**Attendance System using Facial Recognition**

**A Project Report**

**Submitted by:**

**KULDEEP SINGH DUDI (17EBKCS056)**

**KONARK PAREEK (17EBKCS055)**

***in partial fulfillment for the award of the degree***

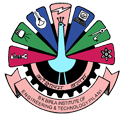
***of***

**BACHELOR OF TECHONOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**At**



**B.K. BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY**

**PILANI, RAJASTHAN (INDIA) – 333031**

**(AFFILIATED TO RAJASTHAN TECHNICAL UNIVERSITY, KOTA, RAJASTHAN (INDIA))**

**JUNE & 2021**

**DECLARATION**

I hereby declare that the project entitled “**Attendance System using Facial Recognition**” submitted for the B. Tech. (CSE) degree is my original work and the project has not formed the basis for the award of any other degree, or any other similar title.

**Signature of the Student**

**Place:**

**Date:**

**B. K. BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY, PILANI (RAJ.)**

**SESSION 2017-2021**



**CERTIFICATE**

This is to certify that the project entitled “**Attendance System using Facial Recognition**” is the bonafide work carried out by **Konark Pareek (17EBKCS55**) & **Kuldeep Singh Dudi (17EBKCS56)**, students of B Tech (CSE) of B. K. Birla Institute of Engineering & Technology, Pilani affiliated to Rajasthan Technical University, Kota, Rajasthan(India) during the academic year 2017-2021, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering ) and that the project has not formed the basis for the award previously of any other degree, or any other similar title.

**Signature of the Guide**

**Place:**

**Date:**

**TABLE OF CONTENTS**

**Title……………………………………………………………………Page No**

*Certificate*

*Acknowledgement*

1. **ABSTRACT……………………………………………………………………………….6**
2. **INTRODUCTION………………………………………………………………………...6-12**

2.1 BACKGROUND

2.2 OBJECTIVE

2.3 FACE DETECTION

2.4 FACE RECOGNITION

2.5 BRIEF ABOUT OPENCV

1. **TOOLS AND ENVIRONMENT USED………………………………………………..12-16**

3.1 HARDWARE REQUIREMENTS

3.2 SOFTWARE REQUIREMENTS

3.3 ANACONDA

3.3.1 Anaconda Navigator

3.3.1.1 Jupyter Notebook

3.3.2 Anaconda Cloud

1. **LANGUAGE USED…………………………………………………………………...16-19**

4.1 PYTHON

4.1.1 Definition

4.1.2 Data types in python

4.1.3 String Handling

4.1.4 Lists

4.1.5 Tuples

4.1.6 Dictionary

4.1.7 Operators in Python

4.1.8 Conditional Statements

**5. METHODOLOGY…………………………………………………………………...20-29**

5.1 IMPORTING LIBRARIES

5.1.1 NumPy

5.1.2 OpenCV

5.1.3 Face Recognition

5.2 PROCEDURE

5.3 TRAINING THE MODEL

**6. PROGRAM CODE………………………………………………………... ………...30-32**

**7. TESTING……………………………………………………………………………...32-37**

7.1 CREATING ENVIRONMENT

7.2 INSTALLING REQUIRED LIBRARIES

7.3 RUNNING THE CODE

**8. RESULT…………………….…………………………………………………………37-38**

**9. REFERENCES……………………………………………………………………….**

**1. ABSTRACT**

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

While initially a form of computer application, facial recognition systems have seen wider uses in recent times on [smartphones](https://en.wikipedia.org/wiki/Smartphone) and in other forms of technology, such as robotics. Because computerized facial recognition involves the measurement of a human's physiological characteristics, facial recognition systems are categorized as biometrics. Although the accuracy of facial recognition systems as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless process. Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance and automatic indexing of images. They are also used widely by law enforcement agencies.

The applications of this sub-domain of computer vision are vast and businesses around the world are already reaping the benefits. The usage of face recognition models is only going to increase in the next few years.

Face recognition is different from Face detection, the differences are laid down further in the report.

**2. INTRODUCTION**

What is Face Recognition?

Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary. Here we will see how we do face recognition using deep learning. So now let us understand how we recognize faces using deep learning. We make use of face embedding in which each face is converted into a vector and this technique is called deep metric learning.

**2.1 Background**

## The dawn of Facial Recognition – 1960s [1]

The earliest pioneers of facial recognition were Woody Bledsoe, Helen Chan Wolf and Charles Bisson. In 1964 and 1965, Bledsoe, along with Wolf and Bisson began work using computers to recognize the human face.

Due to the funding of the project originating from an unnamed intelligence agency, much of their work was never published. However it was later revealed that their initial work involved the manual marking of various “landmarks” on the face such as eye centres, mouth etc. These were then mathematically rotated by a computer to compensate for pose variation. The distances between landmarks were also automatically computed and compared between images to determine identity.

These earliest steps into Facial Recognition by Bledsoe, Wolf and Bisson were severely hampered by the technology of the era, but it remains an important first step in proving that Facial Recognition was a viable biometric.

## Advancing the accuracy of Facial Recognition – 1970s

Carrying on from the initial work of Bledsoe, the baton was picked up in the 1970s by Goldstein, Harmon and Lesk who extended the work to include 21 specific subjective markers including hair colour and lip thickness in order to automate the recognition.

While the accuracy advanced, the measurements and locations still needed to be manually computed which proved to be extremely labour intensive yet still represents advancement on Bledsoe’s RAND Tablet technology.

## Using linear algebra for Facial Recognition – 1980s/90s

It wasn’t until the late 1980s that we saw further progress with the development of Facial Recognition software as a viable biometric for businesses. In 1988, Sirovich and Kirby began applying linear algebra to the problem of facial recognition.

A system that came to be known as Eigenface showed that feature analysis on a collection of facial images could form a set of basic features. They were also able to show that less than one hundred values were required in order to accurately code a normalized facial image.

In 1991, Turk and Pentland carried on the work of Sirovich and Kirby by discovering how to detect faces within an image which led to the earliest instances of automatic facial recognition. This significant breakthrough was hindered by technological and environmental factors, however it paved the way for future developments in Facial Recognition technology.

## FERET Programme – 1990s/2000s

The Defence Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST) rolled out the Face Recognition Technology (FERET) programme in the early 1990s in order to encourage the commercial facial recognition market. The project involved creating a database of facial images. Included in the test set were 2,413 still facial images representing 856 people. The hope was that a large database of test images for facial recognition would inspire innovation and may result in more powerful facial recognition technology.

## Face Recognition Vendor Tests – 2000s

The National Institute of Standards and Technology (NIST) began Face Recognition Vendor Tests (FRVT) in the early 2000s. Building on FERET, FRVTs were designed to provide independent government evaluations of facial recognition systems that were commercially available, as well as prototype technologies. These evaluations were designed to provide law enforcement agencies and the U.S. government with information necessary to determine the best ways to deploy facial recognition technology.

## Face Recognition Grand Challenge – 2006

Launched in 2006, the primary goal of the Face Recognition Grand Challenge (FRGC) was to promote and advance face recognition technology designed to support existing face recognition efforts in the U.S. Government.

The FRGC evaluated the latest face recognition algorithms available. High-resolution face images, 3D face scans, and iris images were used in the tests. The results indicated that the new algorithms were 10 times more accurate than the face recognition algorithms of 2002 and 100 times more accurate than those of 1995, showing the advancements of facial recognition technology over the past decade.

## Social Media – 2010-Current

Back in 2010, Facebook began implementing facial recognition functionality that helped identify people whose faces may feature in the photos that Facebook users update daily. The feature was instantly controversial with the news media, sparking a slew of privacy-related articles. However, Facebook users by and large did not seem to mind. Having no apparent negative impact on the website’s usage or popularity, more than 350 million photos are uploaded and tagged using face recognition each day.

## iPhone X – 2017

Facial Recognition technology advanced rapidly from 2010 onwards and September 12, 2017 was another significant breakthrough for the integration of facial recognition into our day to day lives. This was the date that Apple launched the iPhone X – the first iPhone users could unlock with FaceID – Apple’s marketing term for facial recognition.

**2.2 Objective**

Once considered a thing of science fiction, biometric Facial Recognition is quickly becoming an integrated part of people’s everyday lives.

Several major industries have benefitted from the rapid advancements that have been made in Facial Recognition technology over the past 60 years and these include: law enforcement, border control, retail, mobile technology and banking and finance.

Border controls, airlines, airports, transport hubs, stadiums, mega events, concerts, conferences. Biometrics are playing a growing role not only in the real-time policing and securing of increasingly crowded and varied venues worldwide.

Face recognition can often prove one of the best biometrics because images can be taken without touching or interacting with the individual being identified, and those images recorded and instantly checked against existing databases.

**2.3 Face Detection**

## **What is Face Detection?**

The goal of face detection is to determine if there are any faces in the image or video. If multiple faces are present, each face is enclosed by a bounding box and thus we know the location of the faces

Human faces are difficult to model as there are many variables that can change for example facial expression, orientation, lighting conditions and partial occlusions such as sunglasses, scarf, mask etc. The result of the detection gives the face location parameters and it could be required in various forms, for instance, a rectangle covering the central part of the face, eye centers or landmarks including eyes, nose and mouth corners, eyebrows, nostrils, etc.

In computer vision, one essential problem we are trying to figure out is to automatically detect objects in an image without human intervention. Face detection can be thought of as such a problem where we detect human faces in an image. There may be slight differences in the faces of humans but overall, it is safe to say that there are certain features that are associated with all the human faces.

