	2
	Abstract	3
1.	Introduction	4
1.1	Facial Recognition	4
1.2	Facial Recognition Process	4
1.3	Facial Biometric Systems used for Security	5-10
1.4	Objectives	11
1.5	Proposed System	11
2.	Background Study	12
2.1	Python	12
2.2	Anaconda	13
3.	Technologies Used	14
3.1	Python 3.7	14
3.2	Anaconda 2019.07	14-15
3.3	Pandas	15
3.4	Numpy	16
3.5	Matplotlib	16
3.6	Scikit-learn	17

3.7	OpenCV- Python	17
3.8	Scikit-image	18
3.9	Joblib	19
3.10	Tkinter	19
3.11	Pil	20
3.12	Scipy	20
4.	Installing all dependencies	21
4.1	Python 3	21-26
4.2	Installing Anaconda	26-29
4.3	Accessing Jupyter Notebook	29-30
5.	Requirement for Project	31
5.1	Hardware Requirements	31
5.2	Software Requirements	31
6.	Implementation	32
6.1	Haar Cascade	32
6.2	HOG Descriptor	32
6.3	Datasets	33
6.4	Graphic User Interface	33-34
6.5	Maintaining Records	35
6.6	Working	35-36
6.7	Training a Model	37
6.8	Code	38-52
7.	Concerns and Future Works	53
7.1	The Threats and Concerns About Facial Recognition	53-54
7.2	How to Avoid Facial Recognition	54-57
7.3	What the Future Holds?	57
	References	58

Chapter 1

INTRODUCTION

1.1 Facial Recognition

Face recognition is a biometric solution designed for the purpose of recognizing a human face without any physical contact required. The solution runs through algorithms that match the facial nodes of a person to the ./images saved in the database[1]. Security of any organization or critical location can be enhanced using facial recognition. The versatile nature of facial recognition makes it a preferred choice for added security. Human face detection is preliminary required step of face recognition systems as well as a very important task in many applications, such as security access control systems, video surveillance, human computer interface and image database management. Due to the advancements in face detection technology, it is now possible to detect faces in an image or video, regardless of head pose, lighting conditions, and skin colour[2].

Today it's considered to be the most natural of all biometric measurements. And for good reason – we recognize ourselves not by looking at our fingerprints or irises, for example, but by looking at our faces.

1.2 Facial Recognition Process

Unlike any other identification solution, face recognition identifies the unique features of the human face and then makes a comparison based on the existing database of photographs. Sensors detect and identify face shapes by the colour of the iris, nose shape, and so on. Identifying the human face includes concentrating on certain unique features, such as the jaw, cheekbones, face shape and so on[3]. Once the image in the database matches with the face of the person concerned, the face is verified. It captures, analyses and compares patterns based on the person's facial details.

1. The **face detection** process is an essential step as it detects and locates human faces in images and videos.
2. The **face capture** process transforms an analog information (a face) into a set of digital information (data) based on the person's facial features.
3. The **face match** process verifies if two faces belong to the same person.

Today it's considered to be the most natural of all biometric measurements.

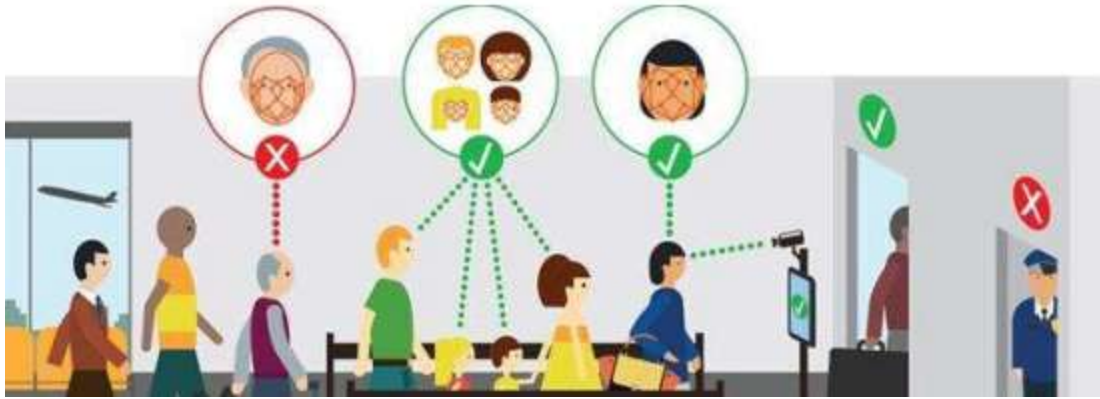


Figure 1.1: Facial Authentication is easy to deploy and implement no physical interaction required by the end-user.

1.3 Facial biometrics system used For Security

Facial biometrics system has been used as a measure of securities in the topmost institutions and workplaces to ensure that there is no scope of any vandalism. This type of software leaves absolutely no room for human error and is a major helping hand. Just by a set of algorithms, the software does geometric and photometric recognition within seconds. These facial biometrics system has emerged as the master of all recognition software due to its easy applicability and low-cost technology. Its non-contact nature is the best thing about it in the sense that a person through facial recognition, even in a crowded place can be recognized, given that his ./images are saved in the database[2]. Facial recognition makes access to information more limited and restricted to those who own it. Facial recognition has made verification relatively easier, with nothing much to equip and a lot of information to access within minutes.



Figure 1.2: Face recognition solution has been as a major component in the field of security.

The Face recognition solution has been as a major component in the field of security. Here's why:

- **Criminal identification**

The authorities can breathe a sigh of relief with the face recognition system. Its database containing all information about criminals makes it easier to catch them. If the sensor identifies the face with the algorithms and if the face matches, it's a win! Face recognition software prevents a crime even before it takes place[5].

- **Surveillance**

Crimes are not committed when you have someone watching over you. Facial recognition keeps a track on everybody in crowded places. Because of CCTV surveillance cameras being installed in crowded areas, the crime rate has been much less[3].



Figure 1.3: Face recognition can identify all threats in real-time.

- **Police Authorities**

Police stations have facial recognition systems to track people who have past criminal records and are wanted. The database when matches with a person's face, it is easier to get hold of criminals through simple algorithms. The police authorities are alerted if the system shows a face match[6].



Figure 1.4: Face recognition makes it easier for authorities to keep track of people on watchlist.

• **Tracking attendance**

Schools and colleges have adopted face recognition both to track attendance and avoid any

malicious activity in premises.



Figure 1.5: Face recognition will make restricted entry seamless.

• **Defence Services**

Defence services use face recognition because of the degree of sensitivity involved. Since only a few people have some confidential information hidden with them, with facial recognition, only they can have access to it.



Figure 1.6: Face recognition can prevent crimes before they happen.

- **Bank Services**

Banks use this product of Artificial intelligence as a security measure to detect any suspects entering without being identified. Basically, the artificial intelligence technology employed in banks is to avoid bank frauds.

- **online payment**

Security also contains safe online payments. Since each face is unique like a fingerprint, there is no chance that your payment will be hacked, as the payment will be made once your face matches.

- **Airport Service**

In many countries, airports use this system of artificial intelligence to recognize faces of passengers so that there are no suspected risks involved. Through facial recognition, the information obtained is authentic and avoids any possibility of error.



Figure 1.7: Face recognition systems can track many people simultaneously and identify all threats in realtime.

1.4 Objectives

To build a prototype model of Surveillance System, for use by institutions and workplaces to ensure that there is no scope of any vandalism. The project involves monitoring and safety authentication on video feeds. The Project is based

- On a model that recognise faces and classifies people.
- On a model that detects gender of recognised people on basis of their faces
- On a model that detects the emotions of basis of their faces

1.5 Proposed System

Need to build a prototype for Surveillance System using a Python 3 and OpenCV. The Proposed System should have the following functions:

- It should draw bounding boxes around the faces detected.
- It should be able to create and manage datasets on its own.
- It should have a GUI through which users can access any one model or either all of the model at same time for recognition.
- It should be able to recognise partial faces.
- It should be able to maintain a record file that at which time particular person is present in the video feed
Create wiring diagram of expected design.
- It should be able to create random colour on basis of the category of detected class/objects and draw their bounding boxes
- It should be able to change pixel values into an image and resize them and save them on hard drive so that they can be used later on

Chapter 2

BACKGROUND STUDY

2.1 Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum. It is used at a number of places as:

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

2.1.1 Reasons for Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.
- Python comes with a huge number of inbuilt libraries. Many of the libraries are for Artificial Intelligence and Machine Learning
- It's easy to experiment with new ideas and code prototypes quickly in a language with minimal syntax like Python

2.2 Anaconda

Anaconda is a python distribution, with installation and package management tools. It provides large selection of packages and commercial support. It is an environment

manager, which provides the facility to create different python environments, each with their own settings. It also provides much greater advantages in the data science platform.

