*A Project report*

*on*

# FACE MASK DETECTION USING DEEP LEARNING

*Submitted in partial fulfillment of the requirements*

*for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

*in*

**Computer Science & Engineering**

*by*

| | |
|---|---|
| **M. LAKSHMI JYOSHNA** | **(184G1A0532)** |
| **P. MANASA** | **(184G1A0541)** |
| **B. PADMINI** | **(184G1A0555)** |
| **G. ARAVIND** | **(184G1A0503)** |
| **S. MUNI VINOD** | **(194G5A0505)** |

Under the Guidance of

**Dr. B. Hari Chandana,** **M.Tech., Ph.D**

Assistant Professor

**Department of Computer Science & Engineering**

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY : ANANTHAPURAMU**

**(Affiliated to JNTUA,Approved by AICTE,New Delhi, Accredited by NAAC with 'A' Grade& Accredited by NBA(EEE, ECE &CSE)**

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu – 515701

**2021-22**

**SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY : ANANTHAPURAMU**

**(Affiliated to JNTUA,Approved by AICTE,New Delhi, Accredited by NAAC with 'A' Grade& Accredited by NBA(EEE, ECE &CSE)**

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu – 515701

# Certificate

This is to certify that the project report entitled Face Mask Detection using Deep Learning is the bonafide work carried out by **M. Lakshmi Jyoshna** bearing Roll Number 184G1A0532, **P. Manasa** bearing Roll Number 184G1A0541, **B. Padmini,** bearing Roll Number 184G1A0555, **R. Aravind** bearing Roll Number 184G1A0503, **S. Muni Vinod** and bearing Roll Number 194G5A0505, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2021-2022.

| | |
|---|---|
| **Signature of the Guide** | **Head of the Department** |
| Dr. B. Hari Chandana, M.Tech.,PhD | Mr. P. Veera Prakash, M.Tech.,(Ph.D) |
| Assistant Professor | Assistant Professor & HOD |

Date: **EXTERNAL EXAMINER**

Place: Ananthapuramu

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that we would like to express our indebted gratitude to our Guide **Dr. B. Hari Chandana, M.Tech.,Ph.D, Assistant Professor, Computer Science and Engineering Department**, who has guided us a lot and encouraged us in every step of the project work. We thank her for the stimulating guidance, constant encouragement and constructive criticism which have made possible to bring out this project work.

We express our deep felt gratitude to **Mr. K. Venkatesh, M.Tech., Assistant Professor, project coordinator** valuable guidance and unstinting encouragement enable us to accomplish our project successfully in time.

We are very much thankful to **Mr. P. Veera Prakash, M.Tech.,(P.hD), Assistant Professor & Head of the Department, Computer Science & Engineering,** for his kind support and for providing necessary facilities to carry out the work.

We wish to convey our special thanks to **Dr. G. Balakrishna, Ph.D, Principal** of **Srinivasa Ramanujan Institute of Technology** for giving the required information in doing our project work. Not to forget, we thank all other faculty and non-teaching staff, and my friends who had directly or indirectly helped and supported us in completing our project in time.

We also express our sincere thanks to the Management for providing excellent facilities**.**

Finally, we wish to convey our gratitude to our family who fostered all the requirements and facilities that we need.

**184G1A0532**
**184G1A0541**
**184G1A0555**
**184G1A0503**
**194G5A0505**

# DECLARATION

We, Ms. M. Lakshmi Jyoshna bearing reg no: 184G1A0532, Ms. P. Manasa bearing reg no: 184G1A0541, Ms. B. Padmini bearing reg no: 184G1A0555, Mr. G. Aravind bearing reg no:184G1A0503, and Mr. S. Muni Vinod bearing reg no: 194G5A0505 students of SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram , hereby declare that the dissertation entitled "FACE MASK DETECTION USING DEEP LEARNING" embodies the report of our project work carried out by us during IV Year Bachelor of Technology under the guidance of Dr.B.Hari Chandana,M.Tech.,PhD, Assistant Professor, Department of CSE, and this work has been submitted for the partial fulfillment of the requirements for the award of Bachelor of Technology degree.

The results embodied in this project report have not been submitted to any other Universities of Institute for the award of Degree.

| | |
|---|---|
| M.LAKSHMI JYOSHNA | Reg no: 184G1A0532 |
| P.MANASA | Reg no: 184G1A0541 |
| B.PADMINI | Reg no: 184G1A0555 |
| G.ARAVIND | Reg no: 184G1A0503 |
| S.MUNI VINOD | Reg no: 194G5A0505 |

# Contents

# List of Figures

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CNN | Convolutional Neural Networks |
| ReLU | Rectified Linear Units |
| MTCNN | Multi-Task Cascaded Convolutional Neural Network |
| ROI | Region Of Interest |
| FC | Fully Connected |

# ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This model presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras , and Scikit-Learn. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion.

The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting. According to this motivation we demand mask detection as a unique and public health service system during the global pandemic COVID-19 epidemic. The model is trained by face mask image and non-face mask image.

**Keywords**: *COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image.*

# CHAPTER - 1
# INTRODUCTION

## 1.1 Introduction

The trend of wearing face masks in public is rising due to the COVID- 19 corona virus epidemic all over the world. Before Covid-19, People used to wear masks to protect their health from air pollution. While other people are self-conscious about their looks, they hide their emotions in the public to hide their faces. More than five million cases were infected by COVID- 19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and overcrowded areas.

We can tackle and predict new diseases by the help of new Technologies such as artificial intelligence, Iot, Big data, and Machine learning. In order to better understand infection rates might be decrease through our technique. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult in public areas. So we will create a automation process for detecting the faces. Here we introduce a facemask detection model that is based on computer vision and deep learning. The proposed model can be integrated with Surveillance Cameras . the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with Tensor flow and Keras. We will achieve the highest accuracy and consume the least time in the process of training and detection. Face mask detection refers to detect whether a person is wearing a mask or not. In fact, the problem is reverse engineering of face detection where the face is detected using different machine learning algorithms for the purpose of security, authentication and surveillance. Face detection is a key area in the field of Computer Vision and Pattern Recognition. A significant body of research has contributed sophisticated to algorithms for face detection in past. The primary research on face detection was done in 2001 using the design of handcraft feature and application of traditional machine learning algorithms to train effective classifiers for detection and recognition.

The problems encountered with this approach include high complexity in feature design and low detection accuracy. In recent years, face detection methods based on deep convolutional neural networks (CNN) have been widely developed to improve detection performance. Although numerous researchers have committed efforts in designing efficient algorithms for face detection and recognition but there exists an essential difference between 'detection of the face under mask' and 'detection of mask over face'. As per available literature, very little body of research is attempted to detect mask over face. Thus, our work aims to a develop technique that can accurately detect mask over the face in public areas (such as airports. railway stations, crowded markets, bus stops, etc.) to curtail the spread of Coronavirus and thereby contributing to public healthcare. Further, it is not easy to detect faces with/without a mask in public as the dataset available for detecting masks on human faces is relatively small leading to the hard training of the model. So, the concept of transfer learning is used here to transfer the learned kernels from networks trained for a similar face detection task on an extensive dataset. The dataset covers various face images including faces with masks, faces without masks, faces with and without masks in one image and confusing images without masks. With an extensive dataset containing 45,000 images, our technique achieves outstanding accuracy of 98.2%. The major contribution of the proposed work is given below:

1. Develop a novel object detection method that combines one-stage and two-stage detectors for accurately detecting the object in real-time from video streams with transfer learning at the back end.

2. Improved affine transformation is developed to crop the facial areas from uncontrolled real-time images having differences in face size, orientation and background. This step helps in better localizing the person who is violating the facemask norms in public areas/ offices.

3. Creation of unbiased facemask dataset with imbalance ratio equals to nearly one.

4. The proposed model requires less memory, making it easily deployable for embedded devices used for surveillance purposes.

## What is deep learning?

Deep learning models introduce an extremely sophisticated approach to machine learning and are set to tackle these challenges because they've been specifically modeled after the human brain. Complex, multi-layered “deep neural networks” are built to allow data to be passed between nodes (like neurons) in highly connected ways. The result is a non-linear transformation of the data that is increasingly abstract. While it takes tremendous volumes of data to ‘feed and build’ such a system, it can begin to generate immediate results, and there is relatively little need for human intervention once the programs are in place.

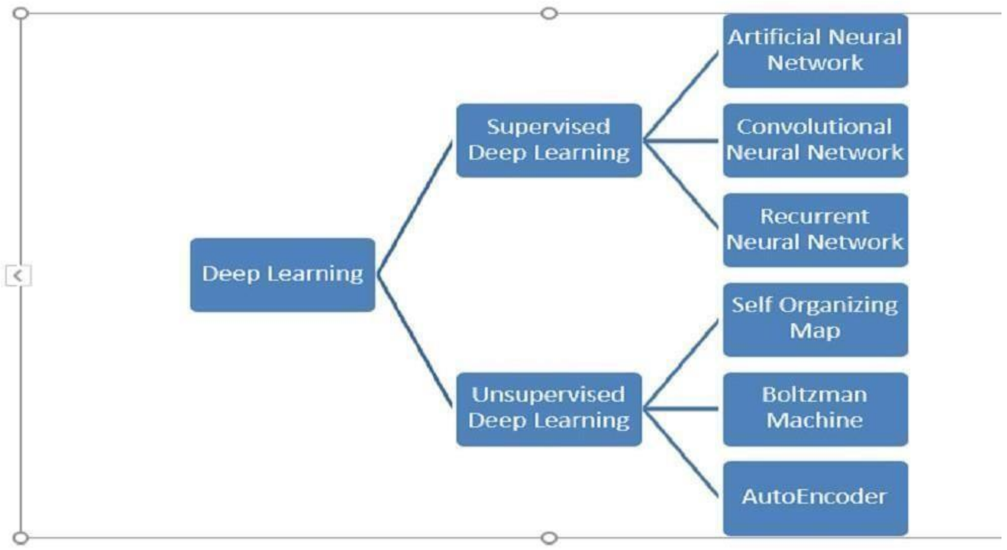


**Fig.1.1. Classification of Deep learning**

CNNs are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of artificial intelligence by offering systems that simulate different types of biological human brain activity. A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds ofcognitive tasks. Like other kinds of artificial neural networks, a convolutional neural network has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. This simulates some of the actions in the human visual cortex.

# CHAPTER – 2
# LITERATURE SURVEY

**[Raza Ali, Saniya Adeel,Akhyar Ahmed-1]:**To mitigate the spread of COVID-19 pandemic, measures need to be taken. We have modeled a face mask detector using SSD architecture and switch learning strategies in neural networks. To train, validate and check the model, we used the dataset that consisted of 1916 masked faces pix and 1919 unmasked faces images. These snap shots were taken from a number sources like Kaggle and RMFD datasets. The model was once inferred on pictures and live video streams. To pick a base model, we evaluated the metrics like accuracy, precision and recall and chosen MobileNetV2 structure with the fantastic performance having a hundred percent precision and 99% recall. It is also computationally efficient using MobileNetv2 which makes it less complicated to set up the mannequin to embedded systems. This face masks detector can be deployed in many areas like shopping malls, airports and different heavy site visitors places to display the public and to avoid the spread of the disorder by checking who is following primary policies and who is not.[1].

**[Z.-Q. Zhao, P. Zheng, S.-t.Xu, and X. Wu, Object detection with deep learning-2] :**Due to its effective learning ability and benefits in dealing with occlusion, scale transformation, and heritage switches, deep learning-based object detection has been a research hotspot in current years. This paper provides a specific overview on deep learning-based object detection frameworks that handle one of a kind subproblems, such as occlusion, clutter, and low resolution, with exceptional levels of modifications on R-CNN. The evaluation starts on commonplace object detection pipelines which grant base architectures for other associated tasks. Then, three other common tasks, namely, salient object detection, face detection, and pedestrian detection, are also briefly reviewed. Finally, we advise a number of promising future directions to gain a thorough grasp of the object detection landscape. This overview is also meaningful for the developments in neural networks and related getting to know systems, which provides precious insights and pointers for future progress.[2]

**[Amit Chavda, Jason Dsouza,SumeetBadgujar Multi-Stage CNN Architecture-3]:** A two-stage Face Mask Detector was once presented. The first stage makes use of a pretrained Retina Face mannequin for robust face detection, after evaluating its performance with Dlib and MTCNN. An unbiased dataset of masked and unmasked faces was created. The 2d stage worried training three different lightweight Face Mask Classifier models on the created dataset and based on performance, the NAS NetMobile based mannequin was once selected for classifying faces as masked or non-masked. Furthermore, Centroid Tracking was once delivered to our algorithm, which helped improve its performance on video streams. In times of theCOVID-19 pandemic, with the world looking to return to normalcy and people resuming inpersonwork, this system can be easily deployed for automated monitoring of the use of face masks atworkplaces, which will help make them safer.[3]

**[AmritKumar, Bhadani,Anurag Sinha A Facemask detector using machine learning and image processing techniques-4]:** As the technology are blooming with emerging traits the availability so we have novel face masks detector which can perhaps make contributions to public fitness care department. The structure consist of MobileNetV2 classifier and ADAM optimizer as the back bone and low computation scenarios. The our face mask detection is trained on CNN mannequin and we are used Tensor Flow , Keras and python to discover whether individual is wearing a masks or no longer . The model have been examined with image and actual is carried out and, the optimization of the model is non-stop process. This particular model should be used as use case of edge analytics International Journal of Computer Engineering and Applications. The structure consist of MobileNetV2 classifier and ADAM optimizer as the back bone it can be used for excessive and low computation scenarios. The our face masks detection is skilled on CNN mannequin and we are used OvenCV, Tensor Flow , Keras and python to observe whether person is sporting a masks or now not The mannequin were examined with photograph and real- time video stream. The accuracy of mannequin is carried out and, the optimization of the mannequin is continuous process. This specific mannequin may want to be used as use case of part analytics.[4]

**[Sammy v. militante,Nanettev.dionisioReal time face mask recognition with alarm system using deep learning-5]:** The manuscript presented a find out about on real-time facemask recognition with an alarm system through deep learning techniques by using way of Convolutional Neural Networks. This method offers a specific and speedily results for facemask detection. The test results exhibit a exceptional accuracy fee in detecting persons carrying a facemask and now not wearing a facemask. The skilled model was in a position to operate its project the use of the VGG-16 CNN mannequin accomplishing a 96% result for performance accuracy. Moreover, the find out about affords a beneficial tool in hostilities the spread of the COVID-19 virus via detecting a character who wears a facemask or no longer and units an alarm if the individual is not sporting a facemask. Future works consist of the integration of physical distancing, whereby the camera detects the man or woman sporting a facemask or no longer and at the same time measures the distance between each person and creates an alarm if the physical distancing does no longer take a look at properly. The integration of quite a few fashions of CNNs and examine every model with the highest overall performance accuracy throughout coaching to make bigger the overall performance in detecting and recognizing human beings carrying facemasks is suggested. [5]

# CHAPTER – 3

# METHODOLOGY

## 3.1. Existing System

1.After Lockdown it has become so difficult to control the public gatherings, especially in educational institutions and in work places. Hence, Face Masks aremade compulsory.