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection can have very useful applications. The most successful application of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

## **Face Detection Methods**

There are two main approaches for Face Detection:

1. Feature Base Approach
2. Image Base Approach

#### **Feature Base Approach**

Objects are usually recognized by their unique features. There are many features in a human face, which can be recognized between a face and many other objects. It locates faces by extracting structural features like eyes, nose, mouth etc. and then uses them to detect a face. Typically, some sort of statistical classifier qualified then helpful to separate between facial and non-facial regions. In addition, human faces have particular textures which can be used to differentiate between a face and other objects. Moreover, the edge of features can help to detect the objects from the face. In the coming section, we will implement a feature-based approach by using OpenCV.

#### **Image Base Approach**

In general, Image-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images. The learned characteristics are in the form of distribution models or discriminant functions that is consequently used for face detection. In this method, we use different algorithms such as Neural-networks, HMM, SVM, AdaBoost learning. In the coming section, we will see how we can detect faces with MTCNN or Multi-Task Cascaded Convolutional Neural Network, which is an Image-based approach of face detection.

**Face detection algorithm**

One of the popular algorithms that use a feature-based approach is the Viola-Jones algorithm.

**Viola-Jones** algorithm is named after two computer vision researchers who proposed the method in 2001, Paul **Viola** and Michael **Jones** in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

**2.4 Face Recognition**

Face detection and Face Recognition are often used interchangeably but these are quite different. In fact, Face detection is just part of Face Recognition. Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary.

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems can be used to identify people in photos, video, or in real-time. Law enforcement may also use mobile devices to identify people during police stops. Face recognition systems use computer algorithms to pick out specific, distinctive details about a person’s face. These details, such as distance between the eyes or shape of the chin, are then converted into a mathematical representation and compared to data on other faces collected in a face recognition database. The data about a particular face is often called a face template and is distinct from a photograph because it’s designed to only include certain details that can be used to distinguish one face from another.

**2.5 Brief about OpenCV**

In the field of Artificial Intelligence, Computer Vision is one of the most interesting and Challenging tasks. Computer Vision acts like a bridge between Computer Software and visualizations around us. It allows computer software to understand and learn about the visualizations in the surroundings. For Example: Based on the color, shape and size determining the fruit. This task can be very easy for the human brain however in the Computer Vision pipeline, first we gather the data, then we perform the data processing activities and then we train and teach the model to understand how to distinguish between the fruits based on size, shape and color of fruit.

Currently, various packages are present to perform machine learning, deep learning and computer vision tasks. By far, computer vision is the best module for such complex activities. OpenCV is an open-source library. It is supported by various programming languages such as R, Python. It runs on most of the platforms such as Windows, Linux and MacOS.

Advantages of OpenCV:

* OpenCV is an open-source library and is free of cost.
* As compared to other libraries, it is fast since it is written in C/C++.
* It works better on System with lesser **RAM**
* T supports most of the Operating Systems such as Windows, Linux and MacOS.

**3. TOOLS AND ENVIRONMENT USED**

Environment and development tools required for the development process:

* Anaconda3 (64-bit)
* Jupyter Notebook
* Microsoft Windows 10 (Operating System)
* Web Browser (FireFox, Chrome, Edge)

**3.1 HARDWARE REQUIREMENTS**

Following are the hardware requirements for the project:

* MEMORY SPACE:

|  |  |  |
| --- | --- | --- |
| • | Minimum | - 32MB |
| • | Recommended | - 64MB |

* HDD

To install the software at least 2 GB and the data storage is depending upon the organizational setup.

|  |  |  |
| --- | --- | --- |
| • | PROCESSOR | - Intel Pentium I3, 1GHZ or above |
| • | RAM | - 256MB or above |
| • | VIDEO | - 1024x768, 24-bit colors |
| • | KEYBOARD | - Standard 104 Keys (QWERTY) |

Blank writable CD to keep the backup of the Package.

**3.2 SOFTWARE REQUIREMENTS**

Following are the software requirements for the project:

• Language : Python

* Operating System: Windows XP

**3.3 ANACONDA [2]**

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, largescale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, which are both not free.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda distribution comes with 1,500 packages selected from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages.

It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the *conda install* command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the *conda build* command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

**3.3.1 Anaconda Navigator**

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using [command-line commands.](https://en.wikipedia.org/wiki/Command-line_interface) Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows,](https://en.wikipedia.org/wiki/Windows) [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux.](https://en.wikipedia.org/wiki/Linux)

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter#JupyterLab)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* QtConsole
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software))
* [Glue](https://en.wikipedia.org/wiki/Glue_(software))
* [Orange](https://en.wikipedia.org/wiki/Orange_(software))
* [RStudio](https://en.wikipedia.org/wiki/RStudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)

**3.3.1.1 Jupyter Notebook**

Jupyter Notebook is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using [Markdown),](https://en.wikipedia.org/wiki/Markdown) mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of [open standard](https://en.wikipedia.org/wiki/Open_standard) output formats [(HTML,](https://en.wikipedia.org/wiki/HTML) [LaTeX,](https://en.wikipedia.org/wiki/LaTeX) [PDF,](https://en.wikipedia.org/wiki/PDF) [ReStructuredText,](https://en.wikipedia.org/wiki/ReStructuredText) [Markdown,](https://en.wikipedia.org/wiki/Markdown) [Python)](https://en.wikipedia.org/wiki/Python_(programming_language)) through "Download As" in the web interface, via the [nbconvert](https://nbconvert.readthedocs.io/) library or "jupyter nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through [NbViewer](https://nbviewer.org/) which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based [REPL](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop) built upon a number of popular [open-source](https://en.wikipedia.org/wiki/Open-source_software) libraries:

* [IPython](https://en.wikipedia.org/wiki/IPython)
* [ØMQ](https://en.wikipedia.org/wiki/%C3%98MQ)
* [Tornado (web server)](https://en.wikipedia.org/wiki/Tornado_(web_server))
* [jQuery](https://en.wikipedia.org/wiki/JQuery)
* [Bootstrap (front-end framework)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
* [MathJax](https://en.wikipedia.org/wiki/MathJax)

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including [Python,](https://en.wikipedia.org/wiki/Python_(programming_language)) [R,](https://en.wikipedia.org/wiki/R_(programming_language)) [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) and [Haskell.](https://en.wikipedia.org/wiki/Haskell_(programming_language))

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple,](https://en.wikipedia.org/wiki/Maple_(software)) [Mathematica,](https://en.wikipedia.org/wiki/Mathematica) and [SageMath,](https://en.wikipedia.org/wiki/SageMath) a computational interface style that originated with Mathematica in the 1980s. According to [*The Atlantic*,](https://en.wikipedia.org/wiki/The_Atlantic) Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

**3.3.2 Anaconda Cloud**

Anaconda Cloud is a package management service by Anaconda where users can find, access, store and share public and private notebooks, environments, and conda and PyPI packages. Cloud hosts useful Python packages, notebooks and environments for a wide variety of applications. Users do not need to log in or to have a Cloud account, to search for public packages, download and install them.

Users can build new packages using the Anaconda Client command line interface (CLI), then manually or automatically upload the packages to Cloud.

**4. LANGUAGE USED**

We have used Python as our programming language in this project. In Jupyter Notebook we can also use R language. But as compared to R language, Python can be easily understood by the developer as well as the user also for basic coding.

**4.1 PYTHON [3]**

In this section we have covered the basics of Python language which includes its definition, its datatypes, string handling, the various operators used in python, as well as the Conditional statements used in Python.

**4.1.1 DEFINITION:**

Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language. Some of the popular systems and applications that have employed Python during development include Google Search, YouTube, BitTorrent, Google App Engine, Eve Online, Maya and iRobot machines.

**4.1.2 DATATYPES IN PYTHON:**

* Int
* String
* List
* Tuple
* Dictionary

There is no need of declaring data types in python.

**4.1.3 STRING HANDLING:**

Some functions-

* capitalize()- to capitalize the given string.
* count()- counts the total no. of occurrence of a given character.
* isalpha()- returns true if string contains only alphabets (no space).
* strip()- this function is used to remove spaces.

**4.1.4 LISTS:**

* Python directly does not support arrays. Instead of arrays we use lists.
* The list items are put inside square brackets [ ] .
* We can say that list is a collection of items enclosed within square brackets.
* Lists are mutable i.e. changeable.

Example-

Marks=[99, 39,56,43,90,45]

**4.1.5 TUPLES:**

* Everything in tuples is same as lists, apart from two differences.
* The items in a tuple are enclosed within round brackets ( ).
* Tuples are immutable that is they cannot be changed once mentioned**.**

Example-

Marks=(22,55,67,33,55,56)

**4.1.6 DICTIONARY:**

* It contains key value pairs enclosed within curly braces { }.
* Values can be of any datatype but generally key should be of integer or string type.
* Key must be unique.

Example-

Marks= {1: “java”, 2: “python”, 3: “c”, 4: “ML”}

* Marks.keys(): print the keys Marks.items(): prints the key value.
* Marks.values(): prints the values.
* Marks.get(1): gives the value at key 1.

**4.1.7 OPERATORS IN PYTHON:**

1. **Arithmetic Operator.**
   * +, -, \*, /, // (floor division), %, \*\* (exponent)
2. **Relational Operator.**
   * >, <, >=, <=, ==, != (result is either true or false)
3. **Bitwise Operator.**
   * & AND
   * | OR
   * ^ EXOR
   * ~ NOT
   * << Shift left • >> Shift right
4. **Boolean Operator.**

It works on only Boolean values.

* + AND
  + OR
  + NOT

1. **Assignment Operator.**

* It is the simplest operator which is used to store the values into variables.
* Symbol is =

1. **Membership Operator.**
   * In
   * Not in

If the item is present in a given sequence like list or tuple, “in” operator will return true else false. Reverse is with “not in” operator.