Conda, the Anaconda's own package manager, is used for updating anaconda and its packages. Conda is a cross platform package and environment manager. It provides installing, executing and updating different packages along with their dependencies. It helps in switching between environments in our local machine

2.2.1 Reasons for Anaconda

- Anaconda python is faster than vanilla python: they bundle Intel MKL and this does make most NumPy computations faster.
- You can easily do a local user install, no need to ask permission from your admin in many cases.
- Under Windows, you don't have a lot of choices and anyway you need a python package installer.
- Anaconda Inc. is a company. This is a plus in a corporate setting. You can get support contracts for instance.
- Anaconda Inc. has historically made a lot of efforts to please the personal, students and academic users. This has basically worked out pretty well for all concerned.
- Anaconda Python is very complete.
- There is no risk of messing up required system libraries

Chapter 3

TECHNOLOGIES USED

3.1 Python 3.7

Before 2008, Python developers had a bit of a headache. The language that started in the 1989 Christmas holidays as the pet project of Guido van Rossum was now growing at a fast pace. Features had been piled on, and the project was now large enough that earlier design decisions were hindering implementation. Because of this, the process of adding new features was becoming an exercise in hacking around the existing code.

The solution was Python 3: the only release that deliberately broke backwards compatibility. At the time, the decision was controversial. Despite the backlash, the decision was taken, giving Guido and the developers a one-off chance to clean out redundant code, fix common pitfalls and re-architect the language. The aim was that within Python 3 there would be only one obvious way of doing things. It's a testament to the design choices made back then that we're still on 3.x releases a decade later.

3.2 Anaconda 2019.07

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling individual data scientists to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews



Fig 3.1: Some of the libraries made available with Anaconda that simplify Data Science Projects.

3.2.1 Technical Specification

- Intel Core i5-6200U dual core chipset
- 2.7GHz Dual-Core Processor
- 8GB RAM
- 64 Bit CPU
- 0.3 MegaPixel WebCam
- 240 GB SSD

3.3 Pandas

IR When it comes to data manipulation and analysis, nothing beats Pandas. It is *the* most popular Python library, period. Pandas is written in the Python language especially for manipulation and analysis tasks. The name is derived from the term “panel data”, an econometrics term for datasets that include observations over multiple time periods for the same individuals.

Installation code- **pip install pandas**

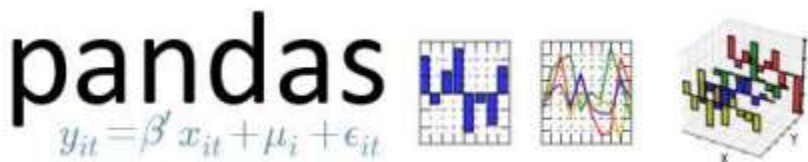


Figure 3.2: Pandas Library Logo

Pandas provides features like:

- Dataset joining and merging
- Data Structure column deletion and insertion
- Data filtration
- Reshaping datasets
- DataFrame objects to manipulate data, and much more!

3.4 Numpy

The NumPy, like Pandas, is an incredibly popular Python library. NumPy brings in functions to support large multi-dimensional arrays and matrices. It also brings in high-level mathematical functions to work with these arrays and matrices. NumPy is an open-source library and has multiple contributors. It comes pre-installed with Anaconda and Python.

installation code: **pip install numpy**



Figure 3.3: NumPy Library Logo.

3.5 Matplotlib

A Matplotlib is the most popular data visualization library in Python. It allows us to generate and build plots of all kinds. This is a standard go-to library for exploring data visually.

installation code: **pip install matplotlib**



Figure 3.4: Matplotlib Library Logo

3.6 Scikit-learn

Like Pandas for data manipulation and matplotlib for visualization, scikit-learn is the Python leader for building models. scikit-learn is built on NumPy, SciPy and matplotlib. It is open source and accessible to everyone and reusable in various contexts.

Scikit-learn supports different operations that are performed in machine learning like classification, regression, clustering, model selection, etc.



Figure 3.5: Scikit-learn Library Logo.

installation code: **pip install scikit-learn**

3.7 OpenCV-Python

When it comes to image processing, OpenCV is the first name that comes to mind. OpenCV-Python is the Python API for image processing, combining the best qualities of the OpenCV C++ API and the Python language. It is mainly designed to solve computer vision problems.

OpenCV-Python makes use of NumPy. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.



Figure 3.6: OpenCV-python Library Logo.

Installation code: **pip install opencv-python**

3.8 Scikit-image

Another python dependency for image processing is Scikit-image. It is a collection of algorithms for performing multiple and diverse image processing tasks. You can use it to perform image segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and much more. We need to have the below packages before installing scikit-image:

- Python (≥ 3.5)
- NumPy ($\geq 1.11.0$)
- SciPy ($\geq 0.17.0$)
- joblib (≥ 0.11)

installation code: **pip install scikit-learn**



Figure 3.7: Scikit-image Library Logo.

3.9 Joblib

Joblib is such an package that can simply turn our Python code into parallel computing mode and of course increase the computing speed. Joblib is a set of tools to provide lightweight pipelining in Python. In particular:

- transparent disk-caching of functions and lazy re-evaluation (memorize pattern) easy simple parallel computing
- Joblib is optimized to be fast and robust in particular on large data and has specific optimizations for numpy arrays

Installation code: **pip install joblib**



Figure 3.8: Joblib Library Logo.

3.10 Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Installation code: **pip install python-tk**



Figure 3.9: Tkinter Library Logo.

3.11 PIL

Python Imaging Library (abbreviated as **PIL**) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. Pillow offers several standard procedures for image manipulation. These include:

- per-pixel manipulations,
- masking and transparency handling,
- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- adding text to images and much more.

Installation code: **pip install Pillow**

3.12 Scipy

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Installation code: **pip install scipy**



Figure 3.11: SciPy Library Logo.

CHAPTER 4

Installing All Dependencies

4.1 Python3

Step 1: Select Version of Python to Install

- The installation procedure involves downloading the official Python .exe installer and running it on your system. It is recommended to download **the latest version of Python 3**.

Step 2: Download Python Executable Installer

- Open your web browser and navigate to the Downloads for Windows section of the official Python website.
- Search for your desired version of Python. At the time of writing, the latest Python 3 release is version 3.7.4.
- Select a link to download either the **Windows x86-64 executable installer** (for 64-bit systems) or **Windows x86 executable installer** (for 32 bit systems).



Figure 4.1: Download Python Executable Installer based on your chipset.

Step 3: Run Executable Installer

- Run the **Python Installer** once downloaded. (In this example, we have downloaded Python 3.7.3.)
- Make sure you select the **Install launcher for all users** and **Add Python 3.7 to PATH** checkboxes.
- Select **Install Now** – the recommended installation options.



Figure 4.2: Run Python Executable Installer.

- For all recent versions of Python, the recommended installation options include **Pip** and **IDLE**. Older versions might not include such additional features.
- The next dialog will prompt you to select whether to **Disable path length limit**. Choosing this option will allow Python to bypass the 260-character MAX_PATH limit. Effectively, it will enable Python to use long path names.

