Face Detection

Project report in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE

Submitted By

Simran Singh	University Roll No. 12019009002048
Sneha Agrawal	University Roll No. 12019009001154
Ahana Sen	University Roll No. 12019009001406
Ankush Khanra	University Roll No. 12019009022027
Snehasish Ghosh	University Roll No. 12019009002157
Siddhant Kumar	University Roll No. 12019009022129

Under the guidance of

Prof. (Dr.) Sibaprasad Sen

Department of Computer Science

UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

University Area, Plot No. III – B/5, New Town, Action Area – III, Kolkata – 700160.

CERTIFICATE

This is to certify that the project titled **Face Detection** submitted by **Simran Singh(University Roll No. 12019009002048),Sneha Agrawal(University Roll No. 12019009001154),Ahana Sen(University Roll No. 12019009001406),Ankush Khanra (University Roll No. 12019009022027),Snehasish Ghosh(University Roll No. 12019009002157)** and **Siddhant Kumar(University Roll No. 12019009022129)**, students of UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA, in partial fulfillment of requirement for the degree of Bachelor of Computer Science, is a bonafide work carried out by them under the supervision and guidance of Prof. (Dr.) Sibaprasad Sen during 4th Semester of academic session of 2020 - 2021. The content of this report has not been submitted to any other university or institute. I am glad to inform that the work is entirely original and its performance is found to be quite satisfactory.

Prof. (Dr.) Sibaprasad Sen	Prof. Sukalyan Goswami
Assistant Professor	Head of the Department
Department of Computer Science	Department of Computer Science
UEM, Kolkata	UEM, Kolkata

ACKNOWLEDGEMENT

We would like to take this opportunity to thank everyone whose cooperation and encouragement throughout the ongoing course of this project remains invaluable to us.

We are sincerely grateful to our guide Prof. (Dr.) Sibaprasad Sen of the Department of Computer Science, UEM, Kolkata, for his wisdom, guidance and inspiration that helped us to go through with this project and take it to where it stands now.

We would also like to express our sincere gratitude to Prof. Sukalyan Goswami, HOD, Computer Science, UEM, Kolkata and all other departmental faculties for their ever-present assistant and encouragement.

Last but not the least, we would like to extend our warm regards to our families and peers who have kept supporting us and always had faith in our work.

Simran Singh

Sneha Agrawal

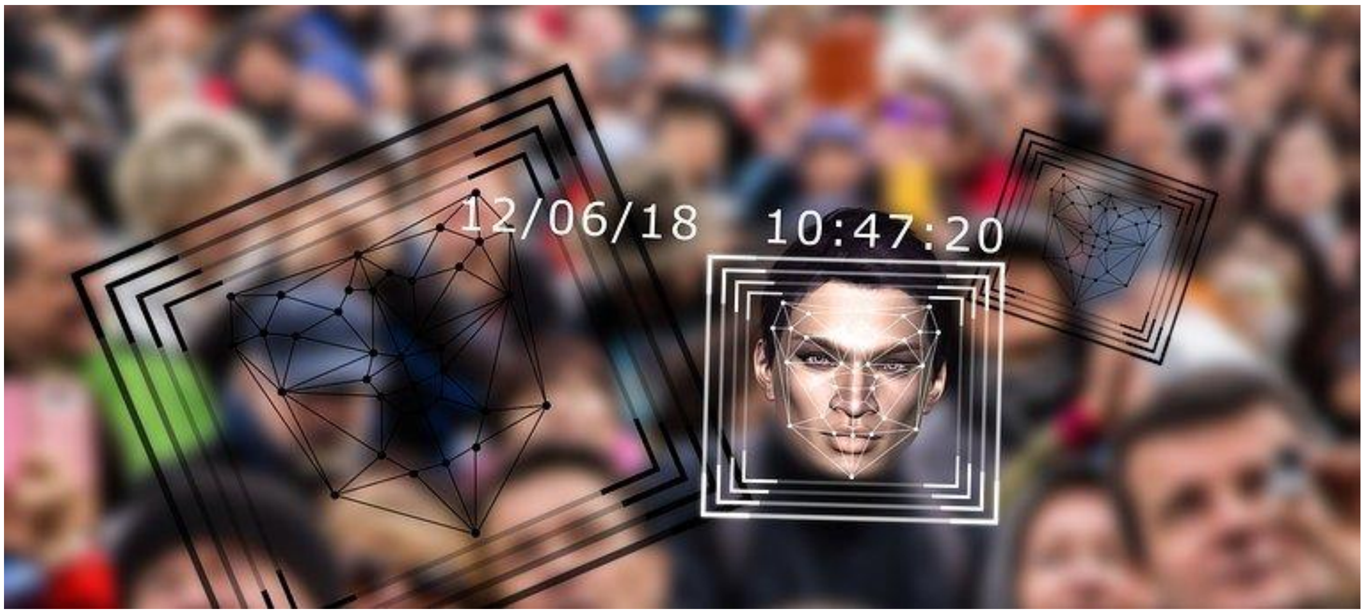
Ahana Sen

Ankush Khanra

Snehasish Ghosh

Siddhant Kumar

TABLE OF CONTENTS


12/06/18
10:47:20

ABSTRACT

One of the simplest ways to tell who someone is by looking at their face. Face recognition is a form of personal identification system that uses a person's personal characteristics to determine their identity. The process of human face recognition is divided into two phases: face identification, which occurs very quickly in humans except in situations where the object is positioned at a short distance away, and presentation, which recognizes a face as an entity. The stage is then repeated and developed as a model for facial image recognition (face recognition), which is a well-studied and developed biometrics technology. Image processing is the focus of this project's face detection system with face recognition.

In the last two decades, face recognition has been a hot topic of study. Many attempts have been made to explain how humans identify each other's faces. Face recognition is widely agreed to rely on both componential (such as eyes, mouth, and nose) and non-componential/holistic (the spatial relationships between these features) information, but how these cues should be optimally integrated is unknown. Using eigen/fisher features of multi-scaled face components and artificial neural networks, a new observer's approach is proposed in this research. The proposed method's basic concept is to build a facial feature vector by down-sampling.

The proposed method's basic concept is to construct a facial feature vector by down sampling face components including eyes, nose, mouth, and whole face with different resolutions based on their importance.

1. INTRODUCTION

The face is one of the most widely accepted biometrics, as well as the most popular means of identification in human visual interactions. Data acquisition has been an issue with authentication systems based on fingerprint, speech, iris, and the most recent gene structure (DNA fingerprint). For example, in order to recognize a fingerprint, the individual should hold his or her finger in the proper position and orientation, and in order to recognize a speaker, the microphone should be in the proper position and distance from the speaker.

However, since the process of obtaining face images is non-intrusive, the face can be used as a biometric trait in a covert scheme (one in which the individual is unaware that he is being monitored). The human face is a universal feature. Face recognition is important not only because of the numerous possible applications in research fields, but also because of the solution's ability to aid in the solution of other classification problems such as object recognition.

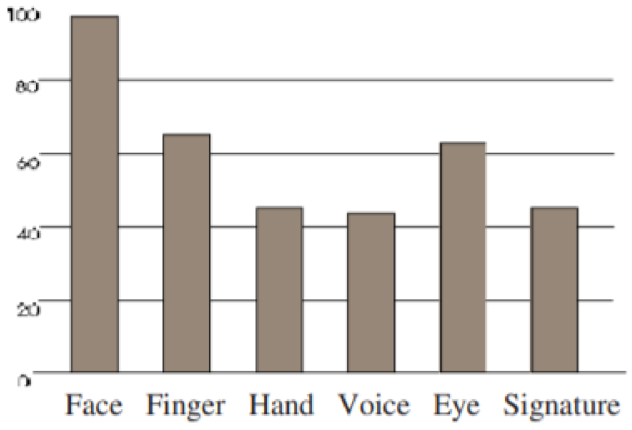


Fig 1: Why face regonition is important (Source: Google Images)

In this project, with the help of pre-loaded dataset we are training the data. The data after being trained is detecting the face in the loaded images.

Face recognition technology has the advantage of being quick to process and requiring no user interaction. Face recognition technology allows businesses to monitor access to facilities without having to wait in long lines. Systems will one day be able to verify identities without stopping anyone. Physical protection isn't the only application; cyber security is also included. Face recognition technology can be used to unlock computers in place of passwords.

One of the most frequently discussed applications of facial recognition technology is law enforcement. The technology is spreading among law enforcement agencies in the United States, according to an NBC storey. Hundreds of state and local law enforcement departments, as well as cities like Los Angeles, New York, and Chicago, are using the technology. In addition, by 2023, US Customs and Border Protection (CBP) plans to use facial recognition to screen 97 percent of foreign passengers. As this CBP study confirms, CBP intends to check the departure of non-immigrant visa holders. CBP has a system in place that tests boarding airline passenger lists at the moment. It does not, however, include biometrics.

Those in favor of the technology cite the following benefits of face recognition:

- Face recognition makes it easier to track down burglars, thieves and trespassers. The technology is capable of analyzing the feed private and public CCTV camera networks.
- The technology is not limited to tracking down criminals. For instance, it could also make it easier to find missing children and seniors.
- Face recognition could make security checkpoints at airports less intrusive to passengers.