1. **Identity Operator.**
   * + Is
     + Is not

If identity of two variables is same, “is” operator returns true.

**4.1.8 CONDITIONAL STATEMENTS:**

Two types:

1. Decision Making
2. Looping

**5. METHODOLOGY**

Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary.

So now let us understand how we recognize faces using deep learning. We make use of face embedding in which each face is converted into a vector and this technique is called deep metric learning. Let me further divide this process into three simple steps for easy understanding:

**Face Detection:**The very first task we perform is detecting faces in the image or video stream. Now that we know the exact location/coordinates of face, we extract this face for further processing ahead.

**Feature Extraction**: Now that we have cropped the face out of the image, we extract features from it. Here we are going to use face embeddings to extract the features out of the face. A neural network takes an image of the person’s face as input and outputs a vector which represents the most important features of a face. In machine learning, this vector is called embedding and thus we call this vector as face embedding.

While training the neural network, the network learns to output similar vectors for faces that look similar. Now after training the network, the network learns to output vectors that are closer to each other(similar) for faces of the same person(looking similar).

**Comparing faces:**Now that we have face embeddings for every face in our data saved in a file, the next step is to recognize a new test image that is not in our data. So the first step is to compute the face embedding for the image using old data and then compare this embedding with the rest of the embeddings we have.

**5.1 IMPORTING LIBRARIES**

Any machine learning project requires the need of importing various in-built libraries which can further help in the development of the project or model. Although there are hundreds of libraries used in Machine Learning, in our model we will be using only four of them.

The libraries that we will be using are:

* NumPy
* OpenCV
* Face recognition

Let’s take a look at all of these libraries.

**5.1.1 NumPy**

NumPy is a library for the [Python programming language,](https://en.wikipedia.org/wiki/Python_(programming_language)) adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices,](https://en.wikipedia.org/wiki/Matrix_(math)) along with a large collection of highlevel [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [opensource software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors. The [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) was not originally designed for numerical computing, but attracted the attention of the scientific and engineering community early on. In 1995 the special interest group (SIG) matrix-sig was founded with the aim of defining an [array](https://en.wikipedia.org/wiki/Array_data_type) computing package; among its members was Python designer and maintainer [Guido van Rossum,](https://en.wikipedia.org/wiki/Guido_van_Rossum) who extended [Python's syntax](https://en.wikipedia.org/wiki/Python_syntax_and_semantics) (in particular the indexing syntax) to make array computing easier.

**5.1.2 OpenCV [4]**

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Let’s start the chapter by defining the term "Computer Vision".

**Computer Vision**

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields −

* **Image Processing** − It focuses on image manipulation.
* **Pattern Recognition** − It explains various techniques to classify patterns.
* **Photogrammetry** − It is concerned with obtaining accurate measurements from images.

**Computer Vision Vs Image Processing**

**Image processing** deals with image-to-image transformation. The input and output of image processing are both images.

**Computer vision** is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

**Applications of Computer Vision**

Here we have listed down some of major domains where Computer Vision is heavily used.

Robotics Application

* Localization − Determine robot location automatically
* Navigation
* Obstacles avoidance
* Assembly (peg-in-hole, welding, painting)
* Manipulation (e.g. PUMA robot manipulator)
* Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people

Medicine Application

* Classification and detection (e.g. lesion or cells classification and tumor detection)
* 2D/3D segmentation
* 3D human organ reconstruction (MRI or ultrasound)
* Vision-guided robotics surgery

Industrial Automation Application

* Industrial inspection (defect detection)
* Assembly
* Barcode and package label reading
* Object sorting
* Document understanding (e.g. OCR)

Security Application

* Biometrics (iris, finger print, face recognition)
* Surveillance − Detecting certain suspicious activities or behaviors

Transportation Application

* Autonomous vehicle
* Safety, e.g., driver vigilance monitoring

Features of OpenCV Library

Using OpenCV library, you can −

* Read and write images
* Capture and save videos
* Process images (filter, transform)
* Perform feature detection
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

This tutorial explains the concepts of OpenCV with examples using Java bindings.

**OpenCV Library Modules**

Following are the main library modules of the OpenCV library.

**Core Functionality**

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

**Image Processing**

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

**Video**

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.video**.

**Video I/O**

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.videoio**.

**calib3d**

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.calib3d**.

**features2d**

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.features2d**.

**Objdetect**

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.objdetect**.

**Highgui**

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, **org.opencv.imgcodecs** and **org.opencv.videoio**.

**A Brief History of OpenCV**

OpenCV was initially an Intel research initiative to advise CPU-intensive applications. It was officially launched in 1999.

* In the year 2006, its first major version, OpenCV 1.0 was released.
* In October 2009, the second major version, OpenCV 2 was released.
* In August 2012, OpenCV was taken by a nonprofit organization OpenCV.org

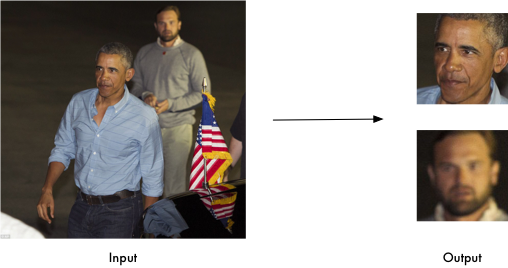
# 5.1.3 Face Recognition package [5]

Recognize and manipulate faces from Python or from the command line with the world’s simplest face recognition library. Built using dlib’s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face\_recognition command line tool that lets you do face recognition on a folder of images from the command line!

# Features

### Find faces in pictures

Find all the faces that appear in a picture:



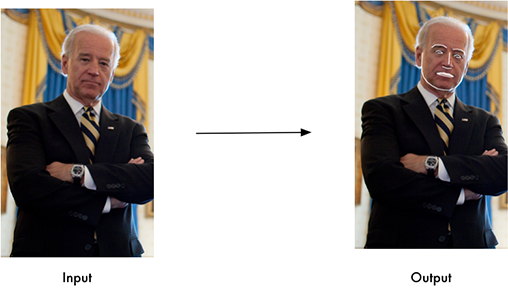
import face\_recognition

image=face\_recognition.load\_image\_file("your\_file.jpg")

face\_locations=face\_recognition.face\_locations(image)

#### Find and manipulate facial features in pictures

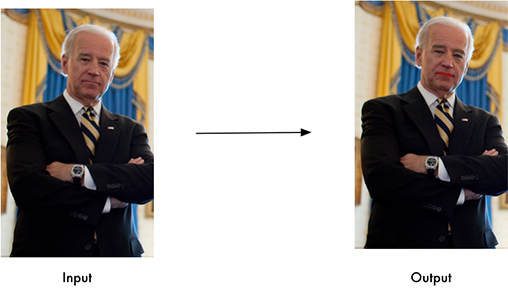
Get the locations and outlines of each person’s eyes, nose, mouth and chin.



import face\_recognition

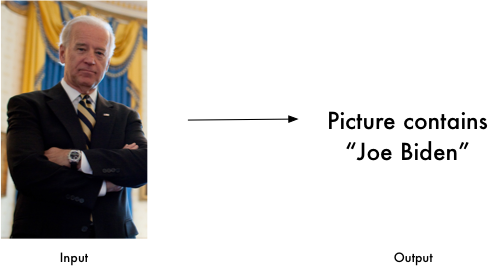
image=face\_recognition.load\_image\_file("your\_file.jpg")

face\_landmarks\_list=face\_recognition.face\_landmarks(image)



#### Identify faces in pictures

Recognize who appears in each photo.



import face\_recognition

known\_image=face\_recognition.load\_image\_file("biden.jpg")

unknown\_image=face\_recognition.load\_image\_file("unknown.jpg")

biden\_encoding=face\_recognition.face\_encodings(known\_image)[0]

unknown\_encoding=face\_recognition.face\_encodings(unknown\_image)[0]

results=face\_recognition.compare\_faces([biden\_encoding],unknown\_encoding)

**5.2 Procedure**

In order to understand how Face Recognition works, let us first get an idea of the concept of a feature vector.

Every Machine Learning algorithm takes a dataset as input and learns from this data. The algorithm goes through the data and identifies patterns in the data. For instance, suppose we wish to identify whose face is present in a given image, there are multiple things we can look at as a pattern:

* Height/width of the face.
* Height and width may not be reliable since the image could be rescaled to a smaller face. However, even after rescaling, what remains unchanged are the ratios – the ratio of height of the face to the width of the face won’t change.
* Color of the face.
* Width of other parts of the face like lips, nose, etc.

Clearly, there is a pattern here – different faces have different dimensions like the ones above. Similar faces have similar dimensions. The challenging part is to convert a particular face into numbers – Machine Learning algorithms only understand numbers. This numerical representation of a “face” (or an element in the training set) is termed as a *feature vector*. A feature vector comprises of various numbers in a specific order.

As a simple example, we can map a “face” into a feature vector which can comprise various features like:

* Height of face (cm)
* Width of face (cm)
* Average color of face (R, G, B)
* Width of lips (cm)
* Height of nose (cm)

Essentially, given an image, we can map out various features and convert it into a feature vector like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Height of face (cm) | Width of face (cm) | Average color of face (RGB) | Width of lips (cm) | Height of nose (cm) |
| 23.1 | 15.8 | (255, 224, 189) | 5.2 | 4.4 |

So, our image is now a vector that could be represented as (23.1, 15.8, 255, 224, 189, 5.2, 4.4). Of course there could be countless other features that could be derived from the image (for instance, hair color, facial hair, spectacles, etc). Now we have encoded each image into a feature vector. Clearly, when we have 2 faces (images) that represent the same person, the feature vectors derived will be quite similar. Put it the other way, the “distance” between the 2 feature vectors will be quite small.