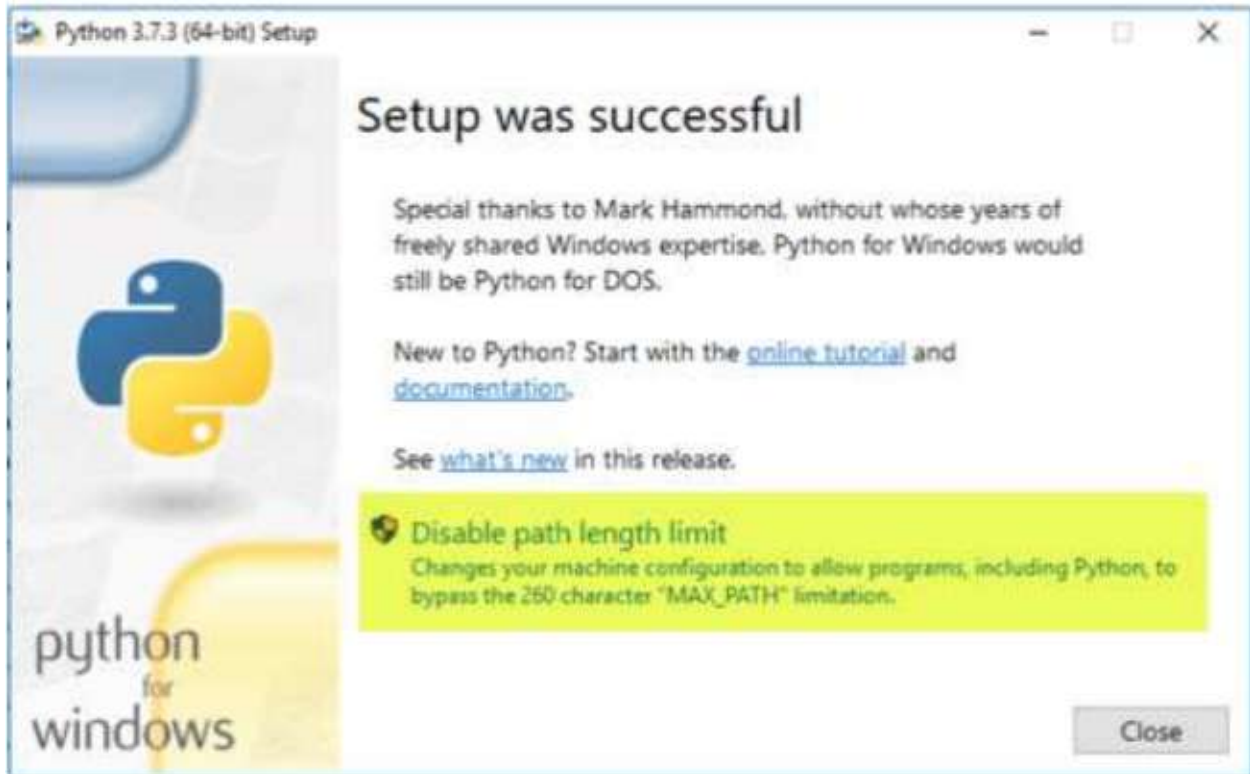
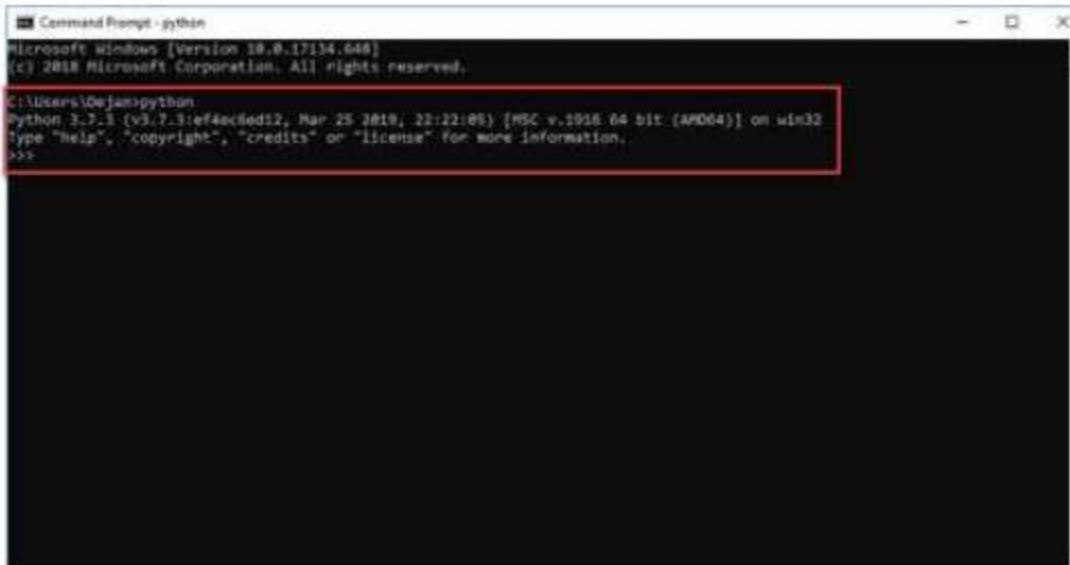


Figure 4.3: Disable Path length limit option in Python Executable Installer.

- The **Disable path length limit** option will not affect any other system settings. Turning it on will resolve potential name length issues that may arise with Python projects developed in Linux.

Step 4: Verify Python Was Installed on Windows

- Navigate to the directory in which Python was installed on the system. In our case, it is **C:\Users\Username\AppData\Local\Programs\Python\Python37** since we have installed the latest version.
- Double-click **python.exe**.
- The output should be similar to what you can see below:



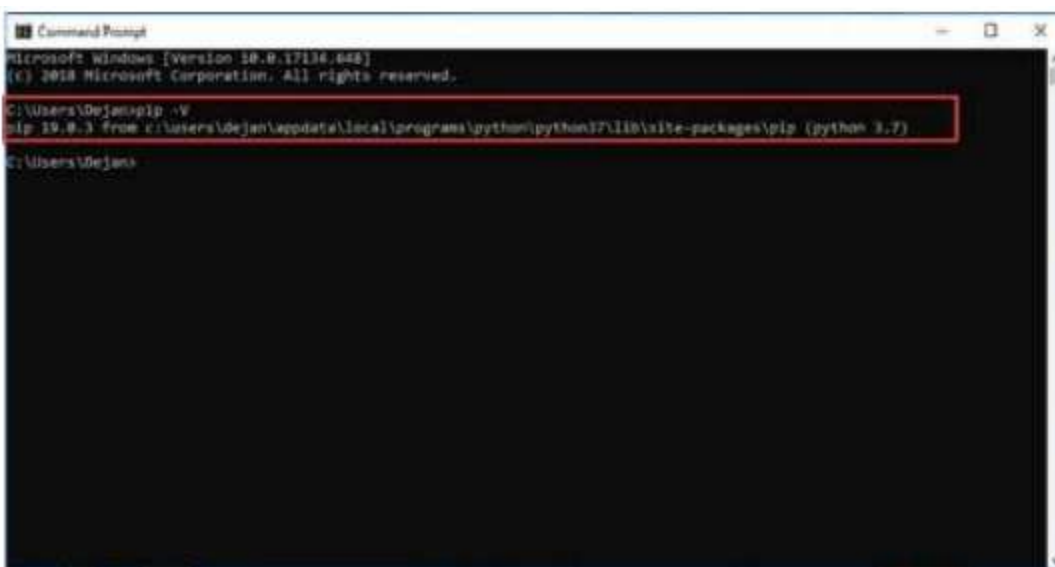
```
Command Prompt - python
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dejan>python
Python 3.7.3 (tags/v3.7.3:ef4aced12, Mar 25 2018, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Figure 4.4: Verify Python Was Installed on Windows

Step 5: Verify Pip Was Installed

- Open the **Start** menu and type “cmd.”
- Select the **Command Prompt** application.
- Enter **pip -V** in the console. If Pip was installed successfully, you should see the following output:



```
Command Prompt
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dejan>pip -V
pip 19.0.3 from c:\users\dejan\appdata\local\programs\python\python37\lib\site-packages\pip (python 3.7)

