ABSTRACT

The recent recommendations related to prevention against the spread of COVID-19 include wearing a face mask as an added measure of protection hence, the main focus of this project is to detect whether a person is wearing a mask or not, without getting close to them. With the computer vision techniques and image processing algorithms of face recognition, this can be made possible. Major applications on exercising this idea can be seen in any populated areas (like malls, Markets, Grocery stores, etc.), residential districts, large-scale manufacturers, and other enterprises where mass gatherings are expected.

Implementing a Python script to train a face mask detector/classifier with the dataset using Keras (deep learning framework) and TensorFlow (software library) and serializing it to the disk completes the training process. Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as ‘with mask’ or ‘without a mask’ and this is how the deployment is done.

TABLE OF CONTENTS

CHAPTER 1:

CHAPTER 2:

LIST OF FIGURES:

FACE MASK DETECTION USING IMAGE PROCESSING

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence (AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyse those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques. The process flow depicted here represents how machine learning works

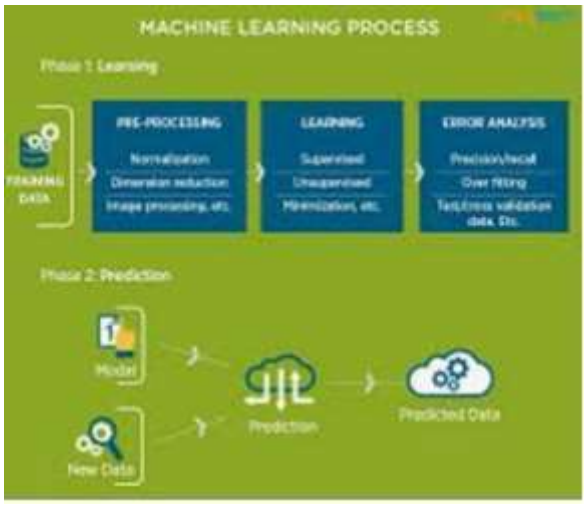


Figure 1: The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyse huge volumes of data Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analysing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data.They input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning:

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called
binary classification, such as determining whether or not someone will default on a loan.
Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

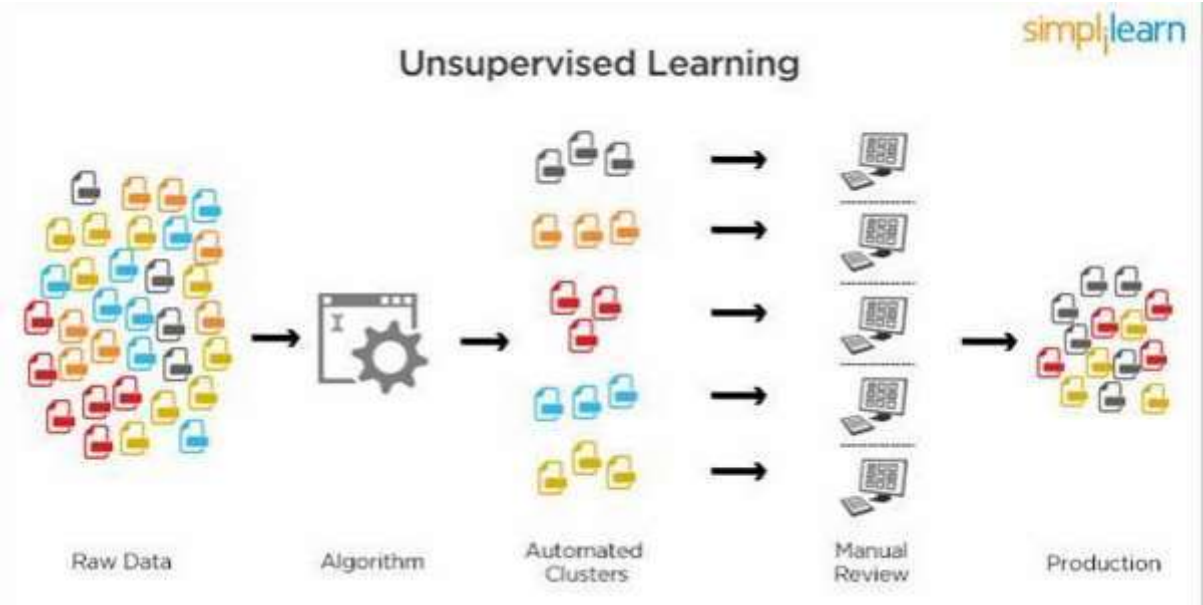


Figure 2: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text Topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labelled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labelled data with a large amount of unlabeled data.

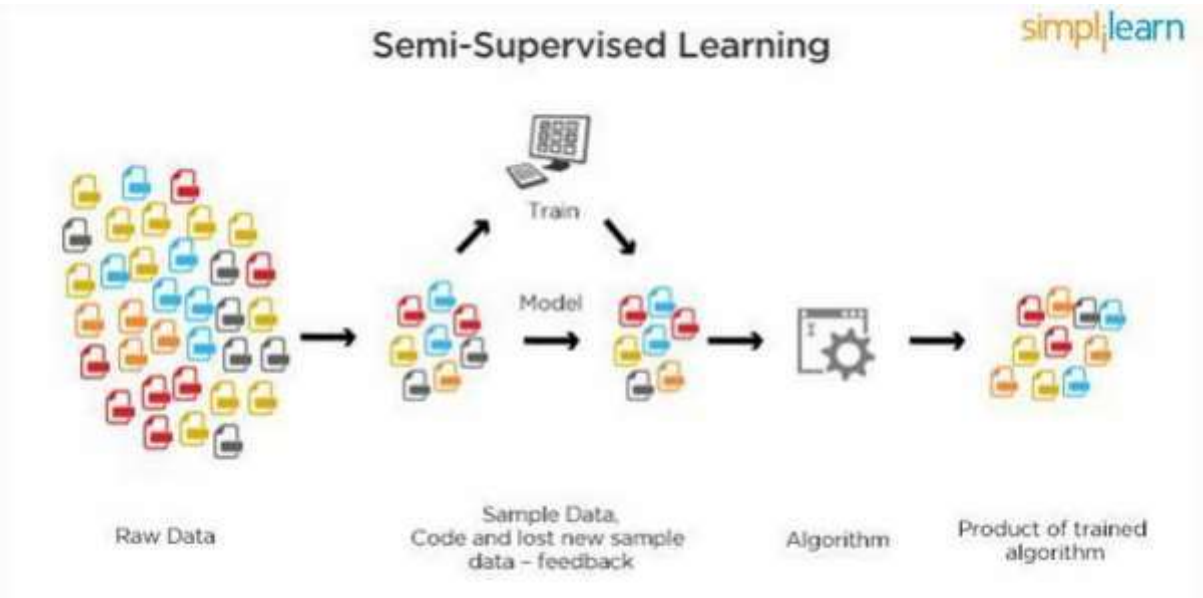


Figure 3: Semi Supervised Learning

1.5 DEEP LEARNING:

Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network. At a very basic level, deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information.

1.5.1 INTRODUCTION OF DEEP LEARNING:

Imagine you work for a loan company, and you need to build a model for predicting, whether a user (borrower) should get a loan or not? You have the features for each customer like age, bank balance, salary per annum, whether retired or not and so on.

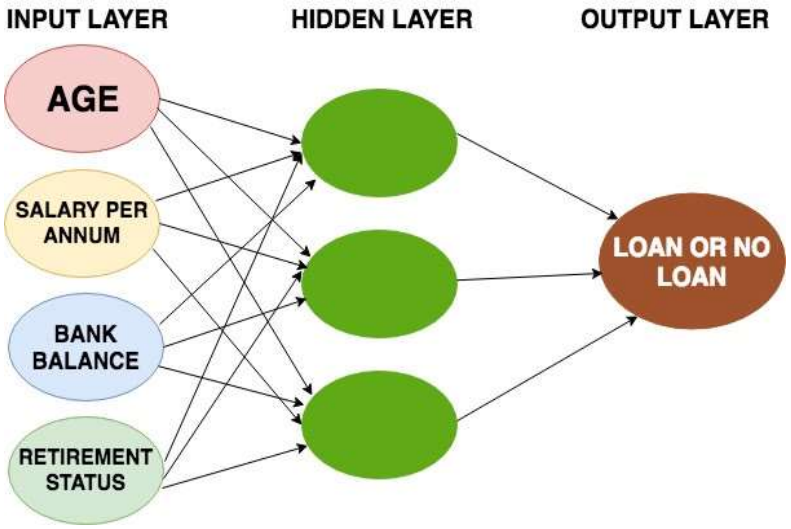


Figure 4: Interactions in neural networks.

The layers apart from the input and the output layers are called the hidden layers. Technically, each node in the hidden layer represents an aggregation of information from the input data; hence each node adds to the model's capability to capture interactions between the data. The more the nodes, the more interactions can be achieved from the data.

1.5.2 FORWARD PROPAGATION:

To understand the concept of forward propagation let's revisit the example of a loan company. For simplification, let's consider only two features as an input namely age and retirement status, the retirement status being a binary (0 - not retired and 1 - retired) number based on which you will make predictions.

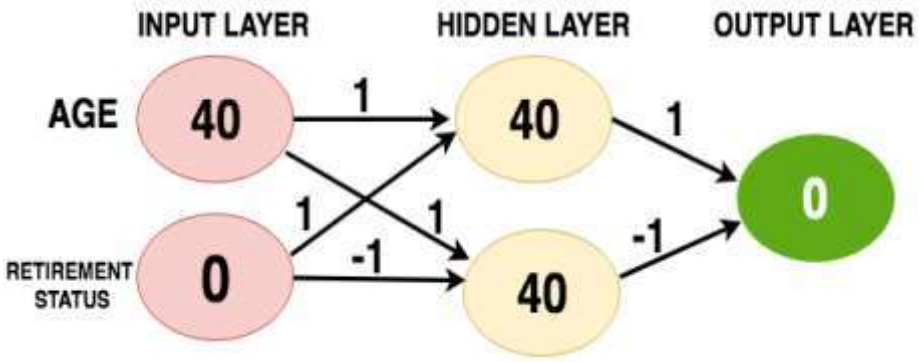


Figure 5: Layers

The above figure shows a customer with age 40 and is not retired. The forward propagation algorithm will pass this information through the network/model to predict the output layer. The lines connect each node of the input to every other node of the

hidden layer. Each line has a weight associated with it which indicates how strongly that feature affects the hidden node connected to that specific line.