Fig 2: CCTV regonizes face (Source: Pixabay)

The public's anger about unjustified stops and searches is a constant source of annoyance and controversy for the police — yet another field where facial recognition technology could be useful. Face recognition technology could help reduce stops and searches of law-abiding citizens by identifying criminals within crowds.

Furthermore, simply being aware of the existence of a face recognition system acts as a deterrent, especially in the case of petty crime.

Another advantage of face recognition technology is its speed of processing and the fact that it does not require any user interaction. Users must recall passwords, present I.D. cards, and endure other inconveniences with existing identity authentication methods.

Face recognition technology allows businesses to monitor access to facilities without having to wait in long lines. Systems will one day be able to verify identities without stopping anyone.

Physical protection isn't the only application; cyber security is also included. Face recognition technology can be used to unlock computers in place of passwords.

Despite the numerous security steps taken by financial institutions around the world, banking fraud continues to be an issue. Face recognition online banking may be able to help solve the problem.

Although banks have advanced in their use of one-time passwords to gain access to accounts and allow transactions, there is still room for improvement. Biometric authentication via facial recognition technology is one potential solution.

Fig 3: Storing employees details via face regonition (Source: Pixabay)

You might authorize transactions by looking at your smart phone or device instead of those annoying one-time passwords. Another advantage of face recognition is biometric online banking.

Face recognition eliminates the possibility of hackers gaining access to passwords. Even if hackers stole your picture database, they'd be useless because “liveness detection” prevents them from being used for impersonation.

The technology can be used in physical bank branches and ATMs, in addition to online. Using debit cards and signatures can soon be obsolete as a result of it.

Although a growing number of consumers shop online, physical stores continue to attract a significant portion of the market. Although foot traffic has decreased, the value of each visit has increased threefold. One theory is that consumers schedule their purchases online and then complete them in the shop. In addition, searches for "near me" have increased by 50%.

Some of these benefits of facial recognition in retail are:

- Shorter Lines: Forget long checkout lines with slow payments, new “face pay” technologies can shorten them.
- Loss prevention: Shoplifting is a notable problem for retailers. Now imagine that retailers could identify subjects of interest when they are coming into stores. The technology still has some way to go to prevent misidentifications and biases, but the potential is there.
- Increase customer loyalty: Imagine if when entering your store, a kiosk recognizes customers, points them in the right direction, or makes suggestions based on their history. More efficient and personalized experiences foster comebacks and repeat purchases.

Time theft is still one of the most prevalent work ethics breaches on industrial floors around the world. The vast majority of employees are trustworthy. Even so, a small percentage of employees use buddy favors from coworkers or security personnel to get out of work (and still get paid). Not only is time fraud bad for businesses, but it's also bad for good employees.

Another advantage of face recognition technology is that it has the potential to eliminate time fraud. Here's how:

- To sign in for work, everyone must pass a face-scanning system. From now until checkout, you'll be charged for your time (Also with facial recognition).
- Employees don't have to prove their identities or clock in with plastic cards, so the process is fast.

Fig 4: Face Regonition for attendance at job (Source: Pixabay)

Face Recognition research could one day allow computers to see and recognise objects and people in the same way that humans do. As a result, new opportunities for interaction will emerge.

Computer vision research is crucial to the viability of innovations like self-driving vehicles. We should expect more interest in computer vision and face recognition research in the future, as well as more breakthroughs.

Advantages of Face Recognition

a. Helps find missing people

b. Protects business against theft

c. Strengthens security measures

d. Reduces the number of touch points

e. Makes shopping more efficient

f. Improves photo organization

g. Improves medical treatment

Disadvantages of Face Recognition

a. Threatens individual and societal privacy

b. Imposes on personal freedom

c. Violates personal rights

d. Creates data vulnerabilities

e. Provides opportunities for fraud and other crimes

f. Technology is imperfect

g. Innocent people could be charged

h. Technology can be fooled

2. LITERATURE SURVEY

Face detection is a technique with a wide range of applications, including face tracking, poses estimation, and compression. Face detection is a two-class problem in which we must determine whether or not there is a face in a picture. This method can be thought of as a simpler solution to the issue of face recognition.

Face recognition has been an active research area over last 40 years. The face recognition research has several disciplines such as image processing, machine learning approach, pattern recognition, computer vision, and neural networks. Face recognition has many applications in the fields of biometrics, security system, surveillance systems, and access control and law enforcement. The limitation of face recognition system can be stated as given still face images or video of a scene, identifying or detecting one or more persons in the given video by using a stored database of face images [R. Chellappa et. al. 1995]. Classification is the main problem. In the process of face recognition it includes, to train the face images from the known individuals and then to classify the newly coming test images into one of the classes. The problem of face recognition is easily solved by humans where limited memory can be the main problem. The problems or limitations for a machine learning face recognition system are:

1. Facial expression change 2. Illumination variation 3. Ageing 4. Pose change 5. Scaling factor (i.e. size of the image) 6. Frontal vs. profile 7. Presence and absence of spectacles, beard, mustache etc. 8. Occlusion due to scarf, mask or obstacles in front.

In automatic face recognition system the main complicated task is that it involves detection of faces from a cluttered background, facial feature extraction, and face recognition. A complete face recognition system has to solve all sub-problems, where each one is a separate research problem. Image template based and geometry feature-based are the two classes of face recognition system algorithms. In template based method it (Robert J. 1981) compute the correlation between a face image and one or more model of face image templates to estimate the face image identity from the database. Brunelli and Poggio (R. Brunelli, 1993) suggest the optimal strategy for face recognition system which is holistic and corresponds to template matching. The statistical tools such as Support Vector Machines (SVM) (E. Osuna, 1997), (Vladimir N, 1995) Independent component Analysis, Principal Component Analysis (PCA) (L. Sirovich, 1987), (Matthew Turk, 1991), Linear Discriminant Analysis (LDA) (Peter N. Belhumeur et.al, 1997), kernel methods (Bernhard Scholkopf et.al, 1998), (M. H. Yang, 2002), and neural networks (A. Jonathan, 1995), (Steve Lawrence, 1998), (T. Poggio, 1994) used to construct a suitable database of face image templates.

Other than neural network approach and statistical approach there are other approaches known as hybrid approaches which are the combination of both statistical pattern recognition techniques and neural network systems. Examples for hybrid approaches include the combination of PCA and Radial Basis Function (RBF) neural network (M. J. Er, 1999), (C. E. Thomaz et. al, 1998). Among other methods, people have used range (R. Chellappa, 1995), infra-red scanned (Y. Yoshitomi et. al, 1997) and profile (Z. Liposcak, 1999) images for face recognition. While templates can be viewed as features, they mostly capture global features of the face image. Facial occlusion (Face images with goggles, specs, scarf etc) and low resolution is often difficult to handle in these given approaches. In the geometry feature based methods the explicit local facial features are found, and their geometric relationships. Cootes et al. (Andreas Lanitis et.al, 1997) have presented an active shape model which was the extending approach by Yuille (Alan L, 1991). Wiskott et al.(Laurenz Wiskott, 1997) developed an elastic bunch graph matching algorithm for face identification. Penev et al. (P. Penev, 1996) developed PCA into Local Feature Analysis (LFA). This technique is one of the most successful and useful commercial face recognition systems, FaceIt.

Template based Methods :Template matching is conceptually related to holistic approach which attempts to identify faces using global representations (J. Huang, 1998). These types of methods approach the face image as a whole and try to extract features from the whole face region and then classify the image by applying a pattern classifier. One of the methods used to extract features in a holistic system, is based on statistical approaches.

Statistical Approaches :There are some techniques that identify, parameterize and analyze linear subspaces. Other than linear subspaces there are some statistical face recognition techniques which are based on nonlinear subspaces (like kernel-PCA and kernel-LDA), transformation (like DCT, DCT & HMM and Fourier Transform) and Support Vector Machine (SVM). Appearance-based approaches for face recognition like PCA, LDA, and probabilistic subspace view a 2D face image as a vector in image space.

Neural Network based Approaches :Artificial Neural Network (ANN) (B. Yegnanarayana, 1999) is a most successful tool for pattern recognition problems. In Kohonen's associative map (T. Kohonen, 1998), one of the earliest demonstrations of neural network for face image recall applications is reported. Using a small set of face images, accurate recall was reported even when input image is very noisy, low resolution and dimension or when portions of the images are missing.

Hop-field memory model: In (Y. Dai, 1998), a Hop-field memory model for the facial images is organized and the optimal procedure of learning is determined. A method for face recognition using Hop-field memory model combined with the pattern matching is proposed. It shows better performance of database having 20 faces of 40 subjects.

3. PROBLEM STATEMENT

Given an image of any particular scene, identify face of one person in the scene using a stored database of faces.

4. PROPOSED SOLUTION

4.1 Using Deep Learning

Deep learning is a subfield of machine learning that deals with artificial neural networks, which are algorithms inspired by the structure and function of the brain.

Machine Learning, on the other hand, is a subset of Artificial Intelligence, and Deep Learning is a subset of Machine Learning. Artificial intelligence (AI) is a broad term that refers to methods that enable computers to imitate human behavior. All of this is made possible by machine learning, which is a series of algorithms trained on data.