C:\Users\Dejan>
```

Fig 4.5: Verify pip Was Installed on Windows.

Step 6: Add Python Path to Environment Variables (Optional)

- We recommend you go through this step if your version of the Python installer does not include the **Add Python to PATH** checkbox or if you have not selected that option.
- Setting up the Python path to system variables alleviates the need for using full paths. It instructs Windows to look through all the PATH folders for “python” and find the install folder that contains the python.exe file.
- Open the **Start** menu and start the **Run** app.

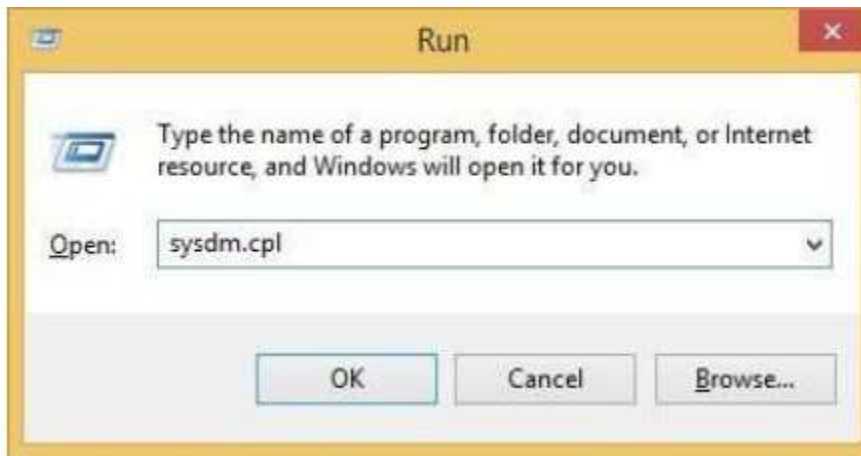


Fig4.6: Add Python Path to Environment Variables.

- Type **sysdm.cpl** and click **OK**. This opens the **System Properties** window.
- Navigate to the **Advanced** tab and select **Environment Variables**.
- Under **System Variables**, find and select the **Path** variable.
- Click **Edit**.
- Select the **Variable value** field. Add the path to the **python.exe** file preceded with a **semicolon (;)**. For example, in the image below, we have added “**;C:\Python34.**”



Figure 4.7: Edit System Variables

- Click **OK** and close all windows.
- By setting this up, you can execute Python scripts like this: **Python script.py**
- Instead of this: **C:/Python34/Python script.py**

4.2 Installing Anaconda

Anaconda is an open source distribution of the Python and R programming languages and it is used in data science, machine learning, deep learning-related applications aiming at simplifying package management and deployment. Anaconda Distribution is used by over 7 million users, and it includes more than 300 data science packages suitable for Windows, Linux, and MacOS.

Step 1: Visit Anaconda website to download anaconda for windows. Click on Download



Figure 4.8: Visit Anaconda website and choose python 3.

Step 2: Next, double click on setup and click on next.



Figure 4.9: Run Anaconda Installer.

Step 3: Click on “I agree” to Accept the agreement.

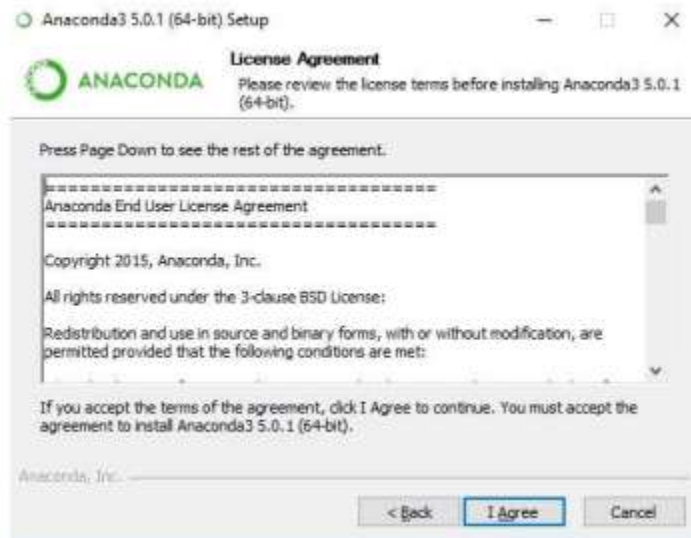


Figure 4.10: Agree Anaconda Agreement.

Step 4: Retain default location and click next.

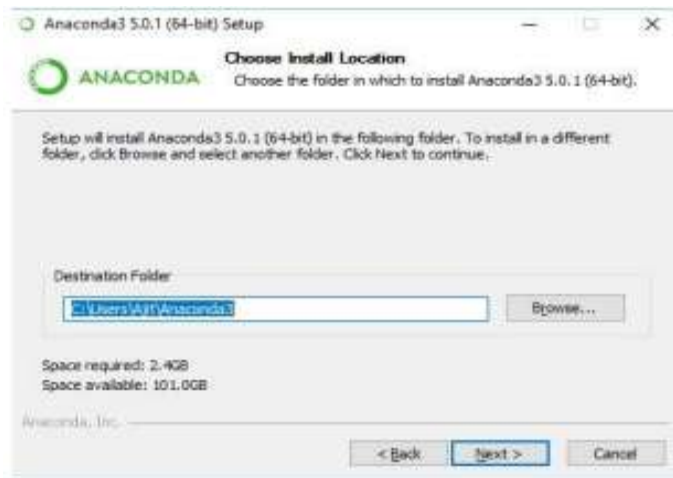


Figure 4.11: Choose Anaconda install location.

Step 5: Click on “Add anaconda to my Path environment variable”.

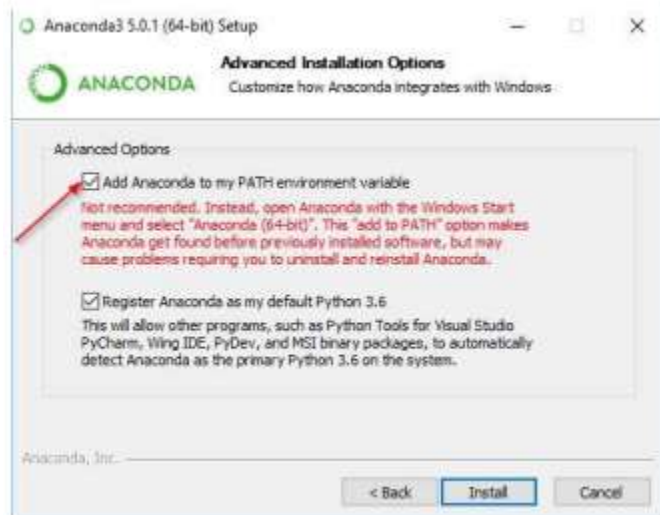


Fig 4.12: Add Anaconda to Environment Variables

Step 6: Click on the Install Button and wait till the installation is complete.



Figure 4.13: Install Anaconda.

Step 7: Click on finish.

4.3 Accessing Jupyter Notebook

To access Jupyter notebook, continue the below steps.

Step 1: Open command prompt and type “jupyter notebook”.

```
Command Prompt - jupyter notebook
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

E:\Users\Ajit>jupyter notebook
[I 16:35:15.317 NotebookApp] writing notebook server cookie secret to C:\Users\Ajit\AppData\Local\Temp\jupyterlab\jupyterlab_cookie_secret
[I 16:35:15.092 NotebookApp] JupyterLab alpha preview extension loaded from C:\Users\Ajit\AppData\Local\Temp\jupyterlab\jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 16:35:15.099 NotebookApp] Running the core application with no additional extensions or labextensions
[I 16:35:15.208 NotebookApp] Serving notebooks from local directory: C:\Users\Ajit\AppData\Local\Temp\jupyterlab\jupyterlab
[I 16:35:15.208 NotebookApp] 0 active kernels
[I 16:35:15.209 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=499305c7f8f6aec5a564aadd0be303f39df0edd740ffb761
[I 16:35:15.209 NotebookApp] Use Control-C to stop this server and shut down all kernels
[C 16:35:15.216 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=499305c7f8f6aec5a564aadd0be303f39df0edd740ffb761
[I 16:35:15.658 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

Figure 4.14: Open command prompt and type “jupyter notebook”.

Step 2: Once the command is given, the jupyter notebook will be opened by the browser automatically.



Figure 4.15: Jupyter Notebook Homepage.

Step 3: Click on the “New” tab and select “Python 3” to create your first file.



Figure 4.16: Create New jupyter notebook.

Step 4: On clicking the python 3 file we get a screen as shown below where we can type the scripts and execute it.

Chapter 5

REQUIREMENT FOR PROJECT

5.1 Hardware requirement

The hardware and software components required for the development of the Smart Car System.

- 64-bit Computer Running Microsoft Windows
- Webcam
- RGB Display
- 4 GB ram
- Dual Core Processor
- 2 GB free space
- Processor 2 GHz or better
- Internet connection

5.2 Software requirements

- Python3
- Anaconda
- OpenCV-Python
- Pandas
- NumPy
- Matplotlib
- Scikit-learn
- Scipy
- PIL
- Tkinter
- Joblib
- Scikit-image

Chapter 6

IMPLEMENTATION

6.1 Haar Cascade

A Haar Cascade is basically a classifier which is used to detect particular objects from the source. The `haarcascade_frontalface_default.xml` is a haar cascade designed by OpenCV to detect the frontal face. This haar cascade is available on [github](#)[7]. A Haar Cascade works by training the cascade on thousands of negative images with the positive image superimposed on it. The haar cascade is capable of detecting features from the source.

6.2 HOG Descriptor

After The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images? In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well[8]. This is done by extracting the gradient and orientation (or you can say magnitude and direction) of the edges.

Additionally, these orientations are calculated in ‘localized’ portions. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections.

Finally the HOG would generate a Histogram for each of these regions separately[9]. The histograms are created using the gradients and orientations of the pixel values, hence the name ‘Histogram of Oriented Gradients’

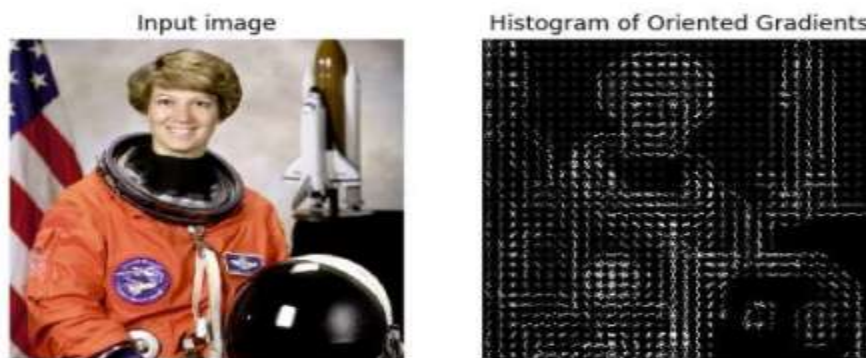


Figure 6.1: Image processing through HOG.

6.3 Datasets

6.3.1 Gender Dataset

The gender dataset was downloaded from the internet but it contained the images of various resolutions. We need a fixed size image to train a model so first we had to resize them according to our need. We wrote a python code to do all that for us. In the same code we also trained a neural network to detect the gender by capturing the webcam feed using opencv and python. We used “mlpclassifier” which is a classifier based on neural network and saved the trained model for the further use with name neural_gender_model.pkl

6.3.2 Emotion Dataset

The Emotion dataset was also downloaded from the internet but it was in form of csv file with three columns i.e. category, pixel values and purpose[11]. We only used the data from the two columns, category and pixel values. The pixel values were the values of pixel that would create a 48*48 image. First we created the images and saved them into their respective folder for further use. After that we loaded the images and on basis of them we trained our model to identify emotions. We wrote a python code to do all that for us. In the same code we also trained a neural network to detect the gender by capturing the webcam feed using opencv and python. We used “mlpclassifier” which is a classifier based on neural network and saved the trained model for the further use with name neural_emotion_model.pkl)

6.3.3 Facial Recognition Dataset

The Facial Recognition Dataset was created by ourselves. We created a GUI through which any user can update their data inot the dataset wnd which will further train a new model and save it for further uses. In doing so, we can use this model to recognize the newly added person in future. It also maintains a record file which contains the data regarding at a given time a person was present over there or not. We used “mlpclassifier” which is a classifier based on neural network and saved the trained model for the further use with name neural_model.pkl.