2.Previously, A model for Non-Masked Face recognition and Masked Face recognition accuracy using principle component Analysis(PCA) to recognize a person.

3.But, There are some drawbacks using PCA as the recognition rate is good for onlyNon-Masked Face.

4.This makes the model less efficient and less accurate. Hence, there is a scope for theemergence of new model.

## 3.2. Proposed System

1.This system is capable to train the dataset of both persons wearing masks and withoutwearing masks.

2.After training the model the system can predicting whether the person is wearing themask or not.

3.It also can access the webcam and predict the result.The proposed method consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons. The algorithm for face mask detection is as follows:

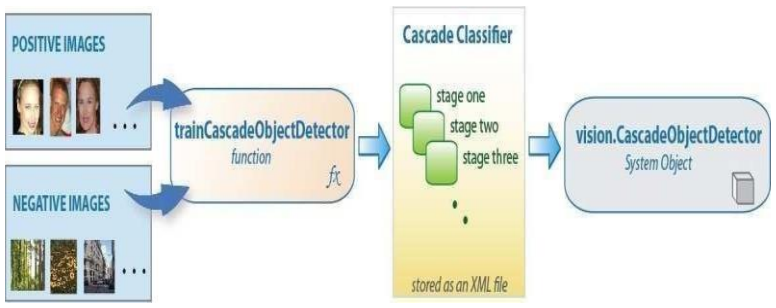
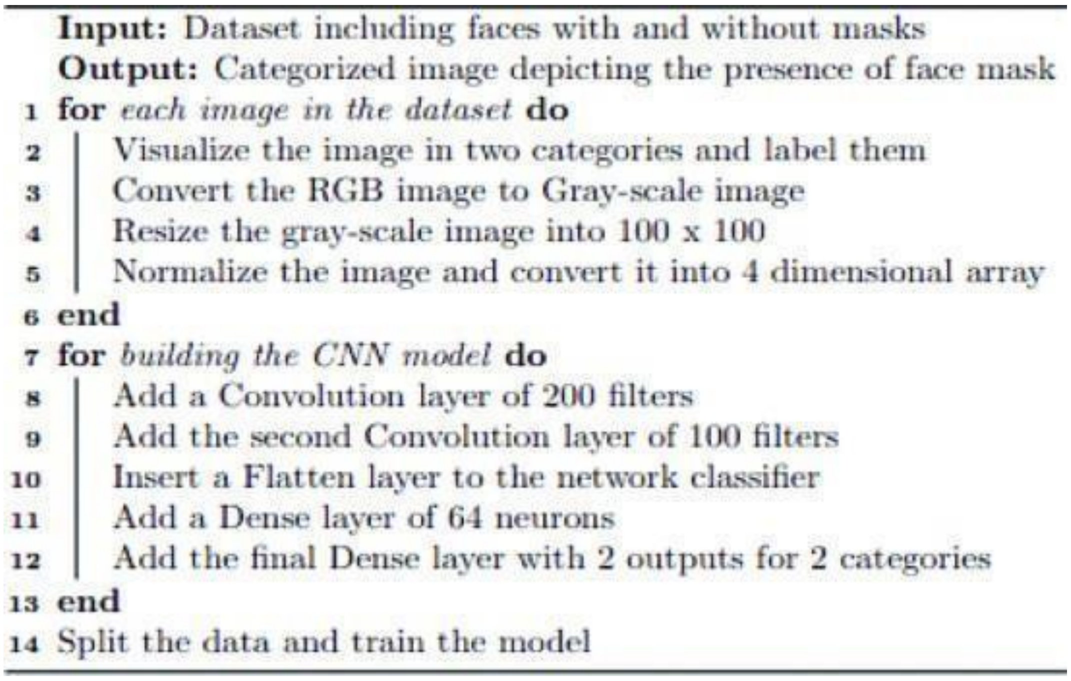


**Fig 3.1. Cascade Classifier**

**Algorithm 1:** Face Mask Detection



```
Input: Dataset including faces with and without masks
Output: Categorized image depicting the presence of face mask
for each image in the dataset do
    Visualize the image in two categories and label them
    Convert the RGB image to Gray-scale image
    Resize the gray-scale image into 100 x 100
    Normalize the image and convert it into 4 dimensional array
end
for building the CNN model do
    Add a Convolution layer of 200 filters
    Add the second Convolution layer of 100 filters
    Insert a Flatten layer to the network classifier
    Add a Dense layer of 64 neurons
    Add the final Dense layer with 2 outputs for 2 categories
end
Split the data and train the model
```

**Data Processing**

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities [6]. The proposed method deals with image and video data using Numpy and OpenCV.

**Data Visualization**

Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in thedataset. The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’.The statement categories=os.listdir(data_path) categorizes the list of directories in the specified data path. The variable categories now looks like: [‘with mask’, ‘without mask’]Then to find the number of labels, we need to distinguish those categories using labels=[i for i in range(len(categories))]. It sets the labels as: [0, 1] Now, each category is mapped to its respective label using label_dict=dict(zip(categories,labels)) which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently. The mapped variable label_dict looks like: {‘with mask’: 0, ‘without mask’: 1}

**Conversion of RGB image to Gray image**

Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from colorto-grayscale. This is because the color-to-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously.

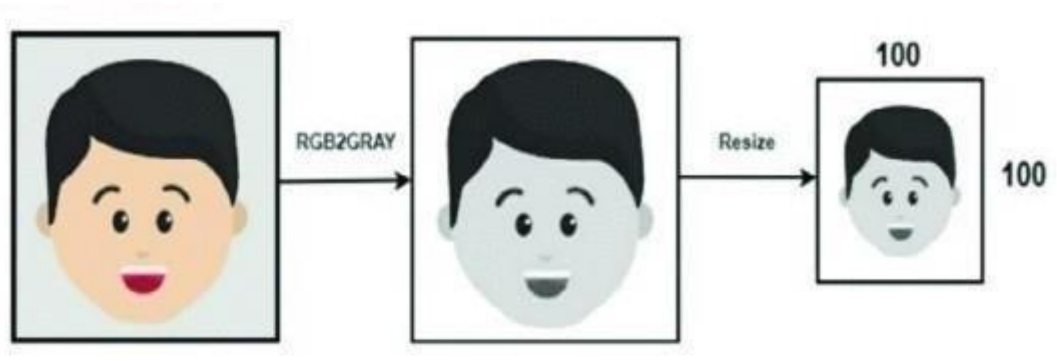


**Fig 3.2. Conversion of a RGB image to a Gray Scale image of 100x100 size**

We use the function cv2.cvtColor(input_image, flag) for changing the color space. Here flag determines the type of conversion [9]. In this case, the flag cv2.COLOR_BGR2GRAY is used for gray conversion. Deep CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset. Using cv2.resize() the gray scale image is resized into 100 x 100.

## Image Reshaping

The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors [10]. Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint. [11]. The images are normalized to converge the pixel range between 0 and 1. Then they are converted to 4 dimensional arrays using data=np.reshape(data,(data.shape[0], img_size,img_size,1)) where 1 indicates the Grayscale image. As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

## Training of Model

### Building the model using CNN architecture

CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN. The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, input_shape is specified as data.shape[1:] which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions [14]. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool_size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape_of_output = (input_shape – pool_size + 1) / strides), where strides has default value (1,1) [15].

The second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

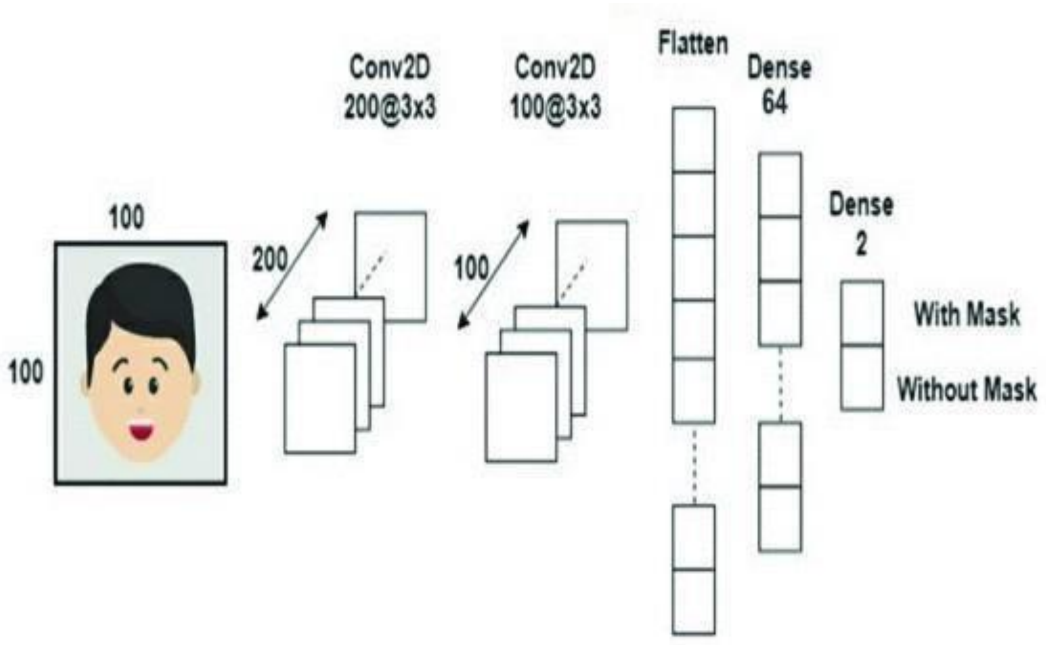


**Fig3.3. Convolutional Neural Network Architecture**

The learning process needs to be configured first with the compile method [13]. Here "adam" optimizer is used. categorical_crossentropy which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize). As the problem is a classification problem, metrics is set to "accuracy".

## Splitting the data and training the CNN model

After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized train_test_split help to produce accurate results while making a prediction. The test_size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using ModelCheckpoint. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting. Fig. 5 depicts visual representation of the proposed model.

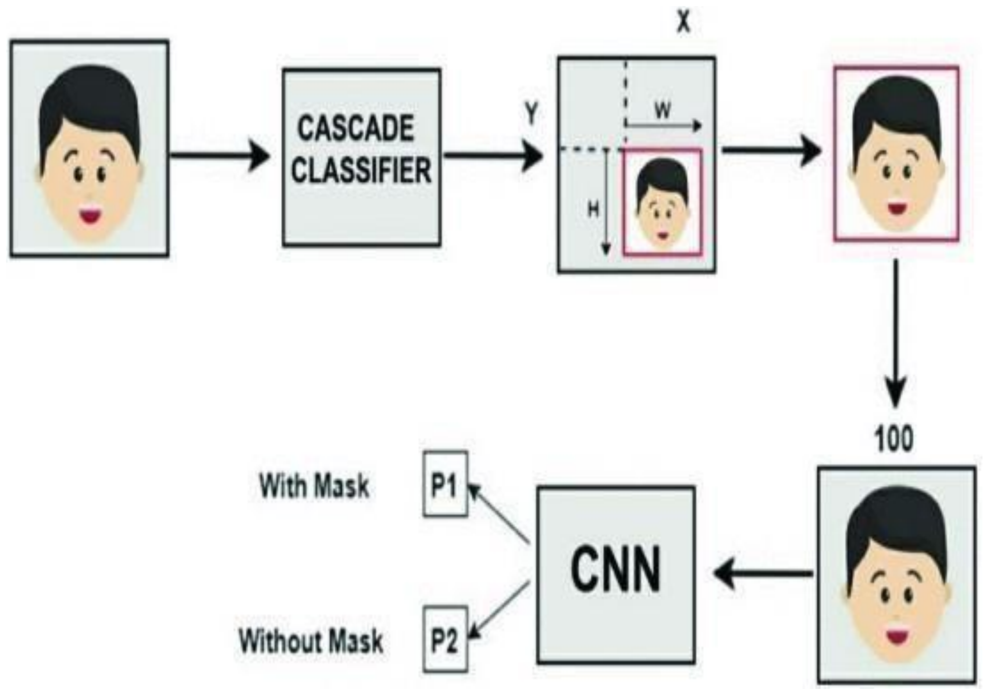


**Fig3.4. overview of the model**

## 3.3. Python Libraries:

### Tensorflow Framework:

1.Tensor flow is an open-source software library.

2.Tensor flow was originally developed by researchers and engineers.

3.It is working on the Google Brain Team within Google’s Machine Intelligence research organization the purposes of conducting machine learning and deep neural networks research.

4.It is an opensource framework to run deep learning and other statistical and predictive analytics workloads.

5.It is a python library that supports many classification and regression algorithms andmore generally deep learning.

6.TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks.

7.It is a symbolic math library, and is also used for machine learning applications suchas neural networks.

8.It is used for both research and production at Google, TensorFlow is Google Brain's second-generation system.

9.Version 1.0.0 Was Released On February 11, While The Reference Implementation Runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).

10.Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and Ios.

## Opencv:

1.It is a cross-platform library using which we can develop real-time computer visionapplications.

2.It mainly focuses on image processing, video capture and analysis including feature like face detection and object detection.

3.Currently Open CV supports a wide variety of programming languages like C++, Python, Java etc. and is available on different platforms including Windows, Linux, OS X, Android, iOS etc.

4.Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. Open CV-Python is the Python API of Open CV.

5.It combines the best qualities of Open CV C++ API and Python language.

6.OpenCV (Open-Source Computer Vision Library) is an opensource computer vision and machine learning software library. OpenCV was built to provide a commoninfrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCVmakes it easy for businesses to utilize and modify the code.

7.The library has more than 2500optimized algorithms, which includes a comprehensive set of both classic and state-of -the-art computer vision and machine learning algorithms.

8.Algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented. reality, etc.

## Numpy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is opensource software and has many contributors.

The Python programming language was not initially designed for numericalcomputing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric also variously called Numerical Python extensions or NumPy Hugunin, a graduate student at Massachusetts Instituteof Technology (MIT) joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer.

In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported num-array's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

## Matplot

Matplot 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, WX Python, Qt, or GTK+. There is also a procedural "Pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and further joined by Thomas Caswell.

## Python

What exactly is Python? You may be wondering about that. You may be referring to this book because you wish to learn editing but are not familiar with editing languages. Alternatively, you may be familiar with programming languages such as C, C ++, C #, or Java and wish to learn more about Python language and how it compares to these "big word" languages.

## Pandas

Pandas is an open-source library that  is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users. Pandas is a python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it  has the broader goal of becoming the most powerful and flexible opensource data analysis/manipulation tool available in any language. It is already well on its way toward this goal. Explore data analysis with Python. Pandas Data Frames make manipulating your data easy, from selecting or replacing columns and indices to reshaping your data. Pandas is well suited for many different kinds of data:

1.Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet.

2.Ordered and unordered (not necessarily fixed-frequency) time series data.
3.Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.

## Keras

KERAS is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

### Four Principles:

1.Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.

2.Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.

3.Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.