Machine Learning can help us here with 2 things:

1. *Deriving the feature vector*: it is difficult to manually list down all of the features because there are just so many. A Machine Learning algorithm can intelligently label out many of such features. For instance, a complex features could be: ratio of height of nose and width of forehead. Now it will be quite difficult for a human to list down all such “second order” features.
2. *Matching algorithms*: Once the feature vectors have been obtained, a Machine Learning algorithm needs to match a new image with the set of feature vectors present in the corpus.

**5.3 TRAINING THE MODEL**

To train the model, we used the following 2 sample images:

EXTRACTED FACE

Image 1

Image 2

**6. PROGRAM CODE**

import cv2

import numpy as np

import face\_recognition

import os

from datetime import datetime

path = 'images'

images = []

personNames = []

myList = os.listdir(path)

print(myList)

for cu\_img in myList:

current\_Img = cv2.imread(f'{path}/{cu\_img}')

images.append(current\_Img)

personNames.append(os.path.splitext(cu\_img)[0])

print(personNames)

def faceEncodings(images):

encodeList = []

for img in images:

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

encode = face\_recognition.face\_encodings(img)[0]

encodeList.append(encode)

return encodeList

def attendance(name):

with open('Attendance.csv', 'r+') as f:

myDataList = f.readlines()

nameList = []

for line in myDataList:

entry = line.split(',')

nameList.append(entry[0])

if name not in nameList:

time\_now = datetime.now()

tStr = time\_now.strftime('%H:%M:%S')

dStr = time\_now.strftime('%d/%m/%Y')

f.writelines(f'\n{name},{tStr},{dStr}')

encodeListKnown = faceEncodings(images)

print('All Encodings Complete!!!')

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

faces = cv2.resize(frame, (0, 0), None, 0.25, 0.25)

#converted bgr value into rgb

faces = cv2.cvtColor(faces, cv2.COLOR\_BGR2RGB)

#it will find the face location in the current frame and store it in the facescurrentframe

facesCurrentFrame = face\_recognition.face\_locations(faces)

#now we will be encoding the faces of the camera and the images

encodesCurrentFrame = face\_recognition.face\_encodings(faces, facesCurrentFrame)

#now we will write the program to find if the faces are matching or not

# we are reading the values

#now we will find the face distance

for encodeFace, faceLoc in zip(encodesCurrentFrame, facesCurrentFrame):

#here we are matching the faces

matches = face\_recognition.compare\_faces(encodeListKnown, encodeFace)

#here we find the distance

faceDis = face\_recognition.face\_distance(encodeListKnown, encodeFace)

# print(faceDis)

matchIndex = np.argmin(faceDis)

if matches[matchIndex]:

name = personNames[matchIndex].upper()

# print(name)

y1, x2, y2, x1 = faceLoc

y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.rectangle(frame, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)

cv2.putText(frame, name, (x1 + 6, y2 - 6), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 2)

attendance(name)

cv2.imshow('Webcam', frame)

if cv2.waitKey(1) == 13:

break

cap.release()

cv2.destroyAllWindows()