There are a total four weights between input and hidden layer. The first set of weights are connected from the top node of the input layer to the first and second node of the hidden layer; likewise, the second set of weight are connected from the bottom node of the input to the first and second node of the hidden layer.

Remember these weights are the key in deep learning which you train or update when you fit a neural network to the data. These weights are commonly known as parameters.

To make a prediction for the top node of the hidden layer, you consider each node in the input layer multiply it by the weights connected to that top node and finally sum up all the values resulting in a value 40 (40 * 1 + 0 * 1 = 40) as shown in the above figure. You repeat the same process for the bottom node of the hidden layer resulting in a value 40. Finally, for the output layer you follow the same process and obtain a value 0 (40 * 1 + 40 * (-1) = 0). This output layer predicts a value zero. That's pretty much what happens in forward propagation. You start from the
input layer move to the hidden layer and then to the output layer which then gives you a prediction score. You pretty much always use the multiple-add process, in linear algebra this operation is a dot product operation. In general, a forward propagation is done for a single data point at a time.

1.5.3 ACTIVATION FUNCTIONS:

The multiply-add process is only half part of how a neural network works. To utilize the maximum predictive power, a neural network uses an activation function in the hidden layers. An activation function allows the neural network to capture non linearities present in the data.

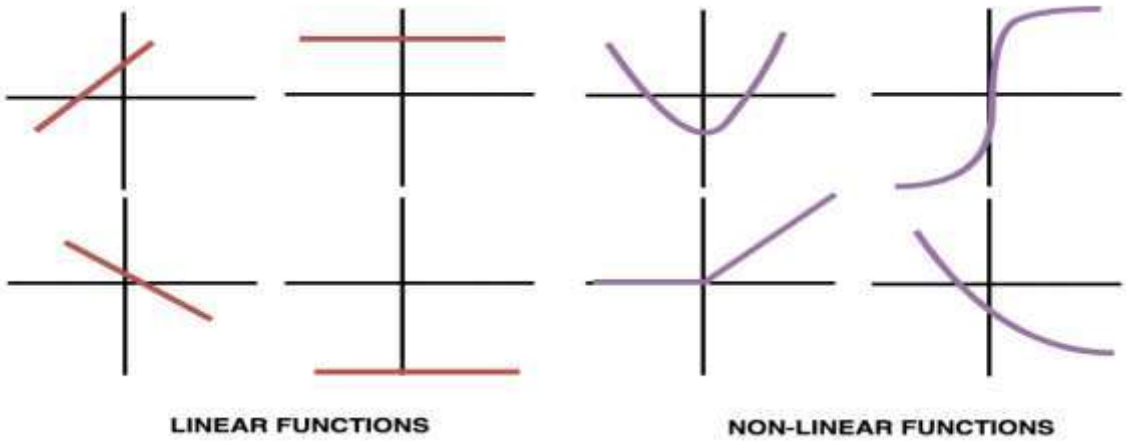


Figure 6: Functions

In neural networks, often time the data that you work with is not linearly separable and to find a decision boundary that can separate the data points you need some non-linearity in your network. For example, A customer has no previous loan record compared to a customer having a previous loan record may impact the overall output differently.

If the relationships in the data aren't straight or linear, then you need a non-linear activation function to capture the non-linearity. An activation function is applied to the value coming into a node which then transforms it into the value stored in that node or the node output.

1.5.4 DEEPER NETWORKS:

The significant difference between traditional neural networks and the modern deep learning that makes use of neural networks is the use of not just one but many successive hidden layers. Research shows that increasing the number of hidden layers massively improves the performance making the network capable of more and more interactions.

The working in a network with just a single hidden layer and with multiple hidden layers remain the same. You forward propagate through these successive hidden layers as you did in the previous example with one hidden layer.

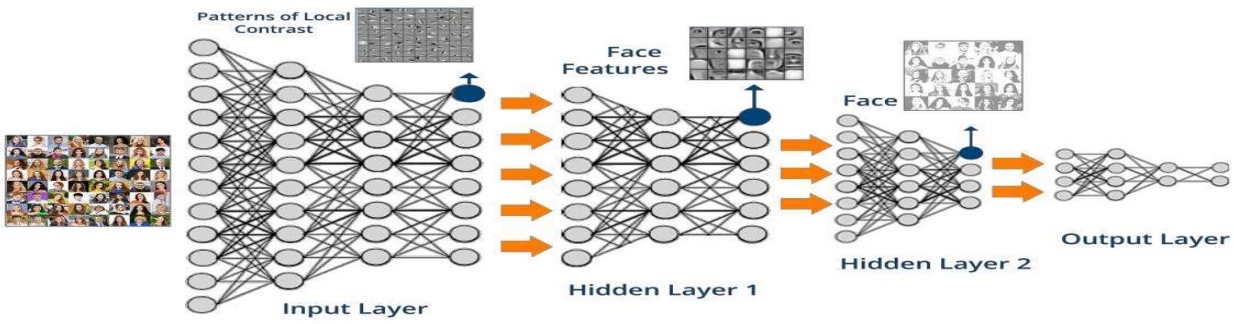


Figure 7: Hidden layers

- Deep Learning networks are capable of internally building up representations of the pattern in the data that are essential for making accurate predictions;

- The patterns in the initial layers are simple, but as you go through successive hidden layers or deep into the network the network starts learning more and more complex patterns;

- Deep learning networks eliminate the need for handcrafted features. You do not need to create better predictive features which you then feed to the deep learning network, the network itself learns meaningful features from the data and using which it makes predictions;

- Deep learning is also called Representation Learning since the subsequent layers in the network build increasingly sophisticated representations of the data until you reach to the final layer where it finally makes the prediction.

- When you train the network, the neural network has weights that are learned to find the relevant patterns to make accurate predictions. The learning process in neural networks off course is a gradual process in which the network undergoes multiple pieces of training before it can learn to make better predictions.

1.6 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning

CHAPTER 2

PYTHON

Basic programming language used for machine learning is: PYTHON

2.1 INTRODUCTION TO PYTHON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general-purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7, it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax,this allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to maintain.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- Databases: Python provides interfaces to all major commercial databases.

- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation (using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.

- Download python from www.python.org

- When the download is completed, double click the file and follow the instructions to install it.

- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

Figure 8: Python download 16

2.4.2 Installation (using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open-source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager that quickly installs and manages packages.
- In WINDOWS:
- In windows
- Step 1: Open Anaconda.com/downloads in a web browser.
- Step 2: Download python 3.4 version for (32-bits graphic installer/64 -bit graphic installer)
- Step 3: select installation type (all users)
- Step 4: Select path (i.e., add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

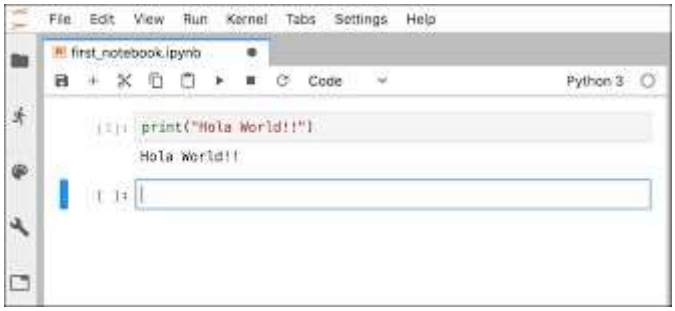


Figure 9: Jupyter notebook

2.4.3 Installation (using PYCHARM):

PYCHARM is a hybrid- platform developed by JetBrains as an IDE for Python.

1. It is commonly used for python application development.
2. It provides a wide range of essential tools for python developers, tightly integrated to create a convenient environment for productive python, web, and data science development.

Figure 10: PyCharm downloaded

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types
- numbers
- Lists
- Tuples
- Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.

- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator an the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data types.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ])and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no

append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]). 21

- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e. ()). Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments are the same as return None.

2.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOPs CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

- Data member: A class variable or instance variable that holds data associated with a class and its objects.

- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class

- Defining a Class:

    - We define a class in a very similar way how we define a function.

    - Just like a function, we use parentheses and a colon after the class name (i.e. () :) when we define a class. Similarly, the body of our class is indented like a functions body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 11: Defining a Class

2.7.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

- The init method has a special name that starts and ends with two underscores: __init__ ()

CHAPTER 3

LITERATURE SURVEY

- **R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCC), Jalandhar, India, 2018, pp. 278-282, doi: 10.1109/ICSCCC.2018.8703316.**

Deep Learning algorithms are designed in such a way that they mimic the function of the human cerebral cortex. These algorithms are representations of deep neural networks i.e. neural networks with many hidden layers. Convolutional neural networks are deep learning algorithms that can train large datasets with millions of parameters, in the form of 2D images as input and convolve it with filters to produce the desired outputs. In this article, CNN models are built to evaluate its performance on image recognition and detection datasets. The algorithm is implemented on MNIST and CIFAR-10 dataset and its performance is evaluated. The accuracy of models on MNIST is 99.6 %, CIFAR-10 is using real-time data augmentation and dropout on CPU units.