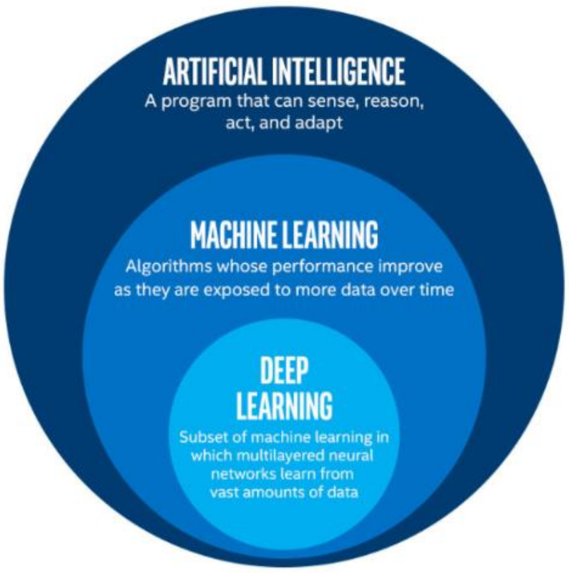

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Fig 5: Deep Learning is a subset of AI & ML (Source: TowardsDataScience)

Deep Learning, on the other hand, is a form of Machine Learning that is influenced by the human brain's structure. Deep learning algorithms analyse data with a predetermined logical framework in order to draw similar conclusions as humans. Deep learning does this by using a multi-layered system of algorithms known as neural networks.

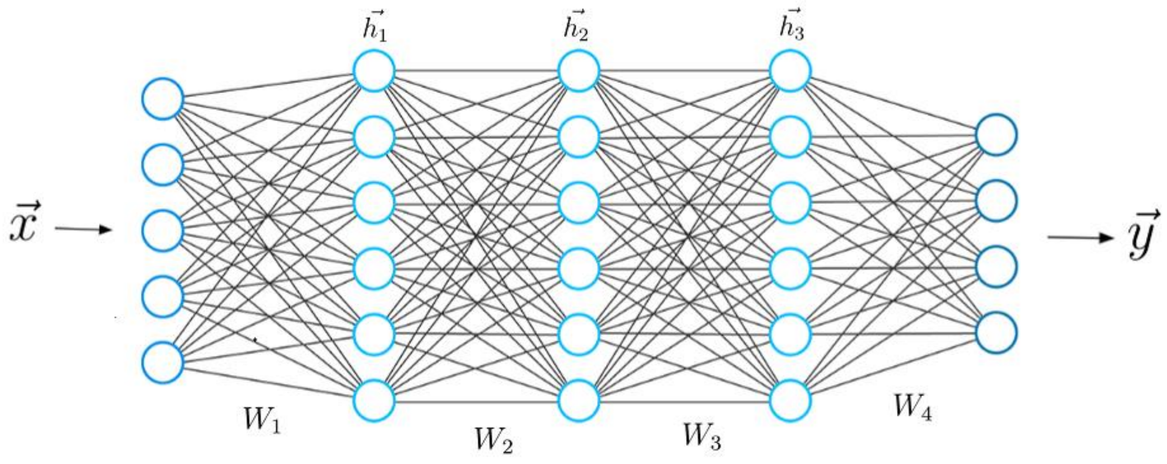


Fig 6: Layers in Deep Learning (Source: TowardsDataScience)

The neural network's architecture is inspired by the structure of the human brain. Neural networks can be trained to perform the same tasks on data as our brains do when identifying patterns and classifying various types of knowledge.

Individual layers of neural networks may also be thought of as a kind of filter that operates from the most obvious to the most subtle, raising the probability of detecting and producing a correct result.

The human brain operates in a similar manner. When we obtain new knowledge, our brain attempts to equate it to previously encountered objects. Deep neural networks make use of the same principle.

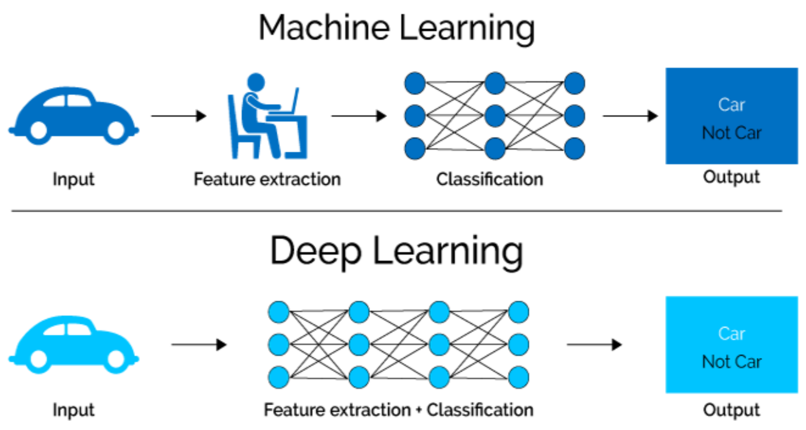


Fig 7: Difference between ML and Deep Learning (Source: Morioh)

We may use neural networks to perform a variety of tasks, such as clustering, sorting, and regression. We can use neural networks to group or sort unlabeled data based on similarities between the samples. Alternatively, in the case of classification, we can train the network on a named dataset in order to categories the samples in the dataset.

Deep learning models can solve tasks that machine learning models can't. Artificial neural networks have specific features that allow deep learning models to solve tasks that machine learning models can't.

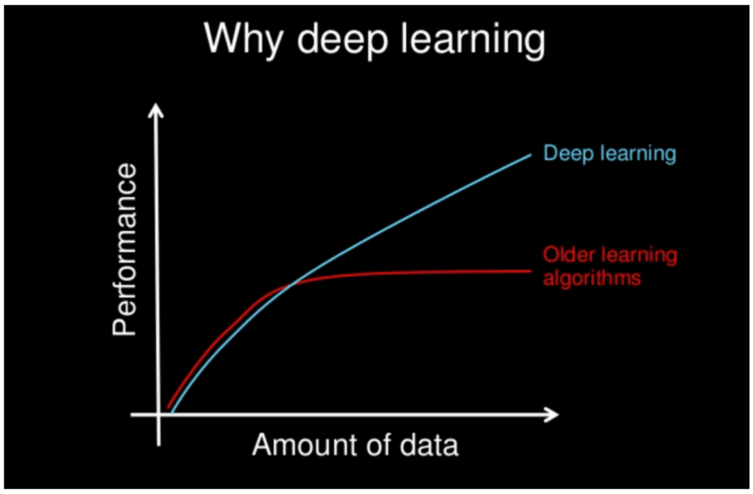


Fig 8: Why deep Learning is advantageous (Source: TowardsDataScience)

4.2 Using Convolution Neural Networks (CNN)

A Convolution Neural Network (CNN) is a multilayered neural network with a unique architecture for detecting complex data features. Image recognition, robot vision, and self-driving cars have all benefited from CNNs.

After constructing a CNN, it can be used to label the contents of various images. All we have to do now is feed the model those images. CNNs, like ANNs, was inspired by the human brain's operations. CNNs can distinguish images by detecting features, much as the human brain does to recognize objects by detecting features.

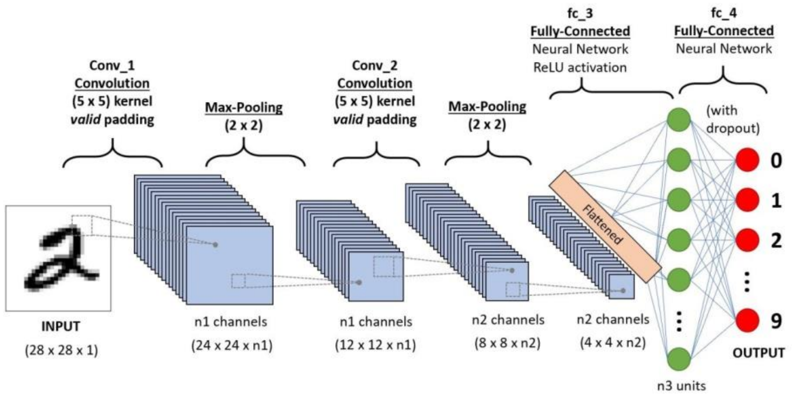

Conv_1 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
Conv_2 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
fc_3 Fully-Connected Neural Network ReLU activation
fc_4 Fully-Connected Neural Network
(with dropout)
INPUT (28 x 28 x 1)
n1 channels (24 x 24 x n1)
n1 channels (12 x 12 x n1)
n2 channels (8 x 8 x n2)
n2 channels (4 x 4 x n2)
Flattened
n3 units
0
1
2
9
OUTPUT

Fig 9: Convolution Neural Network (Source: Google Images)

Source Code Representation:

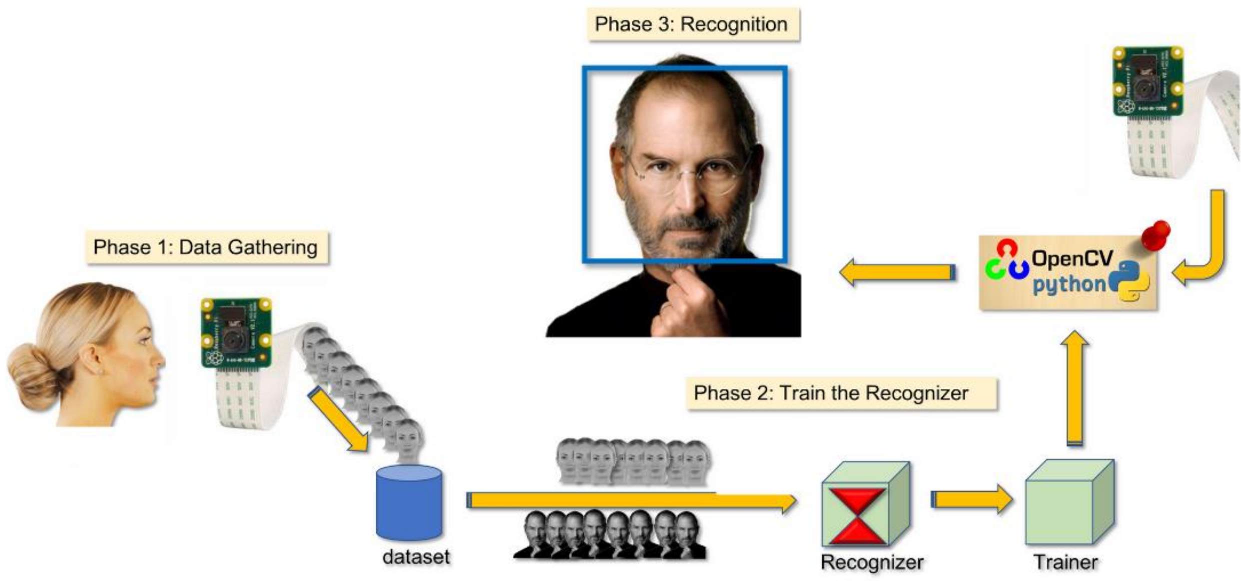


Fig 10: Diagramatical representation of source code (Source: Google Images)

Step-Wise Explanation:

Step 1: Importing Libraries
Python modules can get access to code from another module by importing the file/function using import. The import statement is the most common way of invoking the import machinery.

Matplotlib: It allows you to generate high quality line plots, scatter plots, histograms, bar charts, and much more. Each plot presents data in a different way and it is often useful to try out different types of plots before settling on the most informative plot for your data.

NumPy: NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilized to perform a number of mathematical operations on arrays.

Tensorflow: Tensorflow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. Tensorflow was originally developed for large numerical computations without keeping deep learning in mind.

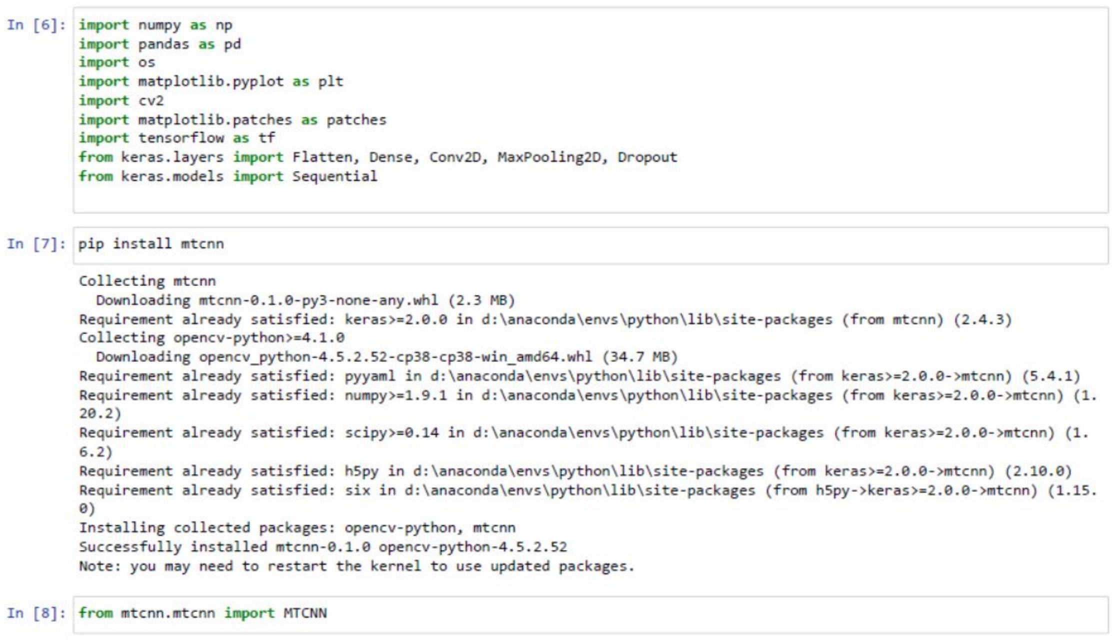
OS: The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

Cv2: OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imread() method loads an image from the specified file.

Pandas: Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

Keras: Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models.

MTCNN: MTCNN (Multi-task Cascaded Convolutional Neural Networks) is an algorithm consisting of 3 stages, which detects the bounding boxes of faces in an image along with their 5 Point Face Landmarks.

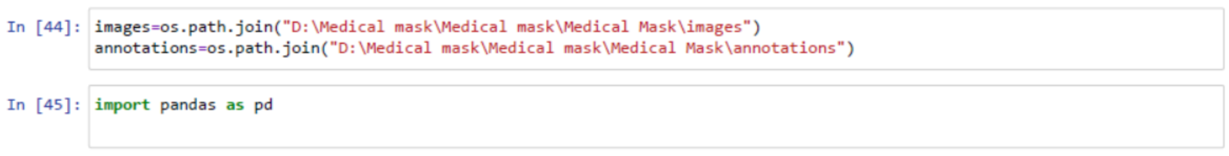


```
In [6]: import numpy as np
        import pandas as pd
        import os
        import matplotlib.pyplot as plt
        import cv2
        import matplotlib.patches as patches
        import tensorflow as tf
        from keras.layers import Flatten, Dense, Conv2D, MaxPooling2D, Dropout
        from keras.models import Sequential

In [7]: pip install mtcnn

        Collecting mtcnn
          Downloading mtcnn-0.1.0-py3-none-any.whl (2.3 MB)
        Requirement already satisfied: keras>=2.0.0 in d:\anaconda\envs\python\lib\site-packages (from mtcnn) (2.4.3)
        Collecting opencv-python>=4.1.0
          Downloading opencv_python-4.5.2.52-cp38-cp38-win_amd64.whl (34.7 MB)
        Requirement already satisfied: pyyaml in d:\anaconda\envs\python\lib\site-packages (from keras>=2.0.0->mtcnn) (5.4.1)
        Requirement already satisfied: numpy>=1.9.1 in d:\anaconda\envs\python\lib\site-packages (from keras>=2.0.0->mtcnn) (1.
        20.2)
        Requirement already satisfied: scipy>=0.14 in d:\anaconda\envs\python\lib\site-packages (from keras>=2.0.0->mtcnn) (1.
        6.2)
        Requirement already satisfied: h5py in d:\anaconda\envs\python\lib\site-packages (from keras>=2.0.0->mtcnn) (2.10.0)
        Requirement already satisfied: six in d:\anaconda\envs\python\lib\site-packages (from h5py->keras>=2.0.0->mtcnn) (1.15.
        0)
        Installing collected packages: opencv-python, mtcnn
        Successfully installed mtcnn-0.1.0 opencv-python-4.5.2.52
        Note: you may need to restart the kernel to use updated packages.

In [8]: from mtcnn.mtcnn import MTCNN
```