6.4 Graphic User Interface

We used tkinter library to create the GUI of the for our project. The GUI was created so that anyone can use it regardless of any prior experience. It contained a text field where users can enter their name to update their data. The data was updated by creating a folder with name as given in text field and which will be used to identify the users later on. We used OS module to check whether any data with same name is existing already or not. If it was already there, it would be updated otherwise a new data would be created. Once the user finishes updating their data, the function to update the facial recognition model is called internally so that later on the new person can also be identified.

The code below is responsible for walking through the given directory and store the name of all folder's in a list.

```
In [2]: #loading the required things
temp=os.walk('./orl_face\\orl_face')
users=[]
for i,j,k in temp:
    if j!=[]:
        for _ in j:
            users.append(_)
```

the code for the GUI and the interface itself are given below.

```
In [10]: root = Tk()
frame=Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack(side = BOTTOM )
label = Label(frame, text="new user's name")
e = Entry(frame, bd =5)
```

```
In [11]: updb=Button(frame, text='update',command = update)
modb=Button(bottomframe, text='model',command = model)
recogf=Button(bottomframe, text='facial',command = recognition1)
recogg=Button(bottomframe, text='gender',command = gender)
recoge=Button(bottomframe, text='emotion',command = emotion)
recoga=Button(bottomframe, text='all models',command = all_models)
label.pack(side = LEFT)
e.pack(side = LEFT)
updb.pack(side = LEFT)
modb.pack( side = LEFT)
recogf.pack( side = LEFT)
recogg.pack( side = LEFT)
recoge.pack( side = LEFT)
recoga.pack( side = LEFT)
```

```
In [12]: root.mainloop()
```



Figure 6.2: Tkinter GUI.

6.5 Maintaining Records

We wrote another function that maintains a record about at a particular time which of the individuals are present in video. It uses datetime library to create a row which contains as many columns as the number of registered users. We store the probability prediction of each class for every frame and then. The time is used as index for that particular row. If the probability is greater than 50%, we mark the person as present or otherwise we mark him as absent. Once the recognition window is closed we add this list at the end of previous records and save it permanently in a file named data.csv. If a new user is added later on, than a new column is created for the user but the instances before the registration of the new user would be empty and this all works dynamically.

6.6 Working

Installing the various modules and repositories given in the earlier chapters, we would then proceed to create the datasets to create various models. To identify the frontal face of humans in images we would use HAAR cascade classifier and HOG descriptor. The whole process can be divided into following steps-

- We read the video captured by webcam with the help of opencv and then read that video feed frame by frame.

```
In [4]: def recognition1():
        count=0

        vid=cv2.VideoCapture(0)
        face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

        #Loading saved model
        clf2 = joblib.load('./neural_model.pkl')

        color={i:(randint(0,255),randint(0,255),randint(0,255)) for i in users}
        color.update({'unknown':(255,255,255)})

        while(True):
            f,frame=vid.read()
            if f==True:
```

- We then resize the frame according to our need. You can keep the original resolution to but it would take a lot more resources in processing later on.

```
#resize frame, change to gray scale, and get the face details in form of list
frame=cv2.resize(frame, (640,480))
im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
```

- After resizing the frame we then pass the frame through the Haar cascade which in turn returns us a 2-d list with n elements where n is the number of faces present in the frame. Each of the elements have 4 elements itself which corresponds to the x, y co-ordinates of the top left corner, width and height of face.

```
face=face_cascade.detectMultiScale(im1)

#processing the data of the faces
for x,y,w,h in face:
```

- Now using these locations we draw a bounding boxes around all the faces in frame. The color of the bounding boxes are generated dynamically using various logics.

```
#draw rectangle, and display the no. of pics taken
cv2.rectangle(frame,(x,y),(x+w,y+h),color,4)
cv2.putText(frame,'face no. '+str(i),(x,y),cv2.FONT_ITALIC,1,color,2,cv2.LINE_AA)
```

- After getting the location of the face in the we crop these faces one by one.

- o If we have to register a new person we save the image by creating a new folder with their name.

```
#creating the new folder if not present and then storing captured image in it
path1='./orl_face\orl_face\%s'%(e.get())
path2='./orl_face\orl_face\%s\1 (%d).png'%(e.get(),i)
if not os.path.exists(path1):
    os.makedirs(path1)
cv2.imwrite(path2,im_f)
```

- o If we are going to use any kind of recognition, we pass these images through HOG descriptor to simplify images.

- We will reshape the array containing image data so that it becomes one dimensional and then pass it to the model to make prediction. The class with the highest probability is regarded as our result.

```

f=feature.hog(im_f)
f.reshape(1,-1)

#predict the probability of each cases and change the result in 12 list
for j in clf2.predict_proba([f]):
    y_pred=[int(i*100) for i in j]

u=users[y_pred.index(max(y_pred))]
cv2.rectangle(frame,(x,y),(x+w,y+h),color[u],2)

```

6.7 Training a Model

We are currently using three models and the GUI can be used to access either any one of them or all of them at once. For all the models we are using MLPClassifier which is a neural network based classifier. Most of the things are almost same in each model except the number of neurons in each layer and number of layers.

```

#define classifier
classifier=MLPClassifier(activation='tanh',
                        hidden_layer_sizes=[100,80,60,40,20],
                        verbose=True,solver='adam',
                        max_iter=1000,tol=0.00001)

```

At first we try some other models when we were selecting the models and the result of those models are given below-

	gaussian	multinomial	bernpoulli
10	56.6667	53.8889	6.66667
20	11.875	62.8125	10.9375
30	23.9286	66.0714	15
40	53.3333	69.5833	14.5833
50	63.5	71	18.5
60	68.75	77.5	26.25
70	75	78.3333	24.1667
80	73.75	81.25	27.5
90	80	80	32.5

Table 6.1: Accuracy of gaussian, multinomial and bernoulli models at various test and train sizes.

	80:20	70:30	50:50
linear	92.5	92.5	85.5
poly	80	85	78.5
rbf	80	81.6667	77

Table 6.2: Accuracy of SVM classifier at various kernels and test and train sizes.

6.8 Code

6.8.1 Gender dataset code

```
import os
import cv2
import joblib
import datetime
import numpy as np
import pandas as pd
from tkinter import *
from PIL import Image
import scipy.misc as smp
from sklearn.svm import SVC
from skimage import feature
import matplotlib.image as img
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
# In[2]:
```

```
def resize():
```

```
    #searching for all the non empty folder in the given path #all the folders are
    stored in form of list in variable dirs temp=os.walk('./gender') dirs=[] for
    i,j,k in temp:
```

```
        if j!=[]:
```

```
            for _ in j:
```

```
                dirs.append(_)
```

```
for i in dirs:
```

```
    for j in range(1,1001):
```

```
        #loading the images, resizing them #writing them back
        to storage p1='./gender%s\\1 (%d).jpg'%(i,j)
        im=Image.open(p1) im1 = im.resize((64,64),
        Image.ANTIALIAS) im1.save('./gender%s\\1
        (%d).jpg'%(i,j)) print(j)
```

```
# In[ ]:
```

```
def gender_model():
```

```

#searching for all the non empty folder in the given path #all the folders are
stored in form of list in variable dirs temp=os.walk('./gender') dirs=[] for
i,j,k in temp:
    if j!=[]:
        for _ in j:
            dirs.append(_)

#declaring variables to store data and label data,label=[],[]
for i in dirs:
    for j in range(1,1001):
        p1='./gender%s\\1          (%d).jpg'%(i,j)
        im=img.imread(p1)          f=feature.hog(im)
        f.reshape(1,-1) data.append(f) label.append(i)

#chaining the type of data to numpy array for ease
data=np.array(data)

#declaring variables and creating the train and test sets
train_data,train_label,test_data,test_label=[],[],[],[] for k in
range(0,2000):
    if(k%1000>=900):
        test_data.append(data[k])
        test_label.append(label[k]) else:
            train_data.append(data[k])
            train_label.append(label[k])

#defining classifier
classifier=MLPClassifier(activation='relu',
                        hidden_layer_sizes=[100,80,60,40,20],
                        verbose=True,solver='adam', max_iter=1000,
                        tol=0.00001)

#providing data to the classifier to train to
classifier.fit(train_data,train_label)
y_pred=classifier.predict(test_data)

#saving the classifier for use in future joblib.dump(classifier,
'./neural_gender_model.pkl', compress=9)

#confirmation for everything went accordingly print('trained
and saved successfully')
print(accuracy_score(test_label,y_pred)*100)

```

6.8.2 Emotion dataset

```

import os
import cv2
import joblib
import datetime
import numpy as np
import pandas as pd
from tkinter import *
from PIL import Image
import scipy.misc as smp
from sklearn.svm import SVC
from skimage import feature
import matplotlib.image as img
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

```

In[2]:

```

#reading data from the dataset
data=pd.read_csv("./fer2013.csv")

```

```

#defining the variables to store training and testing datasets
train_data,train_label,test_data,test_label=[],[],[],[]

```

In[3]:

```

#0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral def
createdata():

```

```

    #decalring a list with all possible categories emotion=['Angry', 'Disgust', 'Fear', 'Happy',
    'Sad', 'Surprise', 'Neutral']