- **Pooja S., Preeti S. (2021) Face Mask Detection Using AI. In: Khosla P.K., Mittal M., Sharma D., Goyal L.M. (eds) Predictive and Preventive Measures for Covid-19 Pandemic. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-33-4236-1_16**

Covid-19 has built a completely new frequency, and the people have realized themselves moving into a new world. While currently our society speedily transforming, we need to be quick to be able to answer fresh specifications, which have encircled all of us. Making risk-free surroundings a priority of every person's mind so that life can be conductive just as before. Options need to be taken to secure all those going back to our workplace and to keep ourselves and our loved ones devoid of problems. Brand-new plans have methodized every day in order to meet policies and regulation. Although face masks are getting to be a whole new implemented standard for daily life, yet, to build a safe environment that contributes to public safety, it becomes necessary to be observant throughout the day and to take action against those who have not wearing masks in public places or workplaces. Many parts of society seem to be accepting some Covid tracking tools for safety. One of the most important tools is a face mask detector. This system enables one to identify who is without a required face mask. These systems work with existing surveillance systems along with innovative neural network algorithms to check whether a person has worn a face mask or not. In this chapter, we will briefly discuss artificial intelligence and its subsets namely machine learning and deep learning, deep learning frameworks followed by a simple implementation for face mask detection systems.

CHAPTER 4

CASE STUDY

4.1 PROBLEM STATEMENT:

Now-a-days due to COVID-19(Coronavirus) every person must wear a mask. Our goal is to train a custom deep learning model to detect whether a person is wearing a mask or is not wearing a mask.

4.2 DATASET

Data Set Link: https://github.com/prajnasb/observations

The given data set has the following Parameters

1) Observations

a) Experiments

i) Data

(1) With mask

(2) Without mask

ii) dest Folder

(1) Test

(a) With mask

(b) Without mask

(2) Train

(a) With mask

(b) Without mask

(3) Validation

(a) With mask

(b) Without mask

(c) Test.csv

(d) Train.csv

b) Mask classifier

c) Readme

Figure 12: Dataset

4.3 OBJECTIVE OF CASE STUDY

To get a better understanding and chalking out a plan of solution of the client, we have adapted the view point of looking at product categories and for further deep understanding of the problem, we have considered all the factors of training data, testing data and validation data, which has images of with mask and without mask. The main objective of this case study was to look up the factors of the data set and its classification and draft outcome report of mask classification.

CHAPTER 5

MODEL BUILDING

5.1 VISUALIZATION OF DATA:

5.1.1 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm. We are importing os for reading the directories and matplotlib for plotting the graphs and images.

OS MODULE: -

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or
Linux.

This module provides a portable way of using operating system dependent functionality. The *os* and *os. path* modules include many functions to interact with the file system.

MATPLOTLIB: -

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. ... SciPy makes use of Matplotlib.

Matplotlib. pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Figure 13: Importing libraries

5.1.2 LOADING THE DATA SET:

The data is loaded by git clone command which is inbuilt in google Collaboratory, so the data is imported from GitHub directly. It reads all the data and stores in the content directory, it loads all the images and files.

Git Python is a Python code library for programmatically reading from and writing to Git source control repositories and reading from a local cloned Git repository.

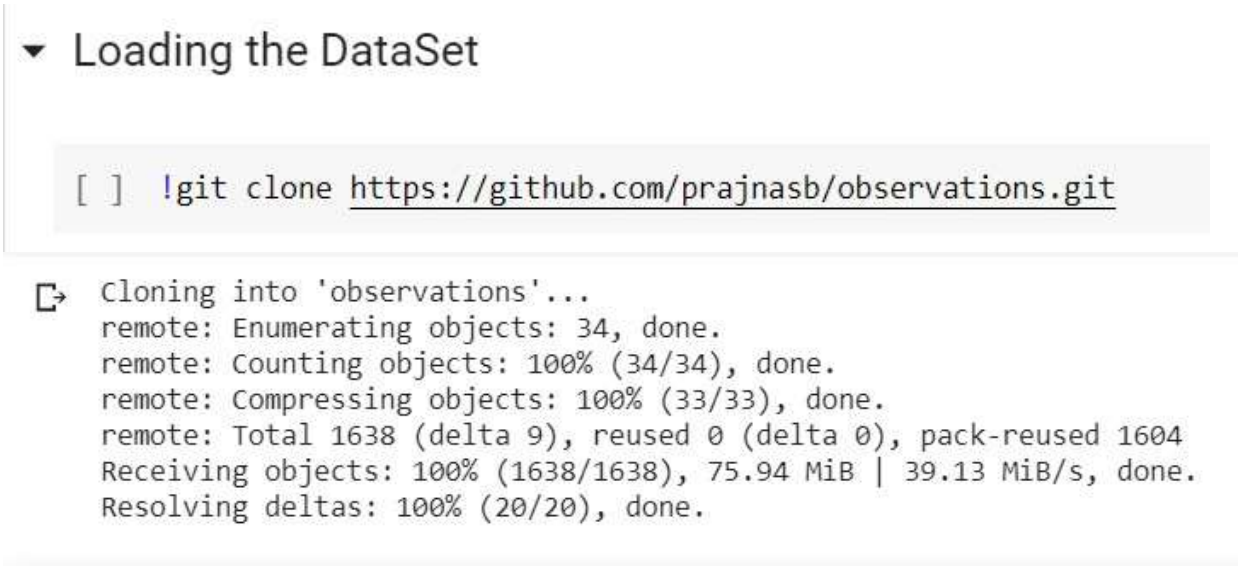


Figure 14: Loading Data set

5.1.3 READING DIRECTORIES:

The data is stored in the content directory so I need to read the directory.

OS LISTDIR: -

os.listdir() method in python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then list of files and directories in the current working directory will be returned. Syntax: os.listdir(path).

Python method listdir() returns a list containing the names of the entries in the directory given by path. The list is in arbitrary order. It does not include the special entries '. ' and '..' even if they are present in the directory.

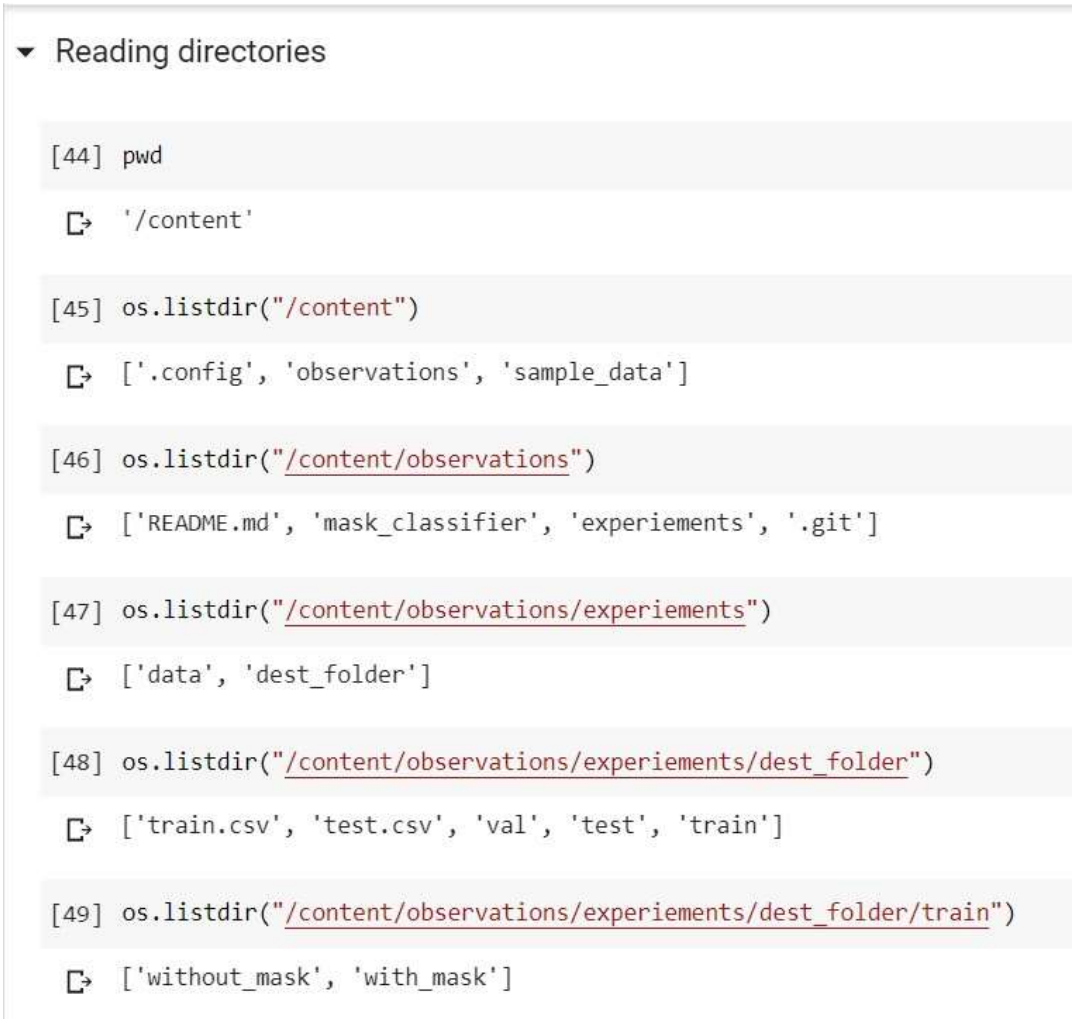


Figure 15: reading directories

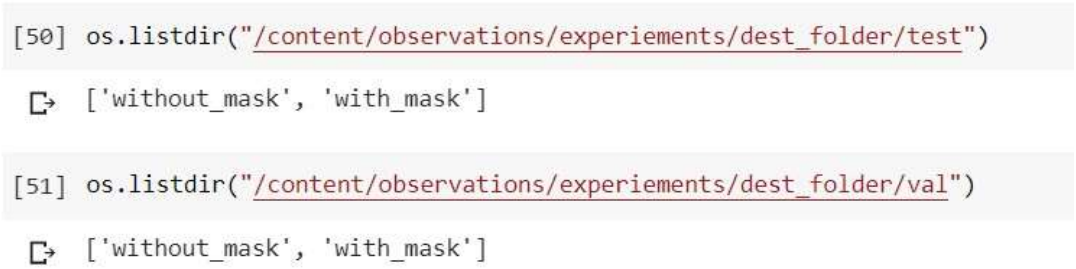


Figure 16: Reading Directories

Using the command os. listdir we are reading the directories, at first it in content directory, and there are 3 parts config, observations, sample data.