**7. TESTING**

**7.1 CREATING ENVIRONMENT**

Creating new environment in anaconda promt

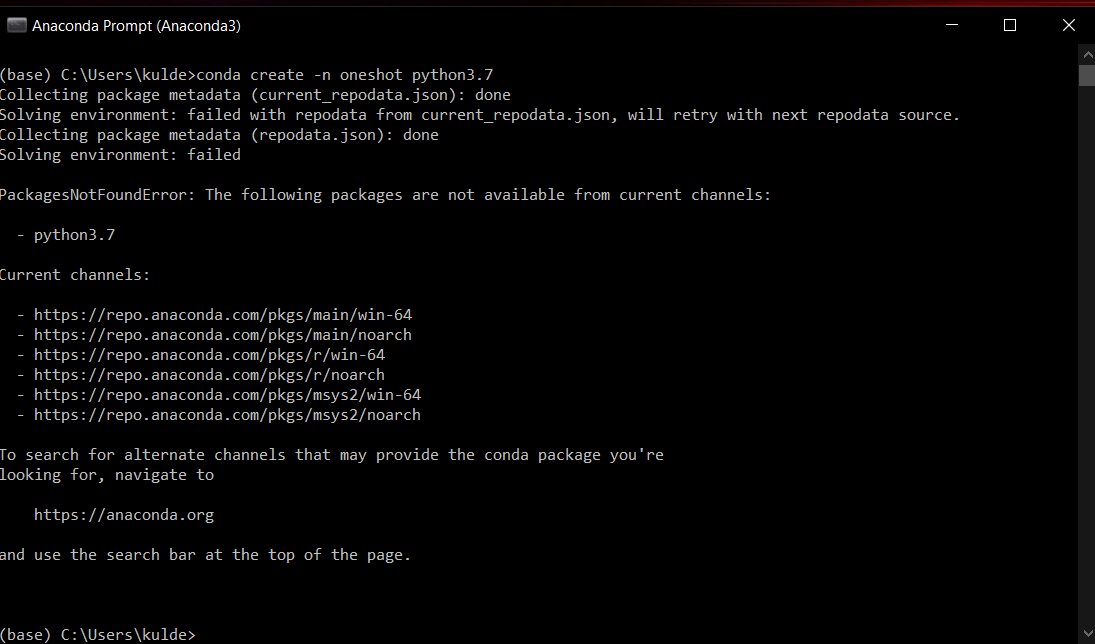
****

Fig 1

**7.2 INSTALLING REQUIRED LIBRARIES**

Installing required libraries in the environment

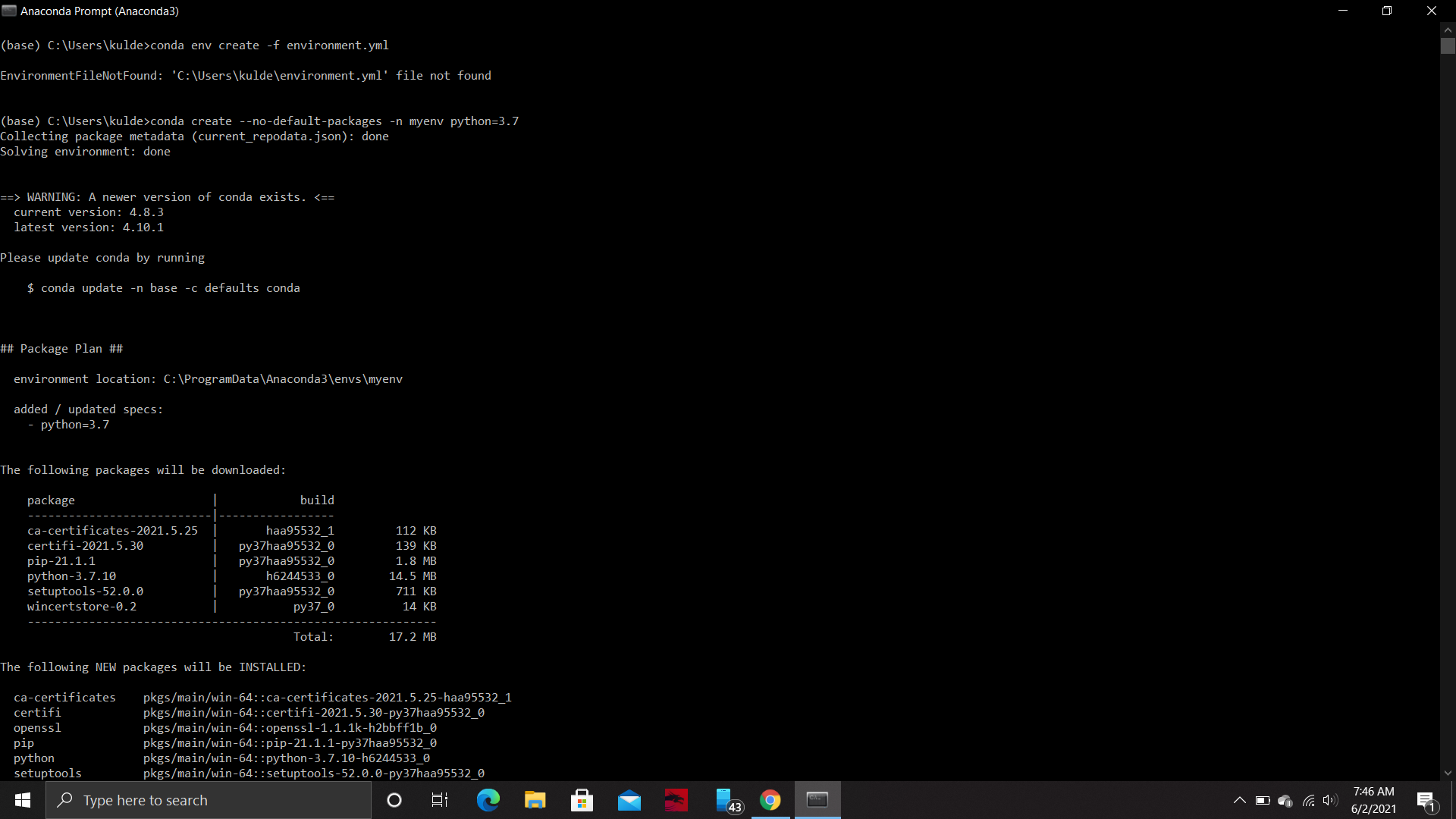


Fig. 2

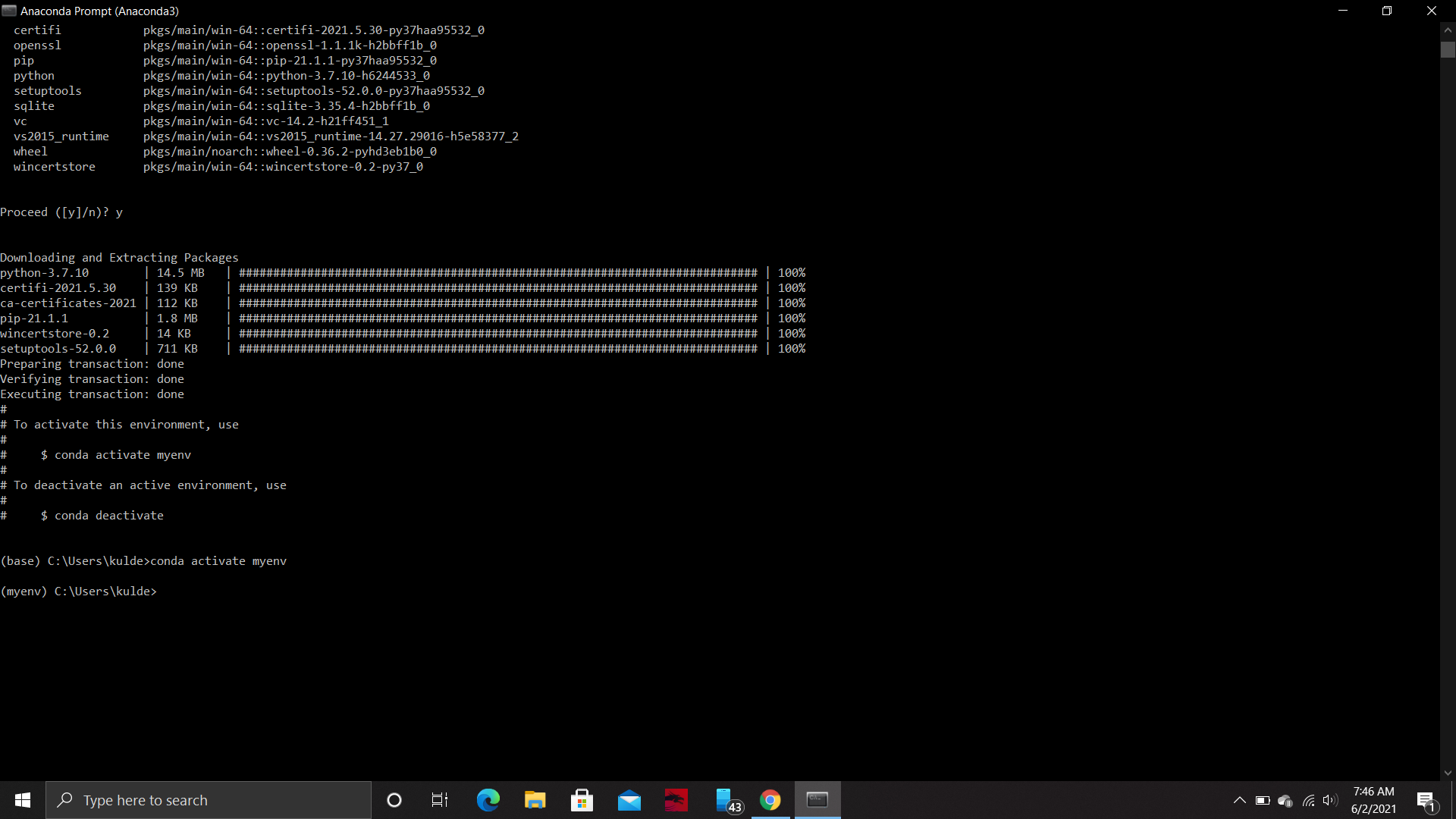


Fig. 3

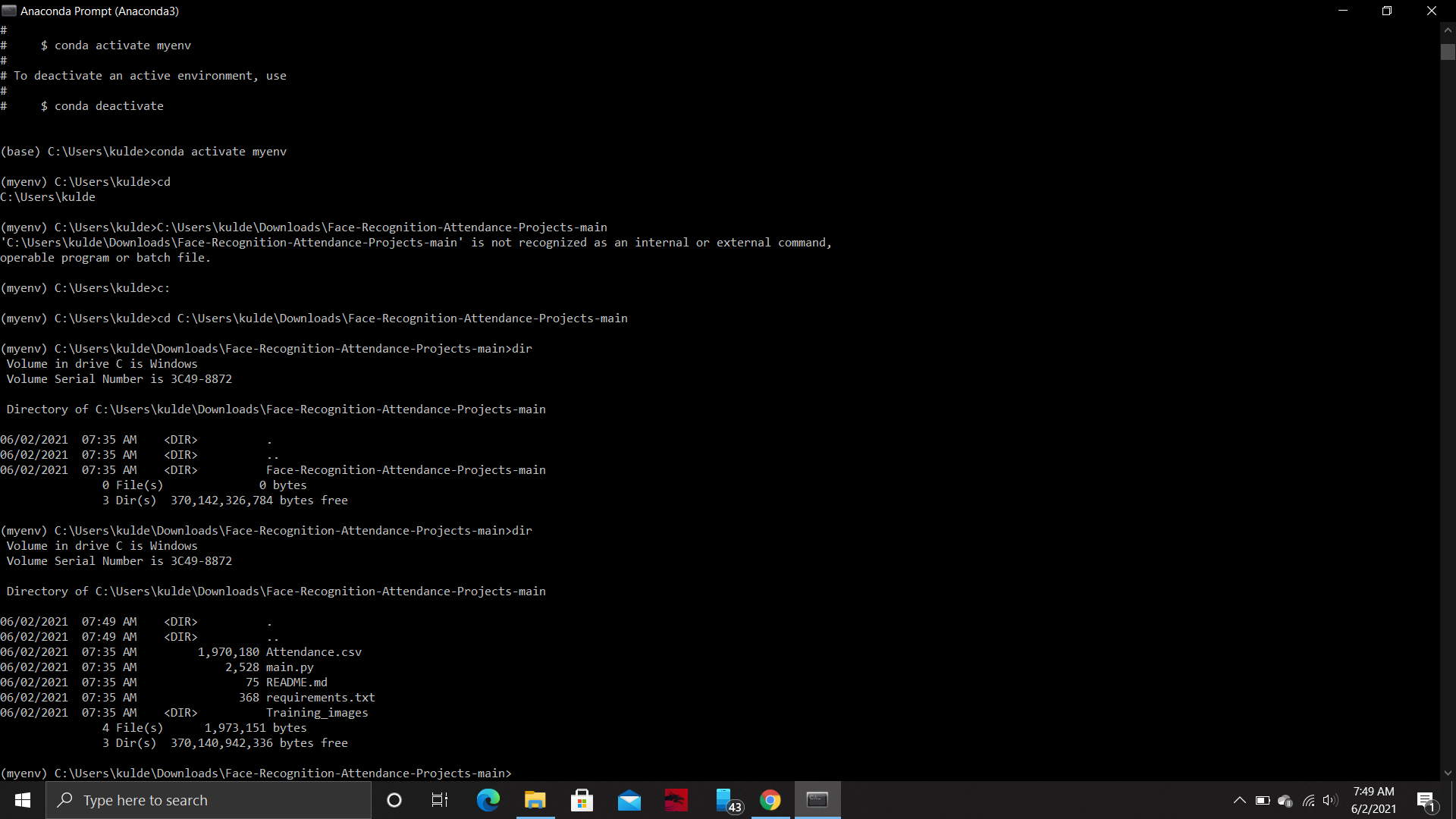


Fig. 4

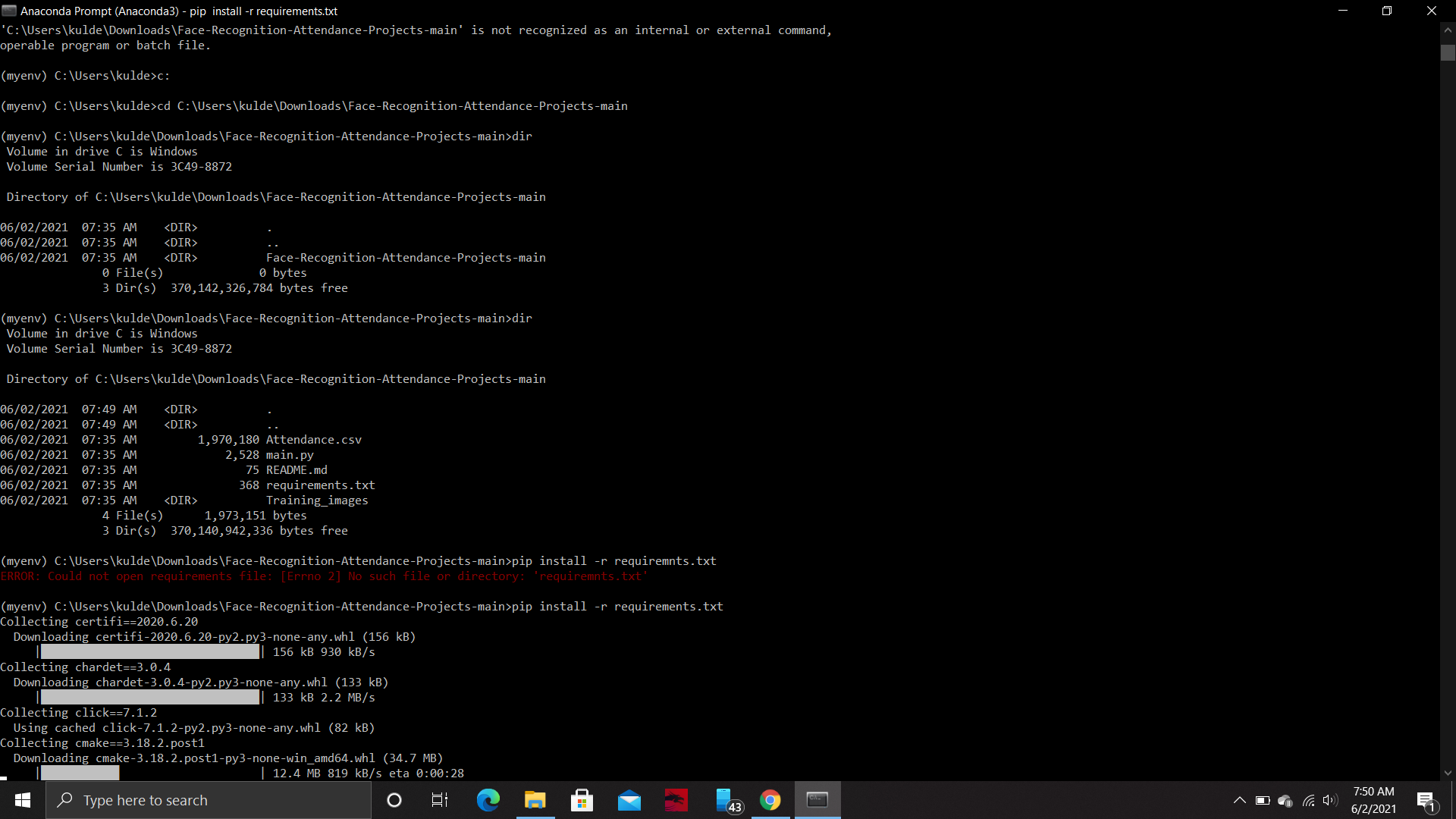


Fig. 5