Fig 11: Snippet from source code of project

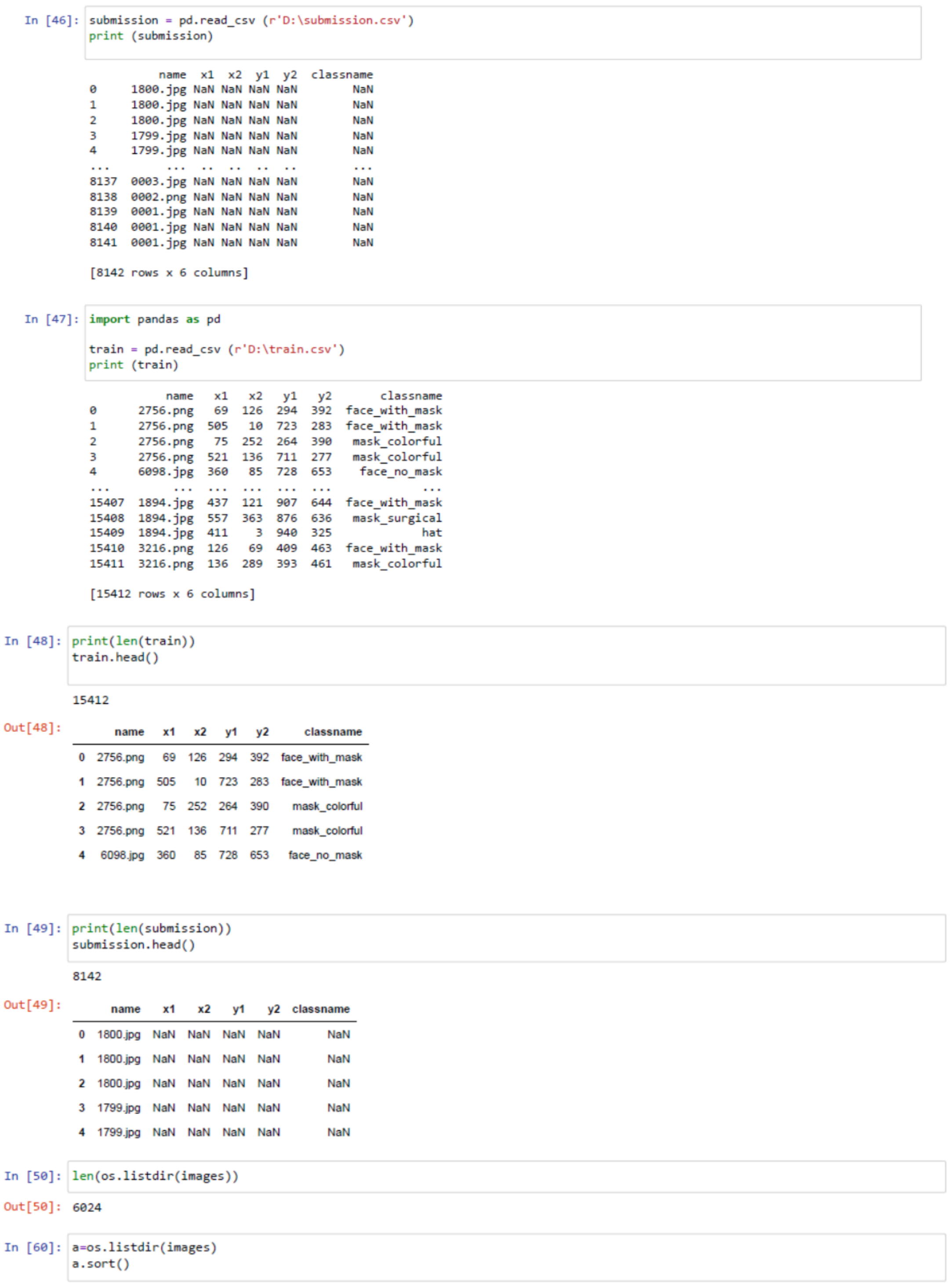
Step 2: Loading Dataset

The source folder is the input parameter containing the images. We are importing all the data in our project (source code).



```
In [44]: images=os.path.join("D:\Medical mask\Medical mask\Medical Mask\images")
         annotations=os.path.join("D:\Medical mask\Medical mask\Medical Mask\annotations")

In [45]: import pandas as pd
```

Fig 12: Snippet from source code of project



```
In [46]: submission = pd.read_csv (r'D:\submission.csv')
         print (submission)

              name  x1  x2  y1  y2  classname
0         1800.jpg NaN NaN NaN NaN        NaN
1         1800.jpg NaN NaN NaN NaN        NaN
2         1800.jpg NaN NaN NaN NaN        NaN
3         1799.jpg NaN NaN NaN NaN        NaN
4         1799.jpg NaN NaN NaN NaN        NaN
...            ...  ..  ..  ..  ..        ...
8137      0003.jpg NaN NaN NaN NaN        NaN
8138      0002.png NaN NaN NaN NaN        NaN
8139      0001.jpg NaN NaN NaN NaN        NaN
8140      0001.jpg NaN NaN NaN NaN        NaN
8141      0001.jpg NaN NaN NaN NaN        NaN

[8142 rows x 6 columns]
```

```
In [47]: import pandas as pd

         train = pd.read_csv (r'D:\train.csv')
         print (train)

              name   x1   x2   y1   y2       classname
0         2756.png   69  126  294  392  face_with_mask
1         2756.png  505   10  723  283  face_with_mask
2         2756.png   75  252  264  390   mask_colorful
3         2756.png  521  136  711  277   mask_colorful
4         6098.jpg  360   85  728  653    face_no_mask
...            ...  ...  ...  ...  ...             ...
15407     1894.jpg  437  121  907  644  face_with_mask
15408     1894.jpg  557  363  876  636   mask_surgical
15409     1894.jpg  411    3  940  325             hat
15410     3216.png  126   69  409  463  face_with_mask
15411     3216.png  136  289  393  461   mask_colorful

[15412 rows x 6 columns]
```

```
In [48]: print(len(train))
         train.head()

15412
```

Out[48]:

	name	x1	x2	y1	y2	classname
0	2756.png	69	126	294	392	face_with_mask
1	2756.png	505	10	723	283	face_with_mask
2	2756.png	75	252	264	390	mask_colorful
3	2756.png	521	136	711	277	mask_colorful
4	6098.jpg	360	85	728	653	face_no_mask

```
In [49]: print(len(submission))
         submission.head()

8142
```

Out[49]:

	name	x1	x2	y1	y2	classname
0	1800.jpg	NaN	NaN	NaN	NaN	NaN
1	1800.jpg	NaN	NaN	NaN	NaN	NaN
2	1800.jpg	NaN	NaN	NaN	NaN	NaN
3	1799.jpg	NaN	NaN	NaN	NaN	NaN
4	1799.jpg	NaN	NaN	NaN	NaN	NaN

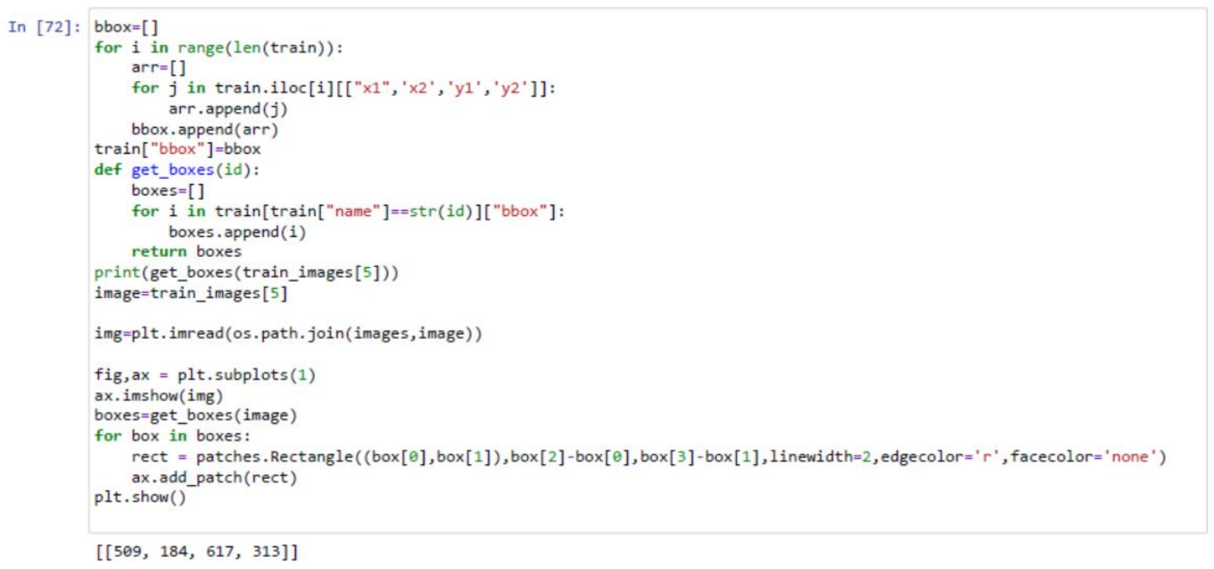
```
In [50]: len(os.listdir(images))

Out[50]: 6024

In [60]: a=os.listdir(images)
         a.sort()
```

Fig 13: Snippet from source code of project

Step 3: Split data into training and test set

A significant aspect of analyzing data mining models is to split data into training and research sets. Usually, much of the data is used for preparation as you split a data set into a training set and a testing set, and a smaller part of the data is used for testing. Research Systems sample the details arbitrarily to help ensure that the types of assessment and training are similar. You can mitigate the impact of data differences and better understand the model's features by using comparable data for preparation and research.

You validate the model by making assumptions against the test set after a model has been processed by using the training set. Since the data already includes known values for the attribute you want to forecast in the research range, it is easy to decide if the guesses of the model are accurate.

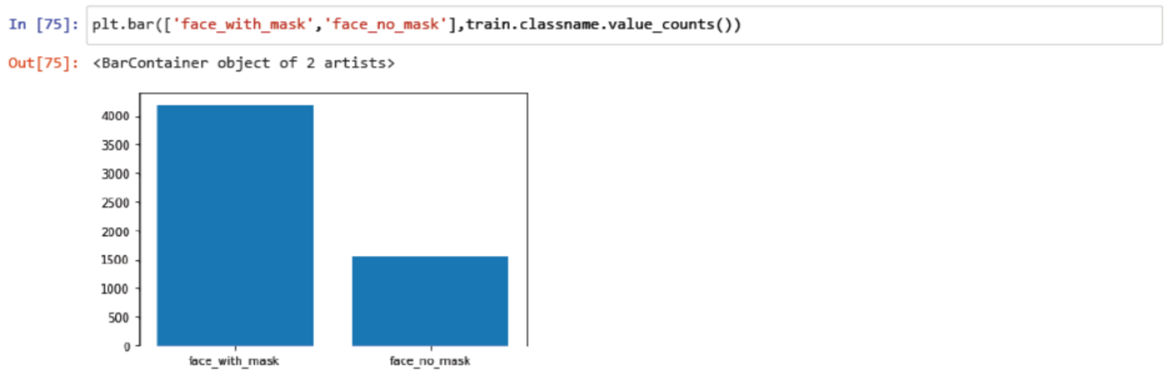


```
In [72]: bbox=[]
         for i in range(len(train)):
             arr=[]
             for j in train.iloc[i][["x1",'x2','y1','y2']]:
                 arr.append(j)
             bbox.append(arr)
         train["bbox"]=bbox
         def get_boxes(id):
             boxes=[]
             for i in train[train["name"]==str(id)]["bbox"]:
                 boxes.append(i)
             return boxes
         print(get_boxes(train_images[5]))
         image=train_images[5]

         img=plt.imread(os.path.join(images,image))

         fig,ax = plt.subplots(1)
         ax.imshow(img)
         boxes=get_boxes(image)
         for box in boxes:
             rect = patches.Rectangle((box[0],box[1]),box[2]-box[0],box[3]-box[1],linewidth=2,edgecolor='r',facecolor='none')
             ax.add_patch(rect)
         plt.show()

         [[509, 184, 617, 313]]
```