4.Python: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

## 3.4. HAAR Cascade Classifiers :

1.It is an Object Detection Algorithm used to identify faces in an image or a real time video. Dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

2.It is an effective way for object detection.

3.In this approach, lot of positive and negative images are used to train the classifier.

4.In this, a model is pre-trained with frontal features is developed and used in this experiment to detect the faces in real-time.

5.HAAR Cascade is a machine learning-based approach where a lot of positive .

The images which we want our classifier to identify. Negative Images: Images of everything else, which do not contain the object we want to detect WHY HAAR FEATURE BASED CASCADE CLASSIFIERS IS PREFFERD?

6.The key advantage of a HAAR-like feature over most other features is its calculation speed. A HAAR-like feature of any size can be calculated in constant Time (Approximately 60 Microprocessor Instructions).

## 3.5. Neural Networks Versus Conventional Networks:

1.Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem-solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem.

2.Neural networks learn by example. They cannot be programmed to is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable. 3.Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

## 3.6. Architecture Of Neural Networks:

### 3.6.1. Feed-Forward Networks:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

### 3.6.2. Feedback Networks:

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organization.

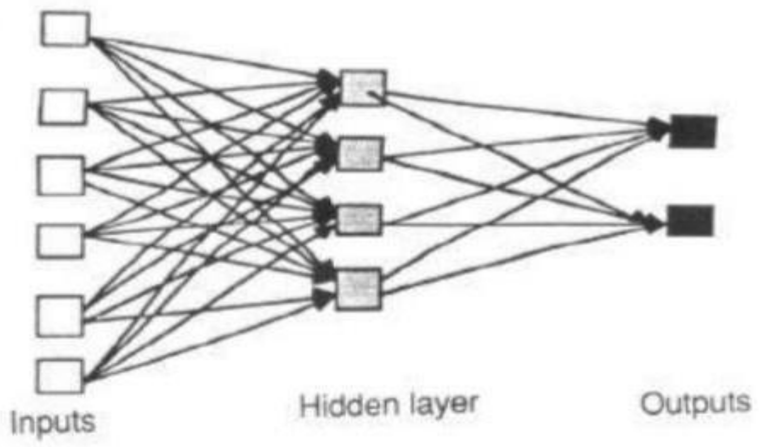


**Fig 3.5. Layers in NN**

### 3.6.3. NETWORK LAYERS:

1.The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of input units is connected to a layer of hidden units, which is connected to a layer of output units. The activity of the input units represents the raw information that is fed into the network.

2.The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units. The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents. Also distinguish single-layer and multi-layer architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

## 3.7. Convolution Neural Network

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

A pre-trained deep CNN model, i.e., is used to extract the deep features of an image. The information about the neural network is strictly concealed by utilizing the lattice-based homomorphic scheme. We implement a real number computation mechanism and a divide-and-conquer CNN evaluation protocol to enable our framework to securely and efficiently evaluate the deep CNN with a large number of inputs.

It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.CNN’s were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits.

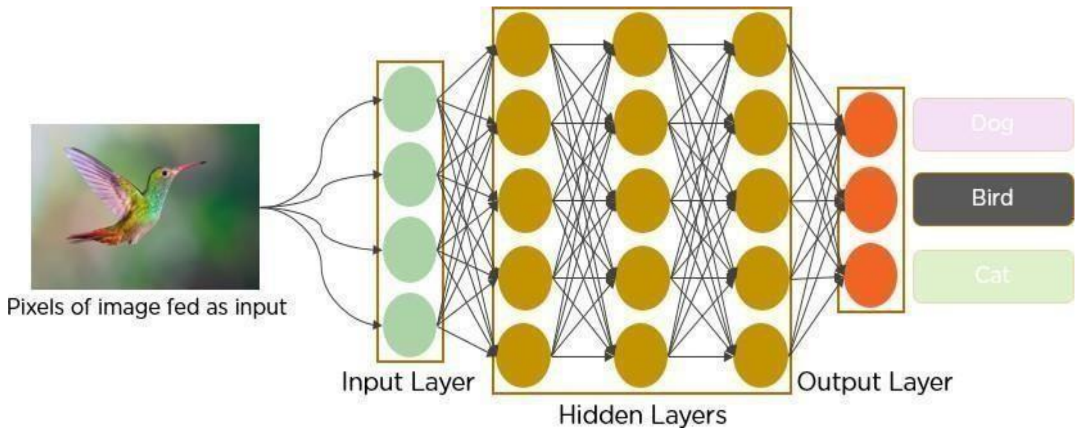


**Fig.3.6. Working structure of CNN Architecture**

In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

Over the years CNNs have become a very important part of many Computer Vision applications. So let's take a look at the workings of CNNs.

## 3.8. Background of CNNs

CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

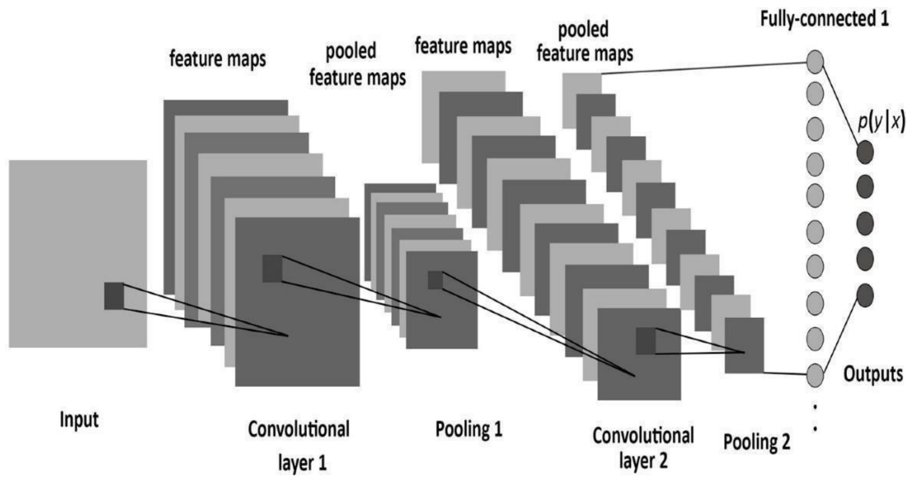


**Fig.3.7. Background of CNN**

The availability of large sets of data, to be more specific ImageNet datasets with millions of labelled images and an abundance of computing resources enabled researchers to revive CNNs.

**What exactly is a CNN?**

In deep learning a convolutional neural network (CNN/ConvNet) is a class deep neural networks, most commonly applied to analyze visual imagery. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

## 3.9. Working of CNN

CNN is composed of three layers where input image is divided into matrix of pixels. Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. Features of a fully connected layer. Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. The three layers of CNN are:

- Convolutional layer
- Pooling layer
- Fully connected layer

**Convolutional Layer**

Convolutional layer is the core building block of the CNN. It carries the main portion of the network's computational load. In this layer, the mathematical operation of convolution is performed between input image , the dot product is taken between the filter and parts of the input image with respect to size of filter(MxM).

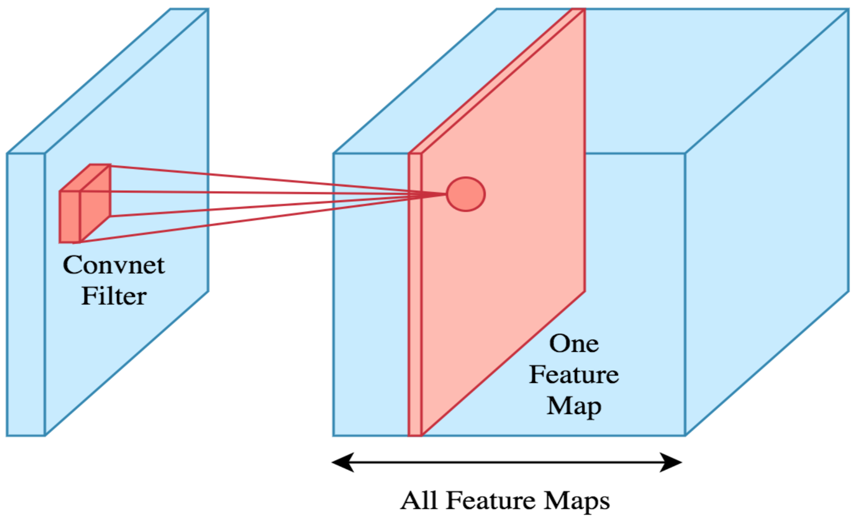


**Fig.3.8. Convolutional layer**

The output of convolutional layer is the Feature map which gives us information about the image such as corners and images.Later this feature map is fed to other layers to learn several other features of the input image.

**Pooling layer**

The primary aim of this layer is to reduce the dimensions of the featured maps thereby, reducing the number of parameters to learn and amount of computation to be performed.

This is performed by decreasing the connections between the layers and independently operates on each feature map.

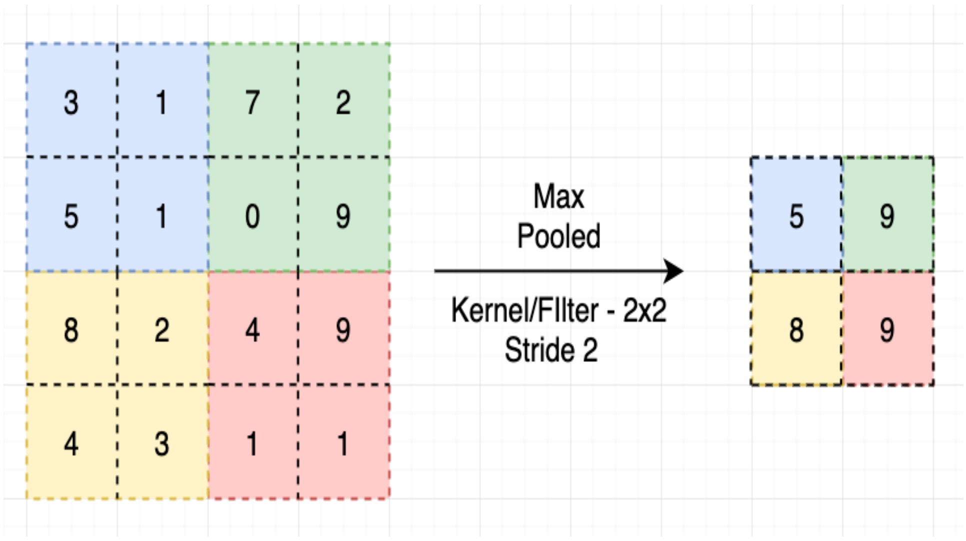


**Fig.3.9. Pooling layer with max pooling**

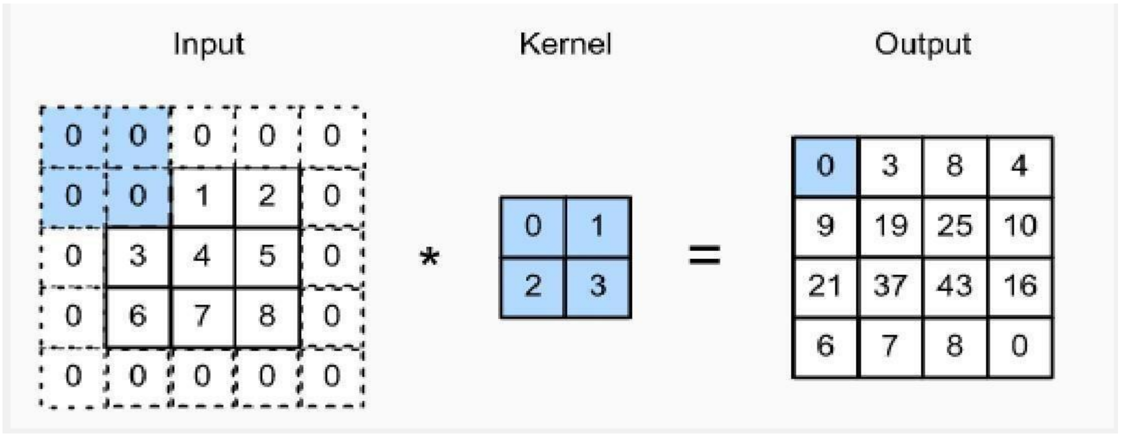
**Padding**

There are two problems arises with convolution:

1. Every time after convolution operation, original image size getting shrinks, as we have seen in above example six by six down to four by four and in image classification task there are multiple convolution layers so after multiple convolution operation, our original image will really get small but we don't want the image to shrink every time.

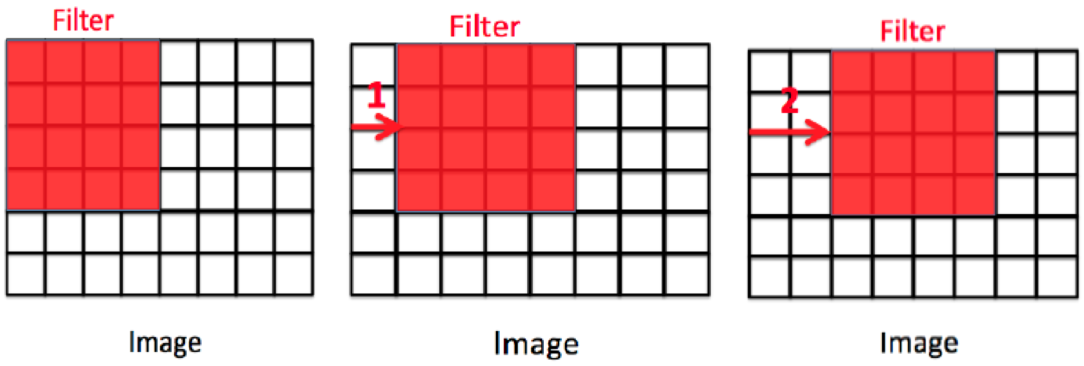
2. The second issue is that, when kernel moves over original images, it touches the edge of the image less number of times and touches the middle of the image more number of times and it overlaps also in the middle. So, the corner features of any image or on the edges aren't used much in the output.

So, in order to solve these two issues, a new concept is introduced called padding. Padding preserves the size of the original image.



**Padded image convolved with 2*2 kernel.**

So if a $n*n$ matrix convolved with an f*f matrix the with padding p then the size of the output image will be (n + 2p — f + 1) * (n + 2p — f + 1) where p =1 in this case.