```

```

#declaring a dictionary to create count of each category count={'Angry':0, 'Disgust':0, 'Fear':0,
'Happy':0, 'Sad':0, 'Surprise':0, 'Neutral ':0}

```

```

    i=0

```

```

    for j in data.values:

```

```

        #splitting the pixel values, converting them to int and mapping them to a list #it is further
        converted into numpy array temp=np.array(list(map(int,str(j[1]).split()))),dtype=int)

```

```

        #changing shape of numy array
        temp.shape=(48,48)

```



```

#converting it to a image img =
smp.toimage( temp )

#checking the category of the given image
temp1=emotion[data['emotion'].loc[i]]

#storing the image into their respective folder
path1='./emotion\\%s'%(temp1) if not os.path.exists(path1):
os.makedirs(path1) count[temp1]+=1 path2='./emotion\\%s\\1
(%d).png'%(temp1,count[temp1]) cv2.imwrite(path2,temp) i+=1

# In[4]:

def load_data():

#searching for all the non empty folder in the given path #all the folders are
stored in form of list in variable dirs temp=os.walk('./emotion') dirs=[] for
i,j,k in temp:
    if j!=[]:
        for _ in j:
            dirs.append(_)

#declaring variables
data,label=[],[] allfiles={}

#searching all of the files in vaious folders in the given directory for _ in dirs:
temp1=os.walk('./emotion/'+_) files=[] for
i,j,k in temp1:
    if k!=[]:
        for __ in k:
files.append(__) allfiles.update({_:files})

#reading the files one by one and creating the dataseet for i in
allfiles.keys(): for j in allfiles[i]:
    p1='./emotion\\%s\\%s'%(i,j)
    im=img.imread(p1) f=feature.hog(im)
    f.reshape(1,-1) data.append(f)
    label.append(i)

data=np.array(data) print("data
loaded",data.shape)

```

```
split_data(data,label)
```

```
# In[5]:
```

```
def split_data(data,label):
```

```
    #shape of data  
    rows,columns=data.shape
```

```
    #creating training and testing datasets for k in  
    range(rows):
```

```
        if(k%1000>=900):  
            test_data.append(data[k])  
            test_label.append(label[k]) else:  
                train_data.append(data[k])  
                train_label.append(label[k])
```

```
    print(len(train_data),len(test_data)) print("training and  
    testing data created")
```

```
    train_feder_model()
```

```
# In[6]:
```

```
#define classifier def  
train_gender_model(l):
```

```
    classifier=MLPClassifier(activation='relu',  
                             hidden_layer_sizes=[100,75,50,25],  
                             verbose=True,solver='adam', max_iter=1000,  
                             tol=0.00001)
```

```
    #providing data to the classifier to train to  
    classifier.fit(train_data,train_label)  
    y_pred=classifier.predict(test_data)
```

```
    #saving the classifier for use in future joblib.dump(classifier,  
    './neural_gender_model.pkl', compress=9)
```

```
    #confirmation for everything went accordingly print('trained and saved successfully')  
    print(accuracy_score(test_label,y_pred)*100)
```

6.8.3 Main code

```
import os
import cv2
import PIL
import joblib
import warnings
import datetime
import pytesseract
import numpy as np
import pandas as pd
from tkinter import *
import scipy.misc as smp
from random import randint
from skimage import feature
from PIL import Image, ImageTk
import matplotlib.image as img
from py_files.gender_dataset import *
from py_files.emotion_dataset import *
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

In[2]:

```
#loading the required things
temp=os.walk('./orl_face\\orl_face') users=[] for i,j,k in
temp:
    if j!=[]:
        for _ in j:
            users.append(_) users.sort()
pred=pd.DataFrame(columns=['time']+users)
ldata=pd.read_csv('data1.csv',index_col='time') pics=40
```

In[3]:

Def update():

```
vid=cv2.VideoCapture(0)
face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
#getting the number of folders in directory
temp=os.walk('./orl_face\\orl_face') dirs=[] for i,j,k in temp:
if j!=[]:
for _ in j:
```

```

dirs.append(_) dirs.sort() num=len(dirs) i
= 0

color=(randint(0,255),randint(0,255),randint(0,255))

#reading the video frame by frame while i <
pics:
    f,frame=vid.read() if f==True:

        #resize frame, change to grac scale, and get the face details in form of list
        frame=cv2.resize(frame, (640,480))
        im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
        face=face_cascade.detectMultiScale(im1)

        #processing the data of the faces for x,y,w,h in
        face:

#draw rectangle, and display the no. of pics taken cv2.rectangle(frame,(x,y),(x+w,y+h),color,4)
cv2.putText(frame,'face no. '+str(i),(x,y),cv2.FONT_ITALIC,1,color,2,cv2.LINE_AA)

    #wait for pressing of q if cv2.waitKey(1) & 0xFF ==
    ord('q'):

        #crop the face from image, and resize it to desired resolution
        im_f=im1[y:y+h,x:x+w] im_f=cv2.resize(im_f,(92,112)) i+=1

        #creating the new folder if not present and then storing captured image in it
        path1='./orl_face\orl_face\%s'%(e.get()) path2='./orl_face\orl_face\%s\1
        (%d).png'%(e.get(),i) if not os.path.exists(path1):
        os.makedirs(path1)
        cv2.imwrite(path2,im_f)

    #showing the frame

    cv2.imshow('frame',frame)
    cv2.waitKey(1)

#releasing resources vid.release()
cv2.destroyAllWindows()

#calling model function to train model model()
return 0

```

```
# In[4]:
```

```

def recognition1():
    count=0

    vid=cv2.VideoCapture(0)
    face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

    #loading saved model clf2 =
    joblib.load('./neural_model.pkl')

    color={i:(randint(0,255),randint(0,255),randint(0,255)) for i in users}
    color.update({'unknown':(255,255,255)})

    while(True):
        f,frame=vid.read() if f==True:

            #resize frame, change to gray scale, and get the face details in form of list
            frame=cv2.resize(frame, (640,480))
            im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
            face=face_cascade.detectMultiScale(im1)

            #processing the data of the faces for x,y,w,h in
            face:

            #draw rectangle, crop the image, change it to desired resolution, and the extra ct features from
            imgae using hog
            im_f=im1[y:y+h,x:x+w]
            im_f=cv2.resize(im_f,(92,112))
            f=feature.hog(im_f) f.reshape(1,-1)

            #predict the prbability of each cases and chnage the result in 12 list

            for j in clf2.predict_proba([f]):
                y_pred=[int(i*100) for i in j]

                u=users[y_pred.index(max(y_pred))]
                cv2.rectangle(frame,(x,y),(x+w,y+h),color[u],2)

                #predicting the current frame and its probability
                if(max(y_pred)>=40):
                    cv2.putText(frame,str(u)+" "+str(max(y_pred)),(x,y),cv2.FONT_ITALIC,1,color[u],1,cv2.LINE_AA) else:
                    cv2.putText(frame,str('unknown'),(x,y),cv2.FONT_ITALIC,1,color[u],2,cv2.LINE_AA) #adding the
                    data to the dataframe with the timestamp pred.loc[count]=[datetime.datetime.utcnow()+y_pred count+=1

            #showing the frame if key q is pressed than loop would be exited
            cv2.imshow('frame',frame) if cv2.waitKey(1) & 0xFF == ord('q'):

```

```
break vid.release() cv2.destroyAllWindows()
```

```
#calling function to update the data data() return  
0
```

```
# In[5]:
```

```
def gender(): count=0
```

```
vid=cv2.VideoCapture(0)  
face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
temp=os.walk('./gender') dirs=[]  
for i,j,k in temp:  
    if j!=[]:  
        for _ in j:  
            dirs.append(_)
```

```
color={i:(randint(0,255),randint(0,255),randint(0,255)) for i in dirs}
```

```
color.update({'unknown':(255,255,255)})
```

```
#loading saved model clf2 =  
joblib.load('./neural_gender_model.pkl') while(True):  
    f,frame=vid.read() if f==True:
```

```
#resize frame, change to gray scale, and get the face details in form of list  
frame=cv2.resize(frame, (640,480))  
im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)  
face=face_cascade.detectMultiScale(im1)
```

```
#processing the data of the faces for x,y,w,h in  
face:
```

```
#draw rectangle, crop the image, change it to desired resolution, and the extra ct features from  
imgae using hog
```

```
im_f=im1[y:y+h,x:x+w]  
im_f=cv2.resize(im_f,(64,64))  
f=feature.hog(im_f) f.reshape(1,-1)
```

```
#predict the prbability of each cases and chnage the result in 12 list for j in  
clf2.predict_proba([f]):  
    y_pred=[int(i*100) for i in j]
```

```

g=dirs[y_pred.index(max(y_pred))]
cv2.rectangle(frame,(x,y),(x+w,y+h),color[g],2)

#predicting the current frame and its probability
if(max(y_pred)>=40):
    cv2.putText(frame,str(g)+"                               "+str(max(y_pred)),(x,y),cv2.FONT_ITALIC,0.5
,color[g],1,cv2.LINE_AA)
    else:
        cv2.putText(frame,str('unknown'),(x,y),cv2.FONT_ITALIC,1,color[g],2,cv2.LINE_AA)