Our data set is named as Observations, so now we are reading the observations directory to know the folders and directories present in it.

On reading observation, we got mask_classifier, experiments, read me and git folders, then we are reading experiments directory, in that we found data and dest_folder.

In dest folder we have train, test, val folders/directories which has the images of with mask and without mask for training, testing and validating the data which are split from the original image data which is in the data directory.

5.1.4 LENGTH OF DATA:

No.of images used in the model , no.of images with mask, no.of images without mask.

LEN() FUNCTION :-

The len() function returns the number of items in an object. When the object is a string, the len() function returns the number of characters in the string.

len() is a built-in function in python. You can use the len() to get the length of the given string, array, list, tuple, dictionary, etc. Value: the given value you want the length of. Return value a return an integer value i.e. the length of the given string, or array, or list, or collections.

Figure 17: Length of Data

In data directory we have 2 folders with mask and without mask, using len and print function we are printing the no.of images of with mask and without mask folders which are 690 images of with mask and 686 images of without mask.

5.1.5: LOADING THE DATA FROM DIRECTORIES:

The dest folder has all the data of training, testing and validation in separate directories, so first we assign base_dir to dest folder using os.path.join which joins/assigns the dest folder path to base_dir.

Now creating train_dir and using base_dir as main path we are assigning the train folder to train_dir using os.path.join. In the same way we are assigning the test folder to test_dir and val folder to validation_dir.

Next we are assigning the with mask folder of train directory to train_with_mask_dir using os.path.join and train_dir as main directory, similarly we also assigned without a mask folder to train_without_mask_dir.

Next we are assigning the with mask folder of test directory to test_with_mask_dir using os.path.join and test_dir as main directory, similarly we also assigned without a mask folder to test_without_mask_dir.

Next we are assigning the with mask folder of validation directory to val_with_mask_dir using os.path.join and validation_dir as main directory, similarly we also assigned without a mask folder to val_without_mask_dir**.**

OS. PATH.JOIN() FUNCTION:

os. path. join () method in Python join one or more path components intelligently. This method concatenates various path components with exactly one directory separator ('/') following each non-empty part except the last path component.

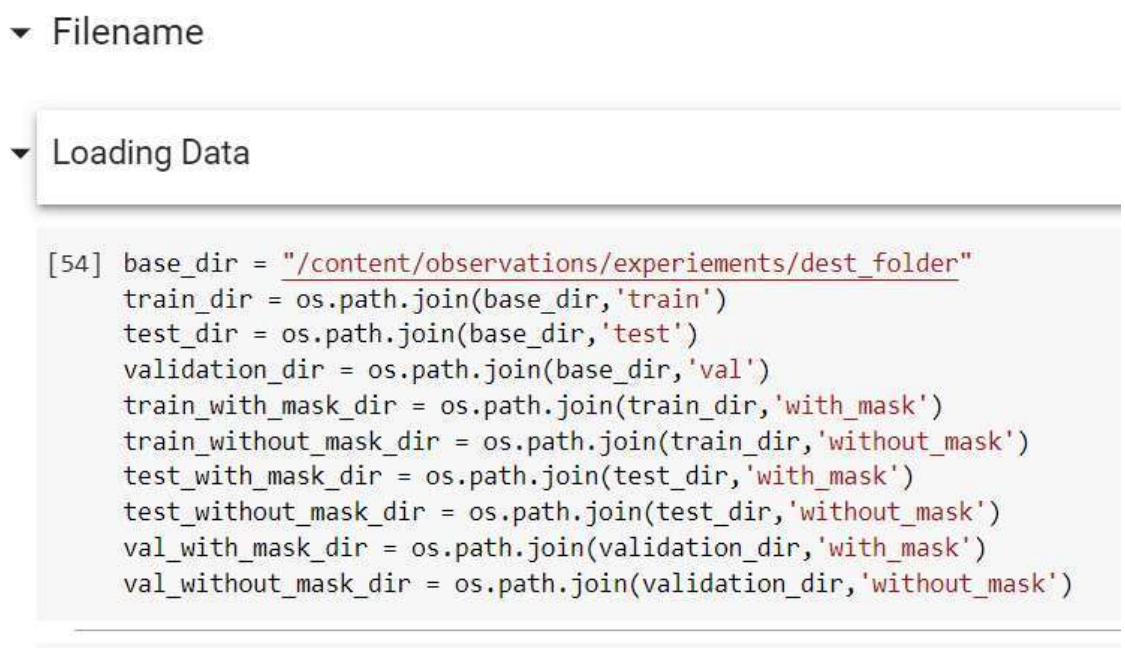


Figure 18: Loading Data

5.1.6 GIVING FILE NAMES:

```
5] #Filenames
   with_mask_filename = os.listdir(train_with_mask_dir)
   with_mask_filename[:4]

['416-with-mask.jpg',
 'augmented_image_90.jpg',
 '76-with-mask.jpg',
 '125-with-mask.jpg']

6] without_mask_filename = os.listdir(train_without_mask_dir)
   without_mask_filename[:10]

['370.jpg',
 'depositphotos-142113624-stock-photo-smiling-indian-man-face.jpg',
 'augmented_image_90.jpg',
 '398.jpg',
 'augmented_image_202.jpg',
 '52.jpg',
 '160.jpg',
 '87.jpg',
 '328.jpg',
 'augmented_image_156.jpg']
```

Figure 19: Giving File Names

In the with_mask_filename , we are reading the data from the directory train_with_mask_dir using the os.listdir() , this command reads all the data in the directory and assigns it and then printing the names of with mask images by calling with mask filename , [:4] is given for the count of names needed to print.

In the without_mask_filename, we are reading the data from the directory train_without_mask_dir using the os.listdir() , this command reads all the data in the directory and assigns it and then printing the names of with mask images by calling without mask filename , [:10] is given for the count of names needed to print.

5.1.7: DISPLAY THE IMAGES:

Displaying the single image of person wearing mask and a person not wearing mask. Here we imported matplotlib as plt. Displayed the image using plot and the imshow key word. From the train data which is train_with_mask_dir, images are read using an imread key word and displayed.

IMSHOW() FUNCTION :

The imshow() function in pyplot module of matplotlib library is used to display data as an image; i.e. on a 2D regular raster. ... X: This parameter is the data of the image. cmap: This parameter is a colormap instance or registered colormap name.

The matplotlib function imshow() creates an image from a 2-dimensional numpy array. The image will have one square for each element of the array. The color of each square is determined by the value of the corresponding array element and the color map used by imshow()

IMREAD() FUNCTION:

imread(filename) reads the image from the file specified by filename , inferring the format of the file from its contents. If filename is a multi-image file, then imread reads the first image in the file.

imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then This method returns an empty matrix.

Figure 20: Display image with mask.

▾ Display image without mask

```
[58] plt.imshow(plt.imread(train_without_mask_dir+'/435.jpg'))
```

```
<matplotlib.image.AxesImage at 0x7fa236df0828>
```

Figure 21: Display image without mask.

▾ Display of set of images with mask

```
[ ] plt.figure(figsize=(16,16))
    j = 1
    for i in range(10):
      img = plt.imread(os.path.join(train_with_mask_dir,with_mask_filename[i]))
      plt.subplot(4,5,j)
      plt.imshow(img)
      plt.title(img.shape)
      plt.axis('off')
      j += 1
```

Figure 22: Set of images with mask

▾ Display of set of images without mask

```
[ ]  plt.figure(figsize=(16,16))
     j = 1
     for i in range(8):
       img = plt.imread(os.path.join(train_without_mask_dir,without_mask_filename[i]))
       plt.subplot(4,4,j)
       plt.imshow(img)
       plt.title(img.shape)
       plt.axis('off')
       j += 1
```

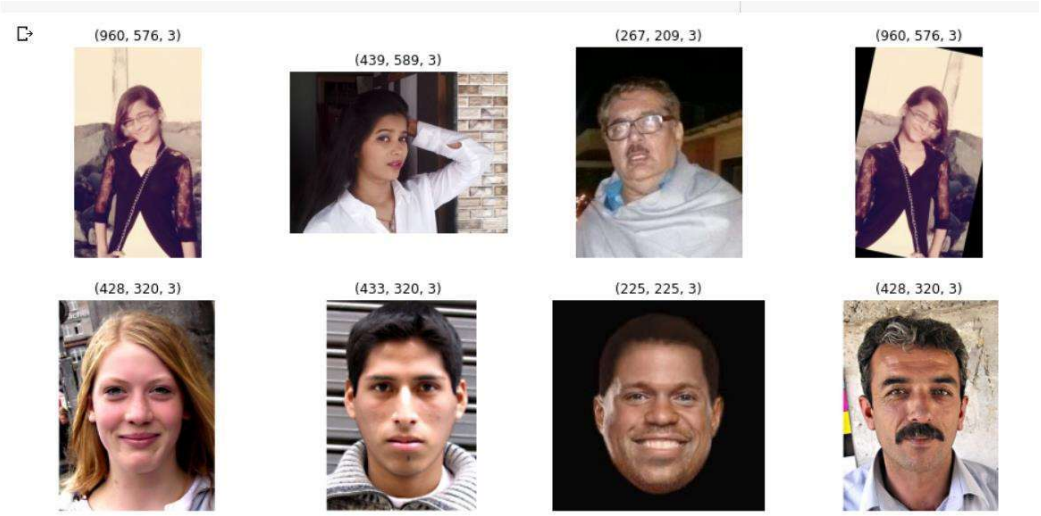


Figure 23: Set of images without mask

The figure size is given 16 . range is for the no .of images need to be printed . Using subplot we can print multiple images and no. of rows and columns given in brackets . Title is the shape of the image.