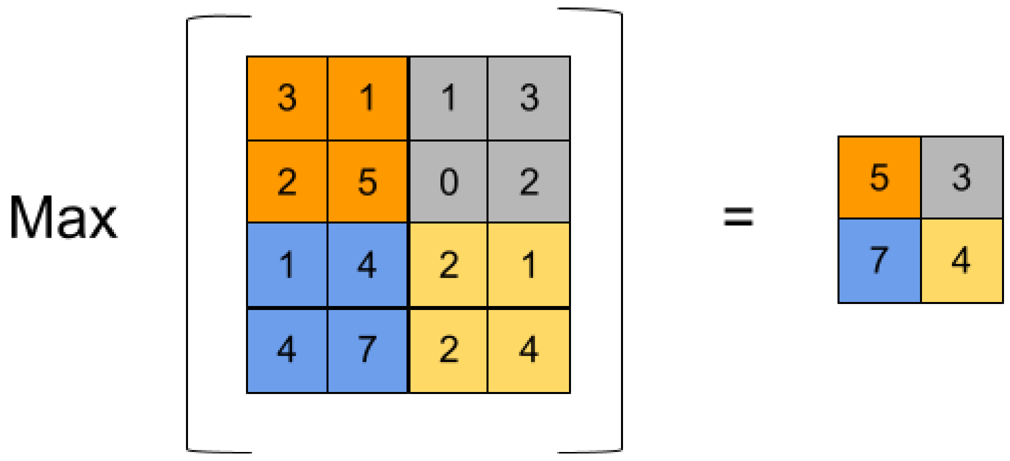


**3.15.3.1. Striding:**

left image: stride =0, middle image: stride = 1, right image: stride =2

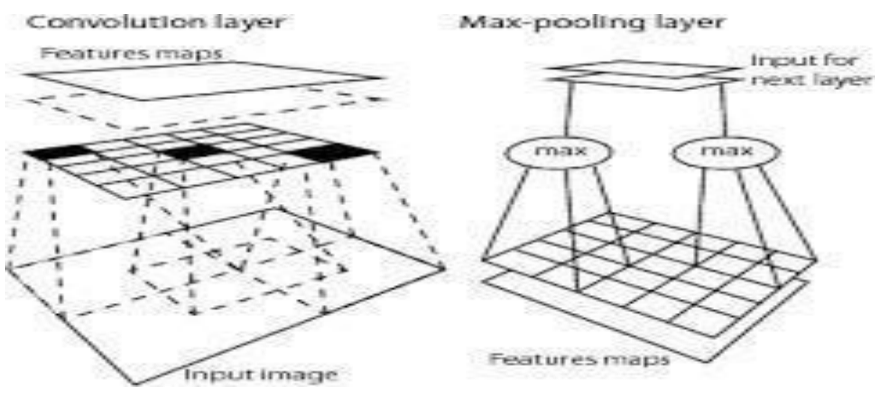
Striding

Stride is the number of pixels shifts over the input matrix. For padding p, filter size $f*f$ and input image size $n*n$ and stride ‘$s$’ our output image dimension will be [ $\{(n + 2p - f + 1) / s\} + 1] * [ \{(n + 2p - f + 1) / s\} + 1]$.Depending upon method used,there are several types pooling operations. However, we are using Max Pooling.In Max-Pooling, the largest element is taken from feature map. When added to a model, max pooling layer reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.



**Representation of max-pooling layer working**

The total sum of elements in a predefined sized image section. The pooling layer usually serves as a bridge between the convolutional layer and the Fully-Connected layer.



**3.10. Max pooling layer working on feature maps**

**Fully connected layer**

The fully connected layer (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form few layers of a CNN Architecture

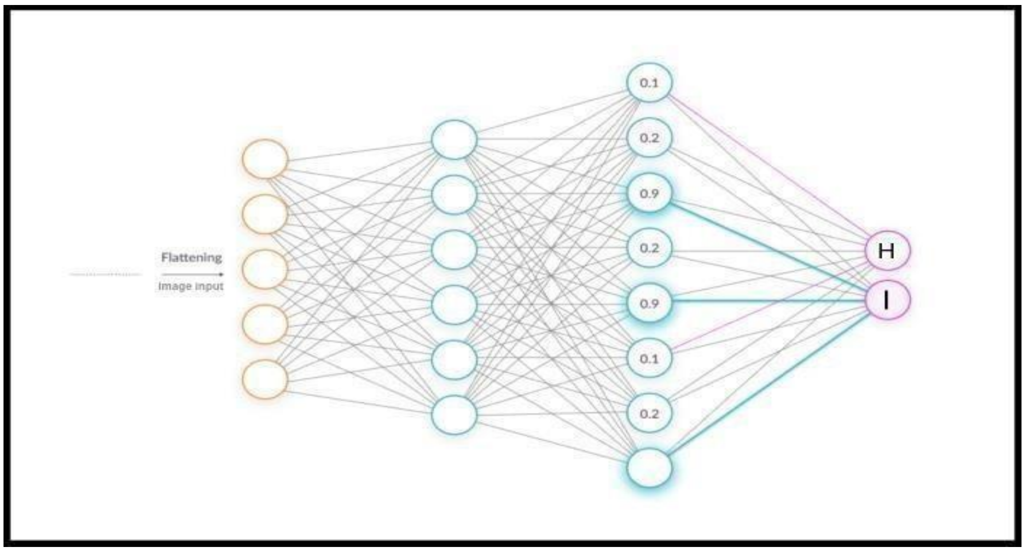
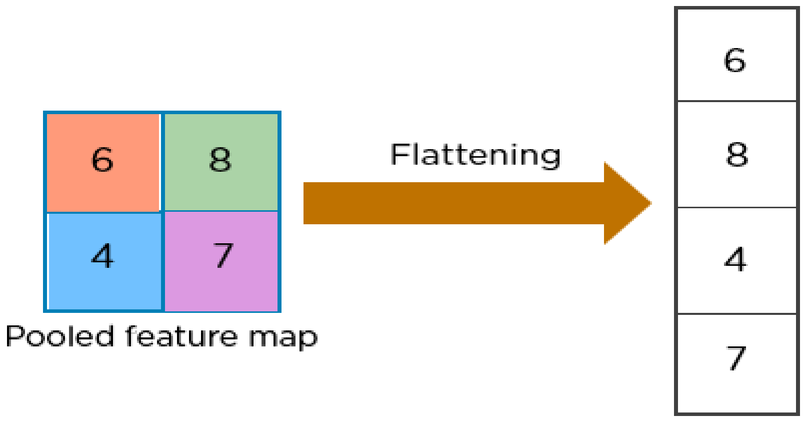


**Fig.3.11. Featuring of FC layer**

In this, the input image from previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, Classification process begins to take place.



**Flattening of pooling layer output to vertical vector**

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long featurevector. And it is connected to the final classification model, which is called a fully-connected layer.

## 3.10. Activation Functions

Finally, one of the most important parameters of the CNN model is the Activation function.

In simple words, it decides which information of the information of the model should fire in the forward direction and which ones should not end of the network.

Here we used ReLU activation function, which maps all negative values to zero and considering only positive values.

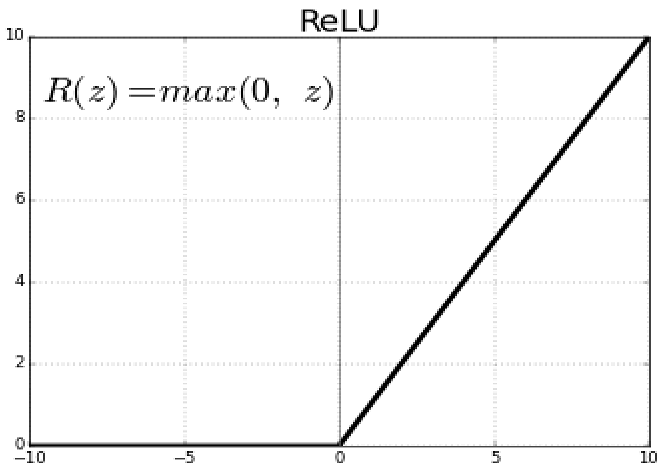


**Fig.3.12. Functioning of ReLU Activation Layer**

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.

## 3.1. LAYERS IN CNN MODEL

Conv2D
MaxPooling2D
Flatten()
Dropout
Dense

1. **Conv2D Layer:**

It has 100 filters and the activation function used is the ‘ReLu’. The ReLu function stands for Rectified Linear Unit which will output the input directly if it is positive, otherwise it will output zero.

2. **MaxPooling2D:**

It is used with pool size or filter size of 2*2.

3. **Flatten () Layer:**

It is used to flatten all the layers into a single 1D layer.

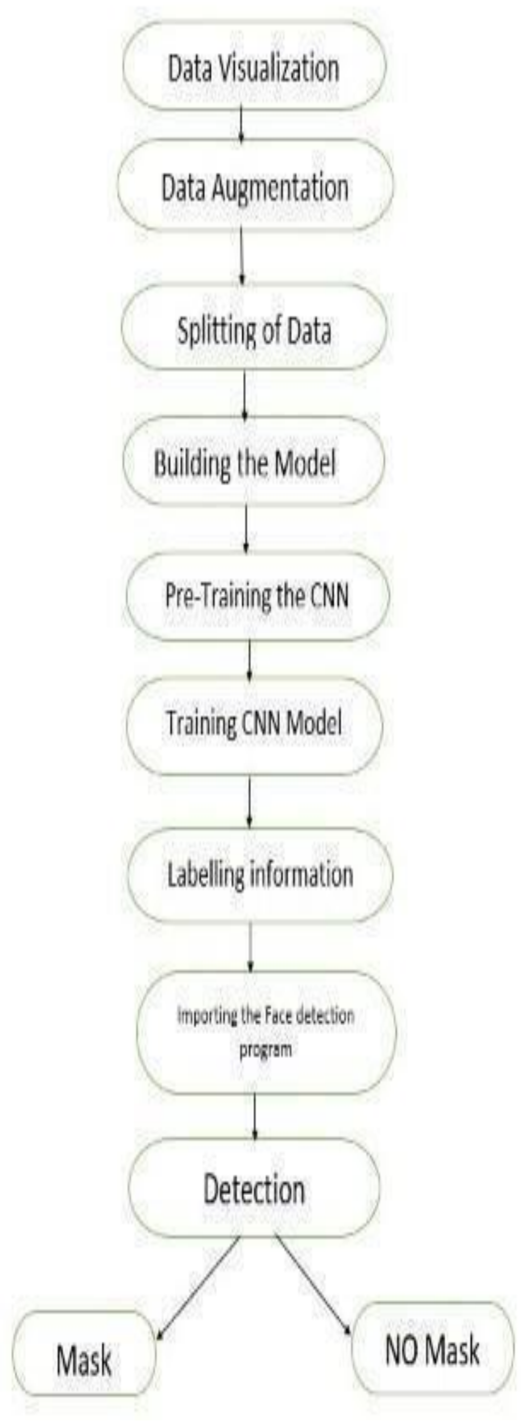
4. **Dropout Layer:**

It is used to prevent the model from overfitting.

5. **Dense Layer:**

The activation function here is soft max which will output a vector with twoprobability distribution values.

## 3.2. SYSTEM ARCHITECTURE



**3.13. System Architecture**

Data Visualization.

Data Augmentation.

Splitting the data.

Labeling the Information.

Importing the Face detection.

Detecting the Faces with and without Masks.

**Data Visualization**

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are ***690*** images in the '***yes***' class and ***686*** images in the '***no***' class.

**Data Augmentation**

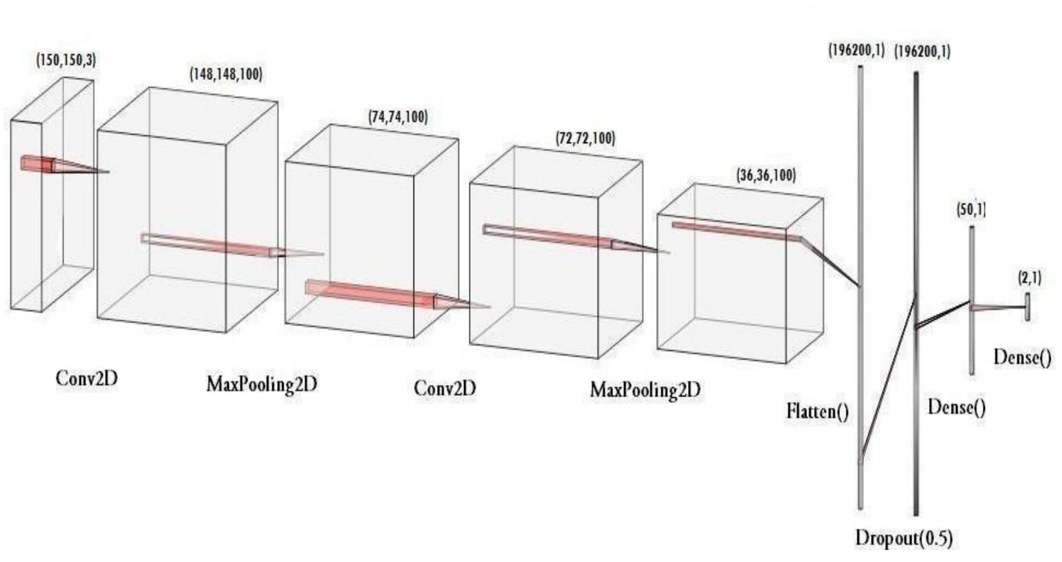
In the next step, we ***augment*** our dataset to include more number of images for our training. In this step of *data augmentation*, we *rotate* and *flip* each of the images in our dataset.

**Splitting the data**

In this step, we ***split*** our data into the ***training set*** which will contain the images on which the CNN model will be trained and the ***test set*** with the images on which our model will be tested.

In the next step, we build our ***Sequential CNN model*** with various layers suchas

*Conv2D, MaxPooling2D, Flatten, Dropout* and *Dense*.



## 3.14. Pre-Training the CNN model

After building our model, let us create the '***train_generator***' and '***validation_generator***' to fit them to our model in the next step.

### Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model we built using ***keras*** library. I have trained the model for ***30 epochs*** (iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs *over-fitting*.

### Labeling the Information

After building the model, we label two probabilities for our results. ***['0'** as **'without_ ask'** and **'1'** as **'with_ mask']***. I am also setting the boundary rectangle color using the RGB values.

**Importing the Face detection Program**

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the *Haar Feature-based Cascade Classifiers* for detecting the features of the face.

**Detecting the Faces with and without Masks**

In the last step, we use the OpenCV library to run an infinite loop to use our webcamera in which we detect the face using the *Cascade Classifier*.

# CHAPTER – 4
# DESIGN

## 4.1 UML Diagrams

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines. the various kinds of diagrams are not strictly enforced. At the same time, somepreclude mixing of different kinds of diagrams,
e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

## 4.2.Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

Scenarios in which your system or application interacts with people, organizations, or external systems.

Goals that your system or application helps those entities (known as actors) achieve.