Fig 14: Snippet from source code of project

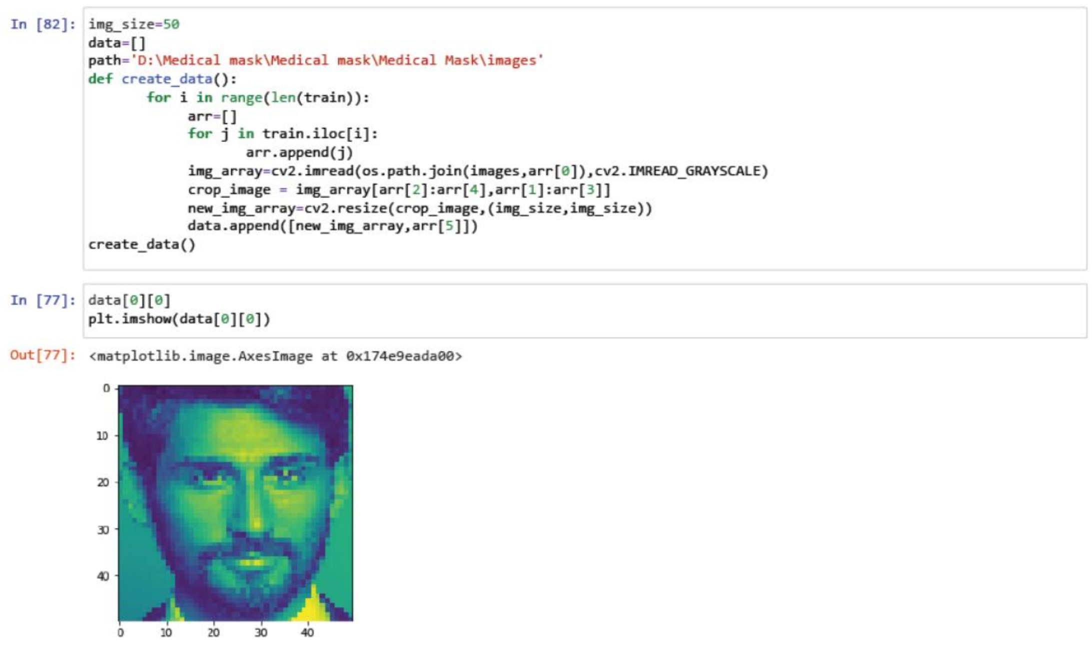
Step 4: Creating Training Dataset

The “training” data set is the general term for the samples used to create the model, while the “test” or “validation” data set is used to qualify performance. Traditionally the dataset used to evaluate the final model performance.



```
In [75]: plt.bar(['face_with_mask','face_no_mask'],train.classname.value_counts())

Out[75]: <BarContainer object of 2 artists>
```

Fig 15: Snippet from source code of project



```
In [82]: img_size=50
         data=[]
         path='D:\Medical mask\Medical mask\Medical Mask\images'
         def create_data():
                for i in range(len(train)):
                     arr=[]
                     for j in train.iloc[i]:
                           arr.append(j)
                     img_array=cv2.imread(os.path.join(images,arr[0]),cv2.IMREAD_GRAYSCALE)
                     crop_image = img_array[arr[2]:arr[4],arr[1]:arr[3]]
                     new_img_array=cv2.resize(crop_image,(img_size,img_size))
                     data.append([new_img_array,arr[5]])
         create_data()

In [77]: data[0][0]
         plt.imshow(data[0][0])

Out[77]: <matplotlib.image.AxesImage at 0x174e9eada00>
```

Fig 16: Snippet from source code of project

Step 5: Model Fitting

Convolution: A convolution is a two-function integration that reveals how one modifies the other. There are three important items to mention in this process: the input image, the feature detector, and the feature map. The input image is the image being detected. The feature detector is a matrix. The aim of this step is to reduce the size of the image and make processing faster and easier. Some of the features of the image are lost in this step. However, the main features of the image that are important in image detection are retained. These features are the ones that are unique to identifying that specific object.

ReLu (Rectified Linear Unit): The rectifier function is used to increase non-linearity in the CNN in this process. Different objects that are not linear to each other are used to create images. The image classification would be viewed as a linear problem if this function is not used, even though it is a non-linear problem.

Pooling: The principle of spatial invariance states that the position of an object in an image has no bearing on the neural network's ability to detect its unique features. Pooling allows the CNN to detect features in multiple images despite differences in lighting and camera angles. There are various forms of pooling, such as maximum pooling and minimum pooling. Placing a 2x2 matrix on the function map and selecting the largest value in that box is how max pooling works. The 2x2 matrix is passed through the entire function map from left to right, picking the largest value in each pass. These values are then combined to create a new matrix known as a pooled function map. Max pooling works to keep the image's key features while shrinking its scale. This helps to avoid over fitting, which occurs when the CNN is given too much data, particularly if the data is irrelevant to classifying the picture.

<u>Flattening:</u> After you've obtained the pooled featured map, you'll need to flatten it. The entire pooled feature map matrix is flattened into a single column, which is then fed to the neural network for processing.

<u>Full Connection:</u> The flattened feature map is then passed through a neural network after flattening. The input layer, the completely connected layer, and the output layer make up this stage. In ANNs, the completely connected layer is identical to the secret layer, but it is fully connected in this case. The predicted groups are found in the output layer. The data is sent through the network, and the prediction error is measured. To boost the prediction, the error is then back propagated through the system.
In most cases, the neural network's final numbers do not add up to one. It is important, however, to reduce these figures to numbers between zero and one, which reflect the likelihood of each class. This is where the Softmax feature comes in.

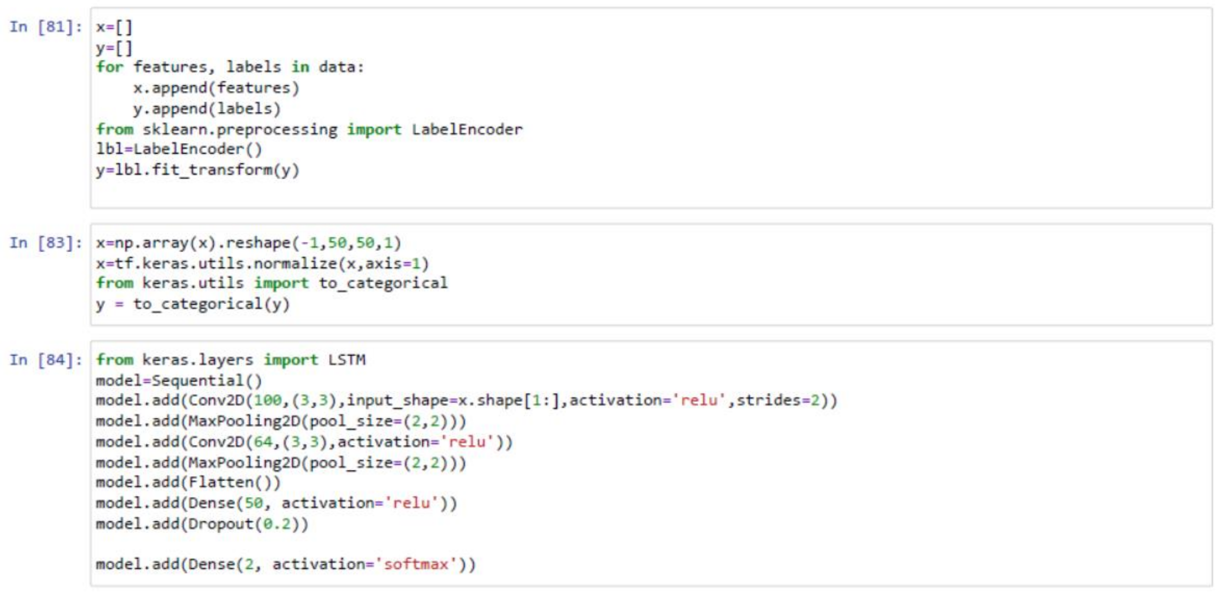
In this step we need to import Keras and other packages that we're going to use in building the CNN. Import the following packages:

- *Sequential* is used to initialize the neural network.
- *Convolution2D* is used to make the convolution network that deals with the images.
- *MaxPooling2D* layer is used to add the pooling layers.
- *Flatten* is the function that converts the pooled feature map to a single column that is passed to the fully connected layer.
- *Dense* adds the fully connected layer to the neural network.

<u>LSTM</u> has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).

<u>To_categorical</u> converts a class vector (integers) to binary class matrix.