5.2 DATA PREPROCESSING:

5.2.1 Creating Train and validation data from Folder:

Creating Train and valiadtion data from Folder

```
[ ]  from tensorflow.keras.preprocessing.image import ImageDataGenerator

     # All images will be rescaled by 1./255
     train_datagen = ImageDataGenerator(rescale=1./255)
     val_datagen = ImageDataGenerator(rescale=1./255)

     # Flow training images in batches of 20 using train_datagen generator
     train_generator = train_datagen.flow_from_directory(
             train_dir,  # This is the source directory for training images
             target_size=(150, 150),  # All images will be resized to 150x150
             batch_size=20,
             # Since we use binary_crossentropy loss, we need binary labels
             class_mode='binary')

     # Flow validation images in batches of 20 using val_datagen generator
     validation_generator = val_datagen.flow_from_directory(
             validation_dir,
             target_size=(150, 150),
             batch_size=20,
             class_mode='binary')
```

```
Found 1315 images belonging to 2 classes.
Found 142 images belonging to 2 classes.
```

Figure 24: Train and validation data

We are importing the libraries required which are tensorflow, keras, and from then we are importing an image data generator.

Using Imagedatagenerator we are scaling the image using the rescale key word.

Since 255 is the maximum pixel value. Rescale 1./255 is to transform every pixel value from range [0,255] -> [0,1]. We are rescaling all the images of train and validation and storing them in train_datagen and val_datagen.

Now we are dividing the images into 20 batches and each batch size is 150x150 using the class mode as binary, the class mode is binary because we have 2 classes for our image data which are with mask and without mask

These divided batches of train data images will be stored in train_generator and validation data images in validation_generator.

Then we got output as 1315 images of 2 classes in train generator and 142 imagesof 2 classes in validation generator

TENSOR FLOW: -

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainlyused for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

KERAS: -

Keras is a neural network library while TensorFlow is the open-source library for a number of various tasks in machine learning. TensorFlow provides both high-level and low-level APIs while Keras provides only high-level APIs. ... Both frameworks thus provide high-level APIs for building and training models with ease.

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML.

Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

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 in just a few lines of code.

We are checking whether the train generator has got all the rescaled values of training data and using that we are displaying an image. We have assigned train generator to imgs and labels.

IMAGE DATA GENERATOR: -

ImageDataGenerator generates batches of image data with real-time data augmentation.
The most basic codes to create and configure ImageDataGenerator and train deep neural

network with augmented images. While the word “augment” means to make something “greater” or “increase” something (in this case, data), the Keras Image Data Generator class actually works by: Accepting a batch of images used for training

5.2.2 DISPLAY OF RANDOM IMAGES

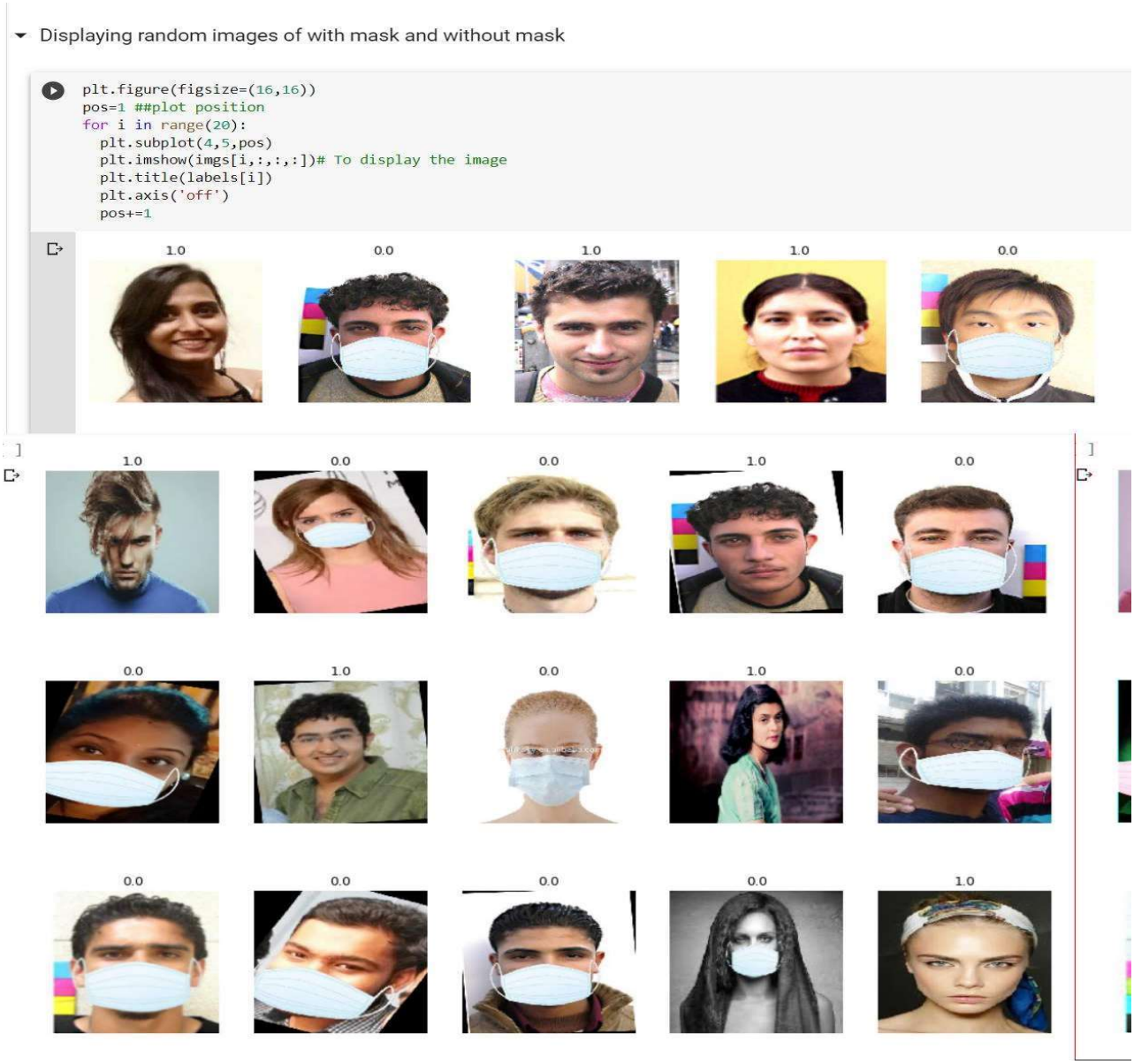


Figure 25: Random images

Printing random images of people with mask and without mask using the train generator which is assigned to images and plotted the images using subplot and imshow().

PLT.FIGURE():-

The purpose of using plt. figure() is to create a figure object. The whole figure is regarded as the figure object. It is necessary to explicitly use plt. figure() when we want to tweak the size of the figure and when we want to add multiple Axes objects in a single figure.

pyplot in matplotlib's plotting framework. That specific import line merely imports the module "matplotlib. pyplot" and binds that to the name "plt". There are many ways to import in Python, and the only difference is how these imports affect your namespace.

SUBPLOT :-

The matplotlib. pyplot. subplots method provides a way to plot multiple plots on a single figure

Creating multiple subplots using plt. subplot. subplots create a figure and a grid of subplots with a single call, while providing reasonable control over how the individual plots are created. For more advanced use cases you can use GridSpec for a more general subplot layout or Figure.

RANGE :-

The range() function is used to generate a sequence of numbers over time. At its simplest, it accepts an integer and returns a range object (a type of iterable).

TITLE:-

The title() method in matplotlib module is used to specify title of the visualization depicted and displays the title using various attributes. Syntax: matplotlib.pyplot.title(label)

5.2.3 HISTOGRAM OF DATA:

▾ Histogram of dataset

```
[ ]  import matplotlib.pyplot as plt
     imgs,labels = train_generator.next()
     plt.figure(figsize=(16,16))
     pos = 1  ## plot position
     for i in range(10):
       plt.subplot(5,2,pos)
       plt.hist(imgs[i,:,:,:].flat) # To display the histogram
       plt.title(labels[i])
       pos += 1
```