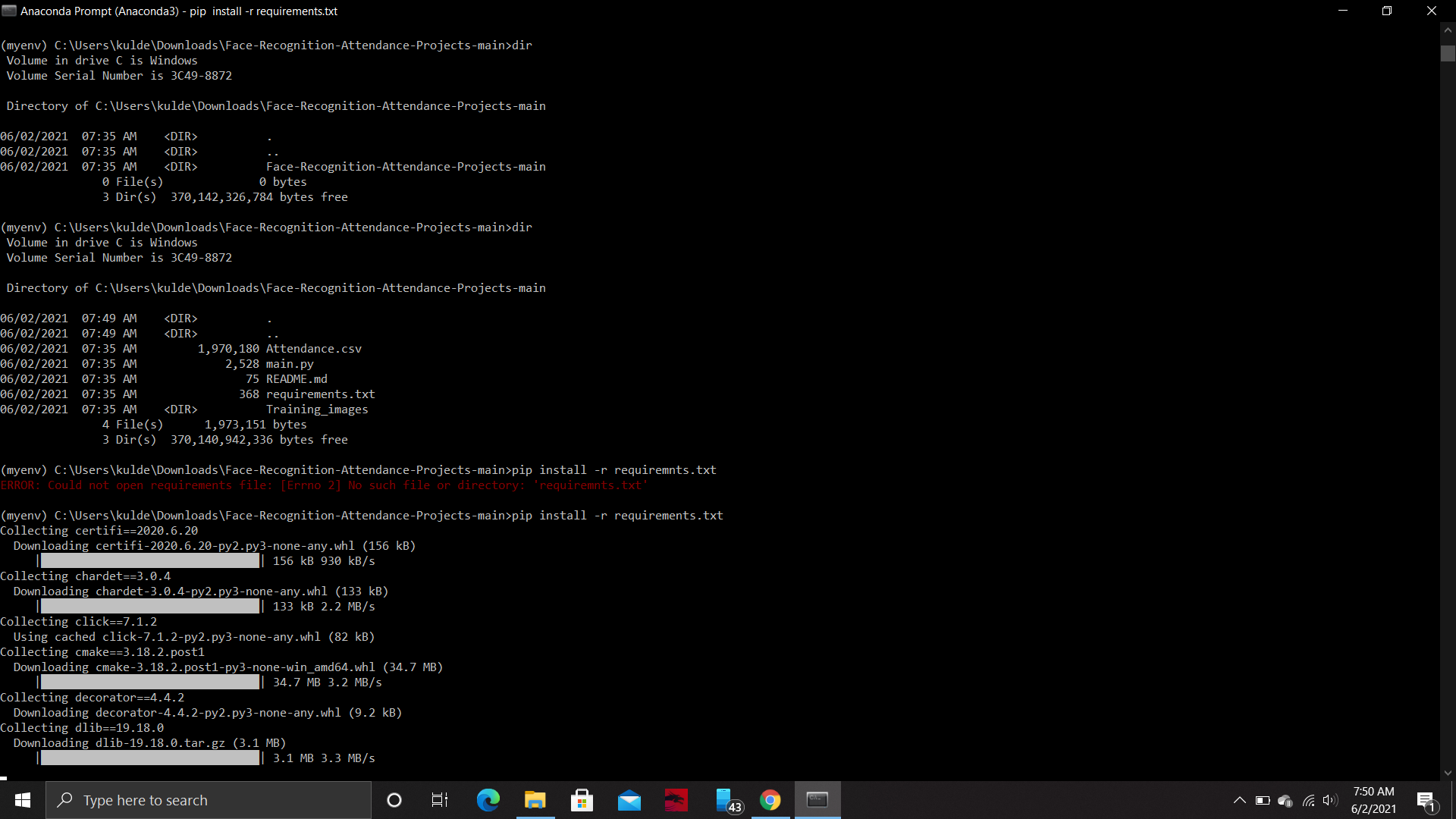


Fig. 6

Downloading Visual Studio for the ‘dlib’ library

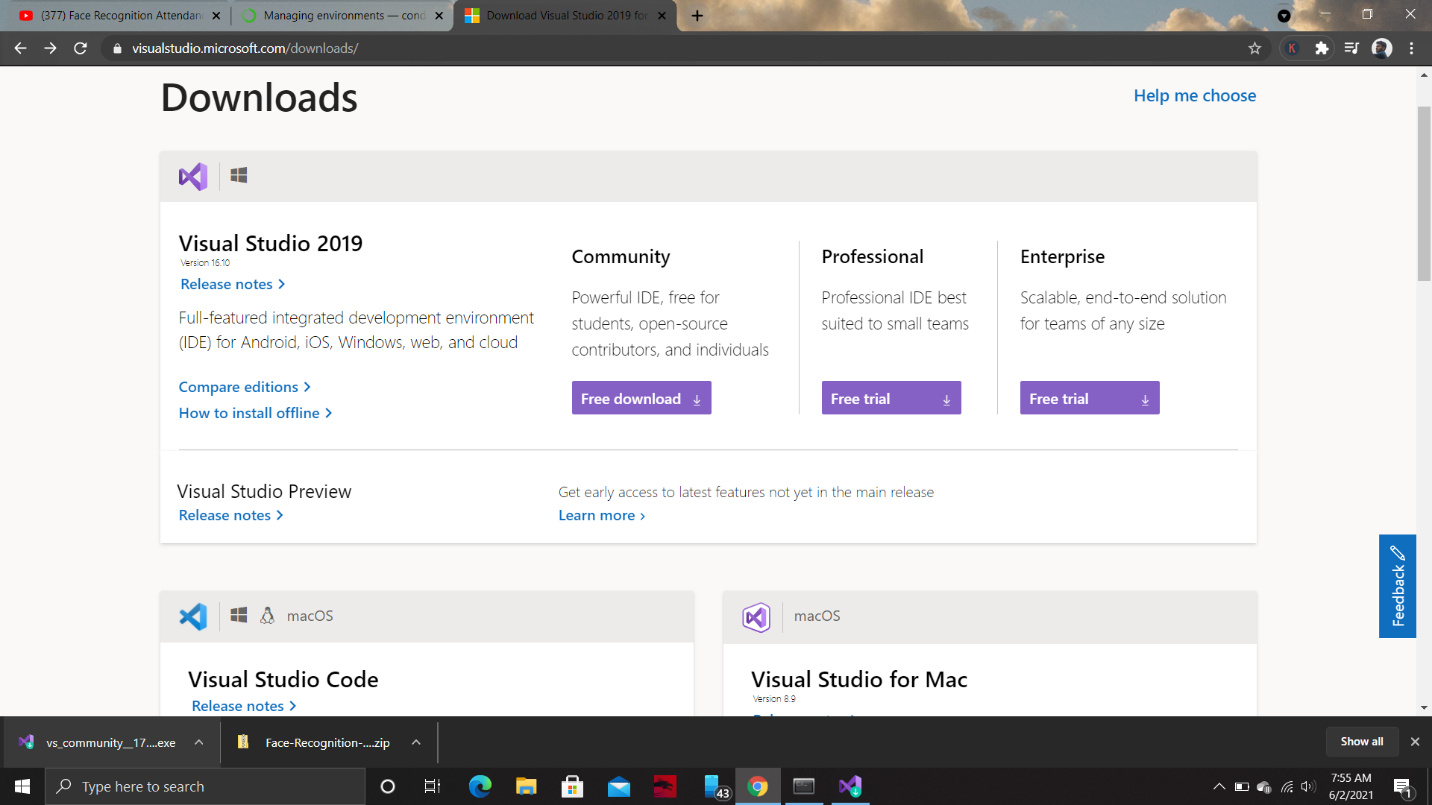


Fig. 7

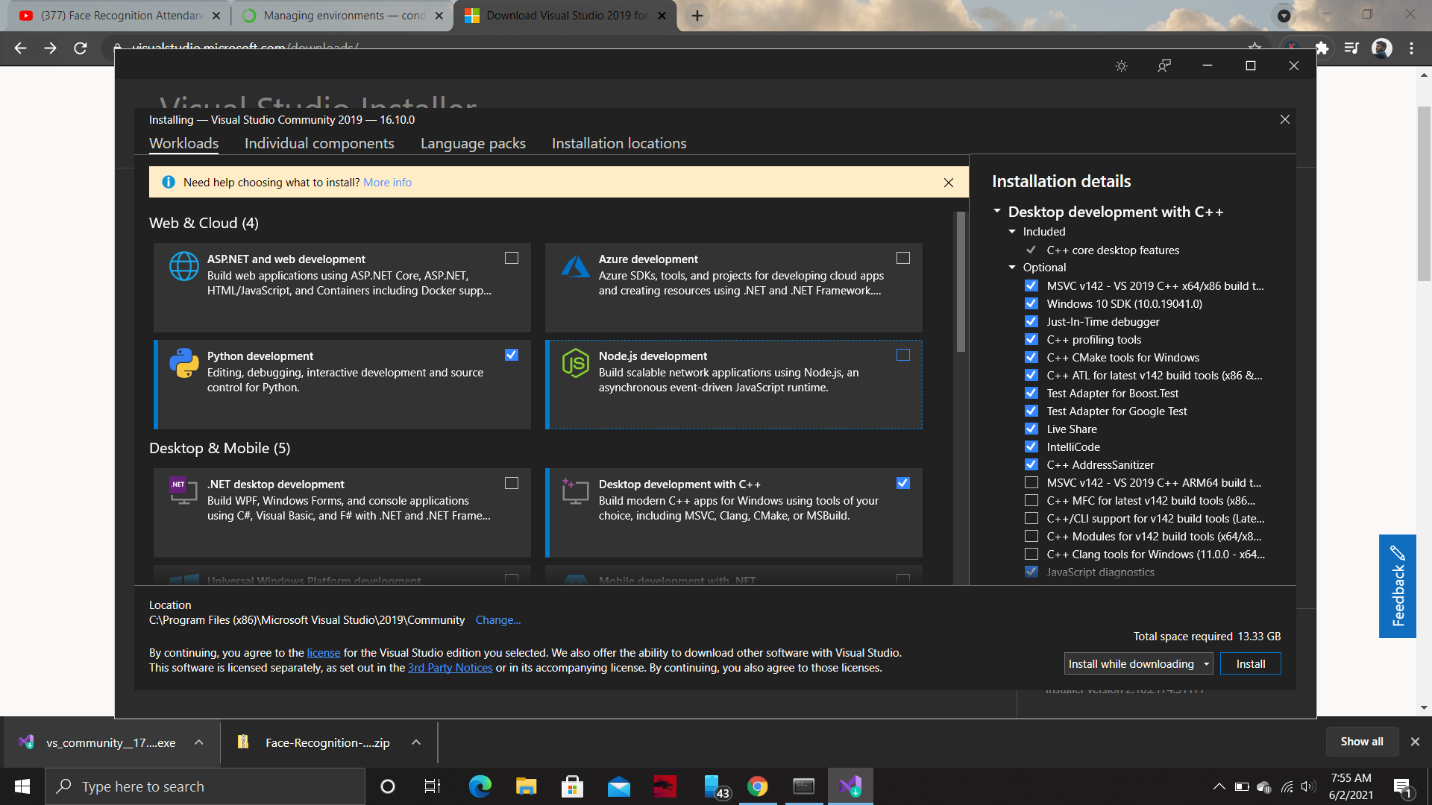


Fig. 8

Installing Cmake library in the environment

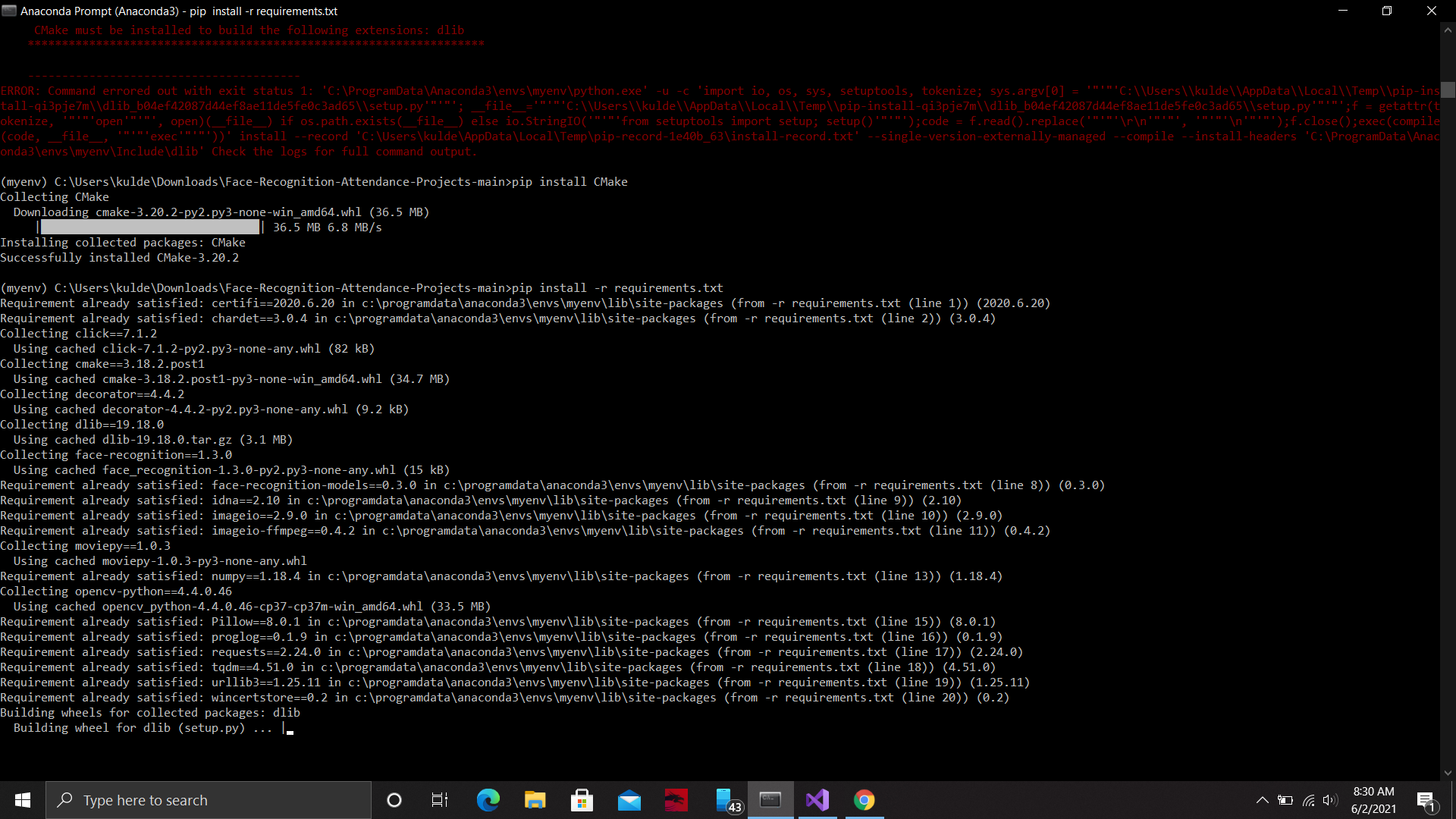


Fig. 9

**7.3 RUNNING THE CODE**

Running the code in the jupyter notebook

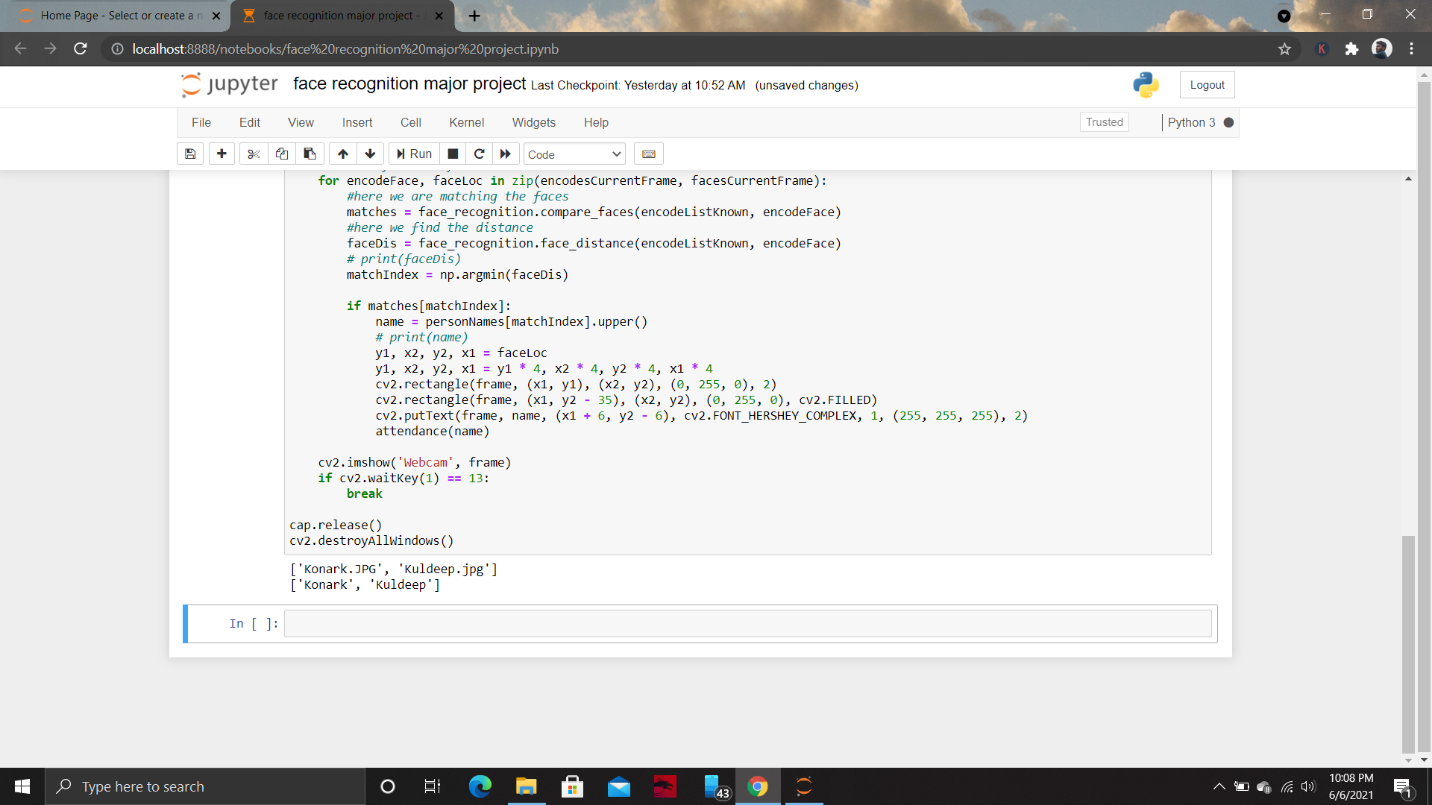


Fig. 10

**8. RESULT**

Output Window