<u>Normalizes</u> takes an array in as an input and normalizes its values between 0 and 1. It then returns an output array with the same dimensions as the input.



```
In [81]: x=[]
         y=[]
         for features, labels in data:
             x.append(features)
             y.append(labels)
         from sklearn.preprocessing import LabelEncoder
         lbl=LabelEncoder()
         y=lbl.fit_transform(y)

In [83]: x=np.array(x).reshape(-1,50,50,1)
         x=tf.keras.utils.normalize(x,axis=1)
         from keras.utils import to_categorical
         y = to_categorical(y)

In [84]: from keras.layers import LSTM
         model=Sequential()
         model.add(Conv2D(100,(3,3),input_shape=x.shape[1:],activation='relu',strides=2))
         model.add(MaxPooling2D(pool_size=(2,2)))
         model.add(Conv2D(64,(3,3),activation='relu'))
         model.add(MaxPooling2D(pool_size=(2,2)))
         model.add(Flatten())
         model.add(Dense(50, activation='relu'))
         model.add(Dropout(0.2))

         model.add(Dense(2, activation='softmax'))
```

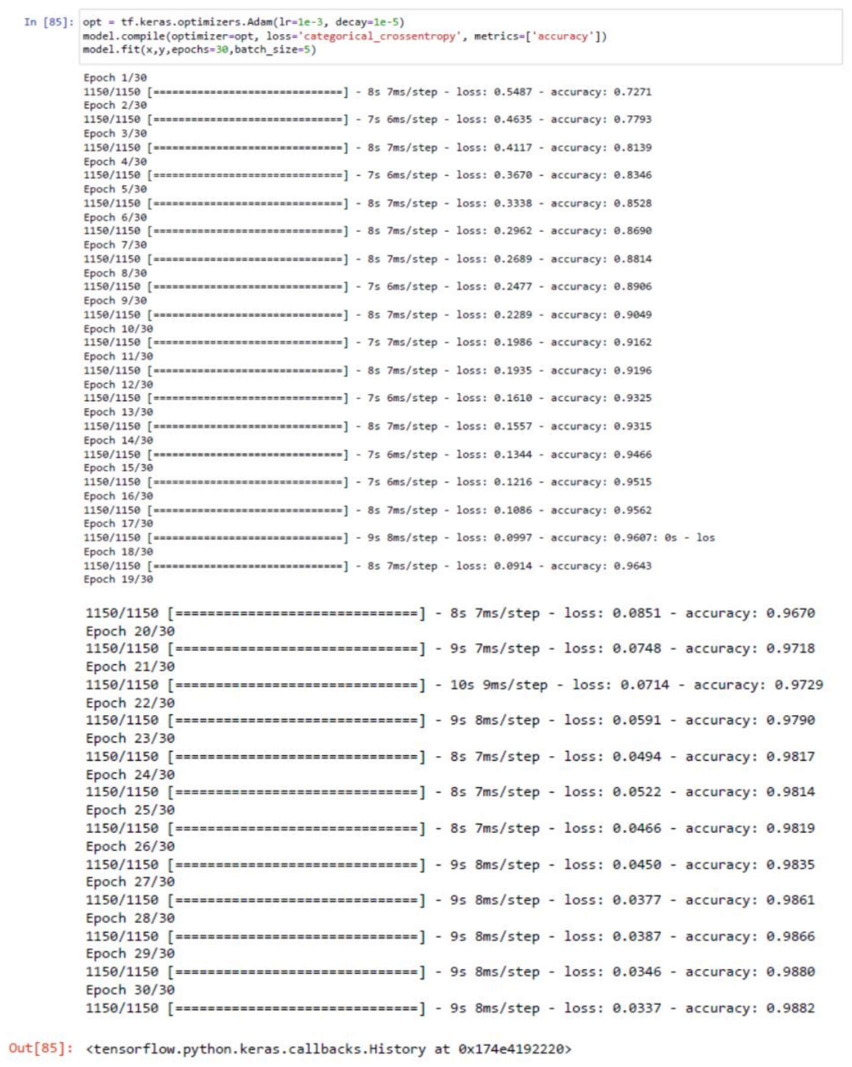
Fig 17: Snippet from source code of project

Adam: The algorithm leverages the power of adaptive learning rates methods to find individual learning rates for each parameter. It is a is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Metric: It is a function that is used to judge the performance of your model. Metric functions are similar to loss functions, except that the results from evaluating a metric are not used when training the model.

Categorical cross entropy: It is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.

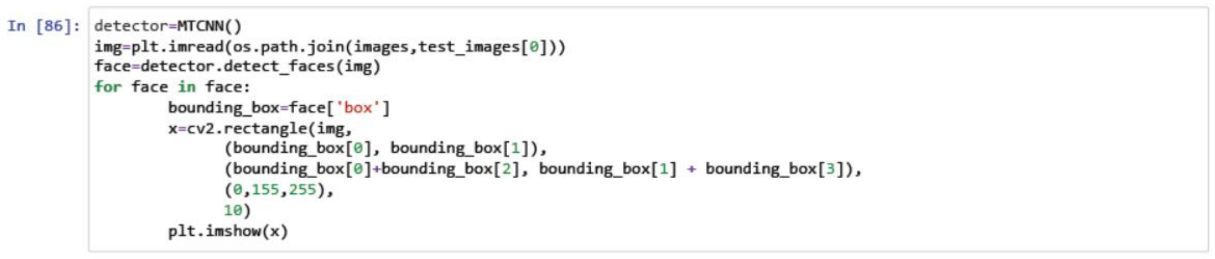
The batch size is a number of samples processed before the model is updated. The number of epochs is the number of complete passes through the training dataset. The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.



```
In [85]: opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)
         model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
         model.fit(x,y,epochs=30,batch_size=5)

         Epoch 1/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.5487 - accuracy: 0.7271
         Epoch 2/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.4635 - accuracy: 0.7793
         Epoch 3/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.4117 - accuracy: 0.8139
         Epoch 4/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.3670 - accuracy: 0.8346
         Epoch 5/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.3338 - accuracy: 0.8528
         Epoch 6/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.2962 - accuracy: 0.8690
         Epoch 7/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.2689 - accuracy: 0.8814
         Epoch 8/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.2477 - accuracy: 0.8906
         Epoch 9/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.2289 - accuracy: 0.9049
         Epoch 10/30
         1150/1150 [==============================] - 7s 7ms/step - loss: 0.1986 - accuracy: 0.9162
         Epoch 11/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.1935 - accuracy: 0.9196
         Epoch 12/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.1610 - accuracy: 0.9325
         Epoch 13/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.1557 - accuracy: 0.9315
         Epoch 14/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.1344 - accuracy: 0.9466
         Epoch 15/30
         1150/1150 [==============================] - 7s 6ms/step - loss: 0.1216 - accuracy: 0.9515
         Epoch 16/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.1086 - accuracy: 0.9562
         Epoch 17/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0997 - accuracy: 0.9607: 0s - los
         Epoch 18/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.0914 - accuracy: 0.9643
         Epoch 19/30

         1150/1150 [==============================] - 8s 7ms/step - loss: 0.0851 - accuracy: 0.9670
         Epoch 20/30
         1150/1150 [==============================] - 9s 7ms/step - loss: 0.0748 - accuracy: 0.9718
         Epoch 21/30
         1150/1150 [==============================] - 10s 9ms/step - loss: 0.0714 - accuracy: 0.9729
         Epoch 22/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0591 - accuracy: 0.9790
         Epoch 23/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.0494 - accuracy: 0.9817
         Epoch 24/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.0522 - accuracy: 0.9814
         Epoch 25/30
         1150/1150 [==============================] - 8s 7ms/step - loss: 0.0466 - accuracy: 0.9819
         Epoch 26/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0450 - accuracy: 0.9835
         Epoch 27/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0377 - accuracy: 0.9861
         Epoch 28/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0387 - accuracy: 0.9866
         Epoch 29/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0346 - accuracy: 0.9880
         Epoch 30/30
         1150/1150 [==============================] - 9s 8ms/step - loss: 0.0337 - accuracy: 0.9882

Out[85]: <tensorflow.python.keras.callbacks.History at 0x174e4192220>
```

Fig 18: Snippet from source code of project

Step 6: Recognizing Face in the preloaded images



```
In [86]: detector=MTCNN()
         img=plt.imread(os.path.join(images,test_images[0]))
         face=detector.detect_faces(img)
         for face in face:
                 bounding_box=face['box']
                 x=cv2.rectangle(img,
                         (bounding_box[0], bounding_box[1]),
                         (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
                         (0,155,255),
                         10)
                 plt.imshow(x)
```