Figure 26: code of histogram

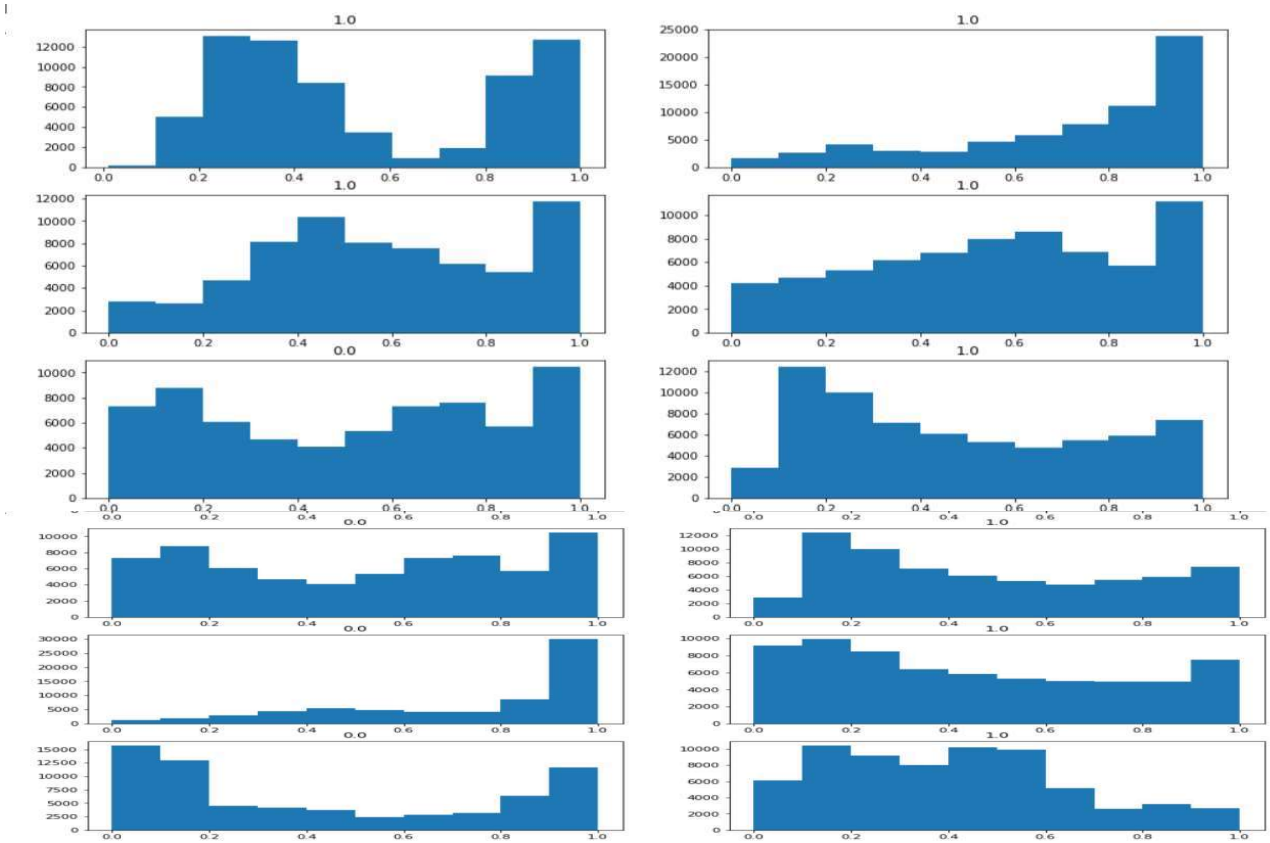


Figure 27: output of histogram

Using plt.hist command we have printed the histogram of different images of train data by using a train generator**.**

PLT.HIST():-

matplotlib. pyplot. hist () Function. The hist() function in pyplot module of matplotlib library is used to plot a histogram

5.3 BUILDING THE MODEL:

CONVOLUTIONAL NEURAL NETWORKS: -

Convolutional Neural Network (ConvNet or CNN) is a special type of Neural Network used effectively for image recognition and classification. They are highly proficient in areas like identification of objects, faces, and traffic signs apart from generating vision in self-driving cars and robots too.

THE CONVOLUTIONAL OPERATOR: -

ConvOp is the main workhorse for implementing a convolutional layer in Theano. ConvOp is used by theano.tensor.signal.conv2d, which takes two symbolic inputs:

• A 4D tensor corresponding to a mini-batch of input images. The shape of the tensor is as follows: [mini-batch size, number of input feature maps, image height, image width].

• A 4D tensor corresponding to the weight matrix *W*. The shape of the tensor is: [number of feature maps at layer m, number of feature maps at layer m- 1, filter height, filter width]

Below is the Theano code for implementing a convolutional layer similar to the one of Figure 1. The input consists of 3 features maps (an RGB color image) of size 120x160. We use two convolutional filters with 9x9 receptive fields.

MAX POOLING: -

Another important concept of CNNs is *max-pooling,* which is a form of non-linear down sampling. Max Pooling partitions the input image into a set of non-overlapping rectangles and, for each such subregion, outputs the maximum value. Max-pooling is useful in vision for two reasons:

1. By eliminating non-maximal values, it reduces computation for upper layers.

2. It provides a form of translation invariance. Imagine cascading a max pooling layer with a convolutional layer. There are 8 directions in which one can translate the input image by a single pixel. If max pooling is done over a 2x2 region, 3 out of these 8 possible configurations will produce exactly the same output at the convolutional layer. For max-pooling over a 3x3 window, this jumps to 5/8. Since it provides additional robustness to position, max-pooling is a "smart" way of reducing the dimensionality of intermediate representations.

Max-pooling is done in Theano by way of theano.tensor.signal.downsample.max_pool_2d. This function takes as input an N dimensional tensor (where N >= 2) and a downscaling factor and performs max-pooling over the 2 trailing dimensions of the tensor.

SEQUENTIAL: -

The Sequential model is a linear stack of layers. The common architecture of ConvNets is a sequential architecture. However, some architectures are not linear stacks.

The sequential API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs.

DENSE: -

Dense layer is the regular deeply connected neural network layer.

It is the most common and frequently used layer. Dense layer does the below operation on the input and return the output.

Each neuron in a layer receives an input from all the neurons present in the previous layer—thus, they're densely connected. In other words, the dense layer is a fully connected layer, meaning all the neurons in a layer are connected to those in the next layer.

FLATTEN: -

The flatten () function is used to get a copy of a given array collapsed into one dimension. Flattening the data in a database means that you store it in one or a few tables containing all the information, with little enforcement of structure. In database lingo, that's called a denormalized schema.

5.3.1 IMPORTING REQUIRED LIBRARIES:

Figure 28: Importing required Methods

The methods Sequential, Conv2D, Dense, Flatten, MaxPooling2D are imported from TensorFlow and keras models.

5.3.2 MODEL:

```
model = Sequential()
## add a conv layer folloed by maxpooling
model.add(Conv2D(16,3,activation='relu',input_shape=(150,150,3)))
model.add(MaxPooling2D(2))
## add a conv layer folloed by maxpooling
model.add(Conv2D(32,3,activation='relu'))
model.add(MaxPooling2D(2))
## add a conv layer folloed by maxpooling
model.add(Conv2D(64,3,activation='relu'))
model.add(MaxPooling2D(2))
# Convert the faeturemap into 1D array
model.add(Flatten())
# Fully connected layer with 512 neurons
model.add(Dense(512,activation='relu'))
## Final output layer
model.add(Dense(1,activation='sigmoid'))

## let us see the the summary
model.summary()
```

Figure 29: Code for Model.

We are invoking the sequential function into model and using the function we are adding the convo layers by max pooling.

We are giving different sizes for different layers of convo using conv 2D and activation as relu

The final output is given with dense 1 as we have only 2 classes and activation as sigmoid.

Finally calling summary for knowing the total parameters of the model. On Line 1 we learn a total of 32 filters. Max pooling is then used to reduce the spatial
dimensions of the output volume. We then learn 64 filters on Line 4. Again, max
pooling is used to reduce the spatial dimensions. The final Conv2D layer learns 128 filters.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               9470464
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
_________________________________________________________________
```

Figure 30: output for model.

The output has 9,494,561 parameters totally which are trainable.

RELU: -

The Rectified Linear Activation Function. The rectified linear activation function (called Relu) has been shown to lead to very high-performance networks. This function takes a single number as an input, returning 0 if the input is negative, and the input if the input is positive.

The main advantage of using the Relu function over other activation functions is that it does not activate all the neurons at the same time. Due to this reason, during the back-propagation process, the weights and biases for some neurons are not updated.

Networks that use the rectifier function for the hidden layers are referred to as rectified networks.

SIGMOID: -

The sigmoid function is a activation function in terms of underlying gate structured in co-relation to Neurons firing, in Neural Networks. The derivative, also acts to be an activation function in terms of handling Neuron activation in terms of NN's. The differential between the two is activation degree and interplay

5.3.3 COMPILING THE MODEL:

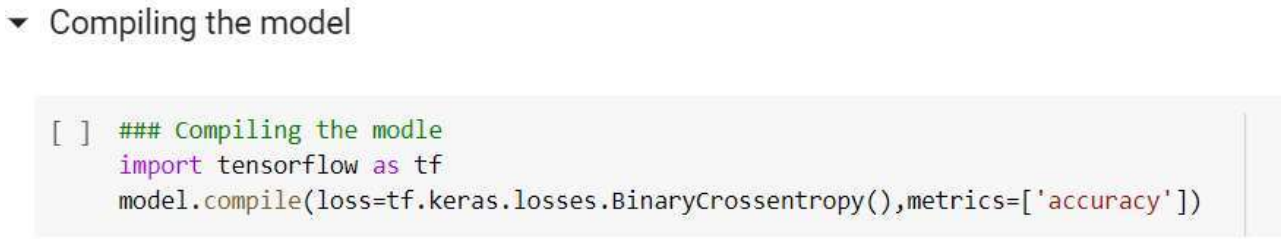


Figure 31: Compiling the model.

The model is compiled using compile function and loss which uses binary cross entropy function and the metrics as accuracy.

MODEL: -

In machine learning, a model is a function with learnable parameters that maps an input to an output. The optimal parameters are obtained by training the model on Data . A well-trained model will provide an accurate mapping from the input to the desired output.

MODEL.COMPILE: -

Compile defines the loss function, the optimizer and the metrics. That's all. It has nothing to do with the weights and you can compile a model as many times as you want without causing any problem to pretrained weights.

LOSS FUNCTION:

The Loss Function is one of the important components of Neural Networks. Loss is nothing but a prediction error of Neural Net. And the method to Calculating the loss is called Loss Function. In simple words, the Loss is used to calculate the gradients.

Loss functions and optimizations. Machines learn by means of a loss function. It's a method of evaluating how well specific algorithms model the given data. If predictions deviate too much from actual results, loss function would cough up a very large number.

CROSS ENTROPY:

Cross-entropy is a measure from the field of information theory, building upon entropy and generally calculating the difference between two probability distributions. ... Cross-entropy can be used as a loss function when optimizing classification models like logistic regression and artificial neural networks.

Binary-Cross-Entropy Loss also called Sigmoid Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss. It's called Binary Cross-Entropy Loss because it sets up a binary classification problem between C′=2 classes for every class in C.

METRICS:

A metric 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 is not used when training the model.

5.3.4 TRAINING THE MODEL:

Figure 32: Training the model.

Using models. fit function we are testing the accuracy of the model which will be best fit or under fit or over fit model

We are using train generator and validation generator to take all the images of train and validation data

We are giving the epochs as 15 and the batch size as 32 which mean with size of 32 it divides into 15 batches.

MODEL.FIT:

When you chose something given what you know from the model, you are making a prediction that what you chose is in fact you want to eat. Model fitting is creating that simplified representation in a way that can generally be used successfully given new data.

We then use Keras to allow our model to train for 15 epochs on a batch_size of 32. When we call them. fit () function it makes assumptions: The entire training set can fit into the Random-Access Memory (RAM) of the computer.