#showing the frame if key q is pressed than loop would be exited
cv2.imshow('frame',frame) if cv2.waitKey(1) & 0xFF == ord('q'):
break vid.release() cv2.destroyAllWindows()

```

```
return 0
```

```
# In[6]:
```

```
def emotion():
```

```
    count=0
```

```
    vid=cv2.VideoCapture(0)
```

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

```
    temp=os.walk('./emotion')    dirs=[]    for
```

```
    i,j,k in temp:
```

```
        if j!=[]:
```

```
            for _ in j:
```

```
                dirs.append(_)
```

```
    color={i:(randint(0,255),randint(0,255),randint(0,255))    for    i    in    dirs}
```

```
    color.update({'unknown':(255,255,255)})
```

```
#loading saved model clf2 =
```

```
joblib.load('./neural_emotion_model.pkl') while(True):
```

```
    f,frame=vid.read() if f==True:
```

```
        #resize frame, change to gray scale, and get the face details in form of list
```

```
        frame=cv2.resize(frame, (640,480))
```

```
        im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
```

```
        face=face_cascade.detectMultiScale(im1)
```

```

#processing the data of the faces for x,y,w,h in
face:

#draw rectangle, crop the image, change it to desired resolution, and the extra ct features from
imgae using hog
im_f=im1[y:y+h,x:x+w]
im_f=cv2.resize(im_f,(48,48))
f=feature.hog(im_f) f.reshape(1,-1)

#predict the prbability of each cases and chnage the result in 12 list for j in
clf2.predict_proba([f]):
y_pred=[int(i*100) for i in j]

#predicting the current frame and its probability

e=dirs[y_pred.index(max(y_pred))]
cv2.rectangle(frame,(x,y),(x+w,y+h),color[e],2)
if(max(y_pred)>=40):
cv2.putText(frame,str(e)+" "+str(max(y_pred)),(x,y),cv2.FONT_ITALIC,0.5
,color[e],1,cv2.LINE_AA)
else:
cv2.putText(frame,str('unknown'),(x,y),cv2.FONT_ITALIC,1,color[e],2,cv2. LINE_AA)

#showing the frame if key q is pressed than loop would be exited
cv2.imshow('frame',frame) if cv2.waitKey(1) & 0xFF == ord('q'):
break vid.release() cv2.destroyAllWindows()

return 0

# In[7]:

def all_models():
count=0 vid=cv2.VideoCapture(0)
face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

#loading saved model clf2 =
joblib.load('./neural_model.pkl')

color={i:(randint(0,255),randint(0,255),randint(0,255)) for i in users}
color.update({'unknown':(255,255,255)})

#loading saved model clf2g =
joblib.load('./neural_gender_model.pkl')

```



```

#loading saved model clf2e =
joblib.load('./neural_emotion_model.pkl')

temp=os.walk('./emotion') dirse=[] for
i,j,k in temp:
    if j!=[]:
        for _ in j:
            dirse.append(_)

temp=os.walk('./gender')

dirsg=[] for i,j,k in temp:
    if j!=[]:
        for _ in j:
            dirsg.append(_)

while(True):
    f,frame=vid.read() if f==True:

        #resize frame, change to grac scale, and get the face details in form of list
        frame=cv2.resize(frame, (640,480))
        im1=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)
        face=face_cascade.detectMultiScale(im1)

        #processing the data of the faces for x,y,w,h in
        face:

        #draw rectangle, crop the image, change it to desired resolution, and the extra ct features from
        imgae using hog
        im_f=im1[y:y+h,x:x+w]
        im_fe=cv2.resize(im_f,(48,48))
        fe=feature.hog(im_fe) fe.reshape(1,-1)

        im_fg=cv2.resize(im_f,(64,64))
        fg=feature.hog(im_fg) fg.reshape(1,-1)

        im_f=cv2.resize(im_f,(92,112))
        f=feature.hog(im_f) f.reshape(1,-1)

        #predict the prbability of each cases and chnage the result in 12 list for j in
        clf2.predict_proba([f]):
            y_pred=[int(i*100) for i in j]

        for j in clf2g.predict_proba([fg]):

```

```

        y_predg=[int(i*100) for i in j]

    for j in clf2e.predict_proba([fe]):
        y_prede=[int(i*100) for i in j]

        #predicting the current frame and its probability

u=users[y_pred.index(max(y_pred))]                e=dirse[y_prede.index(max(y_prede))]
g=dirsg[y_predg.index(max(y_predg))]  cv2.rectangle(frame,(x,y),(x+w,y+h),color[u],2)  st=str(u)+"
"+str(max(y_pred))+ " "+str(e)+" "+str(max(y_prede))+ " "+str(g) + " "+str(max(y_predg))
    if(max(y_pred)>=40):
        cv2.putText(frame,str(st),(x,y),cv2.FONT_ITALIC,0.5,color[u],1,cv2.LINE_AA)
    else:
        cv2.putText(frame,str('unknown'),(x,y),cv2.FONT_ITALIC,1,color[u],2,cv2.LINE_AA)

    pred.loc[count]=[datetime.datetime.utcnow()+y_pred count+=1

    #showing the frame if key q is pressed than loop would be exited
    cv2.imshow('frame',frame) if cv2.waitKey(1) & 0xFF == ord('q'):
break vid.release() cv2.destroyAllWindows()

    data()

    return 0

# In[8]:

def model():

    #getting the number of folders in directory
    temp=os.walk('./orl_face\\orl_face') dirs=[] for i,j,k in
temp:
    if j!=[]:
        for _ in j:
dirs.append(_) dirs.sort() num=len(dirs)

    #reading data from folders
    data,label=[],[] for i in dirs:
        for j in range(1,pics+1):

```

```

p1='./orl_face\\orl_face\\%s\\1          (%d).png'%(i,j)
im=img.imread(p1)      f=feature.hog(im)      f.reshape(1,-1)
data.append(f) label.append(i) data=np.array(data)

#dividing data into train and test
train_data,train_label,test_data,test_label=[],[],[],[] for k in
range(0,num*pics):
    if(k%pics>=36):
        test_data.append(data[k])
        test_label.append(label[k]) else:
            train_data.append(data[k])
            train_label.append(label[k])

#define classifier
classifier=MLPClassifier(activation='tanh',
hidden_layer_sizes=[100,80,60,40,20], verbose=True,solver='adam',
max_iter=1000,tol=0.00001) classifier.fit(train_data,train_label)
y_pred=classifier.predict(test_data)

#saving classifier joblib.dump(classifier, './neural_model.pkl',
compress=9)

#confirmation for everything went accordingly print('trained
and saved successfully')
print(accuracy_score(test_label,y_pred)*100) return 0

# In[9]:

def data():
    global pred global ldata
    pred=pred.set_index('time') pred for i in
    users:

        #setting the values as present or absent at given time for j in
        pred.index:

            j=str(j) if pred[i].loc[j]>=50:
pred[i].loc[j]='present' elif pred[i].loc[j]<50:
    pred[i].loc[j]='absent'

#appending the new and old data ldata=pd.concat([ldata,pred])
ldata.to_csv('data1.csv', encoding='utf-8', index='time') print("no of rows in
data.csv\t",len(ldata),"\t rows added",len(pred))

```

```
# In[10]:
```

```
root = Tk() frame=Frame(root) frame.pack() bottomframe =  
Frame(root) bottomframe.pack(side = BOTTOM ) label =  
Label(frame, text="new user's name") e = Entry(frame, bd  
=5)
```

```
# In[11]:
```

```
updb=Button(frame, text='update',command = update) modb=Button(bottomframe,  
text='model',command = model) recogf=Button(bottomframe, text='facial',command  
= recognition1) recogg=Button(bottomframe, text='gender',command = gender)  
recoge=Button(bottomframe, text='emotion',command = emotion)  
recoga=Button(bottomframe, text='all models',command = all_models)  
label.pack(side = LEFT) e.pack(side = LEFT) updb.pack(side = LEFT) modb.pack(  
side = LEFT) recogf.pack( side = LEFT) recogg.pack( side = LEFT) recoge.pack(  
side = LEFT) recoga.pack( side = LEFT)
```