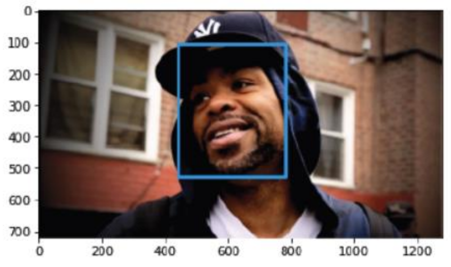
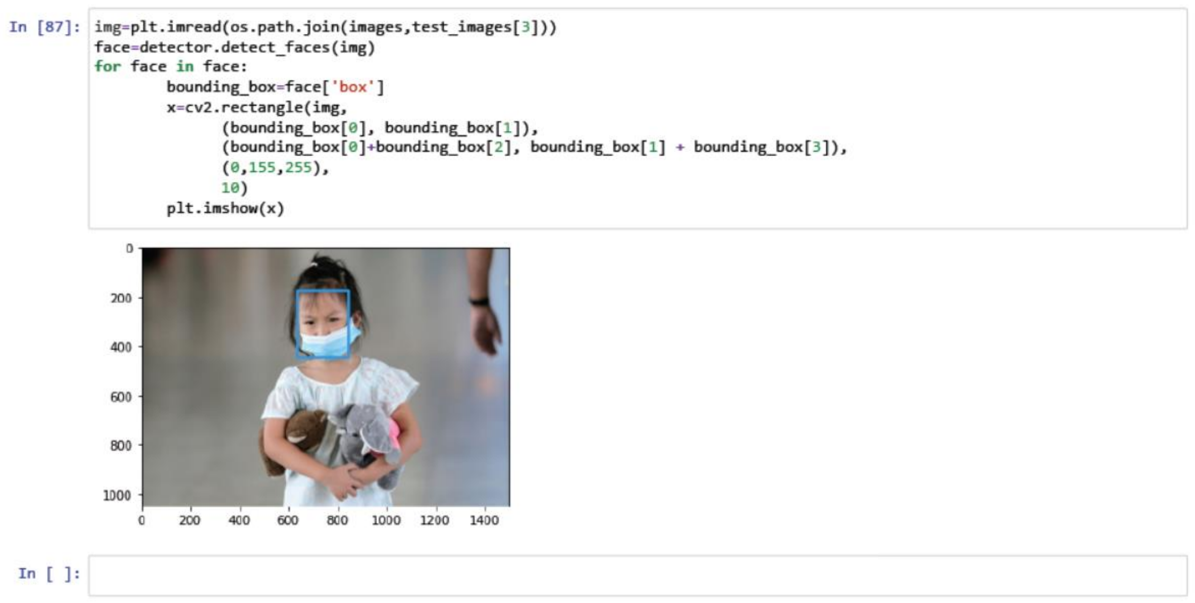


Fig 19: Snippet from source code of project (1st output)



```
In [87]: img=plt.imread(os.path.join(images,test_images[3]))
         face=detector.detect_faces(img)
         for face in face:
                 bounding_box=face['box']
                 x=cv2.rectangle(img,
                         (bounding_box[0], bounding_box[1]),
                         (bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3]),
                         (0,155,255),
                         10)
                 plt.imshow(x)
```

```
In [ ]:
```

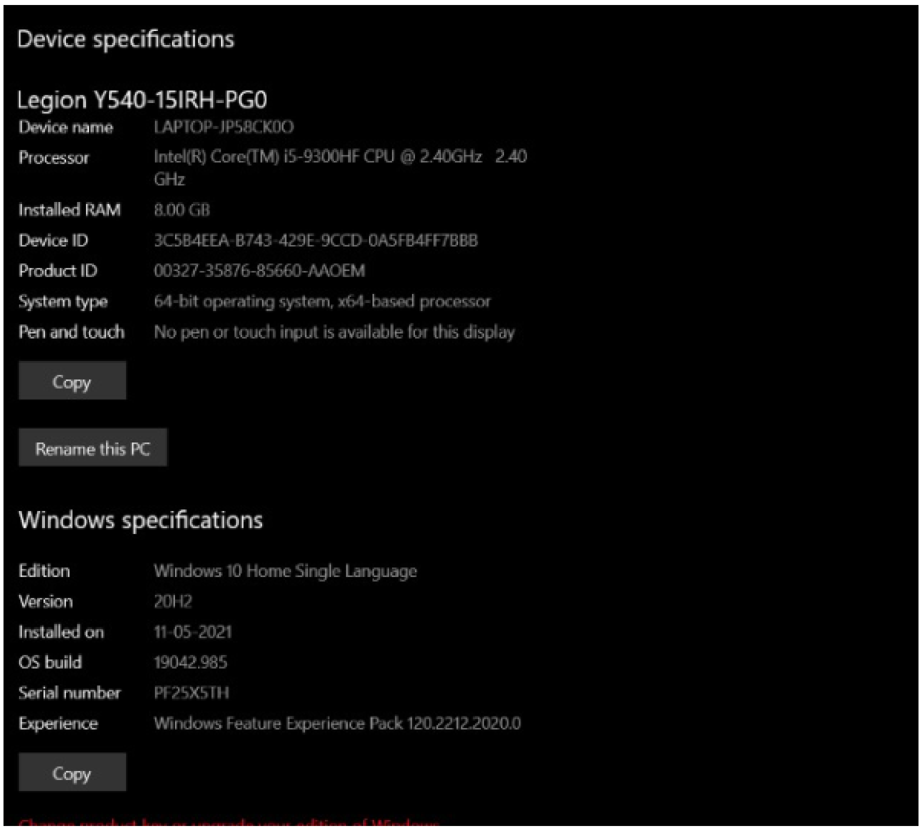
Fig 20: Snippet from source code of project (2nd output)

5. EXPERIMENTAL SETUP AND RESULTS ANALYSIS

5.1 Experimental Setup

Software: Python (version 3.7) distribution Anaconda (version 3) [Jupyter Notebook]

Hardware:



Device specifications

Legion Y540-15IRH-PG0

Device name	LAPTOP-JP58CK0O
Processor	Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz 2.40 GHz
Installed RAM	8.00 GB
Device ID	3C5B4EEA-B743-429E-9CCD-0A5FB4FF7BBB
Product ID	00327-35876-85660-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows 10 Home Single Language
Version	20H2
Installed on	11-05-2021
OS build	19042.985
Serial number	PF25X5TH
Experience	Windows Feature Experience Pack 120.2212.2020.0

Fig 21: Hardware features (where source code has been implemented)

5.2 Result Analysis

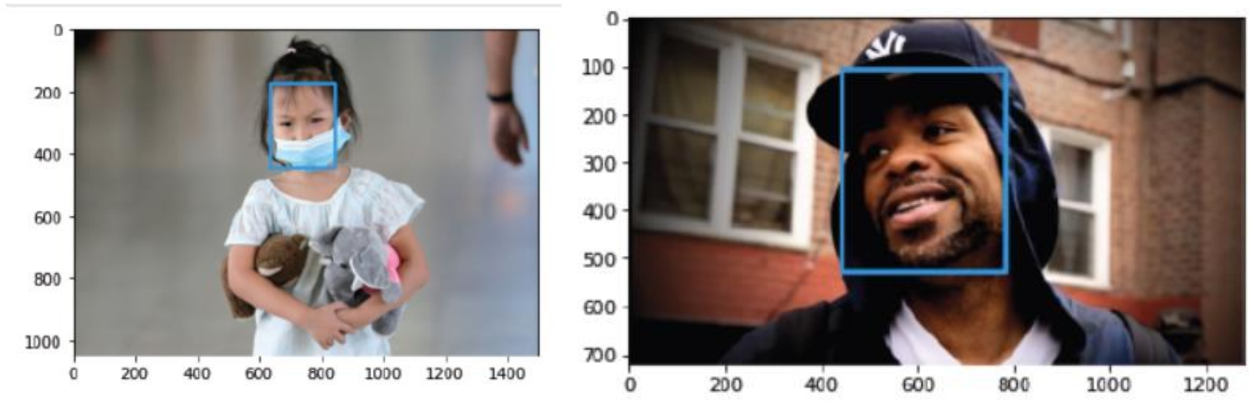


Fig 22: Snippet from source code of project (outcome/result of project)

Face detection is the crux of the face recognition issue. This is a fact that new researchers in this field find perplexing. Face recognition, on the other hand, requires the ability to accurately locate a face and its landmarks. This is essentially a segmentation problem, and in functional systems, this is where the majority of the effort is expended. In reality, recognizing people using features derived from these facial landmarks is only the final step.

6. CONCLUSION AND FUTURE SCOPE

a. Conclusion

In the last two decades, face recognition technology has advanced significantly. Machines can now check identity information for safe purchases, surveillance and security activities, and building access control, among other things. These applications are typically used in structured environments, and recognition algorithms may take advantage of these constraints to achieve high recognition precision. Next-generation face recognition technologies, on the other hand, can see widespread use in smart settings, where computers and machines serve as helpful assistants.

b. Future Scope

- Using webcam we can recognize face and store data.
- In videos we can detect the facial area.

- Since, COVID-19 prevails till date we also have a vision to enable facial recognition even if a person is wearing mask.
- Also, we should be able to distinguish whether a person is wearing a mask or not.

BIBLIOGRAPHY

1. https://www.sciencedirect.com/topics/computer-science/face-detection
2. https://www.forbes.com/sites/bernardmarr/2019/08/19/facial-recognition-technology-here-are-the-important-pros-and-cons/?sh=422344f814d1
3. https://publications.waset.org/7912/face-recognition-a-literature-review
4. https://dl.acm.org/doi/10.1145/954339.954342
5. https://www.manning.com/books/deep-learning-with-python
6. https://www.tutorialspoint.com/python_deep_learning/index.htm
7. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
8. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8
9. https://wiki.pathmind.com/convolutional-network
10. https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/
11. https://theaisummer.com/cnn-architectures/
12. https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist
13. https://www.kaggle.com/omkargurav/face-mask-dataset (dataset)
14. https://www.eff.org/pages/face-recognition
15. https://towardsdatascience.com/
16. https://morioh.com/explore?