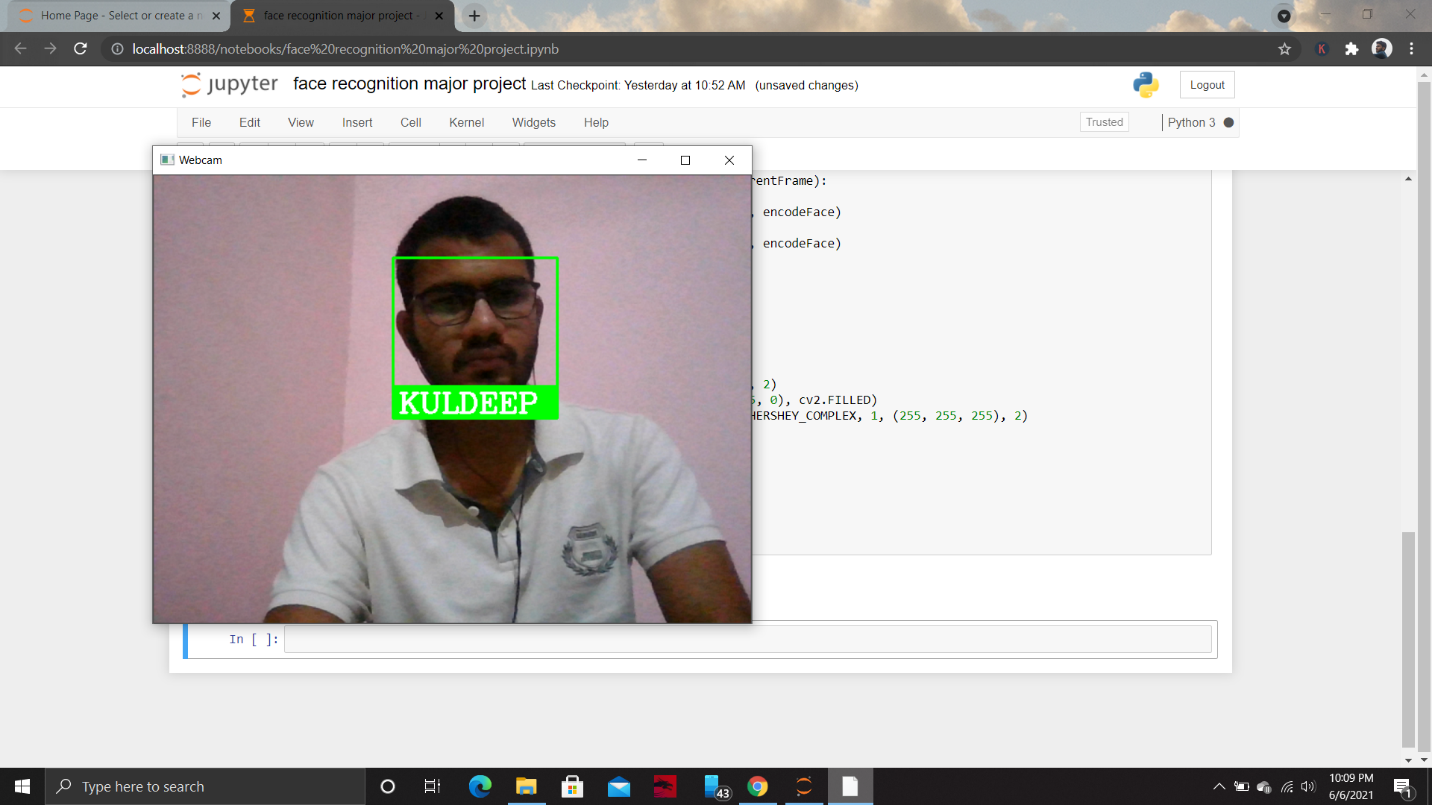


Fig. 11

Marking the attendance in the excel sheet

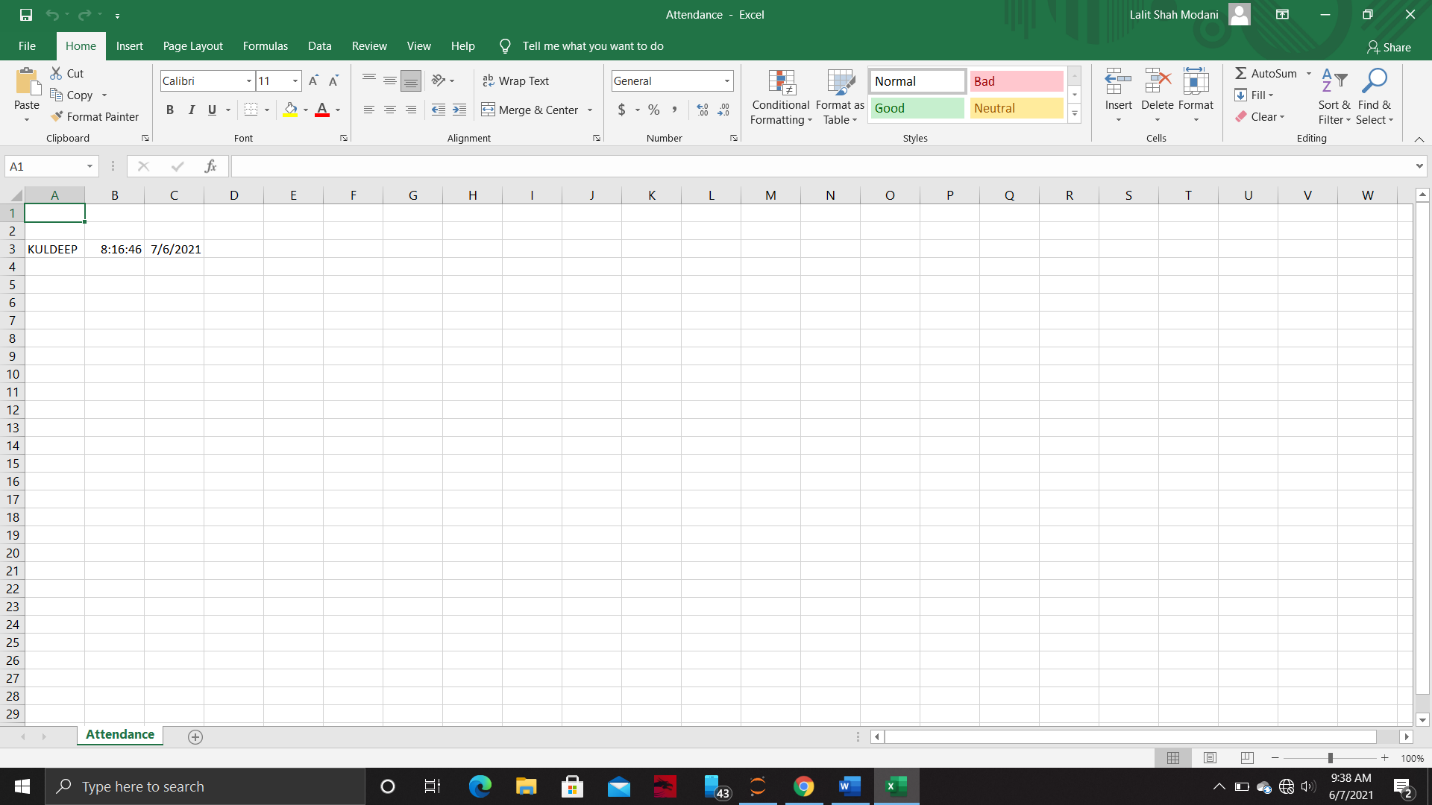


Fig. 12

**9. REFERENCES & WEBSITES USED**

1. Publications & Media – NEC New Zealand
2. Anaconda official documentation
3. Python official documentation
4. OpenCV official documentation
5. Facial Recognition library Official Face Recognition Documentation, Release 1.4.0

Websites:

1. >https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/
2. >https://www.mygreatlearning.com/blog/face-recognition/
3. >https://www.analyticsvidhya.com/blog/2018/08/a-simple-introduction-to-facial-recognition-with-python-codes/
4. >https://www.nec.co.nz/market-leadership/publications-media/a-brief-history-of-facial-recognition/
5. >https://www.eff.org/pages/face-recognition
6. >https://face-recognition.readthedocs.io/en/latest/readme.html

# A brief history of Facial Recognition