```
Epoch 1/20
82/82 [==============================] - 63s 742ms/step - loss: 0.9010 - accuracy: 0.5579 - val_loss: 0.5491 - val_accuracy: 0.8290
Epoch 2/20
82/82 [==============================] - 60s 732ms/step - loss: 0.6820 - accuracy: 0.6610 - val_loss: 0.4623 - val_accuracy: 0.9155
Epoch 3/20
82/82 [==============================] - 60s 734ms/step - loss: 0.5761 - accuracy: 0.7505 - val_loss: 0.3979 - val_accuracy: 0.9395
Epoch 4/20
82/82 [==============================] - 59s 722ms/step - loss: 0.4939 - accuracy: 0.8258 - val_loss: 0.3443 - val_accuracy: 0.9510
Epoch 5/20
82/82 [==============================] - 60s 728ms/step - loss: 0.4369 - accuracy: 0.8520 - val_loss: 0.3005 - val_accuracy: 0.9614
Epoch 6/20
82/82 [==============================] - 60s 732ms/step - loss: 0.3886 - accuracy: 0.8742 - val_loss: 0.2648 - val_accuracy: 0.9718
Epoch 7/20
82/82 [==============================] - 62s 751ms/step - loss: 0.3473 - accuracy: 0.9070 - val_loss: 0.2342 - val_accuracy: 0.9771
Epoch 8/20
82/82 [==============================] - 61s 741ms/step - loss: 0.3251 - accuracy: 0.9028 - val_loss: 0.2096 - val_accuracy: 0.9771
Epoch 9/20
82/82 [==============================] - 61s 740ms/step - loss: 0.2730 - accuracy: 0.9342 - val_loss: 0.1875 - val_accuracy: 0.9791
Epoch 10/20
82/82 [==============================] - 59s 716ms/step - loss: 0.2502 - accuracy: 0.9411 - val_loss: 0.1694 - val_accuracy: 0.9812
Epoch 11/20
82/82 [==============================] - 59s 717ms/step - loss: 0.2253 - accuracy: 0.9508 - val_loss: 0.1552 - val_accuracy: 0.9833
Epoch 12/20
82/82 [==============================] - 59s 717ms/step - loss: 0.2205 - accuracy: 0.9514 - val_loss: 0.1423 - val_accuracy: 0.9844
Epoch 13/20
82/82 [==============================] - 59s 715ms/step - loss: 0.2010 - accuracy: 0.9527 - val_loss: 0.1320 - val_accuracy: 0.9844
Epoch 14/20
82/82 [==============================] - 59s 724ms/step - loss: 0.1824 - accuracy: 0.9596 - val_loss: 0.1229 - val_accuracy: 0.9844
Epoch 15/20
82/82 [==============================] - 61s 739ms/step - loss: 0.1911 - accuracy: 0.9457 - val_loss: 0.1161 - val_accuracy: 0.9854
Epoch 16/20
82/82 [==============================] - 60s 730ms/step - loss: 0.1714 - accuracy: 0.9615 - val_loss: 0.1085 - val_accuracy: 0.9854
Epoch 17/20
82/82 [==============================] - 59s 720ms/step - loss: 0.1567 - accuracy: 0.9669 - val_loss: 0.1023 - val_accuracy: 0.9854
Epoch 18/20
82/82 [==============================] - 59s 721ms/step - loss: 0.1517 - accuracy: 0.9628 - val_loss: 0.0972 - val_accuracy: 0.9854
Epoch 19/20
82/82 [==============================] - 60s 727ms/step - loss: 0.1515 - accuracy: 0.9614 - val_loss: 0.0923 - val_accuracy: 0.9854
Epoch 20/20
82/82 [==============================] - 59s 718ms/step - loss: 0.1383 - accuracy: 0.9636 - val_loss: 0.0876 - val_accuracy: 0.9854
```

Figure 33: output for training model

In the first epoch the loss value is 0.5, accuracy value is 0.7, val loss is 0.1 and val accuracy is 0.9.

By the end of 20th epoch, the loss value has decreased to 0.01 and val loss to 0.015 and the accuracy increased to 0.99 and val accuracy to 0.98.

The overall accuracy of the model is 98% which is a best fit model

```
[ ]  train_acc = history.history['accuracy']
     val_acc = history.history['val_accuracy']
     train_loss = history.history['loss']
     val_loss = history.history['val_loss']

     epochs = list(range(1,16))
     plt.figure(figsize=(16,4))
     plt.subplot(1,2,1)
     plt.plot(epochs,train_acc,label='train_acc')
     plt.plot(epochs,val_acc,label='val_acc')
     plt.title('accuracy')
     plt.legend()
     plt.subplot(1,2,2)
     plt.plot(epochs,train_loss,label='train_loss')
     plt.plot(epochs,val_loss,label='val_loss')
     plt.title('loss')
     plt.legend()
```

Figure 34: Code for Graph of training model

Reading the history of accuracy, val accuracy, loss and val loss using history. history and plotting the graph for history of accuracies and losses. Assigning the history of accuracy to train_acc, val accuracy to val_acc, loss to train_loss, val loss to val_loss for labelling the graph.

First, we wrote the code for accuracy graph using subplot and plot position as 1,2,1 and using plot we are reading all the epochs of train accuracy and val accuracy and giving the labels of train_acc and val_acc which reads all the history of both accuracies.
The plot title is accurate.

Next, we wrote code for loss graph using subplot and plot position as 1,2,2 and using plot we read the epochs of train loss and val loss, giving the labels of train_loss

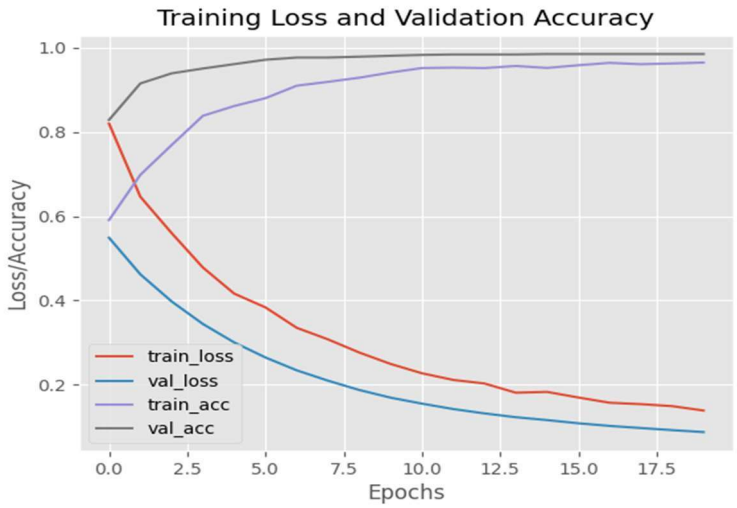


Figure 35: Graph of training model

The graph has accuracy increasing and loss decreasing for training data

Accuracy and loss are fluctuating for validation data.

Overall, it's increasing so it's a best fit model.

5.4 PREDICTING THE IMAGE:

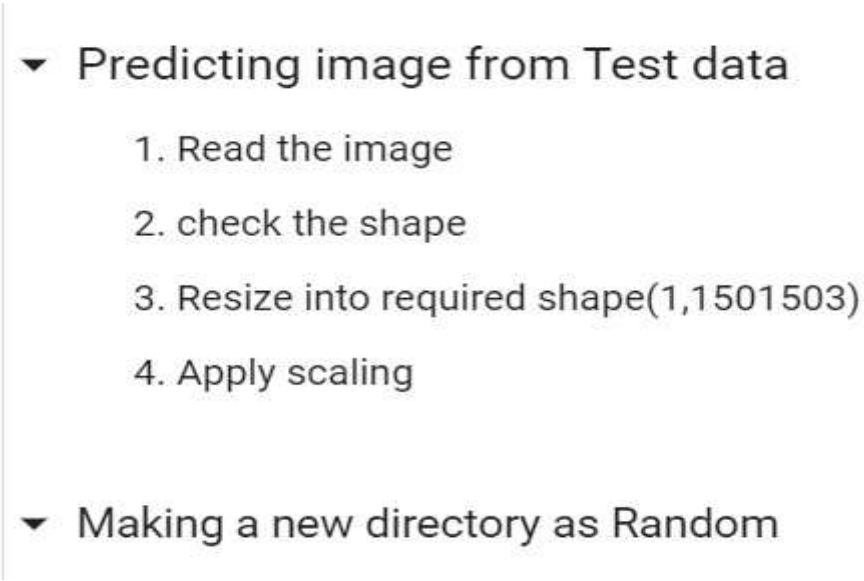


Figure 36: Terms of prediction.

For testing the model, we need to predict the images of with mask and without mask of test data.

But the images of test data are separated in different folders as with mask and without mask, we need to test the images randomly. So for the prediction of images randomly we are making a new directory named "random" and moving all the images of both the with mask and without mask folders to the random directory and then random directory is used to predict the images.

5.4.1 MAKING A NEW DIRECTORY:

Figure 37: New directory

Using mkdir command we are making a new directory "random" and cd command is used for running the further codes in a random directory.

Figure 38: Moving images to random

Using mv command we move all the images of with mask and without mask folders of test directory to random directory. We used os.listdir to read the data of random directory and len function to print no of images in a random directory which are 194 images.

5.4.2: TESTING AND PREDICTING FOR RANDOM IMAGES:

Testing the model for random images

```
: import random, os
path = '/content/random'
random_filename = random.choice([
    x for x in os.listdir(path)
    if os.path.isfile(os.path.join(path, x))
])

print(random_filename)
```

```
179.jpg
```

Figure 39: printing random file name-1

We are importing random and os libraries.

We are assigning the path of random directory to “path” variable

Now we assigning “random_filename” to read any one random image from the directory random using the key work random. choice

In random choice we are reading the path and joining it to “x” variable which reads any one image from 194 images

Finally, we print the name of the image using the print function.