The scope of your system

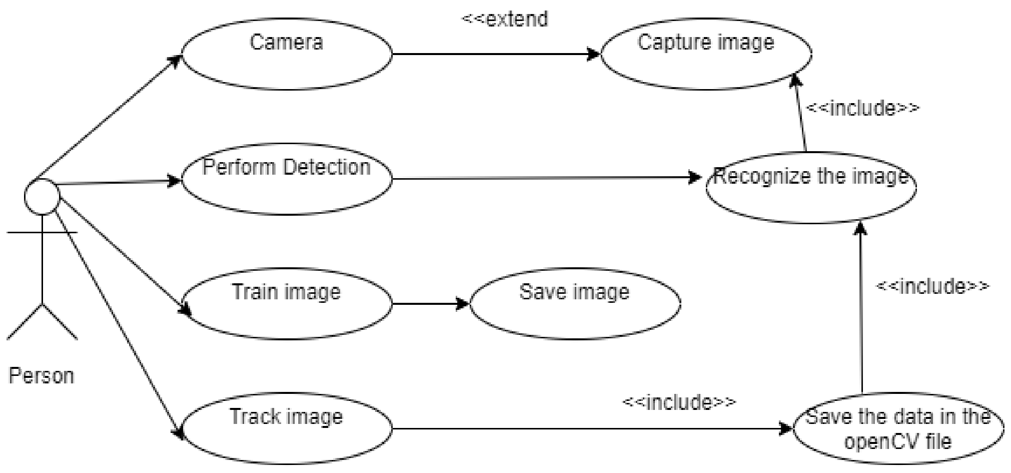


**Fig. 4.1. Use Case Diagram**

## 4.3 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations.

Trydrawing a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

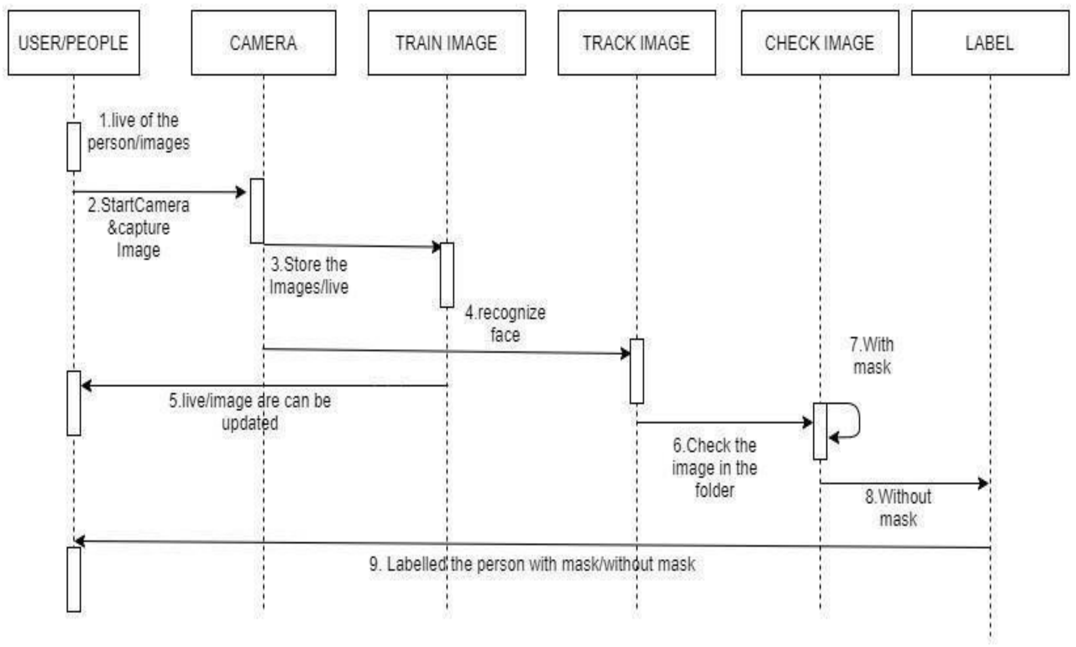

USER/PEOPLE
CAMERA
TRAIN IMAGE
TRACK IMAGE
CHECK IMAGE
LABEL
1.live of the person/images
2.StartCamera &capture Image
3.Store the Images/live
4.recognize face
5.live/image are can be updated
6.Check the image in the folder
7.With mask
8.Without mask
9. Labelled the person with mask/without mask

**Fig 4.2. Sequence Diagram**

## 4.4.Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system.An activity diagram portrays the control flow from a start point to a finish point showing various decision paths that exist while the activity is being executed.

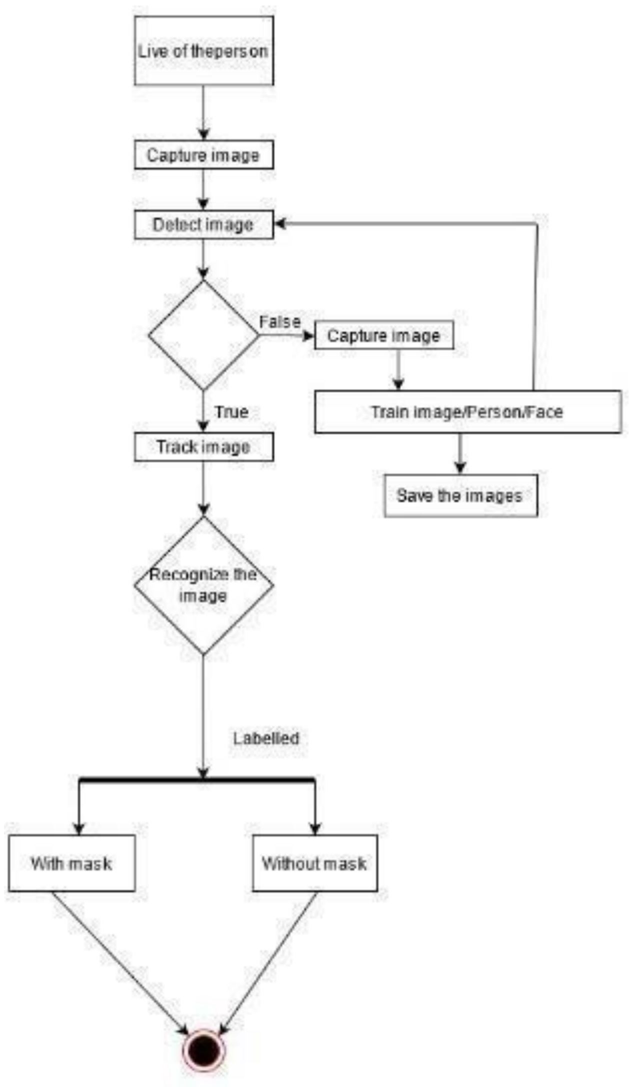


**Fig. 4.3. Activity Diagram**

## 4.5 . Block Diagram

A block diagram is a graphical representation of a system – it provides a functional view of a system. Block diagrams give us a better understanding of a system’s functions and help create interconnections within it.

They are used to describe hardware and software systems as well as to represent processes.

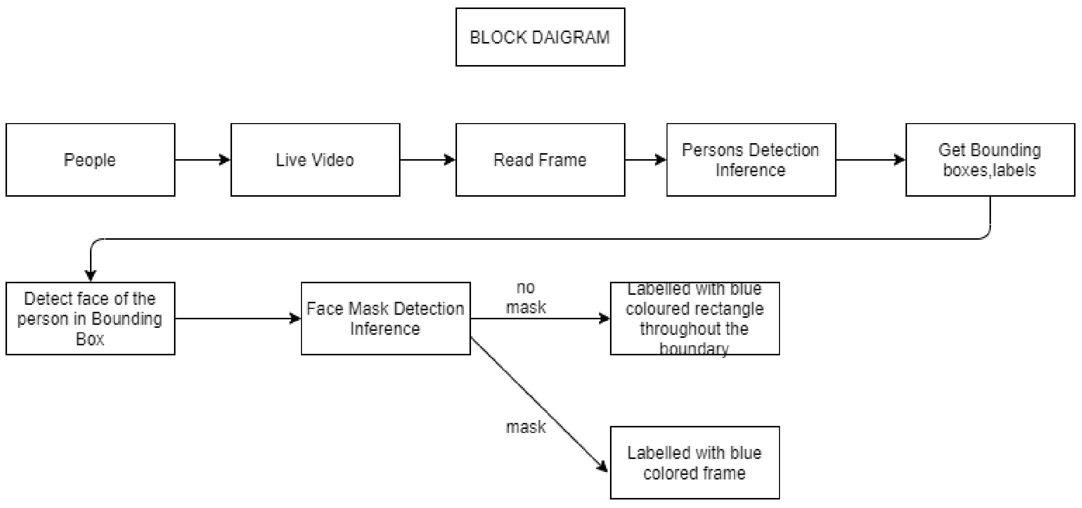

BLOCK DAIGRAM
People
Live Video
Read Frame
Persons Detection Inference
Get Bounding boxes,labels
Detect face of the person in Bounding Box
Face Mask Detection Inference
no mask
Labelled with blue coloured rectangle throughout the boundary
mask
Labelled with blue colored frame

**Fig 4.4. Block Diagram**

## 4.6 Class Diagram

Class diagram is a static diagram. It represents the static view of an application.

Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

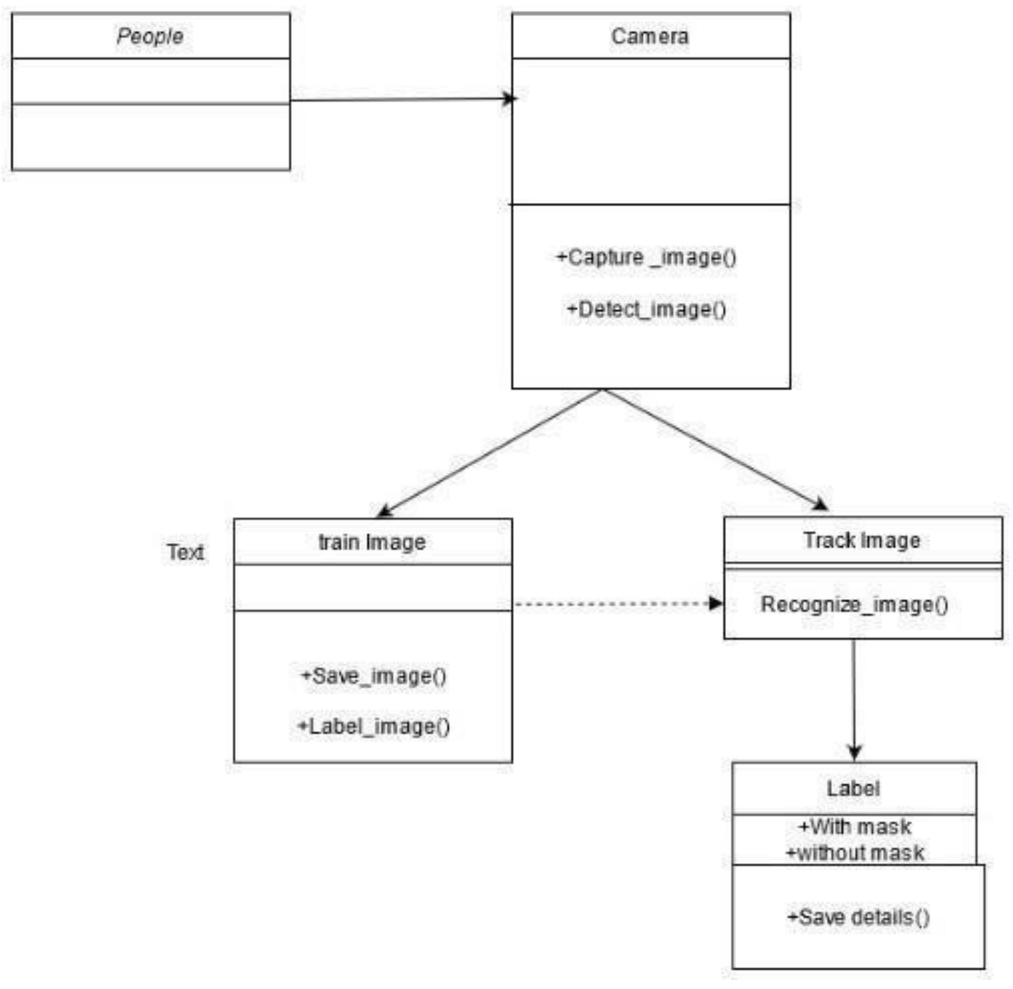


**Fig 4.5.Class Diagram**

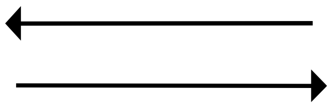
## 4.7. Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

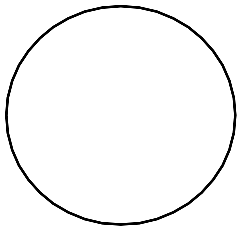
It shows how data enters and leaves the system, what changes the information, and where data is stored.A graphical tool for defining and analyzing minute data with an active or automated system, including process, data stores, and system delays. Data Flow Data is a key and basic tool for the architecture of all other objects. Bubble-bubble or data flow graph is another name for DFD.

DFDs are a model of the proposed system. They should indicate the requirements on which the new system should be built in a clear and direct manner. This is used as a basis for building chart structure plans over time during the design process. The following is the Basic Notation for building DFDs:

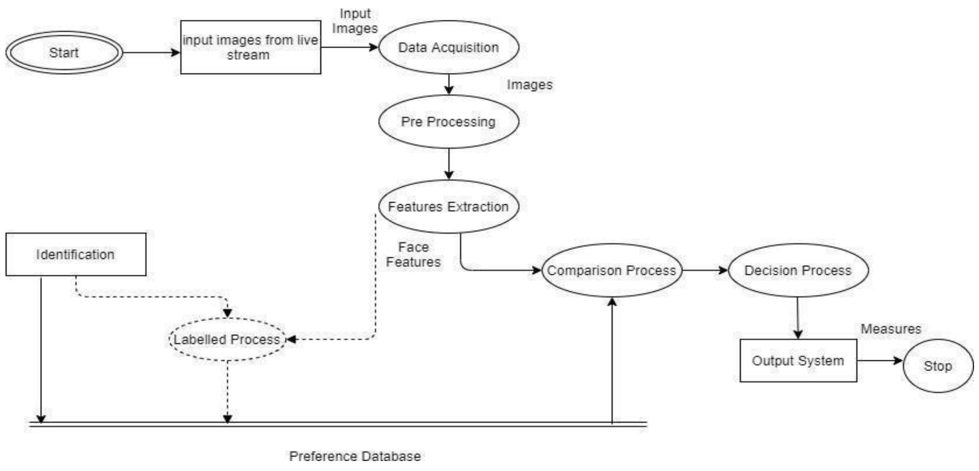
1. **Dataflow:**Data flows in a certain direction from the source to the destination.



2. **Process**: People, processes, or technologies that use or produce(Transforming) Information No information about a body part.



3. **Source:** People, programs, organizations, and other things can be external data sources or locations.