```
# In[12]:
```

```
root.mainloop()
```

Chapter 7

CONCERNS AND FUTURE WORKS

7.1 The threats and concerns about facial recognition

For now, facial recognition seems amazing. It's fast, accurate, and provide outstanding results in no time. Looks like all companies should immediately integrate it with their security systems and start enjoying the benefits from its use. But things are not so bright as they seem.

Here are the biggest concerns about the technology that stands on the way of implementing it in every company.

7.1.1 Breach of privacy

With the help of this technology, the government can track down the criminals. But at the same time, it can actually track down people like you: anytime, anywhere.

The question of ethics and privacy is the most critical one. The USA government is already known to store a certain number of the citizens' pictures without their consent and we don't know about all cases yet[13].

So even though facial recognition indeed brings benefits, there is still an awful lot of work to be done before the technology is 100% used fairly and in accordance with human rights for privacy.

7.1.2 Vulnerability in recognition

Facial recognition technology is indeed very accurate and no one can doubt that.

At the same time, a slight change in the camera angle or even the change of appearance will inevitably lead to an error[12]. Bam — and your new haircut keeps you invisible for the cameras.

So, this is a really serious flaw. You cannot guarantee that a person will stand still and face the camera so the results will not always be correct. And if the person changes the appearance, it would be almost impossible to recognize them.

7.1.3 Massive data storage

Machine Learning technology requires massive data sets to “learn” in order to deliver accurate results. And such data sets require a powerful data storage[15].

So if you are a small or medium-sized company you simply may not have the necessary resources to store all the data. And that might be a problem.



Figure 7.1: Facial Recognition can guess peoples emotions and identify potential threats for future

7.2 How to Avoid Facial Recognition

The first step to avoiding facial recognition online is to take care of where photos of you are uploaded.

Social media sites like Facebook have facial recognition algorithms which analyze photos uploaded to the site to make suggestions for who should be tagged in them[11]. When someone tags you in a photo, they are training the algorithm to identify your face more accurately.

7.2.1 Disabling Facial Recognition on Facebook

You should be able to disable automatic facerecognition on Facebook.

To disable the feature, go to the Facebook website or app and head to **Settings**. Then check the left-hand menu, where you should find **Face recognition** just under **Languages**.

In this menu, click on **Edit**. From the option **Do you want Facebook to be able to recognize you in photos and videos** select **No** from the drop-down menu, then hit **Close**.

This should save your settings and prevent people from tagging you[14].

7.2.2 Use FaceShield When UploadingPhotos

You might still want to share photos online, but not to have them visible to facial recognition software. In that case, you can use the **FaceShield** tool[11].

FaceShield is a filter that you apply to your photos before you upload them to a website. It makes only minor changes to the photo to the human eye, but the developers say that it makes faces less visible to facial recognition software.

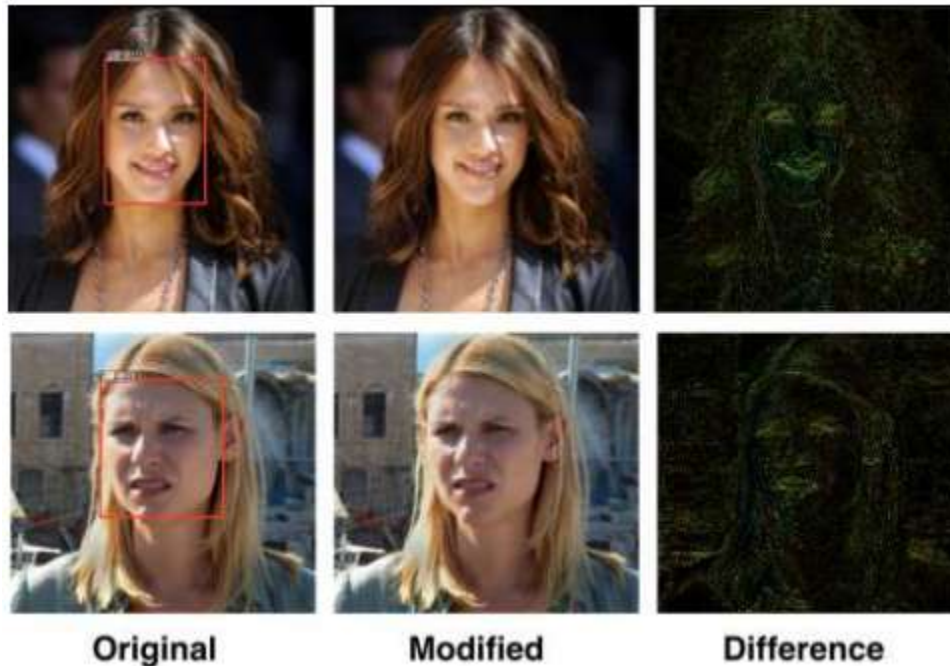


Figure 7.2: FaceShield Tool can make minor changes to photo not visible to human eye but not identifiable by Facial Recognition System.

7.2.3 How to Avoid Facial Recognition in Person

Avoiding facial recognition online is only half the battle, however. You also need to be aware of all the places where facial recognition is happening in person[10].

Facial recognition software is commonly used for security at large events. Law enforcement use software to monitor protests and demonstrations. It's also often found in airports and other high-security locations.

The simplest way to avoid facial recognition in person is to obscure your face with a scarf or balaclava. However, this may be against the law in some places and also has the downside of being very conspicuous. There are few better ways to draw attention to yourself in a crowd than covering your face.

7.2.4 Use Hair and Makeup to Fool Facial Recognition



Figure 7.3: bold hair and makeup forms can confuse facial detection software.

An ingenious way to avoid facial recognition is to use a technique like **CV Dazzle**. This approach uses bold hair and makeup forms which confuse facial detection software and act as camouflage for your face[9].

The looks use high contrast makeup, with dark colors on light skin and light colors on dark skin. The hairstyles often partially or completely obscure the nose bridge region between the eyes, which is key to facial identification. The looks are also asymmetrical, as facial recognition software are used to symmetry between the sides of the face.

7.2.5 Use Clothing to Distract Facial Recognition

HyperFace

False-Face Camouflage

Posted 2017-03-01 in [projects](#), [tagged](#) [Computer vision](#), [Face detection](#), [Face spoof](#), [Face spoofing](#), [Face spoofing](#)

Project is under active development. Sign up for product launch notification at <https://www.hyperface.com>



Figure 7.4: False faces can confuse facial detection software.

Another option is to distract software by overwhelming it with images that look like faces, so it can't see your face. This is the approach taken by the **HyperFace** project.

Hyperface uses prints for clothes and other textiles which create “false faces”. Software sees these prints and struggles to differentiate your real face from the simulated faces, making it harder to track you[8].

These prints aren't publicly available yet, but in the future, they could be a tool against facial recognition.

7.3 What the Future Holds?

The future of facial recognition technology is bright. Forecasters opine that this technology is expected to grow at a formidable rate and will generate huge revenues in the coming years. Security and surveillances are the major segments which will be deeply influenced. Other areas that are now welcoming it with open arms are private industries, public buildings, and schools. It is estimated that it will also be adopted by retailers and banking systems in coming years to keep fraud in debit/credit card purchases and payment especially the ones that are online. This technology would fill in the loopholes of largely prevalent inadequate password system. In the long run, robots using facial recognition technology may also come to foray. They can be helpful in completing the tasks that are impractical or difficult for human beings to complete.

REFERENCES

- [1] Gunturk, B., Batur, A., Altunbasak, Y., III, M.H., Mersereau, R.: Eigenface-domain super-resolution for face recognition. *IEEE Transactions on Image Processing* 12(5), 597–606 (2003)
- [2] Lemieux, A., Parizeau, M.: Experiments on eigenfaces robustness. In: *Proc. ICPR- 2002*, vol. 1, pp. 421–424 (August 2002)
- [3] Wang, X., Tang, X.: Face Hallucination and Recognition. In: Kittler, J., Nixon, M.S. (eds.) *AVBPA 2003*. LNCS, vol. 2688, pp. 486–494. Springer, Heidelberg (2003)
- [4] Jaynes, C., Kale, A., Sanders, N., Grossmann, E.: The Terrascope dataset: scripted multi-camera indoor video surveillance with ground-truth. In: *Proc. Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 309–316 (October 2005)
- [5] Turk, M., Pentland, A.: Eigenfaces for recognition. *Journal of Cognitive Neuroscience* 3(1), 71–86 (1991)
- [6] Wiskott, L., Fellous, J., Krüger, N., Malsburg, C.: Face recognition by elastic bunch graph matching. In: Sommer, G., Daniilidis, K., Pauli, J. (eds.) *CAIP 1997*. LNCS, vol. 1296, pp. 456–463. Springer, Heidelberg (1997)
- [7] Messer, K., Matas, J., Kittler, J., Luetin, J., Maitre, G.: XM2VTS: The Extended M2VTS Database. In: *Proc. AVBPA-1999*, pp. 72–76 (1999)
- [8] Park, S., Park, M., Kang, M.: Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine* 25(9), 21–36 (2003)
- [9] Tsai, R., Huang, T.: Multiframe image restoration and registration. *Advances in Computer Vision and image Processing* 1, 317–339 (1984)
- [10] Baker, S., Kanade, T.: Limits on Super-Resolution and How to Break Them. 24(9), 1167–1183 (2002)
- [11] Baker, S., Kanade, T.: Super Resolution Optical Flow. Technical Report CMU-RI- TR-99-36, The Robotics Institute, Carnegie Mellon University (October 1999)
- [12] Lin, F., Fookes, C., Chandran, V., Sridharan, S.: Investigation into Optical Flow Super-Resolution for Surveillance Applications. In: *Proc. APRS Workshop on Digital Image Computing 2005*, pp. 73–78 (February 2005)
- [13] Black, M., Anandan, P.: A framework for the robust estimation of optical flow. In: *Proc. ICCV-1993*, pp. 231–236 (May 1993)
- [14] Schultz, R., Stevenson, R.: Extraction of High-Resolution Frames from Video Sequences. *IEEE Transactions on Image Processing* 5(6), 996–1011 (1996)
- [15] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *CVPR* (2001)