```
from tensorflow.keras.preprocessing import image
import numpy as np
img = plt.imread(os.path.join('/content/random',random_filename))
print(type(img))
img = tf.keras.preprocessing.image.img_to_array(img)
print(img.shape)
print(type(img))
img = tf.image.resize(img,(150,150))
## Scaling
img = img/255
print(img.shape)
img = np.expand_dims(img,axis=0)
print(img.shape)
plt.imshow(img[0,:,:,:])
```

Figure 40: Code for showing the random image-1

We are importing images from TensorFlow and keras.

We are reading the image which is randomly selected from 194 test images using random_filename and storing it in "img". Then plotting the image after rescaling and resizing it. We print the actual size and the scaled size of the image along with the image.

```
<class 'numpy.ndarray'>
(267, 189, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
```

```
<matplotlib.image.AxesImage at 0x7f6b76ba4908>
```

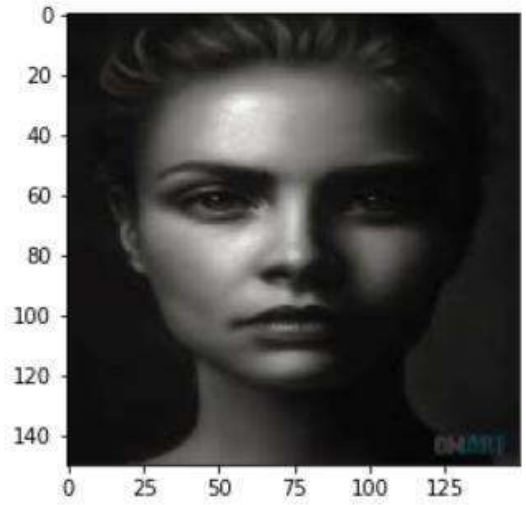


Figure 41: predicting image 1

We are predicting the value of a random image using a model. predict and the range of without mask images are 0-1 if the image is predicted correct it shows around 1 and if not, it goes to less value.

As this image is a person without mask it is giving 0.99 value so the model is predicting correctly

```
]: classes = model.predict_classes(img)
   print(classes)
   if (classes==1):
     print("with out mask")
   else :
     print("with mask")

   [[1]]
   with out mask
```

Figure 42: Reading the class of random image -1

To say whether the image is with a mask or without mask image we use a model. predict _ classes to know the class of image Class of without mask is 1 and with mask is 0, so if class is 1 it prints without mask and else which is 0 it prints with mask.

Here the image is without mask so its class is 1.

NOW TESTING FOR SOME OTHER RANDOM IMAGES:

```
[46] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)

     114-with-mask.jpg
```

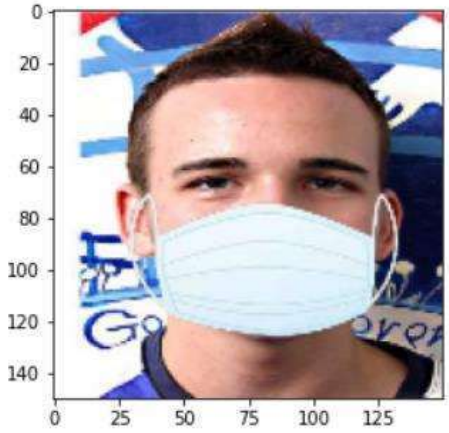
Figure 43: random file name 2

It is an image of person with mask.

```
] from tensorflow.keras.preprocessing import image
  import numpy as np
  img = plt.imread(os.path.join('/content/random',random_filename))
  print(type(img))
  img = tf.keras.preprocessing.image.img_to_array(img)
  print(img.shape)
  print(type(img))
  img = tf.image.resize(img,(150,150))
  ## Scaling
  img = img/255
  print(img.shape)
  img = np.expand_dims(img,axis=0)
  print(img.shape)
  plt.imshow(img[0,:,:,:])
```

Figure 44: code for showing random image 2

```
<class 'numpy.ndarray'>
(433, 327, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d65a0e80>
```




```
[48] model.predict(img)

  array([[8.592877e-20]], dtype=float32)
```

Figure 45: prediction of random image 2

Here the image is a person with mask. The array value is accuracy of predicting the person with mask, if the person wears a mask properly it varies if not it decreases.

```
[49] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")
```

```
[[0]]
with mask
```

Figure 46: class of random image 2

Class is 0 and with mask image.

```
[50] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)
```

```
284.jpg
```

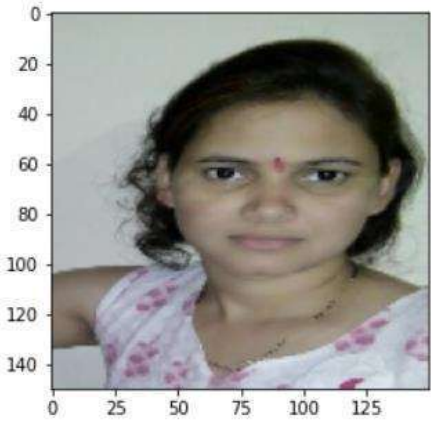
Figure 47: random filename 3

```
[] from tensorflow.keras.preprocessing import image
   import numpy as np
   img = plt.imread(os.path.join('/content/random',random_filename))
   print(type(img))
   img = tf.keras.preprocessing.image.img_to_array(img)
   print(img.shape)
   print(type(img))
   img = tf.image.resize(img,(150,150))
   ## Scaling
   img = img/255
   print(img.shape)
   img = np.expand_dims(img,axis=0)
   print(img.shape)
   plt.imshow(img[0,:,:,:])
```

Figure 48: code for testing random image 3

It is an image of a person without a mask.

```
<class 'numpy.ndarray'>
(576, 398, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d62bc8d0>
```




```
[52] model.predict(img)

array([[1.]], dtype=float32)
```

Figure 49: Prediction of random image 3

The image is without mask and the array value is 1, so the model predicted the person is not wearing a mask.

```
[53] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")
```

```
[[1]]
with out mask
```

Figure 50: class of random image 3

Class is also 1 and its without mask image

▼ Testing the model for random images

```
[57] import random, os
     path = '/content/random'
     random_filename = random.choice([
         x for x in os.listdir(path)
         if os.path.isfile(os.path.join(path, x))
     ])

     print(random_filename)
```

```
464-with-mask.jpg
```

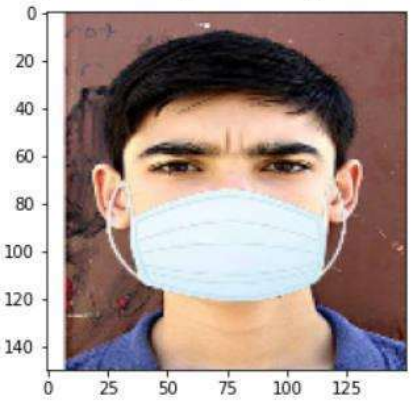
Figure 51: random filename 4

```
[58] from tensorflow.keras.preprocessing import image
     import numpy as np
     img = plt.imread(os.path.join('/content/random',random_filename))
     print(type(img))
     img = tf.keras.preprocessing.image.img_to_array(img)
     print(img.shape)
     print(type(img))
     img = tf.image.resize(img,(150,150))
     ## Scaling
     img = img/255
     print(img.shape)
     img = np.expand_dims(img,axis=0)
     print(img.shape)
     plt.imshow(img[0,:,:,:])
```

Figure 52: code of showing random image 4

The image is a person with a mask.

```
<class 'numpy.ndarray'>
(428, 320, 3)
<class 'numpy.ndarray'>
(150, 150, 3)
(1, 150, 150, 3)
<matplotlib.image.AxesImage at 0x7fe5d6463550>
```




```
[59] model.predict(img)

     array([[5.7457684e-13]], dtype=float32)
```

Figure 53: Predicting random image 4

The array value is less than the previous image because the person is not wearing the mask properly, so the array value is lesser.

```
[60] classes = model.predict_classes(img)
     print(classes)
     if (classes==1):
       print("with out mask")
     else :
       print("with mask")

     [[0]]
     with mask
```

Figure 54: class of random image 4

Class is 0 and the image is with mask

Hence the model has predicted all the images randomly and with correct accuracy, the model is a best fit model.

CONCLUSION:

The proposed work is designed to develop a model to detect, recognize and classify human face. The software used to test the functionality are Anaconda, python 3 and PyCharm. For Face detection dataset we used convolution neural network model for face recognition and classification. The os, matplotlib, tensor flow and other libraries work in support with python programming.

The performance measures are validated with the CNN model designed with an accuracy of 98%. Though we have an imbalanced data i.e., images with mask are 590and without mask are 586 yet we have been able to be successful using metrics as accuracy, we have got validation accuracy as 98.59% and train accuracy as 99.39% and After plotting the graph between accuracy we can observe that our model is precise. With the help of loss as Binary Cross Entropy, we have compiled our model. By looking at the above value after predicting we can see that it gives array having value closest to 1and according to our approach if we get value closest to 1 then it is detecting the person is not wearing a mask and other array values for predicting whether the person is wearing the mask properly or not, if the person wears mask properly it gives a big array value and if a person is not wearing a mask properly then it gives a small array value. So, we are successful in using the approach of CNN.

However, the results prove that the network architecture designed has better advancements and this application are widely used to achieve face classification and recognition.

REFERENCES:

- R. Chauhan, K. K. Ghanshala and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCC), Jalandhar, India, 2018, pp. 278-282, doi: 10.1109/ICSCCC.2018.8703316.
- Pooja S., Preeti S. (2020) Face Mask Detection Using AI. In: Khosla P.K., Mittal M., Sharma D., Goyal L.M. (eds) Predictive and Preventive Measures for Covid-19 Pandemic. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-33-4236-1_16

DATA SET LINK:

https://github.com/prajnasb/observations

TENSORFLOW LINK:

https://www.tensorflow.org/install/errors

https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

MACHINE LEARNING LINK:

https://en.wikipedia.org/wiki/Machine_learning

PYTHON LINKS:

https://www.python.org/

https://en.wikipedia.org/wiki/Python_(programming_language)