## 4.8. Flowchart Diagram

A Flowchart Diagram is a traditional visual representation of the information flows within a system. A neat and clear flowchart diagram can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

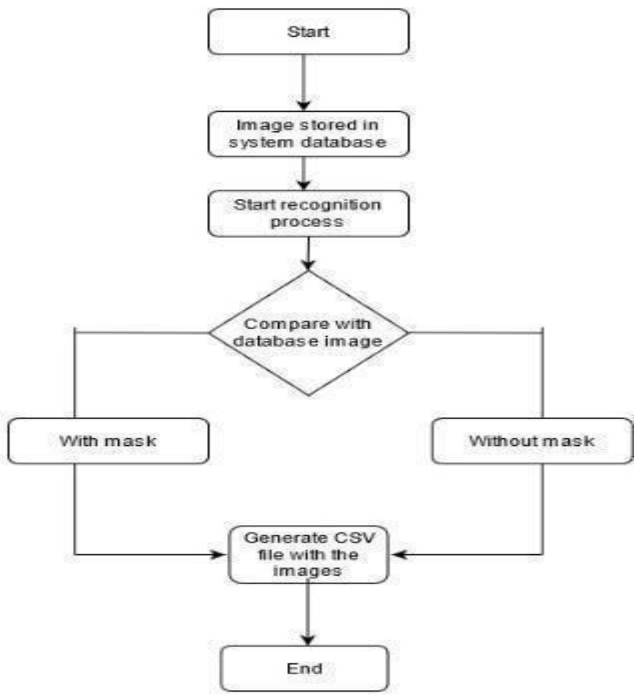

Start
Image stored in system database
Start recognition process
Compare with database image
With mask
Without mask
Generate CSV file with the images
End

**Fig 4.6.Flow Chart Diagram**

# CHAPTER-5

# EXPERIMENT ANALYSIS

## 5.1. Module Description

The entire project is divided into several modules in which they are dependent on one other. Each module perform unique work in the implementation ofthe project. they are:

1.DataCollection

2.DataPreprocessing

3.Model Creation

4.Model Training

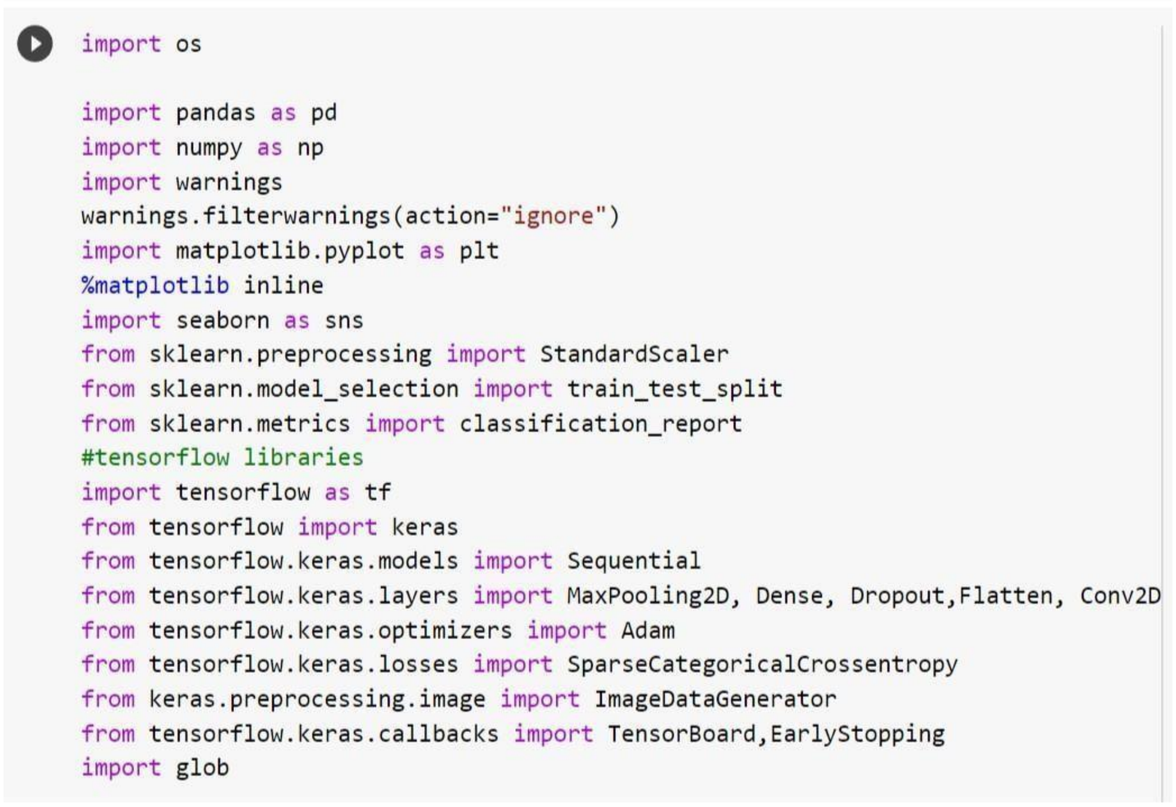
5.Model Evaluation

## 5.2. Dataset

Dataset consists of 7563 images in which 3725 images with people wearing face masks and the rest 3838 images with people who do not wear face masks. The below fig mostly contains front face pose with single face in the frame and with same type of mask having white color only.

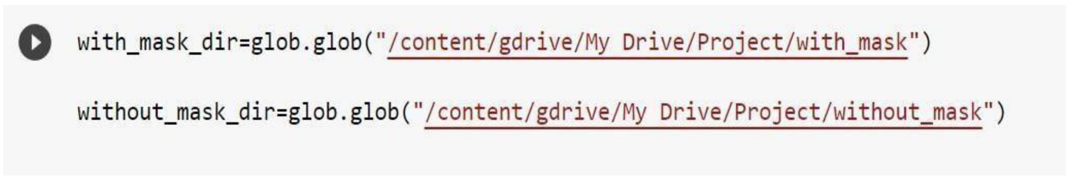
## 5.3. Source Code:

### 5.3.1. Data Collection: Importing libraries:



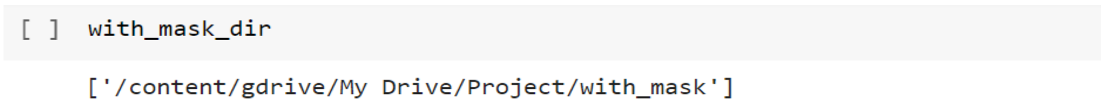
```
import os

import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings(action="ignore")
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
#tensorflow libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import MaxPooling2D, Dense, Dropout,Flatten, Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import TensorBoard,EarlyStopping
import glob
```

### Dataset:

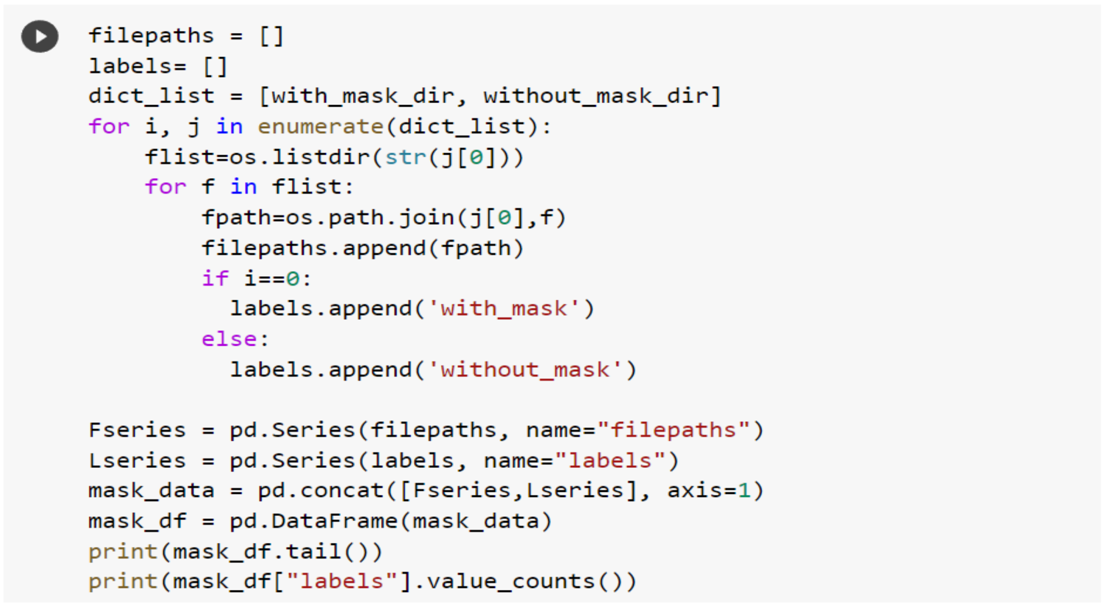


```
with_mask_dir=glob.glob("/content/gdrive/My Drive/Project/with_mask")

without_mask_dir=glob.glob("/content/gdrive/My Drive/Project/without_mask")
```

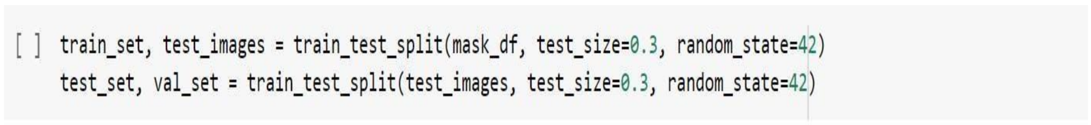
## 5.3.2. Data Preprocessing



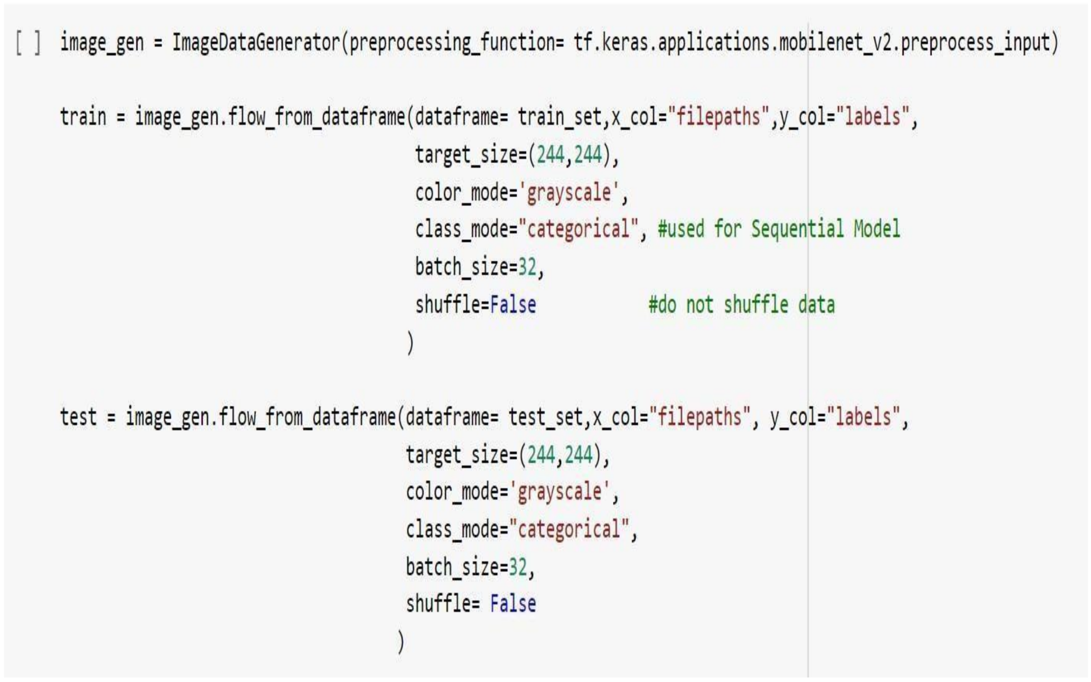
```
[ ]  with_mask_dir

     ['/content/gdrive/My Drive/Project/with_mask']
```



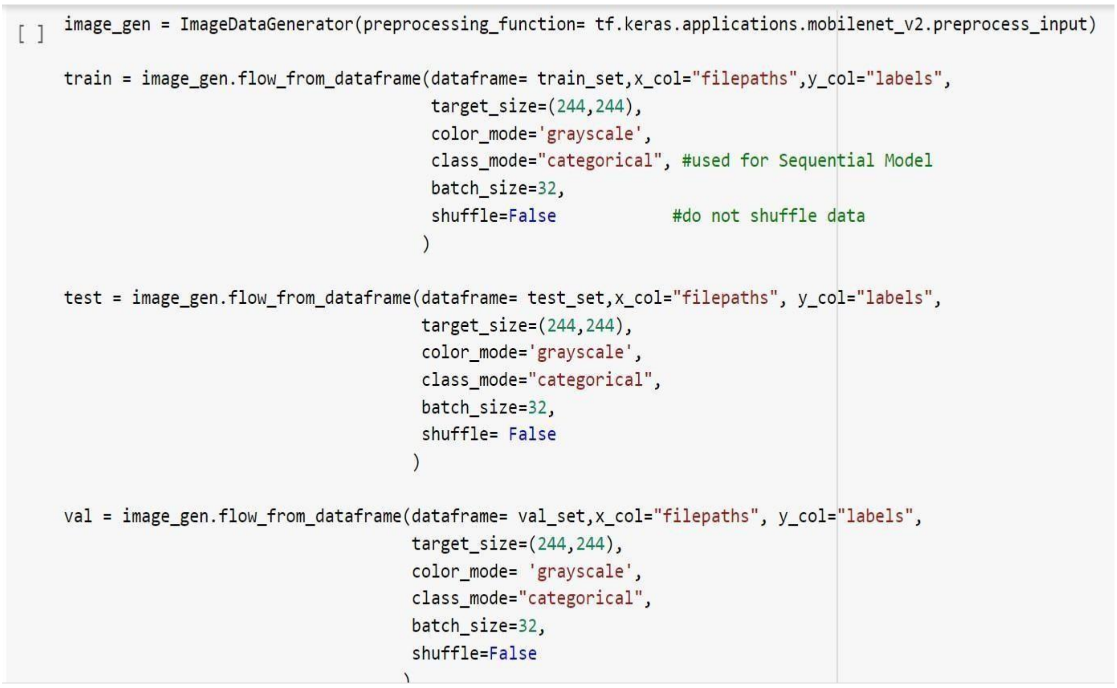
```
filepaths = []
labels= []
dict_list = [with_mask_dir, without_mask_dir]
for i, j in enumerate(dict_list):
    flist=os.listdir(str(j[0]))
    for f in flist:
        fpath=os.path.join(j[0],f)
        filepaths.append(fpath)
        if i==0:
          labels.append('with_mask')
        else:
          labels.append('without_mask')

Fseries = pd.Series(filepaths, name="filepaths")
Lseries = pd.Series(labels, name="labels")
mask_data = pd.concat([Fseries,Lseries], axis=1)
mask_df = pd.DataFrame(mask_data)
print(mask_df.tail())
print(mask_df["labels"].value_counts())
```



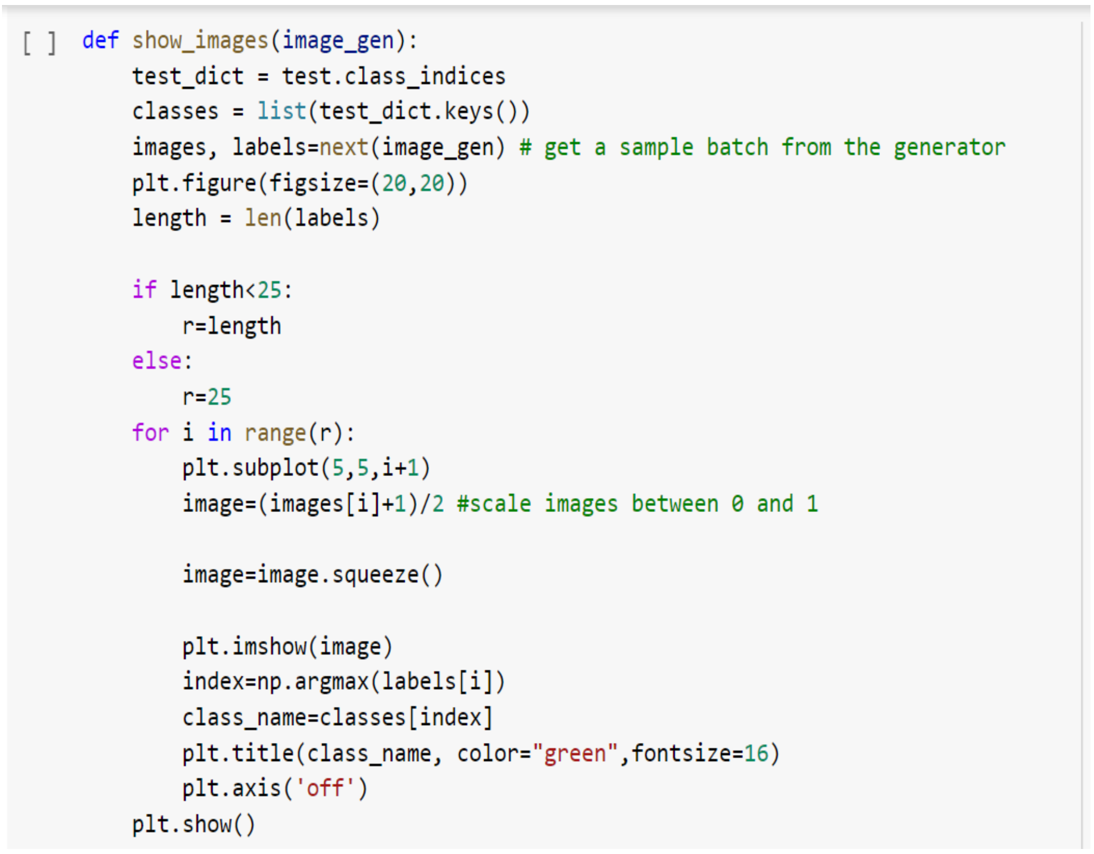
```
[ ]  train_set, test_images = train_test_split(mask_df, test_size=0.3, random_state=42)
     test_set, val_set = train_test_split(test_images, test_size=0.3, random_state=42)
```



```
[ ]  image_gen = ImageDataGenerator(preprocessing_function= tf.keras.applications.mobilenet_v2.preprocess_input)

     train = image_gen.flow_from_dataframe(dataframe= train_set,x_col="filepaths",y_col="labels",
                                           target_size=(244,244),
                                           color_mode='grayscale',
                                           class_mode="categorical", #used for Sequential Model
                                           batch_size=32,
                                           shuffle=False            #do not shuffle data
                                          )

     test = image_gen.flow_from_dataframe(dataframe= test_set,x_col="filepaths", y_col="labels",
                                          target_size=(244,244),
                                          color_mode='grayscale',
                                          class_mode="categorical",
                                          batch_size=32,
                                          shuffle= False
                                         )
```

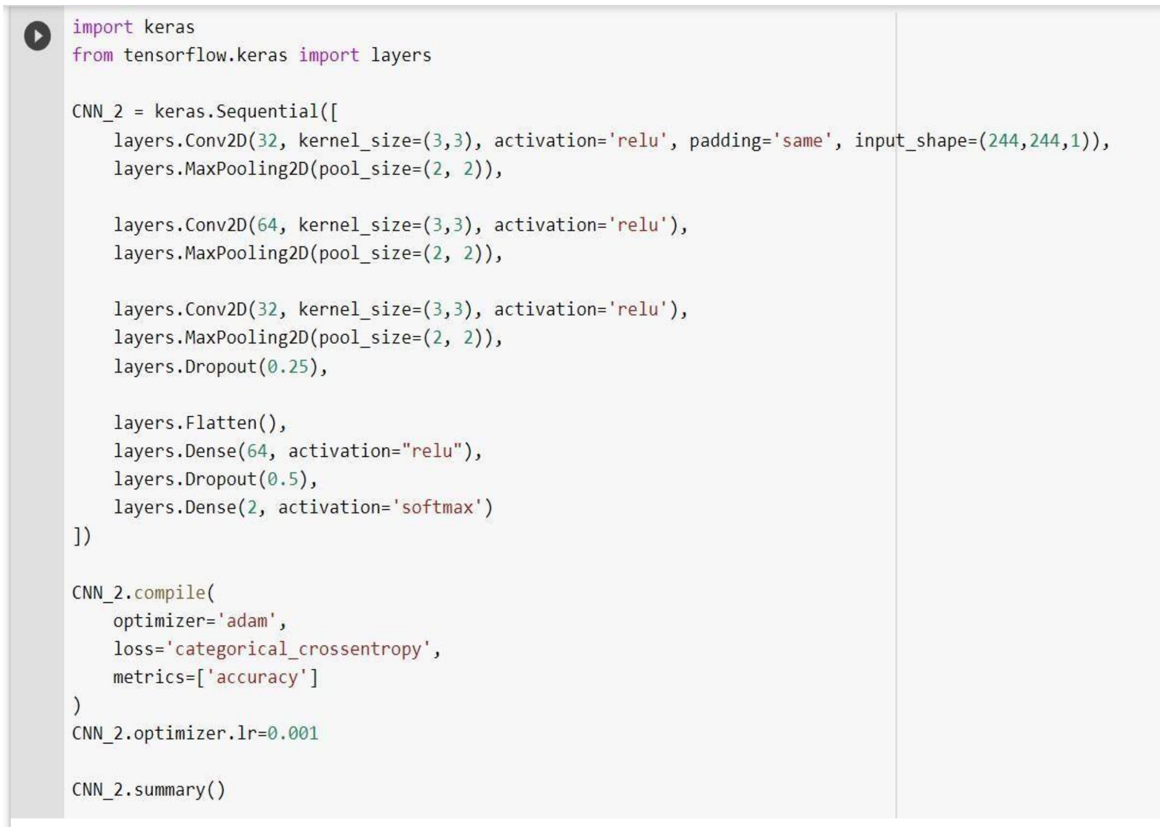


```
[ ]  image_gen = ImageDataGenerator(preprocessing_function= tf.keras.applications.mobilenet_v2.preprocess_input)

     train = image_gen.flow_from_dataframe(dataframe= train_set,x_col="filepaths",y_col="labels",
                                           target_size=(244,244),
                                           color_mode='grayscale',
                                           class_mode="categorical", #used for Sequential Model
                                           batch_size=32,
                                           shuffle=False            #do not shuffle data
                                          )

     test = image_gen.flow_from_dataframe(dataframe= test_set,x_col="filepaths", y_col="labels",
                                          target_size=(244,244),
                                          color_mode='grayscale',
                                          class_mode="categorical",
                                          batch_size=32,
                                          shuffle= False
                                         )

     val = image_gen.flow_from_dataframe(dataframe= val_set,x_col="filepaths", y_col="labels",
                                         target_size=(244,244),
                                         color_mode= 'grayscale',
                                         class_mode="categorical",
                                         batch_size=32,
                                         shuffle=False
```



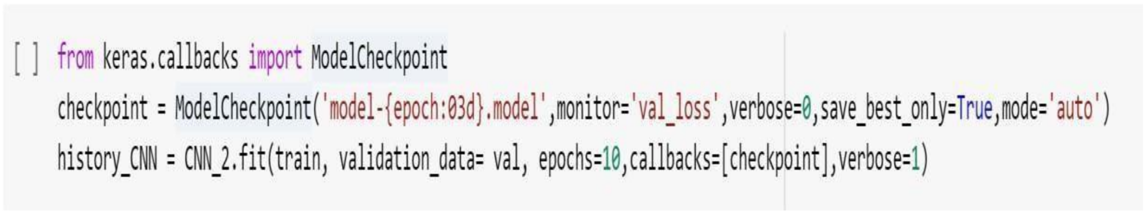
```
[ ]  def show_images(image_gen):
         test_dict = test.class_indices
         classes = list(test_dict.keys())
         images, labels=next(image_gen) # get a sample batch from the generator
         plt.figure(figsize=(20,20))
         length = len(labels)

         if length<25:
             r=length
         else:
             r=25
         for i in range(r):
             plt.subplot(5,5,i+1)
             image=(images[i]+1)/2 #scale images between 0 and 1

             image=image.squeeze()

             plt.imshow(image)
             index=np.argmax(labels[i])
             class_name=classes[index]
             plt.title(class_name, color="green",fontsize=16)
             plt.axis('off')
         plt.show()
```

## 5.3.3 Model Creation:



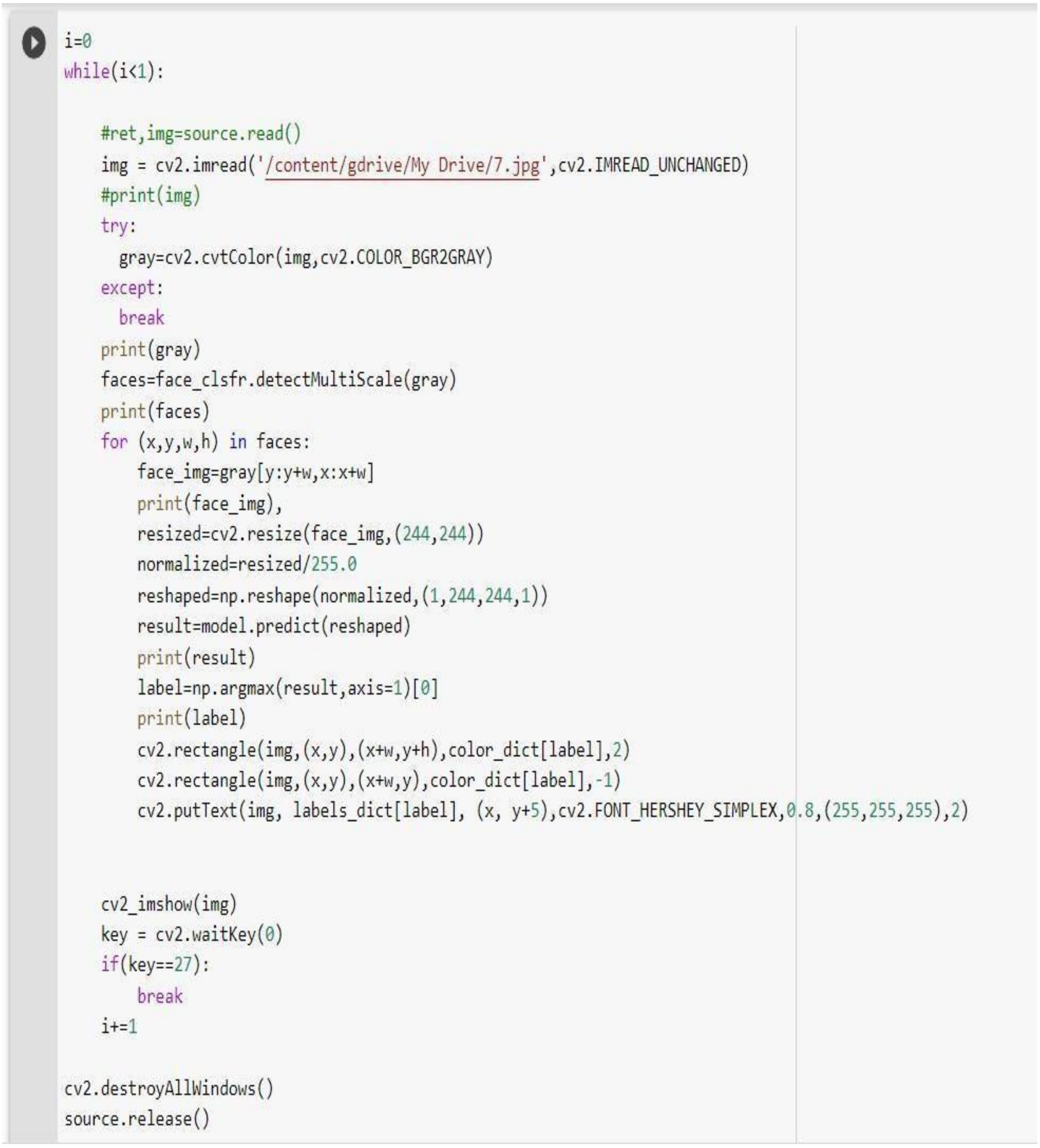
```
import keras
from tensorflow.keras import layers

CNN_2 = keras.Sequential([
    layers.Conv2D(32, kernel_size=(3,3), activation='relu', padding='same', input_shape=(244,244,1)),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),

    layers.Conv2D(32, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Dropout(0.25),

    layers.Flatten(),
    layers.Dense(64, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(2, activation='softmax')
])

CNN_2.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
CNN_2.optimizer.lr=0.001

CNN_2.summary()
```

## 5.3.4. Model Training:



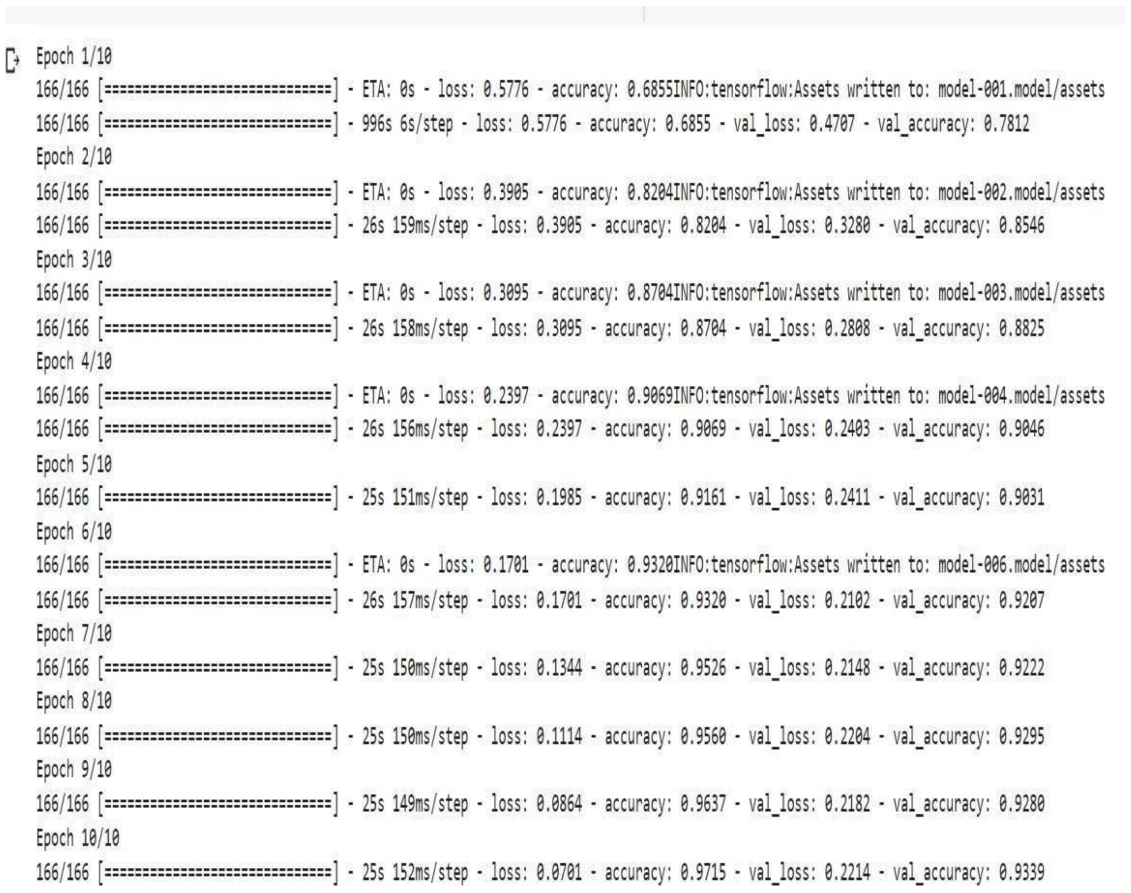
```
[ ] from keras.callbacks import ModelCheckpoint
    checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
    history_CNN = CNN_2.fit(train, validation_data= val, epochs=10,callbacks=[checkpoint],verbose=1)
```

## 5.3.5. Model Testing:



```
i=0
while(i<1):

    #ret,img=source.read()
    img = cv2.imread('/content/gdrive/My Drive/7.jpg',cv2.IMREAD_UNCHANGED)
    #print(img)
    try:
      gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    except:
      break
    print(gray)
    faces=face_clsfr.detectMultiScale(gray)
    print(faces)
    for (x,y,w,h) in faces:
        face_img=gray[y:y+w,x:x+w]
        print(face_img),
        resized=cv2.resize(face_img,(244,244))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,244,244,1))
        result=model.predict(reshaped)
        print(result)
        label=np.argmax(result,axis=1)[0]
        print(label)
        cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(img,(x,y),(x+w,y),color_dict[label],-1)
        cv2.putText(img, labels_dict[label], (x, y+5),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)


    cv2_imshow(img)
    key = cv2.waitKey(0)
    if(key==27):
        break
    i+=1

cv2.destroyAllWindows()
source.release()
```

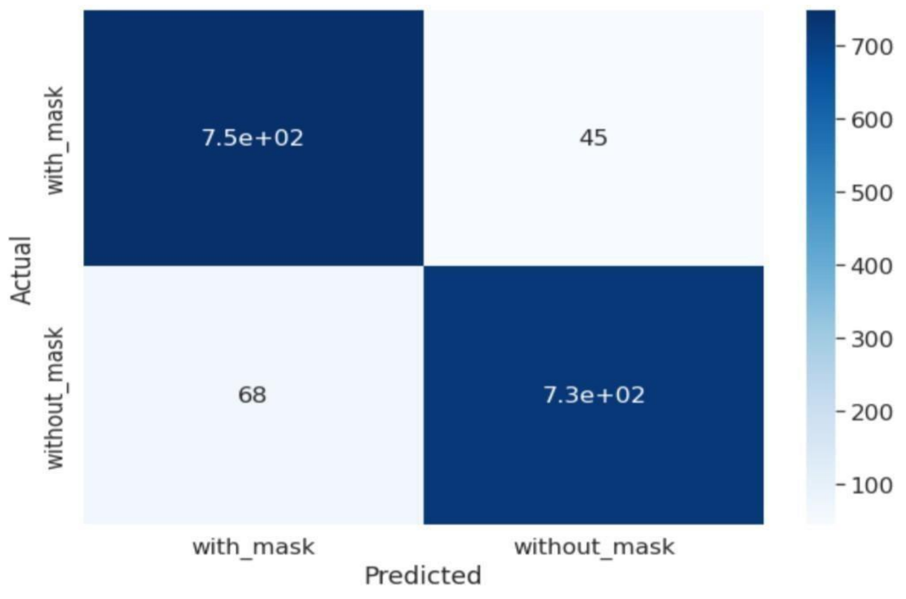
## 5.4. Sample Output:



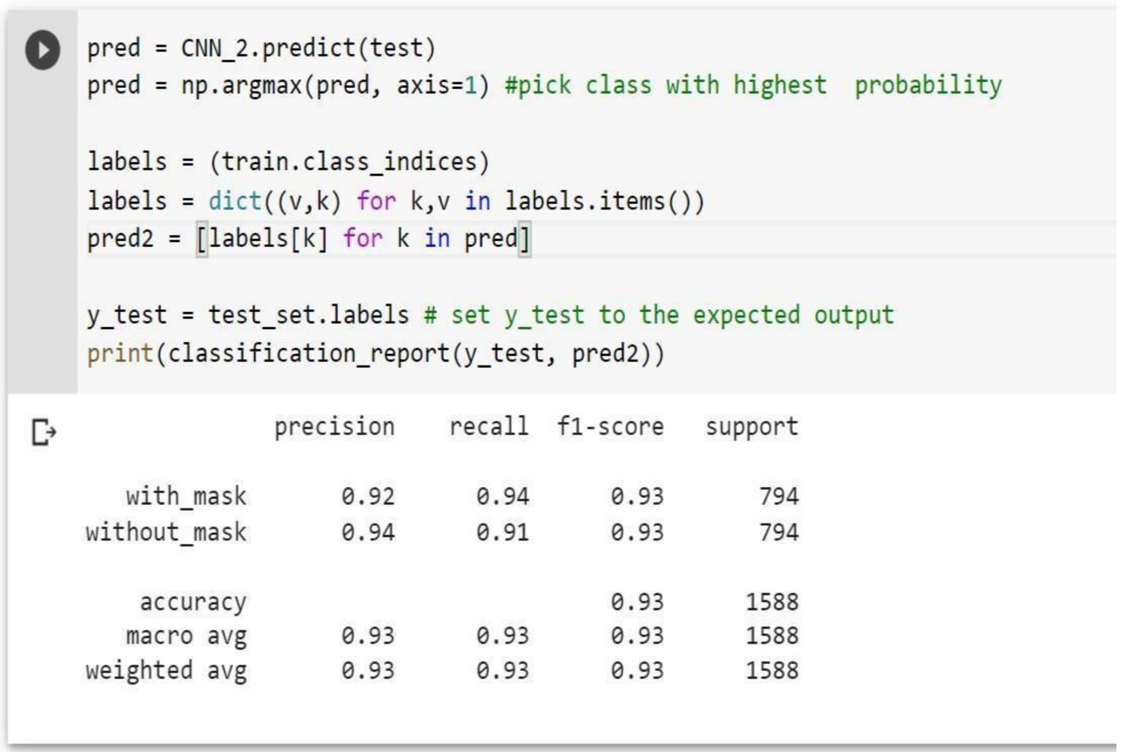
```
Epoch 1/10
166/166 [==============================] - ETA: 0s - loss: 0.5776 - accuracy: 0.6855INFO:tensorflow:Assets written to: model-001.model/assets
166/166 [==============================] - 996s 6s/step - loss: 0.5776 - accuracy: 0.6855 - val_loss: 0.4707 - val_accuracy: 0.7812
Epoch 2/10
166/166 [==============================] - ETA: 0s - loss: 0.3905 - accuracy: 0.8204INFO:tensorflow:Assets written to: model-002.model/assets
166/166 [==============================] - 26s 159ms/step - loss: 0.3905 - accuracy: 0.8204 - val_loss: 0.3280 - val_accuracy: 0.8546
Epoch 3/10
166/166 [==============================] - ETA: 0s - loss: 0.3095 - accuracy: 0.8704INFO:tensorflow:Assets written to: model-003.model/assets
166/166 [==============================] - 26s 158ms/step - loss: 0.3095 - accuracy: 0.8704 - val_loss: 0.2808 - val_accuracy: 0.8825
Epoch 4/10
166/166 [==============================] - ETA: 0s - loss: 0.2397 - accuracy: 0.9069INFO:tensorflow:Assets written to: model-004.model/assets
166/166 [==============================] - 26s 156ms/step - loss: 0.2397 - accuracy: 0.9069 - val_loss: 0.2403 - val_accuracy: 0.9046
Epoch 5/10
166/166 [==============================] - 25s 151ms/step - loss: 0.1985 - accuracy: 0.9161 - val_loss: 0.2411 - val_accuracy: 0.9031
Epoch 6/10
166/166 [==============================] - ETA: 0s - loss: 0.1701 - accuracy: 0.9320INFO:tensorflow:Assets written to: model-006.model/assets
166/166 [==============================] - 26s 157ms/step - loss: 0.1701 - accuracy: 0.9320 - val_loss: 0.2102 - val_accuracy: 0.9207
Epoch 7/10
166/166 [==============================] - 25s 150ms/step - loss: 0.1344 - accuracy: 0.9526 - val_loss: 0.2148 - val_accuracy: 0.9222
Epoch 8/10
166/166 [==============================] - 25s 150ms/step - loss: 0.1114 - accuracy: 0.9560 - val_loss: 0.2204 - val_accuracy: 0.9295
Epoch 9/10
166/166 [==============================] - 25s 149ms/step - loss: 0.0864 - accuracy: 0.9637 - val_loss: 0.2182 - val_accuracy: 0.9280
Epoch 10/10
166/166 [==============================] - 25s 152ms/step - loss: 0.0701 - accuracy: 0.9715 - val_loss: 0.2214 - val_accuracy: 0.9339
```

**Actual Vs Predicted :**

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fef8b2da290>
```



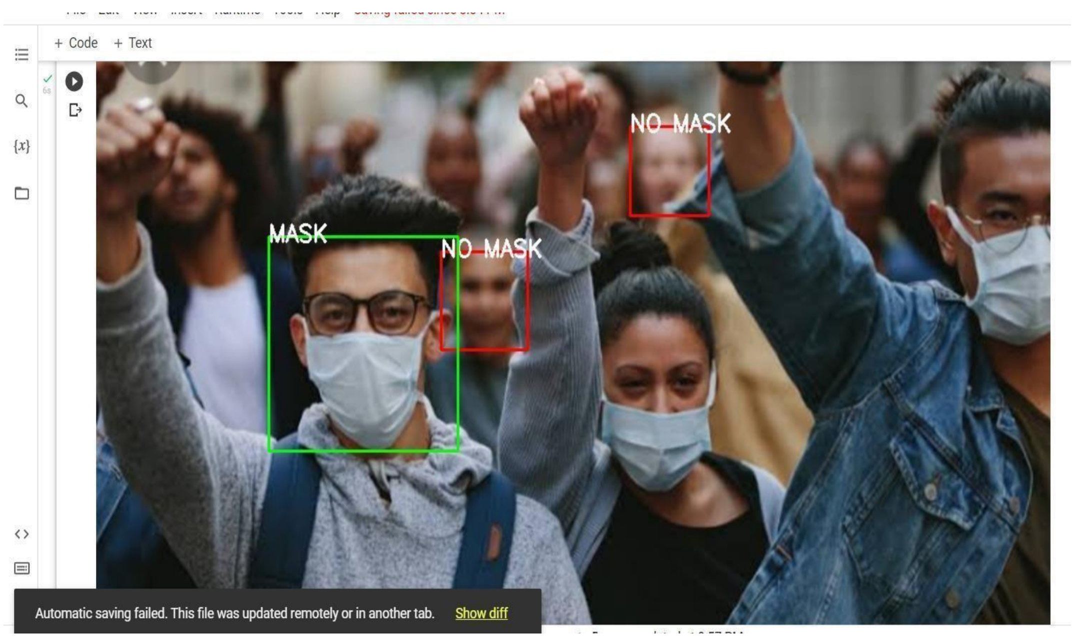
## Predicted Output:



```
pred = CNN_2.predict(test)
pred = np.argmax(pred, axis=1) #pick class with highest  probability

labels = (train.class_indices)
labels = dict((v,k) for k,v in labels.items())
pred2 = [labels[k] for k in pred]

y_test = test_set.labels # set y_test to the expected output
print(classification_report(y_test, pred2))
```

```
              precision    recall  f1-score   support

   with_mask       0.92      0.94      0.93       794
without_mask       0.94      0.91      0.93       794

    accuracy                           0.93      1588
   macro avg       0.93      0.93      0.93      1588
weighted avg       0.93      0.93      0.93      1588
```

## Output of With Mask and Without Mask:



Automatic saving failed. This file was updated remotely or in another tab. Show diff

# CONCLUSION

The main goal of face mask detection system is that automatically identifies if a person is wearing a mask or not. which is an excellent solution for deterring the spread of the COVID-19 pandemic. By using Keras and CNN, the proposed system is able to detect the presence or absence of a face mask and the model gives precise and quick results. The trained model yields an accuracy of around 91%-93%. It can be used for a variety of applications. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

# REFERENCES

[1]T. Ojala M. Pietikainen and T. Maenpaa "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns" IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 24 no. 7 pp. 971-987 July 2002

[2]T.-H. Kim D.-C. Park D.-M. Woo T. Jeong and S.-Y. Min "Multi-class classifier-based adaboost algorithm" Proceedings of the Second Sino- foreign-interchange Conference on Intelligent Science and Intelligent Data Engineering pp. 122-127 2012.

[3]P. Viola and M. J. Jones "Robust real-time face detection" Int. J. Comput. Vision vol. 57 no. 2 pp. 137-154 May 2004.

[4]P. Viola and M. Jones "Rapid object detection using a boosted cascade of simple features" Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 vol. 1 pp. I-I Dec 2001.

[5]J. Li J. Zhao Y. Wei C. Lang Y. Li and J. Feng "Towards real world human parsing: Multiple-human parsing in the wild" CoRR.

[6] Guangchengwang, yumiao"Masked face recognition data sets and application" National natural sciencefoundation of china 2020

[7] Raza Ali, Saniya Adeel,Akhyar Ahmed Face Mask Detector July 2020

[8]Z.-Q. Zhao, P. Zheng, S.-t.Xu, and X. Wu, Object detection with deep learning IEEE transactions onneural networks and learningsystems 2019.

[9] Amit Chavda, Jason Dsouza,SumeetBadgujar Multi-Stage CNN Architecture for Face Mask DetectionSeptember 2020

[10] Amrit Kumar, Bhadani,Anurag Sinha A Facemask detector using machine learning and image processing techniques November 2020 Engineering scienceAnd technology and international confernce.

[11] Sammy v. militante,Nanettev.dionisio*Real time face mask recognition with alarm system using deep learning* 2020 11th IEEE control and system graduate research colloquium

[12] Mohammad marufurmd.Motalebhossenmd.MilonislamSaifuddinmahmud An automated system to limit covid 19 using facial mask detection in smart city network 2020 IEEE international IOT

[13] Toshanlalmeenpal,Ashutoshbalakrishnan,Amitverma, Face mask detection using semantic segmentation 2019,4th international conference on 4th Internationalcomputing, communications and security(ICCCS)

[14] Wenyunsun,Yusong,Changsheng,Face spoofing detection based on local ternary label supervision in fully convolutional networks IEE transactions on information forensis and security 2020.